Deep Learning of Structured Environments for Robot Search

Jeffrey A. Caley, Nicholas R.J. Lawrance and Geoffrey A. Hollinger

***Abstract*— Robots often operate in built environments containing underlying structure that can be exploited to help predict future observations. In this work, we present a deep learning based approach to predict exit locations of buildings. This technique exploits the inherent structure of buildings to create a model. A convolutional neural network is trained using a database of building blueprints and used to guide a search within a building. This technique is compared to standard frontier exploration and a traditional image processing approach of extracting features through histogram of gradients (HOG) and training a support vector machine (SVM). After validation through simulation, we show that the proposed deep learning technique reduces the amount of building exploration required to find the goal by 36%.**

I. Introduction

A challenging problem in robotics is to predict future observations based on previously-recorded data. Robots often operate in built environments that tend to contain some underlying structure, such that newly-visited locations may appear broadly similar to previously visited locations but differ in individual details. A particularly recognizable instance of this structure is in buildings. Although buildings may differ significantly in their layout, shape, size and contents, there are certain features that tend to be present in the majority of buildings. Common features, such as hallways tend to be long, narrow and contain multiple entrances to other rooms. Further, buildings used for similar purposes, such as office buildings, tend to share more common features such as the frequency and relative location of bathrooms, offices and exits. The work presented in this paper utilizes deep learning techniques to learn the structure of buildings to predict the location of the building exit in the context of a robot search problem (see Fig. 1).

Deep learning has recently shown incredible success in many learning and pattern-recognition problems, particularly in the areas of image classification and speech recognition. Deep learning's ability to learn high level features through multiple layers of lower level features lends itself to the problem of learning the underlying structure of complex images. This is because structure is characterized by the organization of many lower level components, such as rooms, hallways and bathrooms, into larger structures (e.g., buildings). These similarities suggest that deep learning may be beneficial in the robotic search domain.

This work has been funded in part by the Office of Naval Research grant N00014-14-1-0509.

J. Caley, N. Lawrance and G. Hollinger are with the Robotics Program, School of Mechanical, Industrial and Manufacturing Engineering, Oregon State University {caleyj, nicholas.lawrance, geoff.hollinger}@oregonstate.edu.

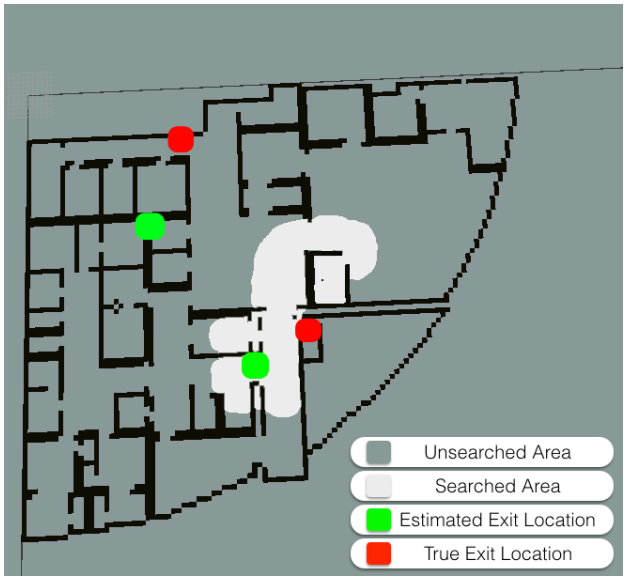


Fig. 1: A TurtleBot exploring a map as it searches a building for an exit. Black pixels represent obstacles, dark gray pixels denote unsearched area, light gray pixels represent the area searched. Red squares denote true locations of exits while green squares represent estimated locations.

The proposed method utilizes deep learning to predict exit location. A mobile robot is placed into a previously unseen building and given access to the building floor plan. Using a convolutional neural network (CNN) to predict exit locations and a simple frontier-based exploration algorithm, we show that the CNN-based prediction finds exit locations with less exploration than frontier-based exploration alone or an alternative image feature classification method. The main novelty of this paper is the use of deep learning techniques to exploit structural information not previously utilized in robot search domains.

The paper is divided into seven sections. Section II outlines the existing work in artificial neural networks, learning structured environments and robot exploration. Section III provides a clear formulation of the problem addressed in this paper. The proposed method is described in Section IV, and Section V describes the building floor plan dataset used for training and testing. Results of simulated exploration and associated analysis is provided in Section VI. Concluding remarks and avenues for future work are suggested in Section VII.

