A Probabilistic Next Best View Planner for Depth Cameras based on Deep Learning

Riccardo Monica[1] and Jacopo Aleotti[1]

Abstract— **In this work we present a novel method to plan the next best view of a depth camera by leveraging on a Convolutional Neural Network (CNN), and on a probabilistic occupancy map of the environment for ray casting operations. In particular, a hybrid approach is introduced that exploits the convolutional encoder-decoder to perform object completion, and an algorithm based on ray casting to evaluate the information gain of possible sensor view poses. Automatic object completion consists of inferring the occupancy probability of the regions of space that have not been observed. A comparison against several methods, including City-CNN, was carried out in 2D and 3D environments on publicly available datasets. In particular, to enable comparison, the original City-CNN algorithm was modified to work with depth cameras. Experiments indicate that the proposed method achieves the best results in terms of exploration accuracy. Results on a real robot are also presented.**

I. INTRODUCTION

A Next Best View (NBV) algorithm predicts the best pose where a sensor should be placed to gather the most possible information about the environment. By applying a NBV algorithm iteratively and by moving the sensor to the planned view poses, it is possible to perform robot exploration tasks. This paper presents a novel method to plan the NBV for a depth camera. The method relies on a volumetric representation of the environment. In particular, we operate on ternary occupancy values, where each cell (voxel) may be empty, occupied or unknown. The proposed NBV algorithm maximizes an information gain that estimates the amount of unknown cells that would turn into occupied or empty if observed from the planned viewpoint. Next best view algorithms are usually computationally expensive, as they need to simulate sensor measurements from many candidate view poses. Moreover, traditional NBV approaches make simple assumptions about the environment, and lack a high-level understanding of the scene being observed. Hence, only simple hypotheses are made about the distribution of occupied space in the currently unknown region.

To address these issues, recent work has proposed the use of deep learning to determine the NBV. Deep learning methods provide the following advantages. First, they are more efficient than traditional NBV algorithms, as sensor simulation is not required. Second, deep learning methods can learn priors about the environment and, therefore, they can predict the NBV by considering the occupancy likelihood of unknown voxels.

However, despite the above advantages, deep learning based view pose prediction does not guarantee that the selected viewpoint has enough information gain for the iterative NBV algorithm to converge. Hence, deep learning methods may not explore the whole environment, but may get stuck in a local optimum. As remarked by Gallos et al. [1] application of CNNs to active vision should focus on modeling uncertainty of predictions over successive steps, instead of looking for where to place the sensor.

Therefore, we propose a hybrid approach, which exploits both prediction of environment priors, thanks to deep learning, as well as a probabilistic NBV planner to compute the optimal sensor pose. In particular, we present a convolutional neural network that generates, at each iteration, a probabilistic occupancy map of the environment, based on the current partially complete environment representation. In the generated probabilistic volumetric map, cells with unknown values are replaced by an occupancy probability value based on the output of the CNN. Then, at each iteration, a probabilistic NBV approach is applied to determine the view pose.

A comparison against several other methods was also carried out, including approaches based on City-CNN [2], and a probabilistic NBV planner [3]. City-CNN is a fully-convolutional approach designed for omni-directional sensors that operates on a volumetric representation, and that does not require the possible sensor viewpoints to be known at training time.

Hence, as further contribution, in order to enable comparison the original City-CNN algorithm was modified to work with depth cameras. In particular, three variants of City-CNN that output not only the sensor position, but also its orientation, were developed and evaluated.

Experimental evaluation was conducted in simulation in 2D environments, on the publicly available Inria Aerial Image Labeling dataset [4], in 3D environments generated from the YCB benchmark [5], and on a real robot. Results indicate that the proposed method outperforms City-CNN based approaches, as well as the standard probabilistic NBV planner, because the proposed method exploits the benefits of the two worlds, i.e. it leverages environment priors prediction and it guarantees full exploration of the environment.

The paper is organized as follows. Section II provides an overview of the state of the art. Section III illustrates the proposed NBV method. Experimental results are reported in Section IV. Finally, Section V concludes the paper.

This research has been financially supported by the Programme "FIL-Quota Incentivante" of University of Parma and co-sponsored by Fondazione Cariparma.

[1] Riccardo Monica and Jacopo Aleotti are with the RIMLab, Department of Engineering and Architecture, University of Parma, Parco Area delle Scienze, 181A, 43124 Parma, Italy {riccardo.monica, jacopo.aleotti}@unipr.it

II. Related Work

Common approaches for robot next best view planning are volumetric algorithms based on 3D occupancy maps and ray casting. An occupancy map approximates the workspace as a regular grid of cubic cells (voxels), or by using hierarchical structures like octrees. Occupancy values can be evaluated either using probabilistic or non-probabilistic methods.

Probabilistic approaches for NBV planning store a value in each occupancy map cell that represents the probability of occupancy [3], [6], [7], [8], [9], [10]. Daudelin et al. [3] proposed a probabilistic next best view framework for a mobile robot to perform incremental reconstruction of the 3D shape of an object. The approach in [3] is similar to the probabilistic NBV planner that we adopted in our hybrid solution. Probabilistic cost functions on 3D occupancy grids were also investigated for NBV planning in tabletop environments [6], [7]. Palomeras et al. [8] presented a probabilistic next best view planner for exploration of underwater environments. In [9] a probabilistic method was introduced for in-hand reconstruction of objects, which did not consider the contribution of other unknown volumes in the environment to compute the information gain.

Non probabilistic approaches were also described in several works [11], [12], [13], [14], [15], [16], [17]. Quin et al. [12] investigated a NBV planner that favors view poses that are close to the current robot configuration. In [13], [14] sampling-based NBV methods were considered. In particular, a receding horizon NBV planner was proposed by Bircher et al. [13] for aerial robotic platforms, and a mixed local and global planner was presented by Selin et al. [14]. In [15] a method was developed to explore objects based on saliency of point cloud segments, while in [16] a NBV planner was introduced to explore regions of the space where changes are more likely to have occurred, due to user manipulation actions. In [17] a NBV planner was proposed for a humanoid robot that exploits body movements primitives to reach favorable points of view.

