Autonomous Exploration, Reconstruction, and Surveillance of 3D Environments Aided by Deep Learning

Louis Ly[1] and Yen-Hsi Richard Tsai[1]

Abstract

We study the problem of visibility-based exploration, reconstruction and surveillance in the context of supervised learning. Using a level set representation of data and information, we train a convolutional neural network to determine vantage points that maximize visibility. We show that this method drastically reduces the on-line computational cost and determines a small set of vantage points that solve the problem. This enables us to efficiently produce highly-resolved and topologically accurate maps of complex 3D environments. We present realistic simulations on 2D and 3D urban environments.

I. INTRODUCTION

We consider the problem of generating a minimal sequence of observing locations to achieve complete visibility (line-of-sight) coverage of an environment. If the environment is initially unknown, the problem is called exploration and reconstruction. This is particularly useful for autonomous agents to map out unknown, or otherwise unreachable environments, such as undersea caverns. If the environment is known, the problem is one of surveillance: *how should a minimal set of sensors be placed to maintain complete surveillance of an environment?*

A. PROBLEM FORMULATION

Let X be the space consisting of all possible environment configurations. Take $\Omega \in X$. Ω is an open set representing the free space and Ω^c is a closed set consisting of a finite number of connected components.

Let $\mathcal{O} = \{x_i\}_{i=0}^k$ be the sequence of vantage points. For each vantage point, the operator $\mathcal{P}_{x_i}\Omega$ is a projection of Ω along x_i. Then $\mathcal{P}_{x_i}\Omega$ is a set of range measurements defined on the unit sphere. The back projection $\mathcal{Q}$ maps the range measurements to the visibility set $\mathcal{V}_{x_i}\Omega := \mathcal{Q}(\mathcal{P}_{x_i})\Omega$; that is, points in this set are visible from x_i. As

[1] Institute for Computational Engineering and Sciences, The University of Texas at Austin {louis,ytsai}@ices.utexas.edu

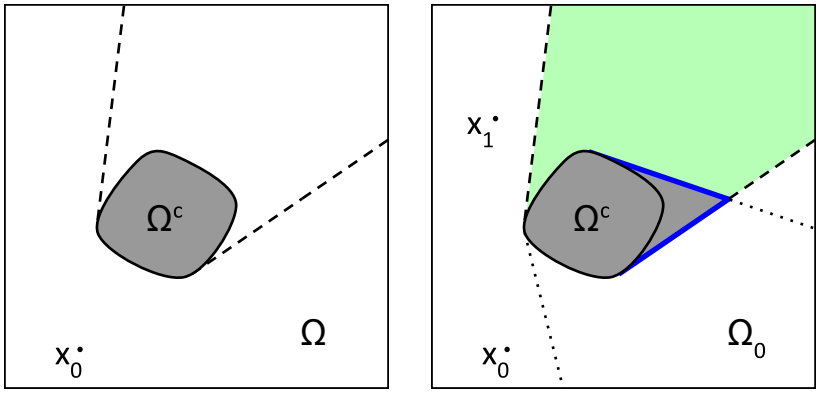

Fig. 1: An illustration of the environment. Dashed and dotted lines are the horizons from x_0 and x_1, respectively. Their shadow boundary, B_1, is shown in thick, solid blue. The area of the green region represents $g(x_1; \Omega_0, \Omega)$.

more range measurements are acquired, the environment can be approximated by the *cumulatively visible set* Ω_k:

$$\Omega_k = \bigcup_{i=0}^{k} \mathcal{V}_{x_i} \Omega$$

$\mathcal{P}$ is reminiscent of the projection operator in computed tomography which maps higher dimensional data into a low dimensional manifold.

By construction, Ω_k admits partial ordering: $\Omega_{i-1} \subset \Omega_i$. For suitable choices of x_i, it is possible that

$$\Omega_n \to \Omega$$

(say, in the Hausdorff distance). We aim at determining a *minimal number of vantage points* from which every point in Ω can be seen.

One may formulate a constrained optimization problem and look for sparse solutions in the following context. Let $D \subset \mathbb{R}^d$, $d = 2, 3$, be the bounded cubic region in which we wish to solve the exploration or surveillance problems. Let I be a real valued function defined on a grid over D, with M nodes in each dimension. The constrained optimization problem is defined as:

$$\min_{I:\mathbb{R}^{Md} \mapsto \{0,1\}} ||I||_0 \quad \text{subject to} \quad \bigcup_{\{I=1\}} \mathcal{V}_x \Omega = \Omega. \tag{1}$$

B. A GREEDY APPROACH

We propose a greedy approach which sequentially determines a new vantage point, x_{k+1}, based on the information gathered from all previous vantage points, $x_0, x_1, \cdots, x_k$. The strategy is greedy because x_{k+1} would be a location that *maximizes the information gain*.

If the environment Ω is known, we define the *gain* function

$$g(x; \Omega_k, \Omega) := |\mathcal{V}_x \Omega \cup \Omega_k| - |\Omega_k|,$$

i.e. the volume of the region that is visible from x but not from $x_0, x_1, \cdots, x_k$. Thus, for surveillance, we consider:

$$x_{k+1} = \arg\max_{x \in \Omega} g(x; \Omega_k, \Omega). \tag{2}$$

In other words, the next vantage point should be chosen to maximize the newly surveyed area. On a Cartesian grid over the domain D, the brute force search is expensive with complexity $O(M^{2d})$ where M is the number of grid points along one dimension.