II. Related Work

A. Artificial Neural Networks

Artificial neural networks (ANNs) are a statistical learning model loosely based on biological neural networks. ANNs

are represented as a system of neurons with weighted connections between them. These weights are tuned as the network is presented with examples, enabling the network to approximate any function. Neural networks have been used to solve a wide variety of tasks, including image classification [1] and speech recognition [2].

B. Convolutional Neural Networks

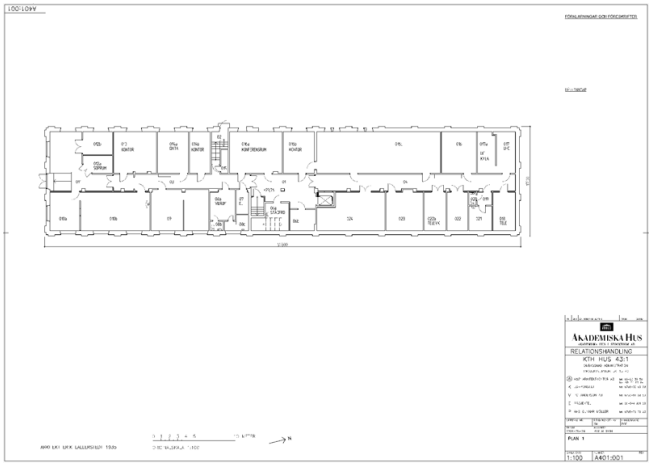
A convolutional neural network (CNN) is a special implementation of an ANN that is specifically designed to handle images as inputs [3]. Recent research has demonstrated CNN implementations solving a variety of previously challenging problems. In 2012, Ciresan et al. used a CNN architecture to achieve near-human performance on the MNIST handwriting dataset and outperformed humans by a factor of two on traffic sign recognition [4]. In 2014, CNNs were used to play the board game Go with excellent results [5] and recently beat the world champion. Using a database of human professional games, the CNN trained via supervised learning was able to predict an expert's move 55% of the time and beat the traditional search program GnuGo in 97% of games. It was also able to match the performance of Monte-Carlo tree search methods while using significantly less computation time.

A recent paper by Levine et al. gives us a look at how CNNs can be incorporated into robotic applications [6]. In that work, a CNN is trained to recognize the position of an end effector and target, then directly translate those predictions into motor commands that move the robot to a desired position. The robot is able to perform simple tasks such as hanging a coat hanger or capping a bottle using only a single neural network for both sensor processing and motor control. Lenz et al. [7] use deep neural networks to learn models for model predictive control. Finally, using state-of-the-art learning techniques, CNNs have been able to match human performance in classifying images on the Large Scale Visual Recognition Challenge dataset [8].

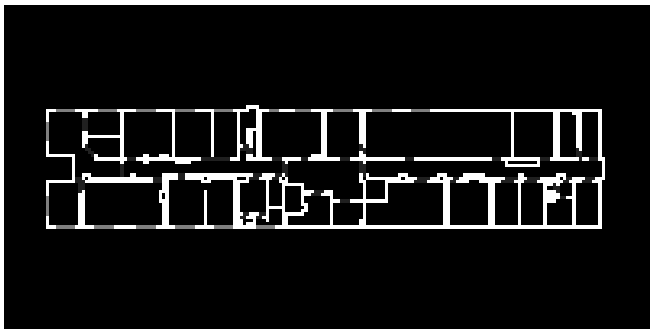
Deep learning has also been applied to planning. The Google DeepMind team created a Q-learning algorithm where a deep neural network replaced the traditional method of performing value iteration with a table of Q-values [9]. The network predicted Q-values given a state-action pair. The actual reward was used with the Bellman update to train the neural net. Thus, the neural network acted as a function approximator for the Q-value table. The major benefit is that the neural net was able to handle very large state-spaces, at the expense of requiring many training examples.

C. Robotic Search and Exploration

Frontier search is one of the most common exploration algorithm in use today. This technique stems from work done by Yamauchi [10]. The robot keeps track of the frontiers between the explored and unexplored regions and then picks the best frontier to explore. A variety of methods to pick the frontier have been studied, including the closest frontier centroid [10], segmenting the frontiers into graphs [11], auctioning frontiers to robots [12], and balancing the benefit



(a) An example floor plan from the KTH dataset. This dataset is used to train and test informed robotic search.

