Surfel-Based Next Best View Planning

Original
Surfel-Based Next Best View Planning / Monica, Riccardo; Aleotti, Jacopo. - In: IEEE ROBOTICS AND AUTOMATION LETTERS. - ISSN 2377-3766. - 3:4(2018), pp. 3324-3331. [10.1109/LRA.2018.2852778]

Availability:
This version is available at: 11381/2849407 since: 2021-10-11T10:59:27Z

Publisher:
Institute of Electrical and Electronics Engineers Inc.

Published
DOI:10.1109/LRA.2018.2852778

Surfel-Based Next Best View Planning

Riccardo Monica and Jacopo Aleotti

Abstract—**Next best view (NBV) planning is a central task for automated three-dimensional (3-D) reconstruction in robotics. The most expensive phase of NBV computation is the view simulation step, where the information gain of a large number of candidate sensor poses are estimated. Usually, information gain is related to the visibility of unknown space from the simulated viewpoint. A well-established technique is to adopt a volumetric representation of the environment and to compute the NBV from ray casting by maximizing the number of unknown visible voxels. This letter explores a novel approach for NBV planning based on surfel representation of the environment. Surfels are oriented surface elements, such as circular disks, without explicit connectivity. A new kind of surfel is introduced to represent the frontier between empty and unknown space. Surfels are extracted during 3-D reconstruction, with minimal overhead, from a KinectFusion volumetric representation. Surfel rendering is used to generate images from each simulated sensor pose. Experiments in a real robot setup are reported. The proposed approach achieves better performance than volumetric algorithms based on ray casting implemented on GPU, with comparable results in terms of reconstruction quality. Moreover, surfel-based NBV planning can be applied in larger environments as a volumetric representation is limited by GPU memory.**

Q1

Index Terms—**Autonomous agents, range sensing, motion and path planning, computer vision for other robotic applications.**

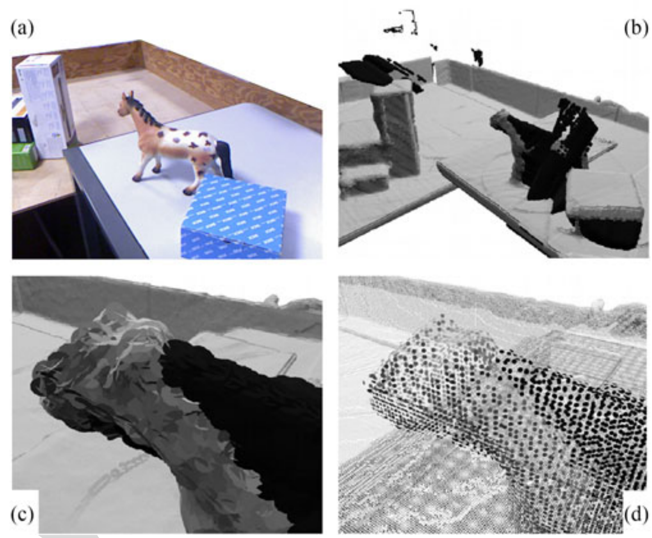


Fig. 1. (a) RGB image of the scene. (b) Incomplete surfel-based representation, with surface surfels (gray shading) and frontier surfels (frontels, black). (c) A closeup view. (d) Closeup view with surfel size reduced to 20%.

I. Introduction

A NEXT Best View (NBV) algorithm computes the best viewpoint of a depth sensor, mounted on a robot, to improve the knowledge of the environment by maximizing the expected information gain. Typical NBV algorithms operate iteratively in two phases: viewpoint generation and viewpoint evaluation. In the first phase, the free configuration space of the robot is explored to retrieve a set of candidate sensor poses. In the second phase, a view is simulated from each candidate sensor pose, given the current model of the environment, and then the most promising viewpoint is selected. The goal of the simulation is to estimate the amount of unknown space visible from the view pose, which in turn predicts the information gain. The view simulation phase is usually the most computationally expensive operation, which may limit the number of poses that can be evaluated in a reasonable time.

Unlike most works that address the NBV problem by using a volumetric representation of the environment, in this work a novel NBV evaluation strategy is proposed that exploits a surfel-based representation. A surfel is a surface element represented as a circular disk without explicit connectivity, described by geometric attributes such as position, radius, normal and color [1]. Multiple surfels may be assembled to describe a surface. Research interest on surfels has risen significantly as a set of surfels can be processed like a point cloud but it contains more information. A surfel cloud is also simpler than a polygon mesh data structure and it requires less memory than a volumetric representation. As an example, the ElasticFusion algorithm [2] has been proven capable of GPU-accelerated real-time 3D reconstruction on a surfel cloud.

In this letter, a novel kind of surfel is introduced, named frontel, in order to represent the frontier between empty and unknown space (Fig. 1). We exploit previous work [3] where the KinFu Large Scale algorithm, an open source implementation of KinectFusion [4] in the Point Cloud Library, was modified to reliably keep track of both empty and unknown voxels. Further changes have been made in this work to generate surfels and frontels in real-time during the KinFu reconstruction process, with minimal overhead. In the view simulation phase, frontels and surfels are rendered to simulate a depth image.

The proposed NBV approach was evaluated in a real setup including a robot arm with a Kinect sensor in eye-in-hand configuration. Results are compared with volumetric methods, where viewpoint evaluation was accelerated on GPU thanks to KinFu internal ray casting. The experimental evaluation shows a

Q2 Manuscript received February 23, 2018; accepted June 17, 2018. Date of publication; date of current version. This letter was recommended for publication by Associate Editor A. Argyros and Editor T. Asfour upon evaluation of the reviewers' comments. *(Corresponding author: Riccardo Monica.)*

The authors are with the RIMLab, Department of Engineering and Architecture, University of Parma, Parma I-43100, Italy (e-mail: riccardo.monica@unipr.it; jacopo.aleotti@unipr.it).