The problem of Exploration is even more challenging since, by definition, the environment is not known. However, we remark that in practice, one is typically interested only in a subset S of all possible environments X. For example, cities generally follow a grid-like pattern. Knowing these priors can help guide our estimate of g for certain types of Ω, even when Ω is unknown initially.

We propose to encode these priors formally into the parameters, θ, of a learned function:

$$g_\theta(x; \Omega_k, B_k) \text{ for } \Omega \in S,$$

where B_k is the part of $\partial\Omega_k$ that may actually lie in the free space Ω. More precisely,

$$B_k = \partial\Omega_k \backslash \Omega^c. \tag{3}$$

See Figure 2 for an example gain function. We shall demonstrate that while training for g_θ, incorporating the shadow boundaries as an additional restriction helps, in some sense, localize the learning of g, and is essential in creating usable g_θ.

II. RELATED WORKS

The surveillance problem is related to the art gallery problem in computational geometry, where the task is to determine the minimum number of guards who can together observe a gallery. For simply-connected polygonal scenes, Chvátal showed the minimum number of guards is upper bounded by $\lfloor n/3 \rfloor$ patrols, where n is the number of vertices in the triangulation of the scene [4]. However, determining the optimal set of observers is NP-complete [21].

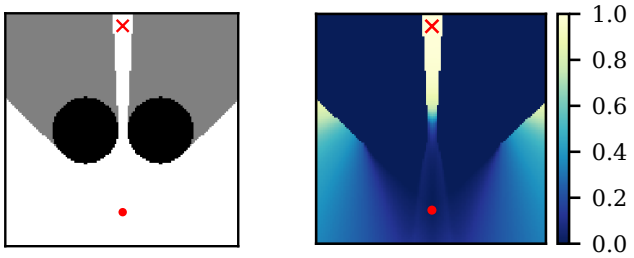

Fig. 2: Left: the map of a scene consisting of two disks. Right: the intensity of the corresponding gain function. The current vantage point is shown as the red dot. The location which maximizes the gain function is shown as the red x.

Goroshin et al. propose an alternating minimization scheme for optimizing the visibility of n observers [5]. They suggest gradually incrementing the number of observers while applying this algorithm in order to find an approximately minimal set. This is very inefficient. Similarly, Kang et al. use a level set framework and system of differential equations to optimize the location and orientation of sensors to maximize surveillance [6]. The number of sensors is assumed to be given.

For the exploration problem, the "wall-following" strategy may be used to map out simple environments. See e.g. [23]. LaValle and Tovar et al. [17], [10], [18] combine wall-following with a gap navigation tree to keep track of gaps, critical events which hide a connected region of the environment that is occluded from a vantage point. Exploration is complete when all gaps have been eliminated. This approach does not produce any geometric representation of the environment upon completion, due to limited information from simple gap sensors.

Landa et al. [9], [7], [8] extend the idea of *chasing the gap* to range sensors, resulting in an algorithm that is efficient, guaranteed to converge in finite time, and reconstructs the environment upon completion. Unfortunately as the algorithm relies on estimation of the curvature of $\partial\Omega_k$, extension to 3D environments seem complicated.

Valente et al. [22] use a heuristic based on the area of the shadow boundary, weighted by the viewing angle from the vantage points, to define potential information gain. This formulation is straight-forward to apply in 3D. However, computing this energy is expensive as it requires evaluating a surface integral at each spatial location.

Bai et al. [1] use convolutional neural networks to choose an action from a fixed set of moves. Their approach terminates when there is no occlusion within view of the agent, even if the global map is still incomplete. Bircher et al. use an occupancy grid and a geometric random tree to determine optimal paths in a receding horizon fashion [2], [3]. Due to high computational complexity, the occupancy maps are coarse. The algorithm executes a second pass in order to reconstruct the environment.

Both Bircher et al. and Bai et al. assume that the sensor range is smaller than the

domain size. They use a notion of information gain that makes no distinction between obstacles and free space. That is, each unmapped voxel is assumed to be unoccupied if it comes within range of the sensor.

Our work is closer to the ideas of Valente et al., using a gain function to steer a greedy approach. We also assume that the visibility range is larger than the domain, which makes the problem more global and challenging. We leverage the convenience of a volumetric representation of the geometries and convolutional neural networks for efficiency. Our non-myopic framework allows for steps of arbitrary distances and takes the geometry of the environment into account.

III. METHODOLOGY

The data needed for the training and evaluation of g_θ are computed by the level set method [14], [16], [13]. The training geometry is embedded by a level set function, denoted by ϕ. For each vantage point x_i, the visibility set is represented by the level set function ψ_{x_i}, which is computed efficiently using the algorithm described in [19].

In the calculus of level set functions, unions and intersections of sets are translated, respectively, into taking maximum and minimum of the corresponding characteristic functions. The cumulatively visible sets Ω_k are represented by the level set function $\Psi_k(x)$, which is defined recursively by $\Psi_k = \max(\Psi_{k-1}, \psi_{x_k})$, point-wise, with $\Psi_0 = \psi_{x_0}$.