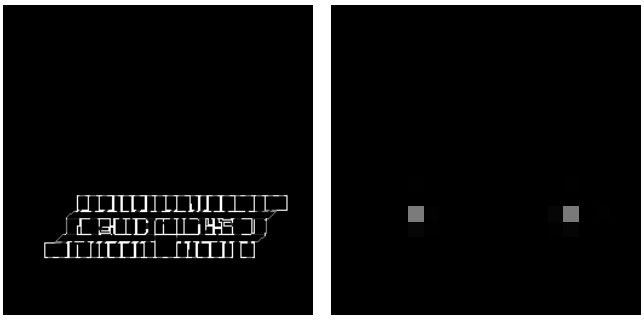


(b) The corresponding floor plan image from the building's XML description. Note that this shows a 256×128 window but the images input to the CNN are all (black padded) 256×256 square images.

Fig. 2: CNN input image generated from an XML description of an office floor plan.

of exploring the frontier with the cost of moving to it [11], [13]. These techniques all rely on the ability to efficiently identify frontiers. Our work dovetails nicely with this work by providing an estimation of the best frontier to expand. By augmenting exploration algorithms with our predictions about the unexplored areas of structured environments, we show that the efficiency of search can be significantly improved.

There is some previous work on identifying patterns based on floor plans. Macé et al. [14] combined building structure heuristics with vision processing techniques to automatically identify rooms and doors from floor plans. However, they do not plan over these identified features. Aydemir et al. [15] takes this one step further by learning common room groupings with a sub-graph based method, then using those groupings to predict the features of unseen rooms. Their method learns based on a dataset of 940 floor plans. They only learn abstract room classifications, but unlike in this paper, they do not extend their work to planning over the learned graphs. Perea Ström et al. [16] use a bag of words approach to match current incomplete maps to a database of previously viewed maps to predict how the environment will expand into unknown areas. This work is similar to ours in



(a) (b)

Fig. 3: (a) An example 256×256 pixel scaled and translated floor plan input image. (b) The corresponding 20×20 pixel image of labeled exit locations. Grey pixels represent exit locations (two in this image). These exit locations are used at the output for training the classification neural network.

that it assumes a common structure between maps, but the approach and goals differ.

III. Problem Formulation

In this work we simulate a robot searching a building for an exit. Starting from a random room located within the building, the robot must search the environment until an exit is found. The goal of the robot is to minimize the area explored to reach an exit.

To explore the building, frontier exploration [10] is used to generate frontiers that guide exploration. The frontier closest to an estimated exit is selected as the robot's next waypoint. As the robot moves, frontiers are continually updated and new waypoints are provided to the robot. The robot keeps exploring in this way until it observes an exit with its laser scanner.

More formally, we assume a bounded planar workspace $\Omega \subseteq \mathbb{R}^2$. The workspace is partitioned into free space ω_{free}, obstacle space ω_{obs} and goal space ω_{goal}. This workspace partitioning is initially unknown to the exploring robot. During exploration along a trajectory $\mathcal{P}$, the laser scanner on the robot reveals the region $\omega_{known}(\mathcal{P})$. Let $\lambda(\cdot)$ denote the area of a region, and $A(\mathcal{P})$ be the proportion of the entire workspace Ω uncovered during execution of path $\mathcal{P}$,

$$A(\mathcal{P}) = \frac{\lambda(\omega_{known}(\mathcal{P}))}{\lambda(\Omega)}. \quad (1)$$

The optimal path $\mathcal{P}^*$ is the path that reveals the smallest region $\lambda(\omega_{known})$ before finding a goal region. Formally stated:

$$\mathcal{P}^* = \arg\min_{\mathcal{P}} A(\mathcal{P}) \quad \text{s.t. } \omega_{known}(\mathcal{P}) \cap \omega_{goal} \neq \emptyset. \quad (2)$$

IV. Method

We describe two methods for exit prediction based on blueprints investigated in this work. One uses a traditional image processing technique of extracting histograms of oriented gradients (HOG) features from the blueprints and then trains a support vector machine (SVM) to predict exit location. The second is our proposed technique of using a deep network to learn exit locations based on the inherent structure of the building.