Digital Object Identifier 10.1109/LRA.2018.2852778

significant performance improvement for the surfel-based NBV algorithm, with a similar quality in 3D reconstruction. Most of the performance improvement is due to the opportunity to fit the whole 3D representation on GPU memory at once, which is not possible with a volumetric representation. Indeed, in volumetric NBV approaches, when observing different regions of space, parts of the environment must be loaded from CPU to GPU memory before viewpoint evaluation. Similarly, due to limited GPU memory, another advantage of surfel-based NBV planning against volumetric approaches is that it can be applied in larger environments. The paper is organized as follows. Related work is reported in Section II. In Section III, the proposed surfel-based NBV algorithm is described. Experiments and results are reported in Section IV. Finally, Section V concludes the work.

II. Related Work

Next best view planning is a well established area of research in robotics [5]–[7]. Here we mainly review recent results, with the aim of showing that the use of volumetric approaches, requiring computationally intensive ray casting operations for viewpoint evaluation, remains prevalent. Most works focus on non model-based methods that do not assume an a priori model of the scene, while model based NBV planning requires both modeling and object recognition [8]. In [9] an approach was proposed which takes into account the position uncertainty of the sensor and that adopts a utility function that considers several factors like perception of unseen areas, navigation distance, reconstruction quality and fast occlusion estimation. Potthast *et al.* [10] presented a variant of the NBV problem that builds a belief model of the unobserved space for cluttered environments with occlusion. In [11] a frontier-oriented algorithm based on volumetric hierarchical ray tracing was introduced to speed up NBV evaluation.

In mobile robotics the view planning problem is constrained in the 2D plane [9], [12]–[18]. Senarathne *et al.* [12] presented a method based on surface frontiers, i.e. the boundary voxels of mapped surfaces adjacent to unmapped space. In [14] an adaptable and probabilistic object reconstruction approach was proposed to evaluate the information gain. An optimization was also introduced, based on a lookup table, for the evaluation of multiple view orientations from the same viewpoint. Isler *et al.* [15] exploited a probabilistic volumetric map and investigated an algorithm that estimates whether a ray is expected to hit the backside of already observed surfaces. Delmerico *et al.* [16] evaluated different information gain metrics for NBV planning using a probabilistic voxel map in terms of completeness and entropy. The comparative analysis indicated that the utility function defined in [9] achieved the best performance, as well as a metric that gives higher weights to unobserved voxels close to already observed surfaces. Patten *et al.* [17] developed a model based system for outdoor active object classification using a mobile robot. Monte Carlo methods for planning new observations were adopted with time and distance constraints together with a non-parametric Bayesian regression classifier.

NBV planning has also been applied to aerial robotics. In [19] a receding horizon method was introduced based on sampling a random tree of candidate viewpoints. The method scales better than a frontier-based planner in large environments. In [20] a 2-stage planning solution was proposed that aims at achieving full coverage of the environment and global optimality of the exploration path.

Surface based approaches for NBV planning were investigated using triangular meshes [21]–[23]. Viewpoints were generated by examining the boundaries of the reconstructed surfaces. These methods do not model the surface between empty and unknown space, and, therefore cannot select the next best view by estimating information gain from unknown volume. Volumetric representations are also preferred for probabilistic approaches [15].

III. Method

The robot task is to perform a 3D reconstruction of the volume around a set of Points Of Interest (POIs), in a tabletop scenario, by taking a sequence of observations. It is assumed that the set of POIs is given as input to the system. For example, a POI may indicate the location of an object or a group of objects to be scanned. At the beginning of the task the robot has an initial, possibly incomplete, representation of the environment and the volume around each POI is cleared (set to unknown), as described in Section III-B. The NBV planning procedure is then executed. At each iteration candidate viewpoints are sampled on a view sphere with fixed radius around each POI, oriented towards the POI itself. In the view simulation phase, surfels and frontels are rendered to a virtual view (Section III-C). Then a score is computed from the rendered image (Section III-D). The score represents the expected information gain for the simulated sensor pose. The poses are attempted using a motion planner in decreasing order of scores. The first feasible solution is executed by the robot.

In the proposed method surfels and frontels are generated on GPU in real-time during robot observations from the KinFu internal representation. Both surfels s and frontels f are circular disks in 3D space. Surfels separate empty from occupied space, while frontels separate empty from unknown space. In particular, surfel/frontel generation is performed during truncated signed distance function (TSDF) volume shifting operations to optimize performance (Section III-A). The TSDF volume is the volumetric representation of the environment used by the KinFu algorithm. As KinFu is able to keep in GPU memory only a limited volume, if the whole environment does not fit inside the TSDF volume KinFu Large Scale shifts the TSDF volume by unloading and loading parts of the TSDF volume from and to GPU memory according to sensor movements.

A. Real-Time Surfel and Frontel Generation

The TSDF volume is organized as a regular voxel grid. The TSDF assigns to each voxel the distance to the nearest surface, negative in occupied space and positive in empty space. The distance is truncated in the interval $[-v_{\max}, v_{\max}]$. Each voxel $c\,(x_c, y_c, z_c)$ holds a TSDF value v_c, i.e. the value of the TSDF at the center of the voxel, and a weight w_c which contains the number of times the voxel has been observed, up to a maximum. KinFu operates on a cubic TSDF volume with edge length $E = c_{\max} e$, where $c_{\max}$ is the volume resolution and e is the voxel size. Whenever a point is observed by the sensor, the voxels on the ray between the observation and the sensor position are updated. However, only the volume inside the TSDF is updated by KinFu. The Large Scale extension of KinFu performs a shifting operation when the distance between the TSDF volume center and a virtual point, at about $E/2$ distance from the sensor along the z axis, is greater than $E/3$. A shifting operation translates the TSDF volume such that it represents a new region in space, centered at the the virtual point. Shifting ensures that the region currently observed by the sensor is the one represented on GPU and that it can be updated. Information contained in the intersection between the old TSDF volume and the new one is kept in GPU memory. Voxels with $w_c > 0$ and $v_c < v_{\max}$ outside the new volume are downloaded from GPU memory to RAM. Then, voxels are converted into points, described by position c and TSDF value v_c, and added to a TSDF point cloud. Points are reloaded to TSDF volume if a subsequent shifting operation moves the volume back in the old region. Due to memory constraints, empty voxels with $v_c = v_{\max}$ are not downloaded by the aforementioned procedure. Therefore, in our previous work [3] an octree Ω was added to store w_c when $v_c = v_{\max}$.