During the last years, few research studies have investigated deep learning approaches for next best view planning [18]. In [19] a reinforcement learning based CNN approach was used to predict the NBV of a depth camera. However, the approach selected the NBV among a discrete action space of pre-sampled viewpoints which must be known at training time. Similarly, a supervised learning method was investigated by Mendoza et al. [20], where a fixed set of viewpoints was defined at training time, and the NBV problem was solved as a classification problem.

In [21] a NBV system for plant phenotyping was proposed, which adopted a CNN to auto-complete point clouds of plants in order to produce compelling predictions about unobserved parts. However, point clouds included only information about occupied space, and the NBV system did not exploit unknown voxels. Instead, in our approach we do not discard information about unknown space at training time.

The work in [22] performed object recognition using a Convolutional Deep Belief Network to auto-complete a partial volumetric representation of the object. A NBV planning approach was also proposed, aimed at selecting the optimal viewpoint for discriminating the object category. Object recognition experiments were carried out in environments comprising a single object, and quantitative evaluation of the NBV algorithm was limited to few views.

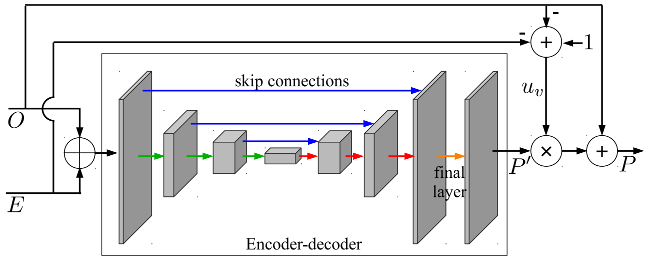

Fig. 1. Proposed CNN architecture. *Conv blocks* and *deconv blocks* are represented by green and red arrows, respectively, and they are numbered from 1 to N_{conv}. Skip connections are displayed in blue. The final convolution layer is displayed in orange.

Ly et al. developed City-CNN [2], a fully-convolutional deep learning method that operates on a volumetric representation of the environment. However, as shown in this paper, end-to-end approaches are unable to guarantee a positive information gain at each NBV.

Deep learning approaches have also been considered for robot navigation and exploration tasks [23][24]. In [23], deep learning was exploited to predict the best action for a mobile robot to explore a 2D environment by selecting a movement from a fixed number of possible actions. In [24], deep learning techniques were adopted to predict the best direction in which a robot manipulator must move a sensor to maximize information gain. Another CNN-based method for object auto-completion was proposed in [25], but this work was aimed at robotic grasping instead of computing a next best view.

III. Method

The NBV exploration approach maintains a partially known representation C of the (2D or 3D) environment as a uniform grid of cells c_v with three possible values: $occupied$, $empty$, and $unknown$. The depth camera is modeled as a standard pinhole model with a maximum range R_{max} and horizontal field of view HFOV (and vertical field of view VFOV in 3D environments). Given the camera pose, the sensor may observe any cell c_v within its field of view, if no other $occupied$ cell is between c_v and the sensor origin, and if the distance between c_v and the origin is lower than the maximum range. When a new observation occurs, any $unknown$ cell which is observed by the sensor is set to $occupied$ or $empty$.

The proposed method is composed of three main steps. In the *prediction* step, the partially known representation C of the environment is fed to an encoder-decoder CNN, to obtain a probabilistic map P where each cell p_v contains a real value in $[0, 1]$, denoting the cell occupancy probability. The encoder-decoder network architecture is described in

Section III-A. Then, in the second step, multiple candidate sensor view poses with different orientations are sampled, originating in each *empty* cell in C. In the final step a probabilistic ray-casting method is used to determine the corresponding information gain.

In order to compute the information gain for a large number of view poses in an efficient way, we take advantage of the fact that frustum volumes originating from the same cell may partially overlap. The candidate sensor view poses evaluation step is split into two sub-steps. First, in the *ray casting* sub-step, omnidirectional ray casting is performed for each cell by traversing a set of uniformly distributed rays, originating from the center of the cell. An information gain is then computed for each ray according to a probabilistic method illustrated in Section III-B. Hence, a directional gain map G_r is defined, where each gain value $G_r(v, h)$ is indexed by one cell index v, and a ray index h.

Finally, in the *view evaluation* sub-step, for each cell v, N_{views} camera view poses, originating in v, are sampled at several orientations. The information gain of each camera view pose is computed by accumulating gains $G_r(v, h)$ of the rays h that are in the view pose field of view, as illustrated in Section III-C. The view pose with the highest gain is selected as the next best view.

A. CNN architecture

The CNN network architecture (Fig. 1) is based on an encoder-decoder inspired by City-CNN [2]. The encoder-decoder receives as input a 2D (or 3D) grid which encodes ternary values (*occupied*, *empty* or *unknown*) for each cell. In particular, the CNN input consists of two separate channels. Elements e_v of the first channel E (empty) are equal to 1 if c_v is *empty*, and 0 otherwise. Elements o_v of the second channel O (occupied) are equal to 1 if cell c_v is *occupied*, and 0 otherwise. In the training phase, a ground truth representation of the environment $D^T = \{d_v\}$ is provided for the output of the decoder, where $d_v = 1$ in occupied cells, and $d_v = 0$ in empty cells. Hence, the network learns to predict the complete environment given a partially known environment representation.