A. HOG features plus SVM

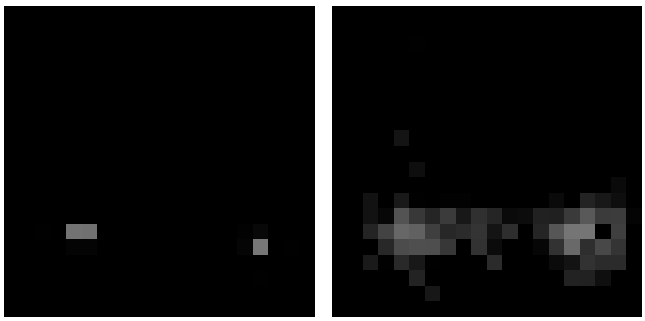
To provide a baseline in which to compare our deep learning method, a traditional image recognition and machine learning technique has been implemented.

The histogram of oriented gradients is a technique that counts occurrences of gradient orientation in localized portions of an image [17]. We use the Matlab toolbox provided by [18] to implement HOG. We start by extracting 12×12 sub-images with 6 pixel overlap across all of our blueprints in the training set and extract HOG features. The bin size used is 6 with 10 orientation bins. This returns 80 features for each sub-image. Each 12×12 sub-image is also labeled with a one if it contains an exit location or a zero for non-exit locations. With this, we now have a set of 80 features with corresponding class labels. With a training set of features and class labels, we use a support vector machine with a linear kernel to learn a model for classification. This model is then used to predict exit locations for unlabeled blueprints. When more exits are predicted in a blueprint than actually exist, the n highest scoring exits are used, where n is the number of exits that exist for the building.

B. Deep Learning

Our proposed method of using structured information to make inferences is to use a convolution neural network (CNN) to predict exit locations. The CNN is trained using a downsampled 256×256 blueprint image as an input. The output is a 20×20 image with pixel values of 0 for non-exit locations and 1 for exit locations. 20×20 images were chosen as the output size for two reasons. A larger output image would increase the complexity the CNN needs to learn and the CNN's ability to predict exit location doesn't achieve an accuracy where more resolution would increase performance. The CNN architecture used in our work is as follows (see Fig. 5):

1) Input layer ($1\times256\times256$)
2) Convolutional layer (16 filters, filter size 9)
3) Max pooling (size 2)
4) RELU Non-Linear Layer



(a) (b)

Fig. 4: (a) Labeled output locations. (b) The output estimate of the CNN. This image is clustered to find an exact location of the exit.

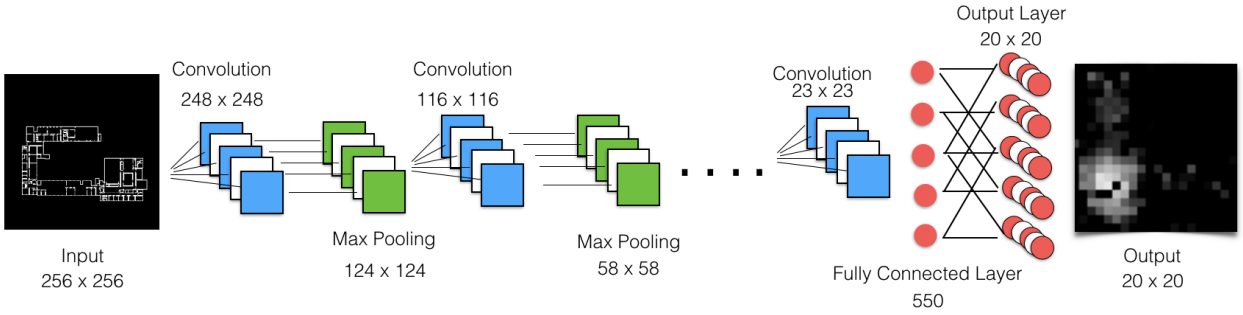


Fig. 5: An illustration of our network structure. A 256×256 image is used as an input into our convolutional neural network structure consisting of four convolution, max pool and RELU layers. This results in a 16×23×23 image that is connected to a 550 node fully connected layer. This is trained through supervised learning and outputs a heat map of estimated exit locations. Pixels in the heat map are clustered to identify exit locations.