Surfels s are circular disks in 3D space, with position p_s, radius r_s and normal n_s. Color is not available in KinFu, but it is not required for the proposed NBV score computation. Similarly, frontels f have position p_f, radius r_f and normal n_f. The surface between the outside and the inside of objects is located in the TSDF volume between voxels having a different sign of v. That is, an empty voxel c is near the surface of an object if the following conditions hold:

$$\begin{cases} v_c \geq 0 \quad \wedge \quad w_c > 0, \\ \exists\, c' \in N_6(c) \mid v_{c'} < 0,\ w_{c'} > 0 \end{cases} \tag{1}$$

where $N_6(c)$ is the 6-neighborhood of voxel c. Similarly, the frontier between empty and unknown space is located where w changes from 0 to a positive value, i.e.

$$\begin{cases} v_c \geq 0 \quad \wedge \quad w_c > 0, \\ \exists\, c' \in N_6(c) \mid w_{c'} = 0 \end{cases} \tag{2}$$

Before a TSDF volume shifting operation occurs, surfels and frontels are generated from the old TSDF volume. A surfel centered on voxel c is generated, i.e. $p_s = c$, if condition (1) holds, whereas a frontel $p_f = c$ is generated if condition (2) holds. Generated surfels/frontels are added to two sets on CPU RAM, here named Σ and Φ respectively. To avoid duplication, existing elements of Σ and Φ inside the region represented by the old TSDF volume are cleared. These elements may have been generated by previous shifting operations in the same region.

The surfel local normal is estimated from the TSDF local gradient, according to the signed distance function properties, as follows (normalization omitted):

$$n_s(c) = \sum_{c' \in N_6} v_{c'} \frac{c' - c}{\|c' - c\|} \tag{3}$$

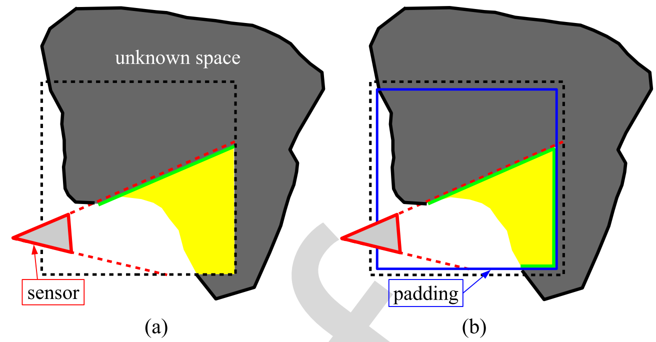


Fig. 2. The current position of the TSDF volume is marked with a black dashed square. The sensor observes towards the right, clearing the yellow unknown area. The green line indicates where frontels are generated (a) without and (b) with padding.

Conversely, a frontel normal is estimated from the 26-neighborhood N_{26} of the voxel c, as follows (normalization omitted):

$$K_w(c) = \begin{cases} 1 & \text{if } w_c > 0 \\ -1 & \text{if } w_c = 0 \end{cases} \tag{4}$$

$$n_f(c) = \sum_{c' \in N_{26}} K_w(c') \frac{c' - c}{\|c' - c\|} \tag{5}$$

The 26-neighborhood was chosen over the 6-neighborhood to reduce the quantization error. Indeed, unlike v_c, $K_w(c)$ as in (4) can assume only two values. Normal n_f is oriented from unknown space towards known space. Equations (1) and (2) place the surfel at the center of an empty voxel next to an occupied or unknown voxel. Therefore, occupied and unknown volumes are enlarged by $\delta_e = \frac{1}{2}e$ per side, i.e half the KinFu voxel edge e. To generate a surface without holes, the surfel/frontel radius must be set as half the distance between the two farthest vertices of a voxel, i.e. $\frac{\sqrt{3}}{2}(e + 2\delta_e)$. Therefore, the surfel/frontel radius may be approximated as $r_s = r_f = \sqrt{3}e$.

The surfel/frontel generation procedure [(1) and (2)] requires knowledge about the neighboring voxels. Voxels in contact with the TSDF volume surface are considered as a special case covered by reducing the active TSDF volume by a one-voxel-wide padding. The padding is downloaded and uploaded from/to GPU as usual, but it is not updated with new data acquired by the sensor. An example of this issue is provided in Fig. 2. In case (a), without padding, the TSDF volume has been set to empty up to its right surface. Therefore, frontels are not generated in that area, leaving a hole in the surface enclosing unknown space. In case (b), the unknown values near the TSDF volume surface are not removed and frontels are correctly generated.

B. Initialization of the Regions of Interest for the NBV Task

Each POI defines a spherical region with origin p_{poi} and radius R_{poi}. At the beginning of the task each spherical region around a POI is cleared and set to unknown. To this purpose the TSDF volume and the surfel/frontel-based representation must be managed in a consistent way. In the part of the sphere currently inside the TSDF volume, w_c is set to 0 whenever $\|c - p_{\text{poi}}\| \leq R_{\text{poi}}$.

Outside the TSDF volume, the update procedure is executed as follows. First, surfels and frontels inside the sphere are deleted from Σ and Φ. Then, frontels must be generated in Φ between the cleared volume and the empty voxels in octree Ω. Let $c \in \Omega$ be a voxel outside the sphere, c has a known neighbor inside the sphere if the following conditions hold:

$$\begin{cases} \|c - p_{\text{poi}}\| > R_{\text{poi}} \quad \wedge \quad w_c > 0, \\ \exists\, c' \in N_6(c) \;\mid\; w_{c'} > 0,\; \|c' - p_{\text{poi}}\| \leq R_{\text{poi}} \end{cases} \tag{6}$$

Therefore, a new frontel is created at position $p_f = c$, with radius r_f and normal $n_f = (p_f - p_{\text{poi}})/\|p_f - p_{\text{poi}}\|$ when conditions in (6) hold. Finally, the voxels inside the sphere are set to unknown in Ω.

C. Surfel Rendering

The standard pinhole model is used for simulating the Kinect sensor. The depth image of a real sensor contains, for each pixel, the distance of the first intersection of a ray cast from the origin of the camera center passing through that pixel. Some pixels are left undefined, since the intersection may go below (or beyond) the minimum (or maximum) sensor range.