The encoder-decoder is organized as N_{conv} *conv blocks* followed by the same number N_{conv} of *deconv blocks*. Each *conv block* is composed of a first convolution layer with stride 1, leaky ReLU activation, a second convolution layer with stride 2, and leaky ReLU activation. Kernel size is 3 for both layers. A 2D convolution is used for 2D environments, and a 3D convolution is used for 3D environments. Due to the value of the second stride (i.e. 2), each *conv block* halves the grid dimensions. The first *conv block* outputs 8 channels (for 2D) and 16 channels (for 3D), each subsequent *conv block* doubles the number of channels. Each *deconv block* is composed of a single transposed convolution (deconvolution) layer, symmetric to the second convolution layer of the *conv block*, with leaky ReLU activation. Each *deconv block* doubles the grid dimensions and halves the number of channels. Skip connections are added between the input of each *conv block* and the output of the corresponding *deconv block*. A single-channel convolution layer with kernel size 1 is used as the final layer of the network, followed by the sigmoid activation function.

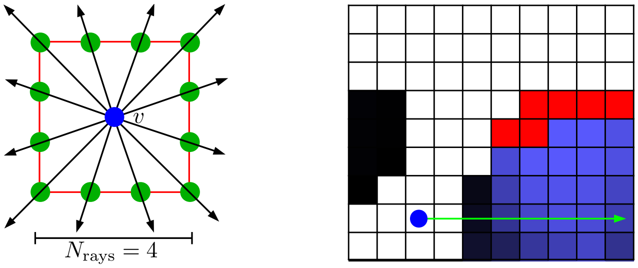

Fig. 2. Left: 2D example rays originating in cell v, with $N_{\text{rays}} = 4$ rays for each square edge. Right: example probabilistic occupancy map in a 2D case, with empty cells (white), occupied cells (red) and unknown cells (from black to blue, with increasing occupancy probability p_v). A ray used by the probabilistic NBV planner is displayed in green, originating in the blue dot.

The output of the encoder-decoder is a single-channel (2D or 3D) grid P', with the same size as the input, and cell values p'_v in the $[0, 1]$ interval. As the status of occupied and empty cells is known, the algorithm focuses only on the prediction of the occupancy probability of unknown cells. Hence, the output is multiplied by a binary mask U whose values u_v are set to 1 if and only if $c_v = \mathit{unknown}$, i.e. cells where $c_v \neq \mathit{unknown}$ are forced to zero. The mask values are computed from the input masks E and O as $u_v = 1 - e_v - o_v$. Also, an occupancy probability equal to 1 is forced in cells where $o_v = 1$. That is, elements p_v of the probabilistic map P are computed as:

$$p_v = u_v \, p'_v + o_v \tag{1}$$

During training against the ground truth $D^T = \{d_v\}$, equation (1) effectively sets to zero backpropagated gradients in the already observed empty and occupied regions, so that the encoder-decoder can focus on the predictions in the unknown regions.

B. Probabilistic NBV approach

For each empty cell v in C, a set $\mathcal{H}_r$ of rays originating from the center of the cell is generated at regular intervals, where each ray has an index h (an illustration for the 2D case is shown in Fig. 2, left). Along each ray samples are taken at 1-voxel intervals, from $t = 0$ to $t = N$, where $N = R_{\text{max}}$ is the maximum range. The samples occur in an ordered sequence of N cells $v_t \in \{v_0 \dots v_N\}$, i.e. for a ray h (originating in v) $v_t = \lfloor v + t\, b_h \rceil$, where b_h is the unit vector of ray h. If the ray goes out of the boundaries of environment C, the sampled cells v_t are considered as already explored and occupied, so that they do not contribute to the information gain.

A ray with origin in v_0 can reach a cell v_t, $0 < t \leq N$, only if all cells in segment $\overline{v_0 v_t}$ are empty, with the possible exception of v_t. Hence, probability $\mathcal{P}(r_t)$, where r_t is the event that cell v_t is observed by the ray, is given by:

$$\mathcal{P}(r_t) = \prod_{i=0}^{t-1} (1 - \mathcal{P}(o_i)) \tag{2}$$

where $\mathcal{P}(o_i) = p_{v_i}$ is the occupancy probability of cell v_i in the probabilistic map P computed in Section III-A. Equation (2) can be computed incrementally for each t during ray traversal.

The information gain is defined as the number of currently unknown cells in C that turn into occupied or empty after observation. In principle, the expected number of unknown cells observed by a single ray can be estimated by the sum of the unknown traversed cells weighted by the probability that a cell is observed by the ray:

$$\sum_{t=0}^{N} u_{v_t} \mathcal{P}(r_t) \tag{3}$$

where $u_{v_t} = 1$ if cell v_t is unknown in C, and 0 otherwise. However, cells near the origin are traversed by multiple rays, so they are over-represented in (3). Ray density decreases with the distance from the origin as $1/t^\nu$, where ν=1 in a 2D grid and ν=2 in a 3D grid. Hence, from (3) the normalized number of unknown cells observed by the ray is:

$$H(v,h) = \sum_{t=0}^{N} u_{v_t}\, \mathcal{P}(r_t)\; t^\nu \tag{4}$$

Conversely, the normalized expected number of cells traversed by a ray that do not turn either into occupied or empty is given by the number of traversed occupied cells, the empty cells, and the unknown cells with probability $(1 - \mathcal{P}(r_t))$:

$$M(v,h) = \sum_{t=0}^{N} \left(e_{v_t} + o_{v_t} + u_{v_t}\, (1 - \mathcal{P}(r_t))\right) t^\nu \tag{5}$$

therefore, the normalized gain $G_r(v,h)$ of ray h originating in cell v is defined as:

$$G_r(v,h) = \frac{H(v,h)}{H(v,h) + M(v,h)} \tag{6}$$

which is always included in the interval $[0,1]$.