5) Convolutional layer (16 filters, filter size 9)
6) Max pooling (size 2)
7) RELU Non-Linear Layer
8) Convolutional layer (16 filters, filter size 5)
9) Max pooling (size 2)
10) RELU Non-Linear Layer
11) Convolutional layer (16 filters, filter size 5)
12) RELU Non-Linear Layer
13) Fully Connected layer (550 nodes)
14) Output layer (400 nodes - 20×20)

The error function used to train the full neural network is calculated by performing a pixel-by-pixel comparison of the image of estimated exit location with the ground truth image of the true exit location (see Algorithm 1). In order to balance positive and negative exit locations, the error is multiplied by a value of 400/3 when a false negative occurs (i.e., when the network estimates there is no exit in a location where one actually exists). This normalization value is due to the fact that on average each exit location image has 3 white pixels, denoting exits, out of 400 total pixels. This normalization factor encourages the CNN to predict a 'heat map' of possible exit locations, as shown in Fig. 4(b), instead of learning to output an all black image.

Algorithm 1 Loss Function

procedure LOSS FUNCTION(Prediction, target)
  $LTT \leftarrow prediction < target$
  $GTT \leftarrow prediction > target$
  $output \leftarrow (\frac{400}{3} LTT + GTT)(target - prediction)^2$
  return output

From the output images of the CNN, a clustering method is used to determine the exact location of the exit prediction. We apply k-means clustering on the output image to find the center of the pixel cluster. When clustering, the effective weight of each pixel is scaled by that pixel's value, in the range $[0, 255]$. Thus, the brighter the pixel, the more it influences where the cluster center is placed. Once the cluster center is identified, we can compare the location of the estimated exit with the actual location of the exit, as shown in Fig. 6.

V. Dataset Generation

The CNN used in this work predicts exit locations after having been trained on a large number of labeled floor plans. The set of floor plans used in this paper is derived from the KTH dataset used in [15]. This dataset consists of 161 hand-labeled PDF blueprints, shown in Fig. 2(a), of building floors on the KTH Royal Institute of Technology's campus. Each blueprint has a corresponding XML file describing the locations of the walls, doors, and centroid for each room shown in the blueprint. Also, each room is labeled with a category, such as hallway, office, classroom, etc. The doorways describe paths between the rooms in the blueprints, either physical doors or logical transitions between them. We augment the provided XML by labeling the exit locations for each floor plan. These exit locations correspond either to stairways leading to lower floors or exterior doors.

In order to transform the data in the dataset into a form usable by the neural network, we construct a 256×256 pixel gray scale image from the wall and window data given in each blueprint's XML file, with pixels values of 255 corresponding to the locations of walls, 128 for windows and 0 for doors and open spaces. Exits are not labeled on the simplified images. This image size and format greatly reduces the complexity of the images and standardizes them across different blueprint notation, while still maintaining enough detail to allow the building's features and structure

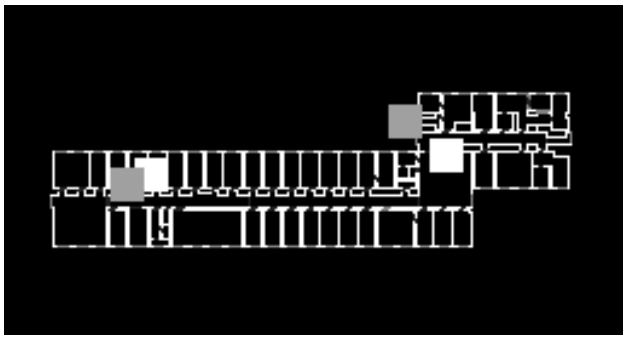


Fig. 6: An example of the CNN's ability to predict exit location. The gray squares are the true exit location, the white squares are the CNN's estimated exit location

to be readily recognized.

Many thousands of examples are required to effectively train a deep neural network, yet only 161 blueprints are included in the data set. In order to overcome this limitation, we generate a series of images where the blueprint has been scaled, translated and rotated on the 256×256 canvas. Through this, we create a dataset of 10,304 images. The data is partitioned into three image sets; a training set consisting of 6907 images, a validation set of 823 and a testing set consisting of 2575 images. Training of the network takes approximately 22 minutes on an Intel i7-4790K with a GeForce GTX TITAN Z GPU.