Thus we have $\Omega = \{\phi > 0\}$, $\mathcal{V}_{x_i}\Omega = \{\psi_{x_i} > 0\}$, and $\Omega_k = \{\Psi_k > 0\}$. The shadow boundaries B_k are approximated by the "smeared out" delta function b_k:

$$b_k(x) := \delta_\varepsilon(\Psi_k) \cdot [1 - H(\delta_\varepsilon(\phi))], \tag{4}$$

where $\delta_\varepsilon(x) = \frac{2}{\varepsilon}\cos^2\left(\frac{\pi x}{\varepsilon}\right) \cdot \mathbb{1}_{[-\frac{\varepsilon}{2},\frac{\varepsilon}{2}]}(x)$, and $H(x)$ is the Heaviside function. In our implementation, we take $\varepsilon = 3\Delta x$ where Δx is the grid node spacing. We refer the readers to [20] for a short review of relevant details.

A. SURVEILLANCE

When the environment Ω is known, we can compute the gain function exactly

$$g(x; \Omega_k, \Omega) = \int H(\psi_x(\xi) - \Psi_k(\xi))\, d\xi. \tag{5}$$

We remark that $H(\psi - \Psi)$ will be 1 where the new vantage point uncovers something not previously seen. Computing g for all x is costly. We approximate it with a function $\tilde{g}_\theta$ parameterized by θ such that

$$\tilde{g}_\theta(x; \Psi_k, \phi, b_k) \approx g(x; \Omega_k, \Omega).$$

B. EXPLORATION

If the environment is unknown, we directly approximate the gain function by learning the parameters θ of a function

$$g_\theta(x; \Psi_k, b_k) \approx g(x; \Omega_k, \Omega) H(\Psi_k)$$

using only the observations as input. Note the $H(\Psi_k)$ factor is needed for collision avoidance during exploration because it is not known *a priori* whether an occluded location y is part of an obstacle or free space. Thus $g_\theta(y)$ must be zero.

C. TRAINING

Ω is randomly sampled from a library. For each Ω, a sequence of data pairs is generated and included into the training set $\mathcal{T}$:

$$\big(\{\Psi_k, b_k\}, g(x; \Omega_k, \Omega) H(\Psi_k)\big), \qquad k = 0, 1, 2, \ldots.$$

The function g_θ is learned by minimizing the empirical loss across all data pairs for each Ω in the training set $\mathcal{T}$:

$$\underset{\theta}{\operatorname{argmin}} \ \frac{1}{N} \sum_{\Omega \in \mathcal{T}} \sum_{k} L\Big(g_\theta(x; \Psi_k, b_k), g(x; \Omega_k, \Omega) H(\Psi_k)\Big)$$

where N is the total number of data pairs. We use the cross entropy loss function:

$$L(p, q) = \int p(x) \log q(x) + (1 - p(x)) \log(1 - q(x)) \ dx$$

D. MODEL ARCHITECTURE

We use convolutional neural networks (CNNs) to approximate the gain function, which depends on the shape of Ω and the location x. CNNs have been used to approximate functions of shapes effectively in many applications. The gain function $g(x)$ does not depend *directly* on x, but rather, x's visibility of Ω, with a domain of dependence bounded by the sensor range. So, we expect certain translation invariance in the computation of the gain function. Thus, we employ a fully convolutional approach for learning g. Finally, CNN's feedforward evaluations are efficient if the off-line training cost is ignored. Our approach can therefore be easily applied to domains of different sizes, with straightforward generalization to 3D.

We base the architecture of the CNN on U-Net [15], which has had great success in dense inference problems, such as image segmentation. It aggregates information from various layers in order to have wide receptive fields while maintaining pixel precision. The main design choice is to make sure that the receptive field of our model is sufficient.

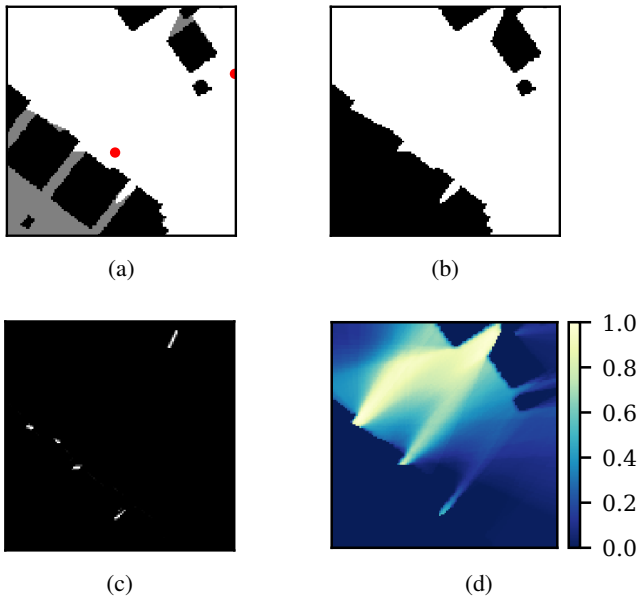

Fig. 3: A training data pair consists of the cumulative visibility and shadow boundaries as input, and the gain function as the output. a) The underlying map with current vantage points shown in red. b) The cumulative visibility of the current vantage points. c) The corresponding shadow boundaries. d) The corresponding gain function.