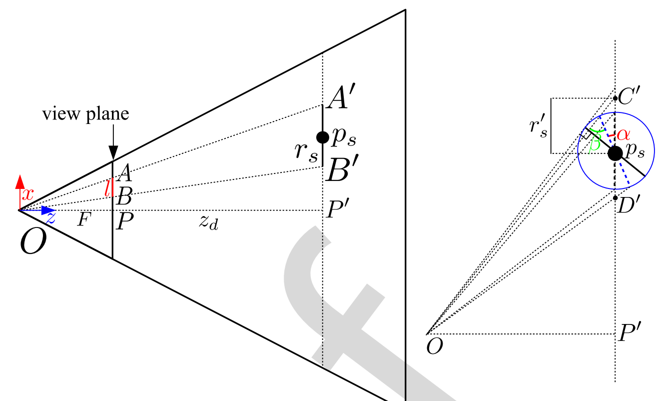
A sensor which follows the pinhole model can be simulated by rendering. For each surfel or frontel, position (3×32-bit floats), normal (3×32-bit floats) and radius (32-bit float) are provided to the rendering pipeline. In total, 28 bytes are required per surfel or frontel. In our approach, rendering generates an image that includes all information needed for NBV score computation. Unlike the real sensor, three possible results may occur for the virtual sensor pixels, i.e. a pixel is marked: a) occupied, if a surfel is rendered; b) unknown, if a frontel is rendered; c) out-of-range, if nothing is rendered or the surfel/frontel is out of the sensor range. A single real value o_{ij} is used for each pixel as the output of the rendering pipeline. Given the z_d depth in camera coordinates of the rendered surfel/frontel, the output value is set as:

$$o_{ij} = \begin{cases} z_d & \text{if (a)} \\ -z_d & \text{if (b)} \\ 0 & \text{if (c)} \end{cases} \tag{7}$$

Therefore, when rendered, surfels and frontels produce positive and negative depths respectively.

Image resolution, center point and focal length are known from the sensor intrinsic calibration. The maximum range of the sensor is used to set the frustum far plane, so that farther points are discarded. By setting background to 0, case (c) is automatically handled for points beyond maximum range. However, since surfels between the camera and minimum range occlude surfels behind them in the real sensor, they must be rendered with output value 0.

In the view evaluation phase TSDF volume shifting operations must be simulated. The predicted TSDF volume is assumed centered on the POI which generated the view. Surfels/frontels outside the predicted position of the TSDF volume are rendered with $o_{ij} = 0$, as when the real sensor measures range data



(a) (b)

Fig. 3. (a) Surfel s, with position p_s and radius r_s, is projected to the view plane. Square size $l \geq \overline{AB}$ must be estimated. (b) A case in which l is underestimated. Distance $\overline{OP}$ is reduced to highlight the effect.

outside the TSDF volume such data do not contribute to the 3D reconstruction.

A point-based rendering technique is adopted to efficiently render surfels as points [24]. The default OpenGL pipeline renders points as squares of size l facing the camera. A vertex shader is executed in parallel on GPU for each point sent to the rendering pipeline. In particular, each point is transformed into camera coordinates and the distance z_d is computed. The OpenGL pipeline is instructed to draw squares with size l by setting the variable `gl_PointSize` in the vertex shader. Each square may span multiple fragments, each corresponding to a pixel. For each fragment, a fragment shader is run by the OpenGL pipeline. The fragment shader computes the 3D distance of each fragment from the center of the surfel which generated it. The fragment is discarded if the distance is greater than the radius r_s.

Estimation of a suitable square size l in the vertex shader is a non-trivial task. Indeed, the size must be large enough to encompass the entire surfel. However, setting a too large square size would cause generation of many useless fragments, thus deteriorating performance. If the surfel is roughly parallel to the image plane [Fig. 3(a)], l can be estimated as follows. Triangles $\triangle OAB$ and $\triangle OA'B'$, and also $\triangle OAP$ and $\triangle OA'P'$ are similar, i.e.:

$$\frac{\overline{OP'}}{\overline{OP}} = \frac{\overline{OA'}}{\overline{OA}} = \frac{\overline{A'B'}}{\overline{AB}} \tag{8}$$

Therefore, given focal length F, edge l is estimated as:

$$l = \frac{2Fr_s}{z_d} \tag{9}$$

A similar estimate was performed in [24]. However, by using such formula, l may be underestimated as shown in Fig. 3(b), in a 2D example. This results in incompletely rendered surfels, cut by the bounding square. In the following, we provide an upper bound for this under-estimation.

Surfel s is rotated by angle α so that it appears slightly bigger, i.e. the projection of its top-most point onto the plane through p_s, parallel to the view plane, is in C' instead of A'. A similar effect happens for the lower-most pixel, which now projects in D'. However, the error for D' is smaller, due to the minor distance from the sensor principal axis. The circle displayed around the surfel in Fig. 3(b) represents the possible positions of the top-most and lower-most points, by varying α values. The maximum distance $\overline{A'C'}$ occurs when the surfel is perpendicular to $C'O$, i.e. $\beta = \pi/2$. In this case, $\alpha = \angle P'OC'$. Therefore, the projected surfel radius $r'_s = \overline{p_s C'}$ is upper bounded as follows:

$$r'_s \leq \frac{r_s}{\cos\left(\angle P'OC'\right)} \tag{10}$$

However, $\angle P'OC'$ can be at most equal to the sensor Half Field Of View (HFOV). Therefore, for a Kinect-like sensor, with HFOV $\approx 30°$, $\cos(\text{HFOV}) = \sqrt{3}/2 \approx 0.866$. Hence, in the worst case a surfel is reduced by less than 14% its size.

D. NBV Score Computation

Standard NBV score computation methods assign a score equal to the number of unknown voxels visible from each pose. The more unknown voxels are visible, the higher is the information gain expected from that pose. In this letter we propose a novel score function where the total area of visible frontels is used to approximate the number of unknown voxels. Since frontels are generated to cover the area of the frontier between empty and unknown space, the total area of visible frontels is (approximately) proportional to the number of voxels in the frontier. The number γ_r of sensor view rays which intersect a frontier is computed from the rendering output (7) as:

$$U_o\left(o_{ij}\right) = \begin{cases} 1 & \text{if } o_{ij} < 0 \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

$$\gamma_r = \sum_{ij} U_o\left(o_{ij}\right) \tag{12}$$

However, each frontel f contributes to γ_r proportionally to its projected area $\mathcal{A}_f^p$. Therefore, frontels close to the camera contribute excessively. This high contribution is not consistent with the standard score function that counts the number of unknown visible voxels. Indeed, in the standard score function a voxel close to the camera counts as 1, like any other voxel. In the following, a more appropriate score function is defined by introducing a weighting factor.