C. View evaluation

For each empty cell, N_{views} camera viewpoints W_j are generated (with $j \in 1,\ldots,N_{\text{views}}$) originating at the center of the cell and oriented outwards. In 2D, viewpoints are sampled at regular intervals around the axis perpendicular to the image plane. In 3D, viewpoints are sampled at regular increments of azimuth, elevation and rotation around the sensor axis. For each viewpoint W_j, the set of rays $\mathcal{H}(W_j) \subset \mathcal{H}_r$ which are in W_j field of view are determined. Then, gain $G_{\text{W}}(v, W_j)$ of view W_j in cell v is computed by summing all gain values $G_r(v,h)$ of the rays in $\mathcal{H}(W_j)$, i.e.:

$$G_{\text{W}}(v, W_j) = \sum_{h \in \mathcal{H}(W_j)} G_r(v,h) \tag{7}$$

and the next best view $W^\star$ is given by:

$$(v^\star, W^\star) = \underset{(v,W)}{\text{argmax}}\; G_{\text{W}}(v, W_j) \tag{8}$$

where $v^\star$ is the cell where the NBV is located.

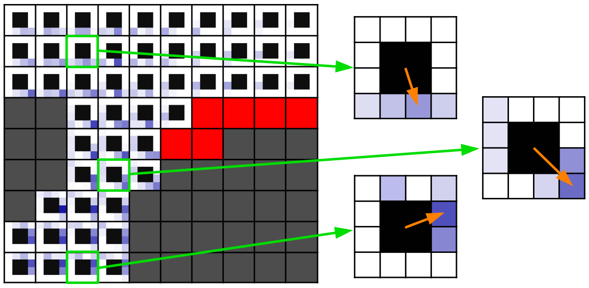

Fig. 3. An example of directional gain grid $G_g(\mu)$, with unknown cells (dark grey), occupied cells (red) and empty composite cells. On the border of composite cells, a deeper blue color means a higher predicted gain in that direction. The center of composite cells is black as it does not contain a useful value. Three composite cells are displayed on the right. The ray direction corresponding to the highest gain is displayed in orange.

D. Directional City-CNN

In order to compare the proposed approach with City-CNN [2], the original City-CNN algorithm (that was designed for omnidirectional sensors) was modified to work with depth cameras. In particular, a Directional City-CNN algorithm (*D-City-CNN*) was developed which predicts the information gain $G_r(v,h)$ of a ray from the current (partially known) 3D environment state. As in [2], the input of *D-City-CNN* is a two-channel grid of cells. The first channel E (empty) contains elements e_v which are equal to 1 if c_v is $empty$, and 0 otherwise. Elements f_v of the second channel F are equal to 1 on the frontier (also called shadow boundary) of the unknown volume, i.e. if cell c_v is $unknown$, and if there is at least an $empty$ adjacent cell.

Since fully-convolutional encoder-decoder CNNs are most effective when working with data organized into grids, the output of the CNN is defined as a (2D or 3D) grid named directional gain grid $G_g(\mu)$. As the rays $h \in \mathcal{H}_r$ (Fig. 2, left) are sampled at regular intervals on the square edges (or cube faces), each cell v is considered as a composite cell. The composite cell is itself a square (or a cube) made of sub-cells, with resolution N_{rays} along each edge. Each ray $h \in \mathcal{H}_r$ is associated to a sub-cell on the border of the composite cell. Resolution of $G_g(\mu)$ is N_{rays} times the voxel resolution of $G_r(v,h)$, i.e. $v = \lfloor \mu / N_{\text{rays}} \rfloor$. Each composite cell corresponds to a possible ray origin, and each $N_{\text{rays}} \times N_{\text{rays}}$ ($\times N_{\text{rays}}$ in 3D) sub-cell on the edge corresponds to a ray direction (Fig. 3). The total number of rays cast from a composite cell is equal to the total number of rays cast from a single cell in the proposed hybrid approach (Section III-A).

The convolutional neural network proposed in Section III-A was changed to predict the directional gain grid $G_g(\mu)$, containing a real value in $[0,1]$ in each sub-cell. In particular, $\log_2(N_{\text{rays}})$ additional *deconv blocks* were added at the end of the network in place of the final layer, to get the required resolution. The central sub-cells of the composite cells are not used, as they are not associated with any ray. Hence, a final layer of the network multiplies the network output by a binary mask which is 0 in the central sub-cells of each composite cell and 1 otherwise. The network output is also

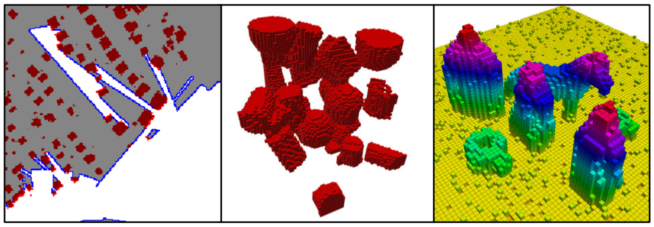

Fig. 4. Left: sample image from the 2D dataset after the simulation of random view poses. Ground truth occupied cells in D^T are displayed in red, empty cells in C are in white, and frontiers in C are in blue. Center: sample ground truth voxel grid from the unstructured 3DU dataset. Right: sample ground truth voxel grid from the tabletop 3DT dataset (cell color changes along the vertical axis).

multiplied by a binary mask E whose values $e_\mu = 1$ if and only if the corresponding cell $c_v = \mathit{empty}$, so that the network generates the next best view only in empty space. As there is a correspondence between each ray $h \in \mathcal{H}_r$ in a cell v and a sub-cell, grid $G_g(\mu)$ is easily converted into $G_r(v, h)$. Then, *D-City-CNN* proceeds as in Section III-C.

IV. EVALUATION

The proposed hybrid CNN-probabilistic (*CNN Prob.*) method was evaluated in 2D and 3D environments against eight other approaches for next best view planning, which are briefly described hereafter.