That is, we want to make sure that the value predicted at each voxel depends on a sufficiently large neighborhood. For efficiency, we use convolution kernels supported in the 3^d-pixel set. By stacking multiple layers, we can achieve large receptive fields. Thus the complexity for feedforward computations is linear in the total number of grid points.

Define a *conv block* as the following layers: convolution, batch norm, leaky `relu` activation, stride 2 convolution, batch norm, and leaky `relu` activation. Each *conv block* reduces the image size by a factor of 2. The latter half of the network increases the image size using *deconv blocks*: bilinear 2x upsampling, convolution, batch norm, and leaky `relu` activation.

Our 2D network uses 6 *conv blocks* followed by 6 *deconv blocks*, while our 3D network uses 5 of each block. We choose the number of blocks to ensure that the receptive field is at least the size of the training images: 128x128 and 64x64x64. The first *conv block* outputs 4 channels. The number of channels doubles with each *conv block*, and halves with each *deconv block*.

The network ends with a single channel, kernel of size 1 convolution layer followed by the sigmoid activation. This ensures that the network aggregates all information into a prediction of the correct size and range.

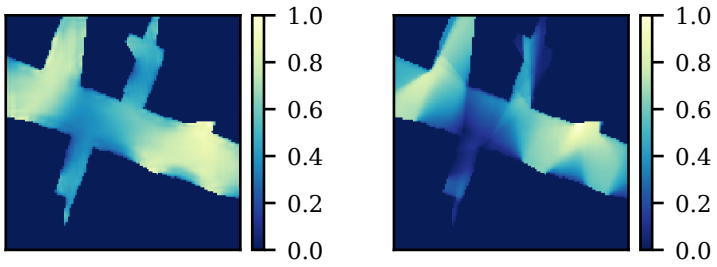

Fig. 4: Comparison of predicted (left) and exact (right) gain function for an Austin map. Although the functions are not identical, the predicted gain function peaks in similar locations to the exact gain function, leading to similar steps.

IV. EXPERIMENTS

We present some experiments to demonstrate the efficacy of our approach. Also, we demonstrate its limitations.

First, we train on 128x128 aerial city blocks cropped from INRIA Aerial Image Labeling Dataset [11]. It contains binary images with building labels from several urban areas, including Austin, Chicago, Vienna, and Tyrol. We train on all the areas except Austin, which we hold out for evaluation. We call this model **City-CNN**. We train a similar model **NoSB-CNN** on the same training data, but omit the shadow boundary from the input. Third, we train another model on synthetically-generated radial maps, such as the one in Figure 10. We call this model **Radial-CNN**.

Given a map, we randomly select an initial location. In order to generate the sequence of vantage points, we apply (2), using g_θ in place of g. Ties are broken by choosing the closest point to x_k. We repeat this process until there are no shadow boundaries, the gain function is smaller than ϵ, or the residual is less than δ, where the residual is defined as:

$$r = \frac{|\Omega \setminus \Omega_k|}{|\Omega|}. \tag{6}$$

We compare these against the algorithm which uses the exact gain function, which we call **Exact**. We also compare against **Random**, a random walker, which chooses subsequent vantage points uniformly from the visible region. We analyze the number of steps required to cover the scene and the residual as a function of the number of steps.

Lastly, we present a simulation for exploring a 3D urban environment. Due to the lack of adequate 3D urban datasets, the model, **3D-CNN**, is trained using synthetically-generated 64x64x64 voxel images consisting of tetrahedrons, cylinders, ellipsoids, and

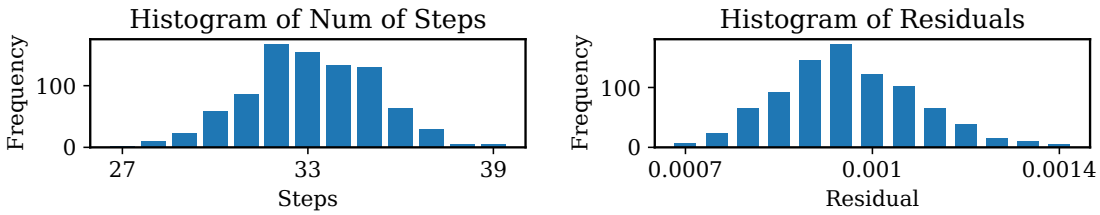

Fig. 5: Distribution of the residual and number of steps generated across multiple runs over an Austin map. The proposed method is robust against varying initial conditions. The algorithm reduces the residual to roughly 0.1 % within 39 steps by using a threshold on the predicted gain function as a termination condition.

cuboids of random positions, sizes, and orientations. In the site[1], the interested reader may inspect the performance of the **3D-CNN** in some other challenging 3D environments.

For our experiments using trained networks, we make use of a CPU-only machine containing four Intel Core i5-7600 CPU @ 3.50GHz and 8 GB of RAM. Additionally, we use an Nvidia Tesla K40 GPU with 12 GB of memory for training the convolutional neural networks and predicting the gain function in 3D scenes.

A. 2D CITY

The **City-CNN** model works well on 2D Austin maps. First, we compare the predicted gain function to the exact gain function on a 128x128-pixel map, as in Figure 4. Without knowing the underlying map, it is difficult to accurately determine the gain function. Still, the predicted gain function peaks in locations similar to those in the exact gain function. This results in similar sequences of vantage points.

