# Learning Autonomous Exploration and Mapping with Semantic Vision

Xiangyang Zhi
ShanghaiTech University
393 Middle Huaxia Road
Shanghai, China
zhixy@shanghaitech.edu.cn

Xuming He
ShanghaiTech University
393 Middle Huaxia Road
Shanghai, China
hexm@shanghaitech.edu.cn

Sören Schwertfeger
ShanghaiTech University
393 Middle Huaxia Road
Shanghai, China
soerensch@shanghaitech.edu.cn

## ABSTRACT

We address the problem of autonomous exploration and mapping for a mobile robot using visual inputs. Exploration and mapping is a well-known and key problem in robotics, the goal of which is to enable a robot to explore a new environment autonomously and create a map for future usage. Different to classical methods, we propose a learning-based approach this work based on semantic interpretation of visual scenes. Our method is based on a deep network consisting of three modules: semantic segmentation network, mapping using camera geometry and exploration action network. All modules are differentiable, so the whole pipeline is trained end-to-end based on actor-critic framework. Our network makes action decision step by step and generates the free space map simultaneously. To our best knowledge, this is the first algorithm that formulate exploration and mapping into learning framework. We validate our approach in simulated real world environments and demonstrate performance gains over competitive baseline approaches.

## CCS Concepts

**•Computing methodologies → Vision for robotics; •Computing methodologies → Deep belief networks**

## Keywords

Robotic Exploration; Computer Vision; Deep Learning; Reinforcement Learning

## 1. INTRODUCTION

For a human who comes to a new environment, one important thing is to look around, acquainting himself with the environment and generating a “map" in his mind. Similarly, for a mobile robot, the process of exploration and mapping, known as autonomous exploration, has been a key problem in robotics for the past several decades. In general, the performance of autonomous exploration is measured by the information gain rate, say the size of the explored area, in a unit time. So as to maximize the efficiency of exploration, the robot should try its best to move towards the unexplored space.


*IVSP 2019,* February 25–28, 2019, Shanghai, China

ACM. ISBN 978-1-4503-6175-0/19/02…$15.00
DOI: https://doi.org/10.1145/3317640.3317652

Most of previous work have been focused on the problem setting for robots equipped with active range sensors, such as frontier-based technique [34]. Despite their success, relying on active sensors leads to several limitations, such as high power consumption and restrictive condition for deployment. By contrast, vision-based navigation using cameras has attracted increasing attention due to their low resource footprint and capacity of capturing richer information on the environment [23]. Early attempts in this direction employed stereo cameras to obtain 3D information [30, 4] and to guide the navigation. However, stereo camera systems based on multi-view geometry are less reliable in real-world environments, especially for textureless or reflective surfaces. Moreover, such geometry-based approaches fail to exploit the semantic cues of the visual scenes, such as object category and spatial layout, which are critical for generating efficient exploration strategies. For instance, if the robot recognizes its current location as a corridor, it may take a specific route to navigate through it.

In this work, we propose to utilize semantic interpretation of the visual scenes along with the camera geometry in tackling the problem of autonomous exploration and mapping. In particular, we aim to exploit the dense semantic segmentation of input frames to build a flexible and confidence-aware map of the traversable space, which in turn yields a rich representation for learning an efficient exploration strategy. Our goal is to maximize the efficiency of autonomous mapping by learning the entire pipeline from an annotated visual environment.

To this end, we develop a deep neural network that consists of three main modules for vision-based exploration and mapping: visual sensing, map construction and exploring action prediction. Specifically, for each input frame, our first module generates a dense pixel-wise semantic segmentation based on a Fully Convolutional Network with dilation convolution [3]. The confidence map of the ground class is sent to the second module, which builds a two-level map representation, ego-centric and bird-view, based on camera pose and scene geometry. The ego-centric map describes the local layout of traversable space while the bird-view representation encodes the global map of the environment. The third module is a convolutional neural network that takes the map representation as input to predict the next movement in robot navigation. A key characteristic of our exploration and mapping network is all three modules are differentiable w.r.t their inputs and parameters, which enable us to train the entire network in an end-to-end fashion.

We formulate the task of autonomous navigation as a sequential decision process, in which we aim to train the deep neural network to generate an efficient exploration policy. To achieve this, we define a reward function that computes the area that was mapped in a fixed number of movements of the mobile robot, and adopt an actor-critic based policy gradient method to maximize the expected

accumulated reward plus a regularization term from the segmentation loss. We evaluate our framework on a large-scale indoor environment [2] and our semantic-driven deep network outperforms other vision-based baselines by a large margin.
The main contributions of our work are three-fold:

- We propose a novel scene representation for vision-based autonomous exploration and mapping, which integrates the semantic image labeling with geometry-based scene models.
- We develop a multi-scale, differentiable map representation that enables us to embed the map construction into a deep neural network.
- We formulate autonomous exploration as a reinforcement learning problem and use a policy gradient method to learn the entire system in an end-to-end manner, which produces a highly efficient exploration policy.