1) *Random*. This approach randomly selects the next best view with a camera pose located in the empty space of the current environment representation.
2) *Flat City-CNN*. This method naively extends City-CNN to work with a depth camera. It provides as output a grid of cells. Each cell v contains a channel (value) for each view W_j sampled in the cell, which represents the predicted information gain for that view $G_{\text{W}}(v, W_j)$. The next best view is selected as the one with the highest channel value.
3) *B-City-CNN*. This method extends City-CNN to output the best camera direction for each cell, in addition to the predicted gain. In particular, the output $Q(v)$ is a 2-channel grid (in 2D), or a 4-channel grid (in 3D). Each channel vector q_v in each cell has values in $[-1, 1]$. The norm $\|q_v\|$ is the predicted gain and the normalized vector $q_v / \|q_v\|$ is the view direction (in 2D), or the view orientation quaternion (in 3D).
4) *D-City-CNN*. It uses the Directional City-CNN approach (Section III-D) to predict information gain $G_r(v, h)$ for N_{rays} rays in each cell. Then, standard view evaluation is carried out as in Section III-C.
5) *Information Gain*. This approach is a standard non-probabilistic next best view planner [11] that does not exploit a CNN. Rays stop at the first non-*empty* cell, and the information gain is equal to the number of unknown cells between the sensor minimum and maximum range.
6) *Probabilistic Information Gain (Prob. IG)*. This method is a pure probabilistic next best view planner (Section III-B) that does not exploit a CNN to predict the occupancy probability of unknown cells. Probability p_v=P_0 is constant for all cells.
7) *CNN Prob. Downsampled (CNN Prob. D)*. A faster variant of *CNN Prob.* where viewpoints are sampled only in a fixed number of empty cells (2000 in 2D and 500 in 3D), chosen at random.
8) *Omniscient (oracle)*. The ground truth NBV given an a priori knowledge about the occupancy state of the environment. It is computed by setting the probabilistic map P equal to the ground truth D^T. This method is not realistic, but it provides an upper bound for the information gain.

TABLE I

PARAMETERS

Symbol	Value (2D/3DU/3DT)	Description
HFOV	60°/60°/60°	Horizontal field of view
VFOV	- /46°/46°	Vertical field of view
R_{max}	128/24/128	Sensor maximum range (cells)
N_{rays}	16/4/4	Square/cube side for ray sampling
N_{views}	60/52/52	Sampled views
P_0	0.15/0.15/0.15	Fixed probability for *Prob. IG*
N_{conv}	6/3/4	No. of conv blocks

TABLE II

APPROXIMATE TRAINING TIME (HOURS)

Method	2D	3DU	3DT
CNN Prob.	2.5	0.5	0.5
B-City-CNN	2.5	1	2
D-City-CNN	13	12.5	2.5
Flat City-CNN	8	16	2

All methods were tested in multiple 2D and 3D environments. At the beginning of each experiment all the cells of C are set to *unknown*. Each experiment completed after the computation of 100 next best views. Since a NBV can be generated only in *empty* cells, the first view pose is randomly generated where the ground truth D_T is empty ($d_v = 0$). The occupancy values of grid C are updated according to the pinhole sensor model. After initialization, grid C contains at least one *empty* cell. Then, the NBV method being tested is executed iteratively. At each iteration a next best view is determined and the occupancy values of C are updated by simulating the sensor observation, and *unknown* voxels are set *empty* or *occupied* according to the sensor model.

The deep learning methods were implemented in Python using Tensorflow. Probabilistic and non-probabilistic NBV methods were implemented in C++, with GPU-accelerated ray casting using OpenCL. Communication between C++ and Python code was achieved using the ROS (Robot Operating System) framework. The software ran on an Intel i7-6700 @ 3.40GHz, 32 GB RAM, with a GeForce GTX 980 Ti, 6 GB RAM. Source code is available at `http://rimlab.ce.unipr.it/Software.html`, under `nbv_3d_prob_cnn`.

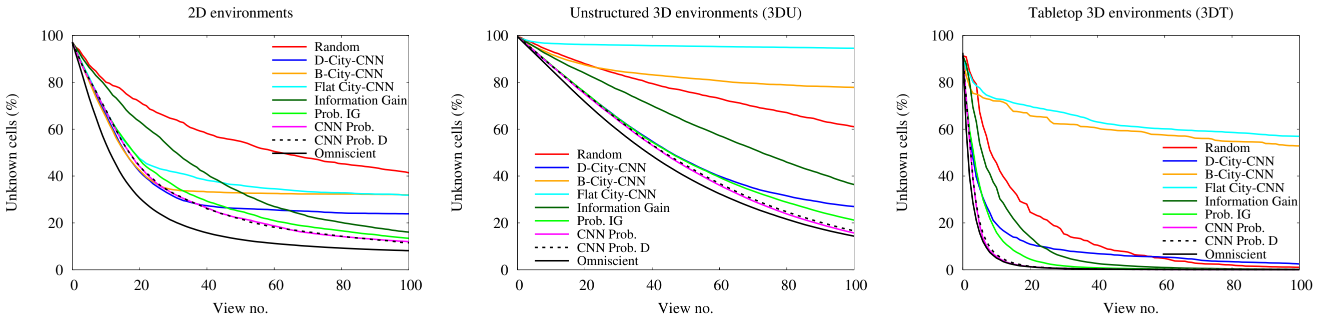

Fig. 5. Percentage of unknown cells at each iteration for each method, averaged over 40 trials, in 2D environments (left), unstructured 3D environments (3DU dataset, center), and tabletop 3D environments (3DT dataset, right).

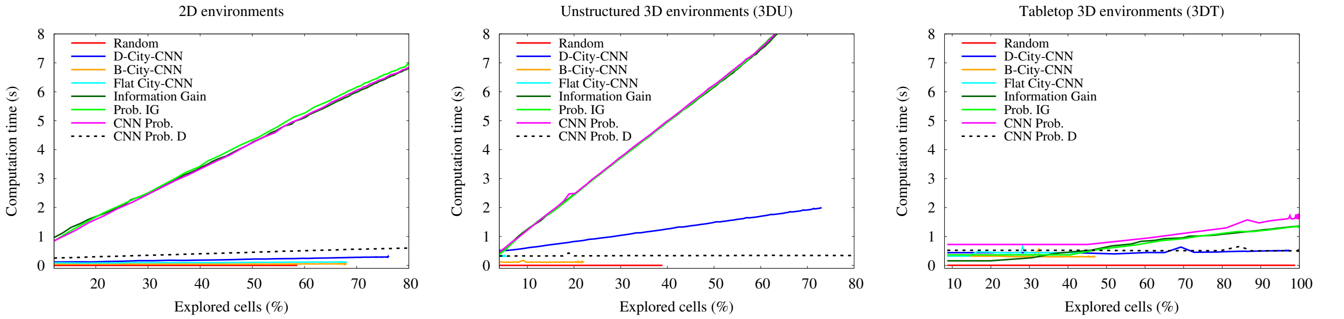

Fig. 6. Computation time for each iteration with respect to the percentage of explored (occupied or free) cells, averaged over 40 trials, in 2D environments (left), unstructured 3D environments (3DU dataset, center), and tabletop 3D environments (3DT dataset, right).