The algorithm is robust to the initial positions. Figure 5 show the distribution of the number of steps and residual across over 800 runs from varying initial positions over a 512x512 Austin map. In practice, using the shadow boundaries as a stopping criteria can be unreliable. Due to numerical precision and discretization effects, the shadow boundaries may never completely disappear. Instead, the algorithm terminates when the maximum predicted gain falls below a certain threshold ϵ. In this example, we used $\epsilon = 0.1$. Empirically, this strategy is robust. On average, the algorithm required 33 vantage points to reduce the occluded region to within 0.1% of the total explorable area.

Figure 6 shows an example sequence consisting of 36 vantage points. Each subsequent step is generated in under 1 sec using the CPU and instantaneously with a GPU.

Even when the maximizer of the predicted gain function is different from that of the exact gain function, the difference in gain is negligible. This is evident when we see

[1]http://visibility.page.link/demo

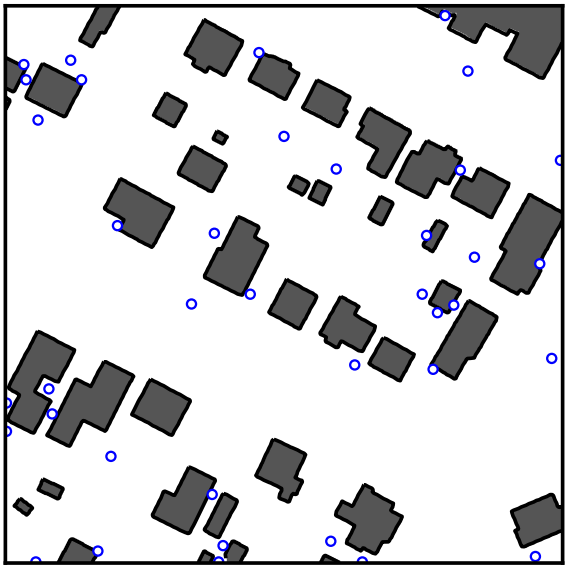

Fig. 6: An example of 36 vantage points (blue disks) using **City-CNN** model. White regions are free space while gray regions are occluded. Black borders indicate edges of obstacles.

the residuals for **City-CNN** decrease at similar rates to **Exact**. Figure 9 demonstrates an example of the residual as a function of the number of steps for one such sequence generated by these algorithms on a 1024x1024 map of Austin. We see that **City-CNN** performs comparably to **Exact** approach in terms of residual. However, **City-CNN** takes 140 secs to generate 50 steps on the CPU while **Exact** takes more than 16 hours to produce 50 steps.

B. EFFECT OF SHADOW BOUNDARIES

The inclusion of the shadow boundaries as input to the CNN is critical for the algorithm to work. Without the shadow boundaries, the algorithm cannot distinguish between obstacles and occluded regions. If an edge corresponds to an occluded region, then choosing a nearby vantage point will reduce the residual. However, choosing a vantage point near a flat obstacle will result in no change to the cumulative visibility. At the next iteration, the input is same as the previous iteration, and the result will be the same; the algorithm becomes stuck in a cycle. To avoid this, we prevent vantage points from repeating. Still, the vantage points tend to cluster near flat edges, as in Figure 7. This clustering behavior causes the **NoSB-CNN** model to be, at times, worse than **Random**. See Figure 9 to see how the clustering inhibits the reduction in the residual.

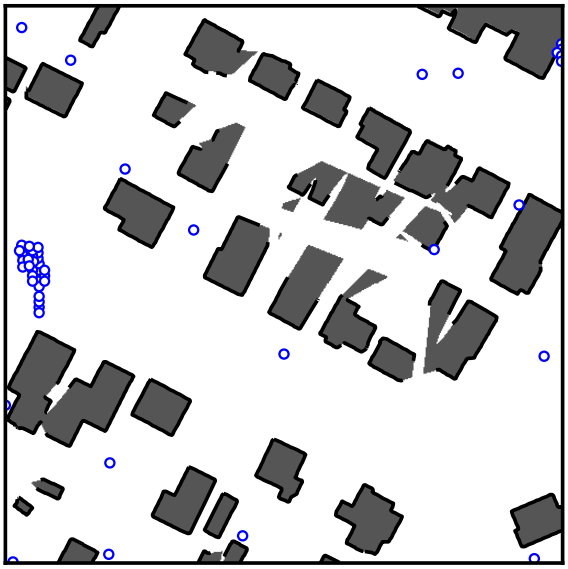

Fig. 7: A sequence of 50 vantage points generated from **NoSB-CNN**. The points cluster near flat edges due to ambiguity and the algorithm becomes stuck. Gray regions without black borders have not been fully explored.

C. EFFECT OF SHAPE

The shape of the obstacles, i.e. Ω^c, used in training affects the gain function predictions. Figure 10 compares the gain functions produced by **City-CNN** and **Radial-CNN**.

D. FREQUENCY MAP

The efficiency of the CNN method allows us to address many surveillance related questions. Here we present one of our studies concerning the exclusivity of vantage point placements in Ω. We generated sequences of vantage points starting from over 800 different initial conditions using **City-CNN** model on a 512x512 Austin map. Then, we model each vantage point as a Gaussian with fixed width, and overlay the resulting distribution on the Austin map in Figure 8. This gives us a frequency map of the most recurring vantage points. These hot spots reveal regions that are more secluded and therefore, the visibility of those regions is more sensitive to vantage point selection.