Equation (9) in Section III-C relates distance z_d and radius r_f of a frontel (roughly parallel to the view plane) to the diameter l of its projection on the view plane. Also, due to (7), $z_d = |o_{ij}|$. Hence, the projected frontel area in pixels can be estimated as:

$$\mathcal{A}_f^p = \pi\left(\frac{l}{2}\right)^2 = \pi\left(\frac{1}{2}\,\frac{2Fr_f}{z_d}\right)^2 = \left(\frac{F}{|o_{ij}|}\right)^2 \mathcal{A}_f^v \tag{13}$$

where $A_f^v = \pi r_f^2$ is the actual frontel area. The weighted NBV score function is then defined as:

$$\gamma = \sum_{ij} U_o\left(o_{ij}\right)\gamma_{ij} \tag{14}$$

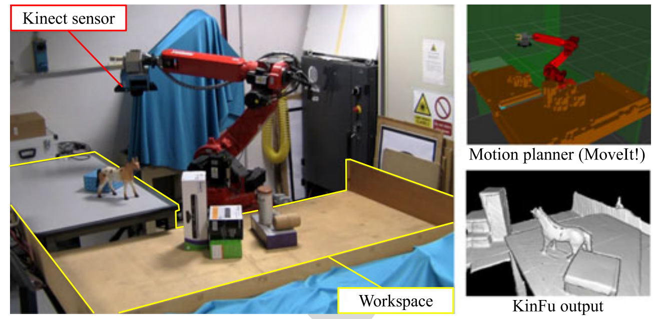


Fig. 4. The experimental setup.

where the weighting factor γ_{ij} is

$$\gamma_{ij} = \frac{\mathcal{A}_f^v}{\max\left\{\mathcal{A}_f^p, 1\right\}} \tag{15}$$

For frontels not too far away from the camera, (15) can be simplified by using (13) so that:

$$\gamma_{ij} = \frac{\mathcal{A}_f^v}{\mathcal{A}_f^p} = \left(\frac{|o_{ij}|}{F}\right)^2 \qquad \text{if } A_f^p \geq 1 \tag{16}$$

i.e. γ_{ij} is set as inversely proportional to the projected surfel area $\mathcal{A}_f^p$. Conversely, for distant frontels where the projection area A_f^p is smaller than a pixel, gain γ_{ij} becomes

$$\gamma_{ij} = \mathcal{A}_f^v = \pi r_f^2 \qquad \text{if } A_f^p < 1 \tag{17}$$

to prevent overestimation. From (15), in the general case, both distance o_{ij} and frontel radius r_f are required to compute area $\mathcal{A}_f^v$ and then gain γ_{ij}. In this work, the frontel radius is constant, as shown in Section III-A. However, our approach can be extended to a variable frontel radius by computing gain γ_{ij} in the fragment shader, where the radius is available.

IV. Experiments

A. Experimental Setup

The experimental setup (Fig. 4) includes an industrial robot arm (Comau SMART SiX) with six degrees of freedom. A Kinect sensor is mounted on the end-effector and calibrated with respect to the robot wrist. KinFu egomotion tracking is disabled and the sensor position is computed from robot forward kinematics. The workspace is a square of about 2.5×2.5 m that encompasses two tables in front of the robot (highlighted in yellow in Fig. 4). KinFu TSDF volume side is set to $E = 1.5$ m and the edge resolution is set to $c_{\text{max}} = 512$ cells. Therefore, the voxel edge length is set to $e = 2.9$ mm, and thus $r_f = 5.1$ mm. As E is lower than the workspace size, shifting operations must be performed while scanning different regions of interest. The same shifting operations must be simulated in the view evaluation phase, to accurately predict information gain. Each experiment is performed as follows. First, the robot scans the environment on a predefined path using KinFu. POIs are given as input to the system, located near the objects to be scanned. Regions of interest are initialized as unknown around each POI as described in Section III-B. Viewpoints are sampled

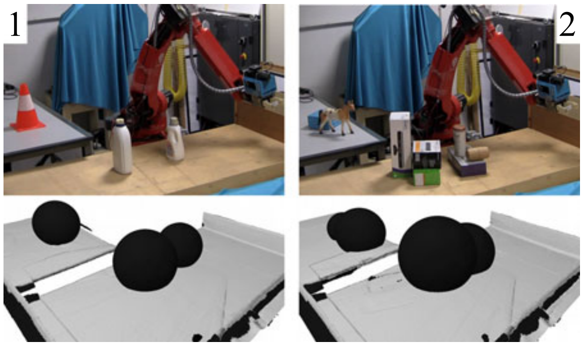


Fig. 5. Top: scenarios 1 and 2. Bottom: the spherical regions of interest centered at each POI, surrounded by black frontels.

TABLE I
EXPERIMENT INITIAL DATA

Scenario	1	2
Initial surfels	∼519k	∼489k
Initial frontels	∼275k	∼302k
Initial unknown	∼4309k	∼7608k

on spheres around each POI, at a fixed distance of 0.8 m. Each POI generates 960 candidate views, sampled at regular intervals, by varying latitude (10 intervals), longitude (12 intervals) and rotation around the sensor axis (8 intervals). Then, the NBV task is started. At every iteration, all viewpoints generated for each POI are evaluated by the NBV algorithm. The MoveIt! ROS framework is used for planning a collision free path towards the viewpoint with the highest predicted information gain. A collision map with precision 5 cm is extracted from KinFu to update the MoveIt! planning scene. The collision map considers both occupied and unknown voxels as obstacles. Viewpoints are attempted in decreasing score order, until the planner succeeds. The robot then moves the sensor in the selected NBV pose and new information is acquired. The Kinect is slightly tilted ($\pm 5^\circ$) to remove artifacts that otherwise would appear due to the emitted IR pattern. Indeed, if the sensor is fed multiple images from the same viewpoint in a static environment ripples appear in the 3D reconstruction. The Kinect depth image has a resolution of 640×480 pixels, with a focal length of about 528 pixels. However, when the sensor is simulated during NBV computation a 560×540 resolution is adopted, with the same focal length. The simulated camera height is increased to account for the tilting motion of the real sensor during acquisition, while the simulated camera width is reduced due to sensor calibration. The software runs on an Intel i7-6700 CPU at 3.40 GHz, 32 GB RAM, NVIDIA GeForce GTX 980 Ti GPU, 6 GB RAM.