A. Dataset and training

In 2D environments (Fig. 4, left), similarly to City-CNN [2], the dataset was taken from ground truth images of the Inria Aerial Image Labeling dataset [4]. The dataset contains 180 black-and-white aerial images of cities, where buildings are labeled in white. Images were downsampled from the original resolution 5000×5000 to 1250×1250. Tests were carried out on two different synthetic 3D datasets, an "unstructured" dataset (3DU), and a "tabletop" dataset (3DT). The 3DU dataset is composed of 180 volumetric grids with resolution $64 \times 64 \times 64$. The environments of 3DU were generated by placing a random number of objects from the YCB benchmark [5] at random position and orientation in the grid (Fig. 4, center). Conversely, the 3DT dataset is composed of 180 volumetric grids with resolution $128 \times 128 \times 96$, representing tabletop scenarios. The environments of 3DT were generated by randomly placing objects from the dataset [26] (named RD) on a planar surface represented by a point cloud with noise (Fig. 4, right). Cell size is about 1.2 cm. Data augmentation was achieved by rotating each environment in steps of 90 degrees. All four possible rotations were used in 2D, for a total of 720 samples. Four rotations around the vertical axis were also used in the 3DT dataset. In the 3DU dataset, only eight rotations around the coordinate axes were selected among the possible ones, for a total of 1440 samples.

For training, empty grid E, occupied grid O and frontier grid F were generated for each environment by simulating random view poses (Fig. 4, left). For each environment, a different random number of view poses was selected between 200 and 400 in 2D, between 10 and 15 in 3DU, and between 30 and 40 in 3DT, to prevent overfitting on the number of poses. Ground truth D^T for the *CNN Prob.* and *CNN Prob. D* methods was the volumetric representation of the environment itself. To train the other CNNs (*Flat City-CNN*, *D-City-CNN* and *B-City-CNN*), ground truth for grids $G_{\mathrm{W}}(v, W)$, $G_g(\mu)$ and $Q(v)$ was computed by simulating each view or ray, and by computing the amount of unknown cells which would be observed. Only $2/3$ of the dataset was used as training set, and the remaining as test set. Training used the Adam optimizer with learning rate 0.001. All CNNs were trained for up to 120 epochs. Training was stopped after 20 epochs for the 3DU and 3DT version of *CNN Prob.* and *B-City-CNN*, as further training would lead to over-fitting. All networks use the Mean Square Error (MSE) loss function.

Relevant parameters are reported in Table I. For *D-City-CNN*, N_{rays} was reduced to 4 to fit into available video RAM. For the same reason, in the larger 3DT environments the output was downsampled to $1/8$ resolution in *D-City-CNN*, *B-City-CNN*, and *Flat-City-CNN*, and three *deconv blocks* were removed accordingly. Hence, for consistency, on the 3DT dataset other methods (*CNN Prob.*, *Information Gain*, *Prob. IG* and *Omniscient*) sampled viewpoints only in $1/8^3$ of the empty cells and a reduction of computation time was obtained to less than 2 seconds, which may be realistic for robotic applications.