E. ART GALLERY PROBLEM

Our proposed approach outperforms the computational geometry solution [12] to the art gallery problem, even though we do not assume the environment is known. The key issue with computational geometry approaches is that they are heavily dependent on

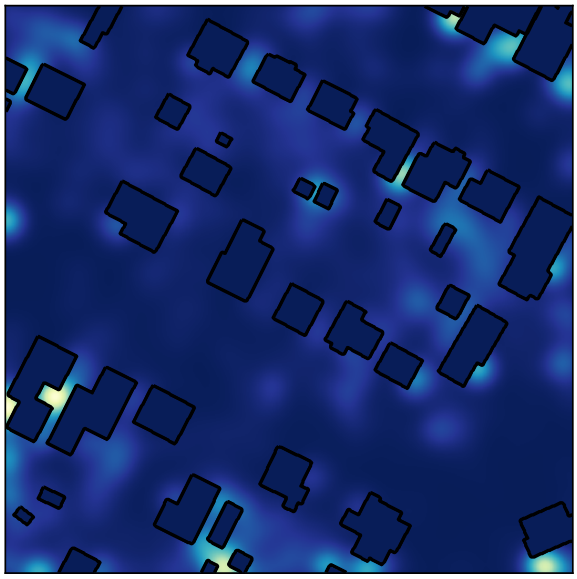

Fig. 8: Distribution of vantage points generated by **City-CNN** method from various initial positions. Hot spots are brighter and are visited more frequently since they are essential for completing coverage.

the triangulation. In an extreme example, consider an art gallery that is a simple convex *n-gon*. Even though it is sufficient to place a single vantage point anywhere in the interior of the room, the triangulation-based approach produces a solution with $\lfloor n/3 \rfloor$ observers.

Figure 11 shows an example gallery consisting of 58 vertices. The computational geometry approach requires 19 vantage points to completely cover the scene, while **City-CNN**, on average, across various initial positions, requires only 8 vantage points, despite not knowing the environment.

F. 3D ENVIRONMENT

We present a 3D simulation of a 250m$\times$250m environment based on Castle Square Parks in Boston. The map is discretized as a level set function on a 768x768x64 voxel grid. At this resolution, small pillars are accurately reconstructed by our exploration algorithm. Each step can be generated in 3 seconds using the GPU or 300 seconds using the CPU. Parallelization of the distance function computation will further reduce the computation time significantly. A map of this size was previously unfeasible. See Figure 12 for snapshots of the algorithm in action. See Supplemental Material for a video clip of the exploration process.

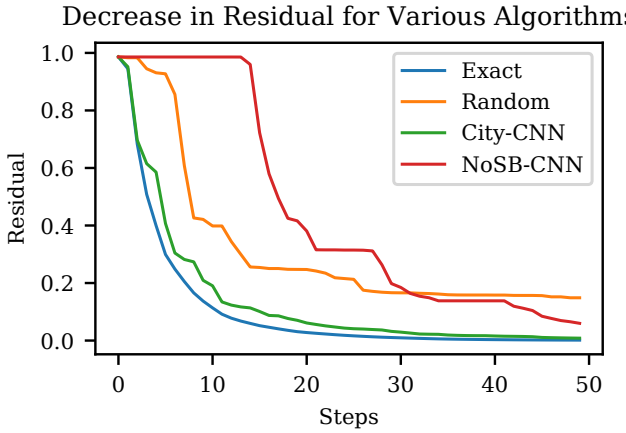

Fig. 9: Graph showing the decrease in residual over 50 steps among various algorithms starting from the same initial position for an Austin map. Without using shadow boundary information, **NoSB-CNN** can at times be worse than **Random**. Our **City-CNN** model is significantly faster than **Exact** while remaining comparable in terms of residual.

V. CONCLUSION

From the perspective of inverse problems, we proposed a greedy algorithm for autonomous surveillance and exploration. We show that this formulation can be well-approximated using convolutional neural networks. The inclusion of shadow boundaries, computed using the level set method, is crucial for the success of the algorithm. Our approach is amenable to a wide range of three dimensional real-world applications.

ACKNOWLEDGMENT

This work was partially supported by NSF Grant DMS-172017. We thank Texas Advanced Computing Center (TACC) for providing the computational resources that made this work possible. Tsai also thanks National Center for Theoretical Sciences (NCTS) for hosting his visits to the center, where part of the research was being conducted.