VI. Results and Discussion

In this section we report on the results achieved in predicting exit locations using HOG features with SVM and the CNN detailed above. We quantify each technique's performance by calculating the average distance between the predicted exit and the true exit. Additionally, we run simulations in ROS stage of a TurtleBot[1] exploring buildings, searching for exits based on estimations from the techniques described in Section IV. We assume the robot is able to localize itself sufficiently well within the building to determine the relative location of the projected exit. All predictions are performed on the test set, which has no overlap of buildings or floors with the training or validation set.

TABLE I: Average distance between estimated exits and true exits in 2575 floorplans

	Random	HOG+SVM	CNN
Distance (m)	17.95	12.95	10.15

When comparing estimated exit location to true exit location, we found the average distance error in the estimated exit location for the test set was 12.95 m when using HOG+SVM and 10.15 m when using the CNN method (see Table 1). This is compared to an average distance of 17.95 m if a random location inside the bounds of the building is selected as an exit location. This shows that HOG+SVM is able to learn some information about exit locations from local features, but the CNN is able to do a better job of identifying exit locations in buildings. Because the CNN is able to construct features that incorporate the whole building, structural information can be encoded and used to help predict exit location.

To understand the impact exit prediction accuracy has on search performance, we perform a series of simulations of a TurtleBot exploring office buildings using ROS [19]. Each simulation starts by placing a TurtleBot in the center of a random room inside the building. The TurtleBot performs frontier exploration with a 4 m range laser scanner. Two methods in which the TurtleBot picks waypoints are explored. As a baseline, generic frontier exploration is used, picking the frontiers that are nearest to the TurtleBot as the next waypoint for exploration. The other method selects the frontier that is closest to the estimated exit. This guides the search towards the estimated exit, but guarantees search completeness.

[1]http://wiki.ros.org/Robots/TurtleBot

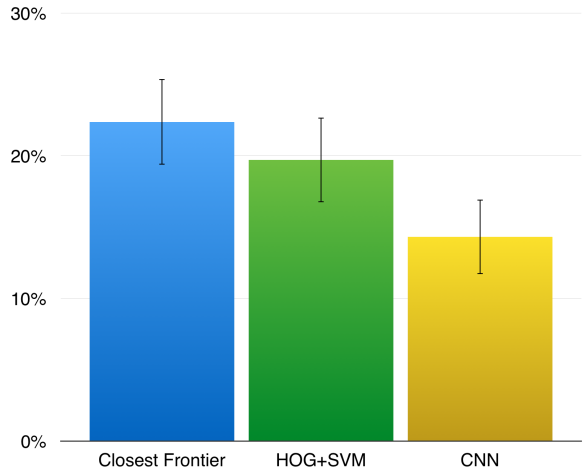


Fig. 7: The mean area of the building explored before the robot discovered an exit. Error bars show one standard error of the mean (SEM) above and below the mean. Averaged over 20 simulated building searches.

Figure 7 shows the results of 20 simulated building searches. As a baseline, standard frontier exploration explores, on average, 22.3% of the building until an exit is found. This has a large variance due to the differences in building size and shape and the random starting locations. Sometimes the robot starts close to the exit location and requires little exploration and sometimes it is far from the exit and needs to explore a large percentage of the building. HOG+SVM performs slightly better than the baseline. This matches its exit location prediction accuracy, performing slightly better than a random search. The Deep Network performs best, needing to only explore 14.3% of the building on average to find the exit. This results in a 35.8% reduction in building exploration, which can be directly translated into a reduction of time and energy the robot needs to use to find the exit.

VII. Conclusion and Future Direction

The results presented in this paper demonstrate the promise of using deep neural nets to perform model-free predictions for robotic exploration. This work shows that a deep neural network can provide a reliable method for increasing the efficiency of current search methods by providing improved predictions about the goal locations and environmental features. Furthermore, little effort is needed to adapt the method to other domains as it directly uses low level sensor and feature inputs without the need for external filtering or processing.

Further work is needed to compare these methods to other model-free prediction algorithms such as Gaussian Processes. While deep neural nets have been used successfully in many areas, the method presented here should be applied to other domains to characterize its performance characteristics and robustness to environmental variation. This work also

suggests several other broad avenues for further studies in the areas of sensor processing and motion planning. Investigating whether deep neural nets are able to plan trajectories in addition to predicting the goal location is of particular interest. Lastly, a direct extension to this work is using partial floor plans, similar to those constructed via SLAM, as inputs into the predictive neural network.