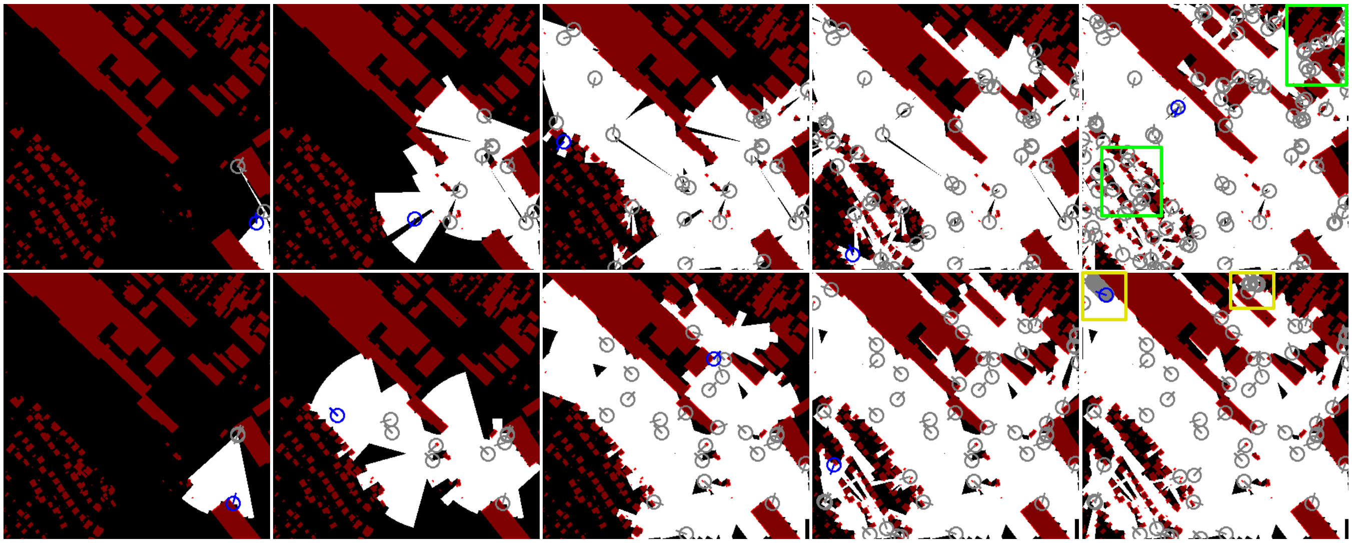

Fig. 7. A 2D execution of a next best view planning task for *CNN Prob.* (top row) and *D-City-CNN* (bottom row). Images show the explored environment after 3, 10, 25, 50, and 100 views (from left to right). Observed empty cells (grid C) are displayed in white, while ground truth occupied cells are displayed in red. The position of the current NBV is displayed as a blue circle (including a segment pointing towards the camera forward direction). All previous next best views are displayed as gray circles. The two green rectangles (top right image) highlight narrow regions that have been successfully explored by the proposed *CNN Prob.* method. The two yellow rectangles (bottom right image) highlight regions where *D-City-CNN* got stuck.

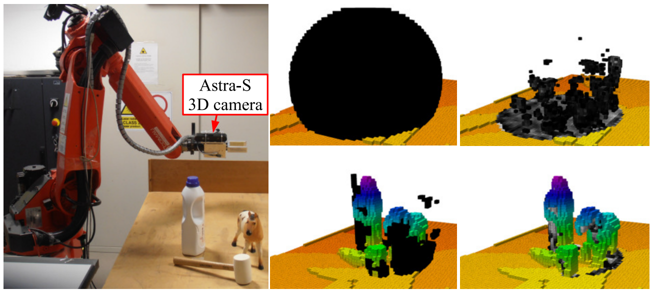

Fig. 8. Experimental setup (left). Environment representation at the beginning of the experiment (center column, top) and after the 5th NBV (center column, bottom). The corresponding probabilistic maps predicted by the encoder-decoder CNN of the proposed *CNN Prob.* method (right column). Unknown cells are displayed in black. Predicted occupied cells are displayed in grayscale with brightness proportional to occupancy probability.

Training time for all CNN methods, for 2D, 3DU and 3DT, is reported in Table II. Training is significantly faster for the proposed *CNN Prob.* method and for *B-City-CNN*, which have a simpler final layer. Also, *Flat City-CNN* and *D-City-CNN* have to be re-trained in case of changes to the number of sampled view poses N_{views} or rays N_{rays}, respectively.

B. NBV evaluation

All methods were evaluated in 40 simulated environments for each of the three datasets. The environments were randomly extracted from the test set. Test images of 2D environments were cropped from their original resolution (1250 × 1250) to 400 × 400 at random coordinates. Each NBV method was executed for 100 iterations, or until the NBV method predicted a null information gain for all the view poses. The percentage of $unknown$ cells in C at each iteration, averaged over the 40 trials, is shown in Fig. 5. The proposed hybrid approach *CNN Prob.* shows the best result. Indeed, *CNN Prob.* achieves the highest amount of explored region, i.e. the lowest amount of unknown cells left after iteration 100, compared to the CNN-based approaches (*D-City-CNN*, *B-City-CNN*, *Flat City-CNN*), and to the standard NBV approaches that are not based on CNNs (*Information Gain*, *Prob. IG*). It can be observed that results for *CNN Prob. D* are slightly worse than *CNN Prob.*, due to the lower number of viewpoints. In some cases, particularly in 2D environments, the number of unknown cells left for City-CNN-based approaches (*D-City-CNN*, *B-City-CNN*, *Flat City-CNN*) decreases rapidly at the beginning of each experiment, in a similar way to the proposed *CNN Prob.* approach. However, the information gain in all City-CNN-based approaches degrades as the exploration progresses, and eventually the next best view predicted by these methods is worse. Among the three City-CNN-based approaches *D-City-CNN* shows the best results, while *Flat City-CNN* gives the worst. This behavior may be explained by recalling that in *D-City-CNN* rays are organized into a grid of composite cells and, therefore, *D-City-CNN* is able to learn neighboring information. As expected, all curves are above the *Omniscient* method, which provides an upper bound for information gain.

Computation time for each iteration is reported in Fig. 6, as a function of the percentage of unknown cells. It can be noticed that the computation time of *Information Gain*, *IG Prob.* and *CNN Prob.* always increases linearly as the experiment progresses, and the environment is explored. Efficiency decreases since the number of evaluated viewpoints is proportional to the number of $empty$ cells. A linear computation time increase is also noticeable for the *D-City-CNN* method. This trend is mostly noticeable in 3DU environments as in 2D the view evaluation step in *D-City-CNN* (from $G_r(v, h)$ in Section III-C) is computationally trivial. In general, results indicate that *D-City-CNN*, *B-City-CNN* and *Flat City-CNN* have a lower computation time than the proposed *CNN Prob.* approach. The computation time of *CNN Prob. D* is also lower than the *CNN Prob.* method.

An example execution of the *CNN Prob.* and *D-City-CNN* algorithms in a 2D environment from the test set is reported in Fig. 7. At the beginning of the task, *D-City-CNN* is able to select slightly better poses and it rapidly explores the large diagonal empty region (second column). However, after 100 next best views the proposed hybrid *CNN Prob.* method managed to explore several narrow regions, while *D-City-CNN* algorithm got stuck several times, as highlighted in the fifth column.

To demonstrate the applicability of the proposed *CNN Prob.* method in a real-world scenario, an experiment was carried out using a COMAU Smart Six robot manipulator equipped with an eye-in-hand depth camera (ORBBEC Astra-S). KinectFusion was used for 3D reconstruction. The robot was tasked to explore a region of space in a tabletop scenario (Fig. 8) containing a few objects of the RD dataset. The *CNN Prob.* network, trained on the 3DT dataset, was used to predict occupancy probability values. The experiment is shown in the accompanying video.