Experiments were performed in two different scenarios, shown in Fig. 5. In the first scenario, three POIs were defined, one for each object, with radius 0.2 m. In the second scenario, four POIs of different sizes were defined, one for each group of objects. Table I reports the initial number of surfels, frontels and unknown voxels inside the POIs. In each scenario, the experiment was repeated three times, by changing the NBV score function. The first score function, named RCV (Ray Casting Voxel count), is the standard NBV algorithm that maximizes the number of unknown visible voxels. KinFu ray casting was exploited to obtain o_{ij} and voxel index c_{ij} for each pixel. To prevent the same voxel index to be counted more than one time, a support boolean 3D matrix is used to track which voxels have already been counted. The matrix has the same size of the TSDF volume and it is indexed by c_{ij}. The second NBV score function (RCP: Ray Casting Pixel gain) exploits KinFu ray casting to compute the pixel values o_{ij}. The distance $|o_{ij}|$ is obtained as the depth of the intersection between the ray and the first non empty voxel. Gain γ (14) is used to evaluate the next best view. RCP only requires knowledge of the pixel values o_{ij}. Finally, the third score function (SRP) uses pixel values o_{ij} obtained by surfel rendering as proposed in Section III-C. The NBV is again estimated by γ. The robot task terminates either when the NBV predicts a gain lower than a threshold or after 10 robot poses. The threshold was set as $\gamma_{\text{th}} = 0.002\ \text{m}^2$ for RCP and SRP. In the RCV case, as the area of a voxel face is approximately $(2.9\ \text{mm})^2$, the threshold was set equal to $0.002\ \text{m}^2/(2.9\ \text{mm})^2 \approx 237$ voxels.

TABLE II
AVERAGE EXECUTION TIME (SECONDS)

Phase	RCV	RCP	SRP
Initialization	-	-	0.02 ± 0.01
Shifting	2.83 ± 0.17	2.81 ± 0.18	-
Rendering	-	-	0.25 ± 0.03
Ray casting	1.49 ± 0.18	1.11 ± 0.12	-
GPU download	1.29 ± 0.04	0.34 ± 0.00	0.71 ± 0.04
Score computation	3.53 ± 0.15	0.56 ± 0.04	0.46 ± 0.02
Total	9.14 ± 0.41	4.83 ± 0.25	1.43 ± 0.06

B. Results

Average execution times of each phase of NBV evaluation are reported in Table II, as well as standard deviations. Results were obtained by averaging the evaluation time of each POI (960 view poses). More than 50 POI evaluations were executed for each approach in the reported experiments. It can be noticed that the total time to evaluate 960 poses is 9.14 s for RCV, 4.83 s for RCP and 1.43 s for SRP. That is, surfel rendering (SRP) outperforms both volumetric approaches (RCV and RCP) since rendering is faster than ray casting. The proposed NBV approach based on surfel rendering (SRP) requires a short initialization phase, since clouds Σ and Φ are reloaded from RAM to GPU memory. For the methods based on KinFu ray casting, instead, the TSDF volume must be shifted and centered on the current POI. Therefore, the shifting operation delays NBV computation by 2.81 s on average. SRP improvement is partially offset by the shorter data download time of RCP, probably due to better CUDA optimization. In general, the RCV approach is the slowest, since it also downloads voxel index c_{ij}. Moreover, in RCV each voxel must be counted only once, thus requiring a more complex score computation method.

The surfel/frontel generation procedure introduces a low overhead at each KinFu shifting operation during 3D reconstruction. Indeed, shifting lasts for additional 132 ms on average, over a total time of about 781 ms. The amount of unknown voxels

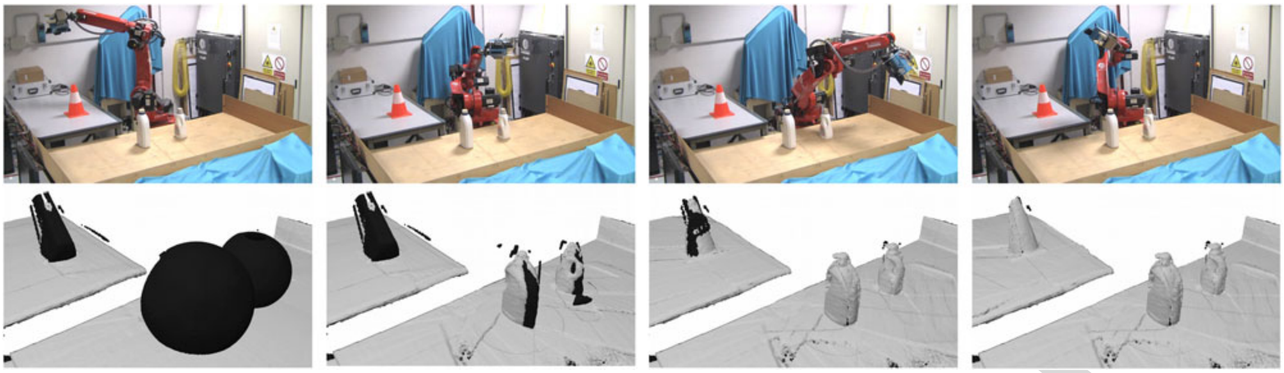


Fig. 6. Top: the first, third, fifth and seventh NBV reached by the robot in the SRP experiment of scenario 1. Bottom: the surfel- and frontel-based representation after each NBV.