## 2. RELATED WORK

There is a large body of literature on autonomous exploration and mapping [32, 13, 27]. Typically, the existing pipelines consist of three stages: identifying the unexplored regions, determining which region to go next and navigating to that region. During the exploration, these three stages are executed repeatedly until the map is built. As a partial map is generated during exploration, there are several map-based navigation methods available for the third stage. Hence most prior work focus on the first two stages.
**Range-sensor based exploration:** For identifying the unexplored regions, one of the most successful methods is based on frontiers, which are the regions on the boundary between free space and unexplored space. In [34], the author assumed that the robot is equipped with range sensors, for instance, sonar or laser range finder, with which the frontiers can be easily identified. Several recent methods employed hand-craft features [33, 22, 21], which also need range sensors. However, if we rely on cameras only rather than range sensors, the exploration becomes much more difficult due to the loss of 3D information.
**Vision based exploration:** Sim and Little [30] adopted a stereo camera to construct an occupancy map and employ a frontier exploration technique. Santosh *et al*. [28] segmented floors from the RGB images: first over-segment the image into lots of super-pixels, then, by assuming that a small region directly in front of the robot is free space, find similar super-pixels and label them as floor. With the segmented floor, the robot can move without collision. However, one obvious disadvantage is that the floor segmentation method is not robust, especially when the color of the floor varies widely. Fraundorfer *et al*. [4] conducted visual SLAM with a stereo camera mounted on a MAV, generating a 3D occupancy map, and then by reducing the 3D occupancy map to 2D occupancy map, used frontier-based exploration method proposed in [34].
**Exploration strategy:** Given the interesting regions of the environment, the next step for a mobile robot is to select one region to go. A naive but efficient algorithm is to choose the nearest region, which is adopted widely in literature [34, 33, 21, 28, 4]. There are also several approaches [24, 5] that consider the utility, i.e., information gain when the robot reach one region of interest, in determining the exploration strategy. However, these existing approaches divide the exploration into several steps, while in this paper we propose an end-to-end exploration pipeline.
**Reinforcement learning:** Reinforcement Learning (RL) has been successfully applied to a variety of robot applications. [16] proposed a RL-based method to make a robot push boxes autonomously. [9] utilized RL to control a helicopter. In [17], Michels *et al*. used RL to detect obstacles with a monocular camera. RL has also been successful in locomotion and motor control for various type of robots [11, 25].
Recently, due to the unprecedented success of deep learning, deep RL has shown remarkable development in tackling complicated problems. [20] proposed Deep q-learning (DQN) which achieves higher scores than human players in ATARI games. Furthermore, [29] proposed a deep RL algorithm with Monte-Carlo tree search (MCTS), which defeats the world champion in the game of Go. To improve learning efficiency, four asynchronous gradient descent methods for deep RL are discussed in [19], including the Asynchronous Advantage Actor-Critic (A3C), which is widely adopted in solving various RL problems. Moreover, Deep RL has also been used in robotics with end-to-end training, such as robotic manipulation [14, 15] and robot navigation [35, 18].
**Relationship to contemporary work:** To our best knowledge, there is little work in learning-based exploration and mapping so far. On the other hand, Deep RL has been used in robot navigation, which is a related problem as it also needs path planning [6, 7]. Among them, perhaps the most related work is [6], in which a mapper also predicts the free space of the environment and accumulates the free space to a map representation. However, our work differs in the following two aspects: first we build an explicit map on traversable space using the semantic segmentation to predict free space and then transforming the free space to bird-view grid map by perspective projection; second, instead of using imitation learning [26], which is difficult to apply to our problem, we adopt an efficient RL approach, A3C, to learn the full pipeline.