V. Conclusion

This work presented an hybrid approach for depth camera next best view planning that exploits a convolutional neural network to predict environment priors, and a probabilistic volumetric map of the environment to compute the optimal sensor pose. The proposed method was compared against several other approaches in both 2D and 3D environments using publicly available datasets. In particular, three variants of the City-CNN algorithm were developed to work for depth cameras. The proposed hybrid approach showed better results in terms of exploration accuracy. As future work, we will explore a more realistic model of the depth camera, taking into account possible missing measurements. Moreover, in the current approach the environment size is limited by the memory occupancy of the input grid. Future work may investigate a local approach, as a cell occupancy prediction depends only on the input in a neighborhood of the cell.

References

[1] D. Gallos and F. Ferrie, "Active vision in the era of convolutional neural networks," in *16th Conference on Computer and Robot Vision (CRV)*, 2019, pp. 81–88.

[2] L. Ly and Y. R. Tsai, "Autonomous Exploration, Reconstruction, and Surveillance of 3D Environments Aided by Deep Learning," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2019, pp. 5467–5473.

[3] J. Daudelin and M. Campbell, "An Adaptable, Probabilistic, Next-Best View Algorithm for Reconstruction of Unknown 3-D Objects," *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1540–1547, 2017.

[4] E. Maggiori, Y. Tarabalka, G. Charpiat, and P. Alliez, "Can Semantic Labeling Methods Generalize to Any City? The Inria Aerial Image Labeling Benchmark," in *IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, 2017, pp. 3226–3229.

[5] B. Calli, A. Singh, J. Bruce, A. Walsman, K. Konolige, S. Srinivasa, P. Abbeel, and A. M. Dollar, "Yale-CMU-Berkeley dataset for robotic manipulation research," *The International Journal of Robotics Research*, vol. 36, no. 3, pp. 261–268, 2017.

[6] C. Potthast and G. S. Sukhatme, "A probabilistic framework for next best view estimation in a cluttered environment," *Journal of Visual Communication and Image Representation*, vol. 25, no. 1, pp. 148 – 164, 2014.

[7] S. Isler, R. Sabzevari, J. Delmerico, and D. Scaramuzza, "An information gain formulation for active volumetric 3D reconstruction," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 3477–3484.

[8] N. Palomeras, N. Hurtós, E. Vidal, and M. Carreras, "Autonomous Exploration of Complex Underwater Environments Using a Probabilistic Next-Best-View Planner," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1619–1625, 2019.

[9] M. Krainin, B. Curless, and D. Fox, "Autonomous generation of complete 3D object models using next best view manipulation planning," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2011, pp. 5031–5037.

[10] S. Kriegel, C. Rink, T. Bodenmüller, and M. Suppa, "Efficient Next-Best-Scan Planning for Autonomous 3D Surface Reconstruction of Unknown Objects," *Journal of Real-Time Image Processing*, vol. 10, no. 4, pp. 611–631, 12 2015.

[11] C. Connolly, "The determination of next best views," in *IEEE Intl Conference on Robotics and Automation (ICRA)*, 1985, pp. 432–435.

[12] P. Quin, G. Paul, A. Alempijevic, D. Liu, and G. Dissanayake, "Efficient neighbourhood-based information gain approach for exploration of complex 3D environments," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2013, pp. 1343–1348.

[13] A. Bircher, M. Kamel, K. Alexis, H. Oleynikova, and R. Siegwart, "Receding Horizon "Next-Best-View" Planner for 3D Exploration," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 1462–1468.

[14] M. Selin, M. Tiger, D. Duberg, F. Heintz, and P. Jensfelt, "Efficient Autonomous Exploration Planning of Large-Scale 3-D Environments," *IEEE Robot. Autom. Lett.*, vol. 4, no. 2, pp. 1699–1706, 2019.

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

[16] R. Monica, J. Aleotti, and S. Caselli, "A KinFu based approach for robot spatial attention and view planning," *Robotics and Autonomous Systems*, vol. 75, pp. 627 – 640, 2016.

[17] R. Monica, J. Aleotti, and D. Piccinini, "Humanoid Robot Next Best View Planning Under Occlusions Using Body Movement Primitives," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 2493–2500.

[18] R. Zeng, Y. Wen, W. Zhao, and Y.-J. Liu, "View planning in robot active vision: A survey of systems, algorithms, and applications," *Computational Visual Media*, vol. 6, no. 3, pp. 225–245, 2020.

[19] X. Han, Z. Zhang, D. Du, M. Yang, J. Yu, P. Pan, X. Yang, L. Liu, Z. Xiong, and S. Cui, "Deep Reinforcement Learning of Volume-Guided Progressive View Inpainting for 3D Point Scene Completion From a Single Depth Image," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 234–243.

[20] M. Mendoza, J. I. Vasquez-Gomez, H. Taud, L. E. Sucar, and C. Reta, "Supervised learning of the next-best-view for 3d object reconstruction," *Pattern Recognition Letters*, vol. 133, pp. 224 – 231, 2020.

[21] C. Wu, R. Zeng, J. Pan, C. C. L. Wang, and Y. Liu, "Plant phenotyping by deep-learning-based planner for multi-robots," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3113–3120, 2019.

[22] Zhirong Wu, S. Song, A. Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and J. Xiao, "3D ShapeNets: A deep representation for volumetric shapes," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1912–1920.

[23] S. Bai, F. Chen, and B. Englot, "Toward autonomous mapping and exploration for mobile robots through deep supervised learning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 2379–2384.

[24] Y. Wang, S. James, E. K. Stathopoulou, C. Beltrán-González, Y. Konishi, and A. Del Bue, "Autonomous 3-D Reconstruction, Mapping, and Exploration of Indoor Environments With a Robotic Arm," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3340–3347, 2019.

[25] J. Varley, C. DeChant, A. Richardson, J. Ruales, and P. Allen, "Shape completion enabled robotic grasping," in *IEEE/RSJ Intl Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 2442–2447.

[26] R. Monica and J. Aleotti, "Point cloud projective analysis for part-based grasp planning," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4695–4702, 2020.