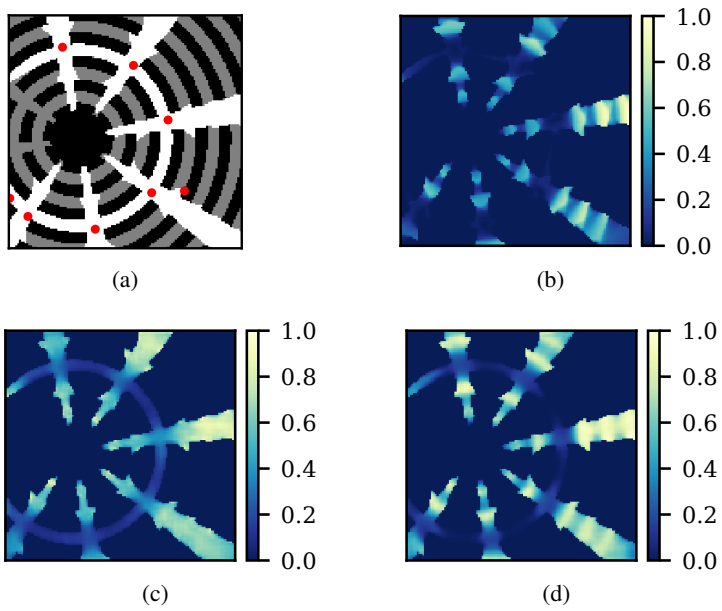

Fig. 10: Comparison of gain functions produced with various models on a radial scene. Naturally, the CNN model trained on radial obstacles best approximates the true gain function. a) The underlying radial map with vantage points show in red. b) The exact gain function c) **City-CNN** predicted gain function. d) **Radial-CNN** predicted gain function.

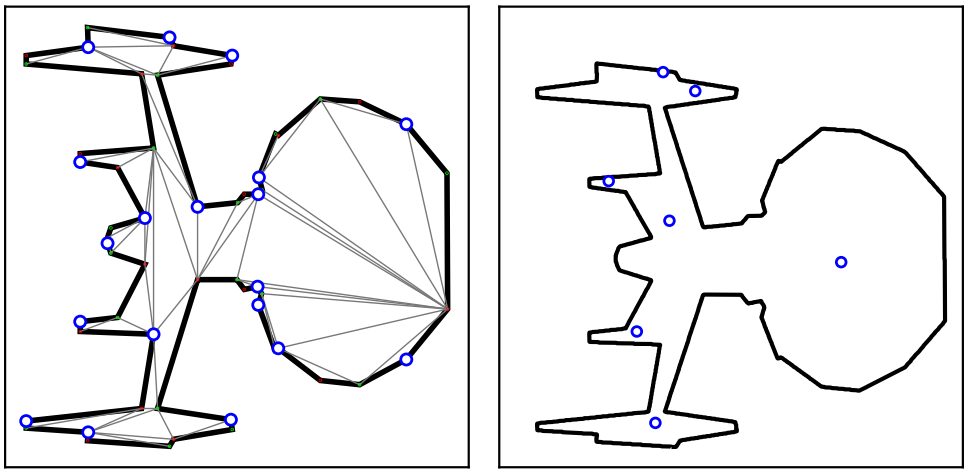

Fig. 11: Comparison of the computational geometry approach and the **City-CNN** approach to the art gallery problem. The blue circles are the vantage points computed by the methods. Left: A result computed by the computational geometry approach, given the environment. Right: An example sequence of 7 vantage points generated by the **City-CNN** model.