The results presented in this paper show that using deep neural nets to inform search of unknown environments has the potential to significantly improve the efficiency and autonomy of robots performing a wide variety of mapping, monitoring, and search missions.

ACKNOWLEDGMENT

Special thanks to William Curran and Kory Kraft for assistance with Robot Operating System.

REFERENCES

[1] E. Kussul and T. Baidyk, "Improved method of handwritten digit recognition tested on MNIST database," *Image and Vision Computing*, vol. 22, no. 12, pp. 971–981, 2004.

[2] K. J. Lang, A. H. Waibel, and G. E. Hinton, "A time-delay neural network architecture for isolated word recognition," *Neural networks*, vol. 3, no. 1, pp. 23–43, 1990.

[3] S. Lawrence, C. L. Giles, A. C. Tsoi, and A. D. Back, "Face recognition: A convolutional neural-network approach," *IEEE Transactions on Neural Networks*, vol. 8, no. 1, pp. 98–113, 1997.

[4] D. Ciresan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," in *Proc. IEEE International Conference on Computer Vision and Pattern Recognition*, 2012, pp. 3642–3649.

[5] C. J. Maddison, A. Huang, I. Sutskever, and D. Silver, "Move evaluation in Go using deep convolutional neural networks," *ArXiv preprint arXiv:1412.6564*, 2014.

[6] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *Journal of Machine Learning Research*, vol. 17, no. 39, pp. 1–40, 2016.

[7] I. Lenz, R. Knepper, and A. Saxena, "DeepMPC: Learning deep latent features for model predictive control," in *Proc. Robotics: Science and Systems Conference*, Rome, Italy, 2015.

[8] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, *et al.*, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[9] X. Guo, S. Singh, H. Lee, R. L. Lewis, and X. Wang, "Deep learning for real-time Atari game play using offline Monte-Carlo tree search planning," in *Proc. Advances in Neural Information Processing Systems*, 2014, pp. 3338–3346.

[10] B. Yamauchi, "Frontier-based exploration using multiple robots," in *Proc. International Conference on Autonomous Agents*, ACM, 1998, pp. 47–53.

[11] K. M. Wurm, C. Stachniss, and W. Burgard, "Coordinated multi-robot exploration using a segmentation of the environment," in *Proc. IEEE International Conference on Intelligent Robots and Systems*, 2008, pp. 1160–1165.

[12] M. Berhault, H. Huang, P. Keskinocak, S. Koenig, W. Elmaghraby, P. Griffin, and A. Kleywegt, "Robot exploration with combinatorial auctions," in *Proc. IEEE International Conference on Intelligent Robots and Systems*, vol. 2, 2003, pp. 1957–1962.

[13] W. Burgard, M. Moors, D. Fox, R. Simmons, and S. Thrun, "Collaborative multi-robot exploration," in *Proc. IEEE International Conference on Robotics and Automation*, vol. 1, 2000, pp. 476–481.

[14] S. Macé, H. Locteau, E. Valveny, and S. Tabbone, "A system to detect rooms in architectural floor plan images," in *Proc. IAPR International Workshop on Document Analysis Systems*, ACM, 2010, pp. 167–174.

[15] A. Aydemir, P. Jensfelt, and J. Folkesson, "What can we learn from 38,000 rooms? reasoning about unexplored space in indoor environments," in *Proc. IEEE International Conference on Intelligent Robots and Systems*, 2012, pp. 4675–4682.

[16] D. Perea Ström, F. Nenci, and C. Stachniss, "Predictive exploration considering previously mapped environments," in *Proc. IEEE International Conference on Robotics and Automation*, 2015, pp. 2761–2766.

[17] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. IEEE International Conference on Computer Vision and Pattern Recognition*, vol. 1, 2005, pp. 886–893.

[18] P. Dollár, *Piotr's Computer Vision Matlab Toolbox (PMT)*, `http://vision.ucsd.edu/~pdollar/toolbox/doc/index.html`.

[19] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: An open-source robot operating system," in *Proc. ICRA Workshop on Open Source Software*, Kobe, Japan, 2009.