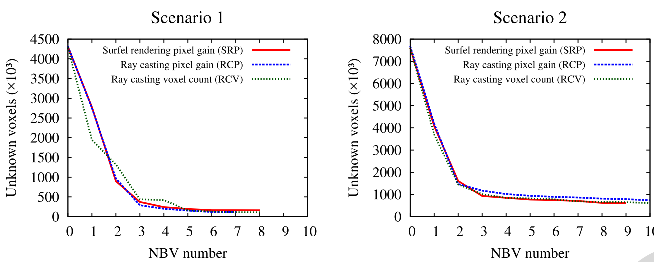


Fig. 7. Number of unknown voxels inside regions of interest, after each NBV.

TABLE III
NBV ITERATIONS, COMPLETENESS AND RECONSTRUCTION QUALITY

Scenario	1			2		
Method	RCV	RCP	SRP	RCV	RCP	SRP
Iterations	8	7	8	10+	10+	9
Completeness	96%	94%	96%	92%	93%	90%
Average Q_t	425.2	411.3	408.5	531.0	514.7	477.3
Q_t Std. dev.	294.7	249.3	295.5	495.6	385.7	428.7

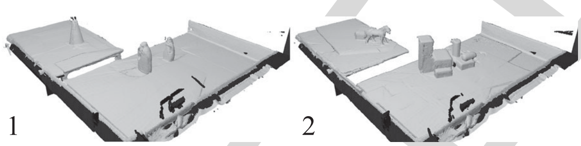


Fig. 8. Surfel- and frontel-based reconstruction at the end of the SRP experiments, for each scenario.

inside regions of interest decreases with the number of NBV iterations, as the robot explores the unknown regions. Fig. 6 shows images of some next best views for SRP in scenario 1, as well as the progressive reduction of unknown space surrounded by frontels (in black). The graphs in Fig. 7 show the number of unknown voxels as the NBV task progresses. All three methods have a similar trend. Table III reports the number of NBV iterations for each experiment. Symbol 10+ marks experiments that where stopped after 10 iterations. The final 3D environment reconstruction for the surfel-based experiments is shown in Fig. 8. Reconstruction quality was assessed by comparing results of the NBV task with the initial scan of the environment. The initial robot scan was performed along a predefined path to scan all POIs and, therefore, it acts as ground truth. We define T the initial point cloud produced by KinFu marching cubes algorithm after the initial scan, inside the regions of interest. Moreover, we define G the set of all points acquired by Kinect during an experiment. Quality Q_t of point $t \in T$ is defined as the number of points in G whose distance to t is lower than 2 cm. The meaning of Q_t is that automated NBV reconstruction quality is higher the more points are closer to ground truth points obtained in a full scan of the POIs. Completeness of reconstruction was evaluated as the fraction of points where $Q_t > 0$. Completeness, average quality across all points in T and quality standard deviation are reported in Table III. In general, completeness is comparable for the three methods. Quality is slightly lower for RCP and SRP than for RCV, which is probably due to the use of an approximated gain, as presented in Section III-D, instead of the exact number of voxels. Fig. 9 shows the effect of volume padding on frontel generation, as described in Section III-A. The sensor is oriented towards a POI centered on the small horse object. However, other POIs are also visible. No data can be acquired outside the TSDF volume, as highlighted by the red line near the boxes. Therefore, the unknown regions surrounding the two POIs on the boxes are only partially carved. Frontels are correctly generated on the surface of the TSDF volume inside the spheres. The methods SRP and RCP generate a slightly different simulated image o_{ij}, due to the different approximations made by the two algorithms. An example is shown in Fig. 10. RCP shows some drawback: the step of the KinFu ray casting algorithm is larger than the size of a single voxel, therefore, thin unknown volumes may be skipped during ray casting, as in the area highlighted with yellow squares. Moreover, some borders appear jagged (yellow circle), as the algorithm may detect the unknown space up to one step deep inside the surface. On the

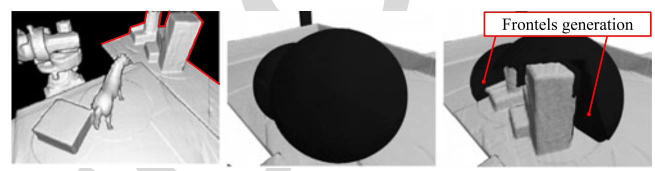


Fig. 9. Effect of the volume padding on frontel generation, during first NBV of scenario 2. Left: KinFu output, with the TSDF volume limit highlighted in red. Center and right: surfels/frontels before and after the NBV.

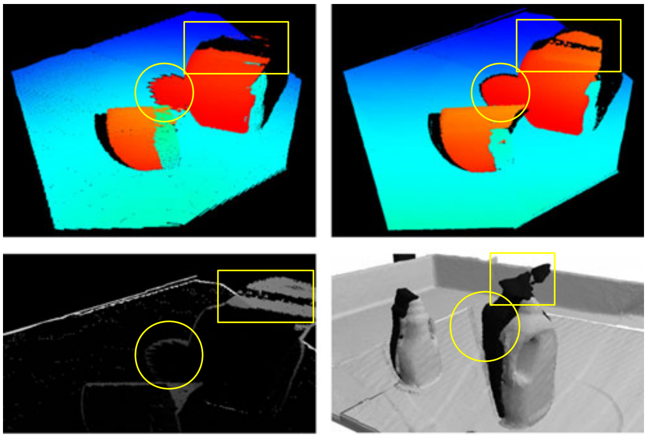


Fig. 10. Top: visible points simulated by RCP (left) and SRP (right) for the third NBV of the SRP experiment in scenario 1. Blue colors represent positive values of o_{ij}, while orange colors negative values. Bottom left: difference between the two simulated images. Bottom right: the surfel-based representation observed from a different viewpoint.

other hand the surfel-based approximation of SRP causes objects to expand slightly, since the surfel/frontel sizes $r_s = r_f$ were over-estimated in Section III-A.

V. Conclusion

In this work a novel method for robot next best view planning has been proposed. The approach is based on a surfel representation of the environment. Surfel rendering is used for NBV evaluation, instead of complex ray casting operations. Results indicate that a score function based on surfels is more efficient to compute and that it achieves comparable results in terms of reconstruction quality and completeness. Another advantage is that surfel-based NBV planning can be applied in larger environments. A limitation of the proposed method is that it still requires an initial voxel based volume representation, which is obtained through KinectFusion, from which surfels and frontels are extracted. In future work we plan to investigate algorithms for surfel and frontel generation that do not require an initial voxel representation of the environment.