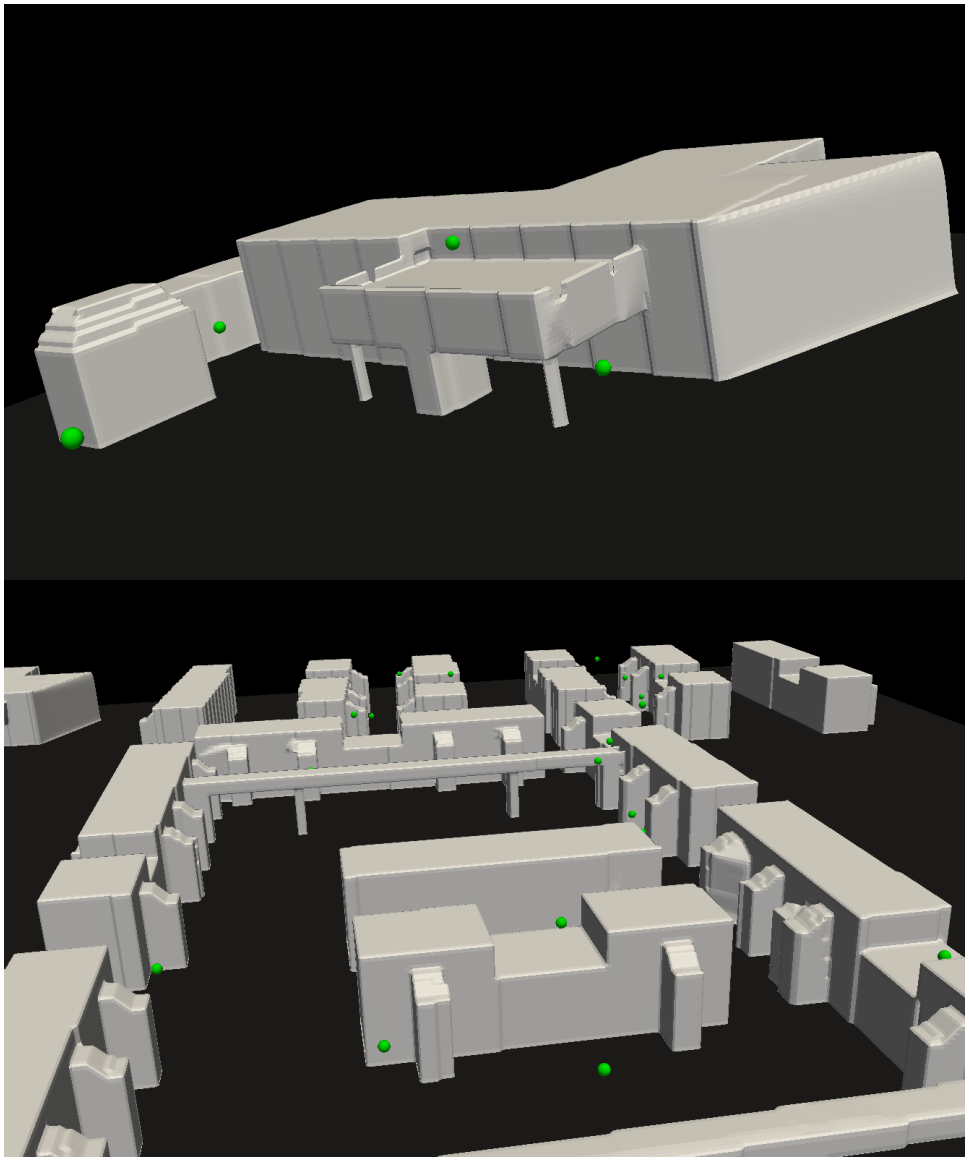

Fig. 12: Snapshots demonstrating the exploration of an initially unknown 3D urban environment using sparse sensor measurements. The green spheres indicate the vantage point. The gray surface is the reconstruction of the environment based on line of sight measurements taken from the sequence of vantage points. New vantage points are computed in virtually real time using **3D-CNN**.

REFERENCES

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

[2] A. Bircher, M. Kamel, K. Alexis, H. Oleynikova, and R. Siegwart. Receding horizon" next-best-view" planner for 3d exploration. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 1462–1468. IEEE, 2016.

[3] A. Bircher, M. Kamel, K. Alexis, H. Oleynikova, and R. Siegwart. Receding horizon path planning for 3d exploration and surface inspection. *Autonomous Robots*, 42(2):291–306, 2018.

[4] V. Chvátal. A combinatorial theorem in plane geometry. *Journal of Combinatorial Theory*, B(18):39–41, 1975.

[5] R. Goroshin, Q. Huynh, and H.-M. Zhou. Approximate solutions to several visibility optimization problems. *Communications in Mathematical Sciences*, 9(2):535–550, 2011.

[6] S. H. Kang, S. J. Kim, and H. Zhou. Optimal sensor positioning; a probability perspective study. *SIAM Journal on Scientific Computing*, 39(5):B759–B777, 2017.

[7] Y. Landa, D. Galkowski, Y. R. Huang, A. Joshi, C. Lee, K. K. Leung, G. Malla, J. Treanor, V. Voroninski, A. L. Bertozzi, Y.-H. R. Tsai, et al. Robotic path planning and visibility with limited sensor data. In *American Control Conference, 2007. ACC'07*, pages 5425–5430. IEEE, 2007.

[8] Y. Landa and Y.-H. R. Tsai. Visibility of point clouds and exploratory path planning in unknown environments. *Communications in Mathematical Sciences*, 6(4):881–913, 2008.

[9] Y. Landa, Y.-H. R. Tsai, and L.-T. Cheng. Visibility of point clouds and mapping of unknown environments. In *International Conference on Advanced Concepts for Intelligent Vision Systems*, pages 1014–1025. Springer, 2006.

[10] S. LaValle. *Planning algorithms*. Cambridge University Press, 2006.

[11] 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)*. IEEE, 2017.

[12] J. O'Rourke. *Art gallery theorems and algorithms*, volume 57. Oxford University Press Oxford, 1987.

[13] S. Osher and R. Fedkiw. *Level set methods and dynamic implicit surfaces*, volume 153. Springer Science & Business Media, 2006.

[14] S. Osher and J. A. Sethian. Fronts propagating with curvature-dependent speed: algorithms based on hamilton-jacobi formulations. *Journal of computational physics*, 79(1):12–49, 1988.

[15] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[16] J. A. Sethian. *Level set methods and fast marching methods: evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science*, volume 3. Cambridge university press, 1999.

[17] B. Tovar, L. Guilamo, and S. LaValle. Gap navigation trees: Minimal representation for visibility-based tasks. In *Algorithmic Foundations of Robotics VI*, pages 425–440. Springer, 2004.

[18] B. Tovar, R. Murrieta-Cid, and S. LaValle. Distance-optimal navigation in an unknown environment without sensing distances. *IEEE Transactions on Robotics*, 23(3):506–518, 2007.

[19] Y.-H. R. Tsai, L.-T. Cheng, S. Osher, P. Burchard, and G. Sapiro. Visibility and its dynamics in a PDE based implicit framework. *J. Comput. Phys.*, 199(1):260–290, 2004.

[20] Y.-H. R. Tsai and S. Osher. Total variation and level set methods in image science. *Acta Numer.*, 14:509–573, 2005.

[21] J. Urrutia. Art gallery and illumination problems. In *Handbook of Computational Geometry*, pages 973–1027. Elsevier, 2000.

[22] L. Valente, Y.-H. R. Tsai, and S. Soatto. Information-seeking control under visibility-based uncertainty. *Journal of Mathematical Imaging and Vision*, 48(2):339–358, 2014.

[23] F. Zhang, A. O'Connor, D. Luebke, and P. Krishnaprasad. Experimental study of curvature-based control laws for obstacle avoidance. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 4, pages 3849–3854. IEEE, 2004.