References

[1] H. Pfister, M. Zwicker, J. van Baar, and M. Gross, “Surfels: Surface elements as rendering primitives,” in *Proc. 27th Annu. Conf. Comput. Graph. Interactive Techn.*, 2000, pp. 335–342.

[2] T. Whelan, S. Leutenegger, R. S. Moreno, B. Glocker, and A. Davison, “Elasticfusion: Dense SLAM without a pose graph,” in *Proc. Robot., Sci. Syst.*, Rome, Italy, Jul. 2015. Q3

[3] R. Monica, J. Aleotti, and S. Caselli, “A KinFu based approach for robot spatial attention and view planning,” *Robot. Auton. Syst.*, vol. 75, Part B, pp. 627–640, 2016.

[4] R. A. Newcombe *et al.*, “KinectFusion: Real-time dense surface mapping and tracking,” in *Proc. IEEE Int. Symp. Mixed Augmented Reality*, Oct. 2011, pp. 127–136.

[5] C. Connolly, “The determination of next best views,” in *Proc. IEEE Int. Conf. Robot. Automat.*, 1985, vol. 2, pp. 432–435.

[6] P. Whaite and F. Ferrie, “Autonomous exploration: Driven by uncertainty,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 3, pp. 193–205, Mar. 1997.

[7] R. Monica and J. Aleotti, “Contour-based next-best view planning from point cloud segmentation of unknown objects,” *Auton. Robots*, vol. 42, no. 2, pp. 443–458, 2018.

[8] S. Kriegel, M. Brucker, Z. C. Marton, T. Bodenmüller, and M. Suppa, “Combining object modeling and recognition for active scene exploration,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2013, pp. 2384–2391.

[9] J. I. Vasquez-Gomez, L. E. Sucar, R. Murrieta-Cid, and E. Lopez-Damian, “Volumetric next-best-view planning for 3D object reconstruction with positioning error,” *Int. J. Adv. Robot. Syst.*, vol. 11, no. 10, p. 159, 2014. Q4

[10] C. Potthast and G. S. Sukhatme, “A probabilistic framework for next best view estimation in a cluttered environment,” *J. Vis. Commun. Image Representation*, vol. 25, no. 1, pp. 148–164, 2014.

[11] F. Bissmarck, M. Svensson, and G. Tolt, “Efficient algorithms for next best view evaluation,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2015, pp. 5876–5883.

[12] P. G. C. N. Senarathne and D. Wang, “Towards autonomous 3D exploration using surface frontiers,” in *Proc. IEEE Int. Symp. Safety, Secur., Rescue Robot.*, 2016, pp. 34–41.

[13] K. Wu, R. Ranasinghe, and G. Dissanayake, “Active recognition and pose estimation of household objects in clutter,” in *Proc. IEEE Int. Conf. Robot. Automat.*, May 2015, pp. 4230–4237.

[14] J. Daudelin and M. Campbell, “An adaptable, probabilistic, next-best view algorithm for reconstruction of unknown 3-D objects,” *IEEE Robot. Automat. Lett.*, vol. 2, no. 3, pp. 1540–1547, Jul. 2017.

[15] S. Isler, R. Sabzevari, J. Delmerico, and D. Scaramuzza, “An information gain formulation for active volumetric 3D reconstruction,” in *Proc. IEEE Int. Conf. Robot. Automat.*, May 2016, pp. 3477–3484.

[16] J. Delmerico, S. Isler, R. Sabzevari, and D. Scaramuzza, “A comparison of volumetric information gain metrics for active 3D object reconstruction,” *Auton. Robots*, vol. 42, no. 2, pp. 197–208, Feb. 2018.

[17] T. Patten, W. Martens, and R. Fitch, “Monte Carlo planning for active object classification,” *Auton. Robots*, vol. 42, no. 2, pp. 391–421, Feb. 2018.

[18] T. Patten, M. Zillich, R. Fitch, M. Vincze, and S. Sukkarieh, “Viewpoint evaluation for online 3-D active object classification,” *IEEE Robot. Automat. Lett.*, vol. 1, no. 1, pp. 73–81, Jan. 2016.

[19] A. Bircher, M. Kamel, K. Alexis, H. Oleynikova, and R. Siegwart, “Receding horizon "next-best-view" planner for 3D exploration,” in *Proc. IEEE Int. Conf. Robot. Automat.*, May 2016, pp. 1462–1468.

[20] Z. Meng *et al.*, “A two-stage optimized next-view planning framework for 3-D unknown environment exploration, and structural reconstruction,” *IEEE Robot. Automat. Lett.*, vol. 2, no. 3, pp. 1680–1687, Jul. 2017.

[21] R. Pito, “A solution to the next best view problem for automated surface acquisition,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 21, no. 10, pp. 1016–1030, Oct. 1999.

[22] S. Chen and Y. Li, “Vision sensor planning for 3-D model acquisition,” *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 35, no. 5, pp. 894–904, Oct. 2005.

[23] S. Kriegel, T. Bodenmüller, M. Suppa, and G. Hirzinger, “A surface-based next-best-view approach for automated 3D model completion of unknown objects,” in *Proc. IEEE Int. Conf. Robot. Automat.*, May 2011, pp. 4869–4874.

[24] M. Botsch, M. Spernat, and L. Kobbelt, “Phong splatting,” in *Proc. First Eurograph. Conf. Point-Based Graph.*, Aire-la-Ville, Switzerland, 2004, pp. 25–32.

GENERAL INSTRUCTIONS

- **Authors: We cannot accept new source files as corrections for your paper. If possible, please annotate the PDF proof we have sent you with your corrections and upload it via the Author Gateway. Alternatively, you may send us your corrections in list format. You may also upload revised graphics via the Author Gateway.**
- Authors: If you need an invoice or have any other billing questions, please contact reprints@ieee.org as they handle these requests.

QUERIES

Q1. Author: Please provide the expansion of the acronym “GPU,” if required.
Q2. Author: Please confirm or add details for any funding or financial support for the research of this article.
Q3. Author: Please provide page range in Ref. [2].
Q4. Author: Please provide full page range in Ref. [9].