## 3. PROBLEM SETUP

As it is very difficult to train on a real robot, we adopt a simulation-based strategy in this work. A number of indoor stimulation environments have been developed, such as SUNCG [31] and AI2-THOR [12]. However, the scenarios of these datasets are synthetic, thus showing significant differences to reality. Inspired by the work of Gupta *et al*. [6], we run our algorithm on the Stanford large-scale 3D Indoor Spaces (S3DIS) dataset [2]. In contrast to AI2-THOR and SUNCG, this dataset is collected by scanning the real environment with a Matterport scanner[1]. It consists of 6 large-scale indoor areas that originate from 3 different buildings of mainly educational and office use. For each area, there is a 3D reconstruction with textured mesh, as well as the corresponding 3D semantic mesh.
We simulate a robot in that environment. Identical to the work of Gupta *et al*., we simplify the robot as a cylinder, equipped with a RGB camera mounted on top of the robot rigidly. In addition, to focus on the high-level action decision, we also assume that the robot has accurate motion and perfect odometry, for example, by employing visual odometry. We assume that the robot is omni-directional using Mecanum wheels or omni wheels, so we specify six actions for the robot: move forward $x$ cm, move backward $x$ cm, move left $x$ cm, move right $x$ cm, turn left by angle $\theta$ and turn right by angle $\theta$. In this paper, $x$ and $\theta$ are set to 40 and $\pi/2$, respectively. With the above setting, we can actually discretize the traversable space into a directed graph of which the nodes denote the traversable spots and directed edges indicate the robot can move from one node to another by one of the above six actions.

## 4. METHODOLOGY

To learn an efficient autonomous exploration and mapping strategy, we develop a deep neural network that consists of three modules according to their functions, and can be trained end-to-end. Figure 1

[1]https://matterport.com/

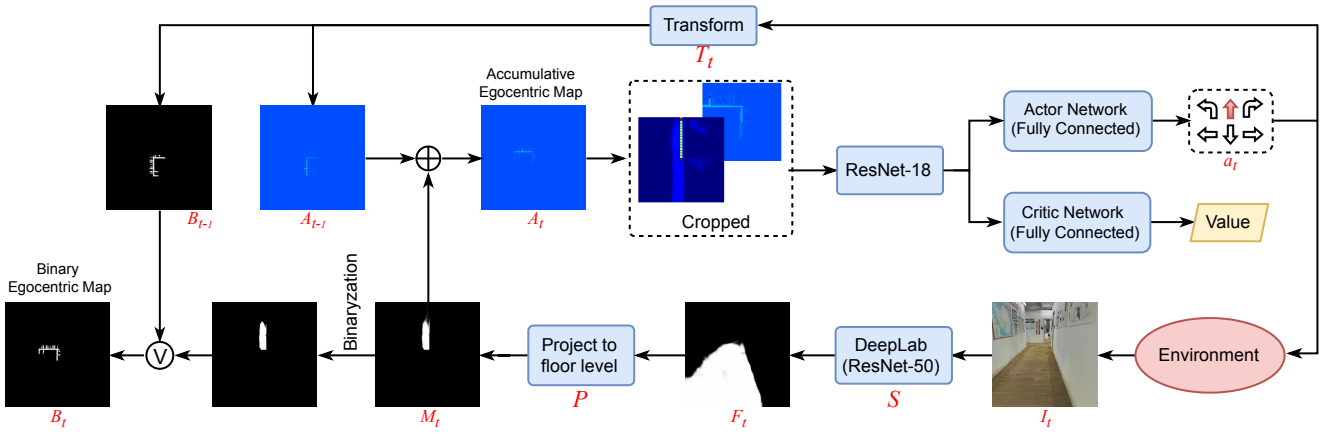


**Figure 1: Diagram of proposed exploration and mapping method.**

illustrates the overview of our network architecture. The first module of our pipeline is a semantic segmentation network, which is designed to obtain semantic features from the RGB image, especially the free space. The second module is mapping of free space, which extends a bird-view map step by step. And the third module is an action decision network determining the next movement of the robot. After executing the action, the robot moves to a new position, starting the whole procedure again. The entire network is learned based on the actor-critic framework. We now introduce the three modules one by one.

## 4.1 Semantic Segmentation

As the robot needs to build up a map of the environment, it is necessary to extract free space from the image captured by the camera. To achieve this, we first employ a semantic segmentation network to extract pixel-level semantic information from the image. In addition to free space (represented by the floor class), we also generate segmentation of other object classes as they provide informative context cues. Specifically, we group the rest of object labels into three super-classes and segment each image into four semantic categories. As we have the ground-truth semantic labeling in the training dataset, we compute the loss of the semantic segmentation network and integrate it into the exploration reward.

Mathematically, at time step $t$, the robot takes a photo of the environment, say $I_t$. After the semantic segmentation function $S$, we get free space image $F_t$, *i.e.* $F_t = S(I_t)$.

## 4.2 Mapping

For an indoor scenario, 2D grid map is usually enough for most applications. In this paper, the map generated is also in 2D. Although the free space has been segmented in the image, we cannot map it directly, because the depth of the pixels are unknown. However, as the robot operates indoor, it is reasonable to assume that the ground level is planar. With this assumption, we can determine the floor corresponding to the free spaces in the images.

We assume the camera is well calibrated, *i.e.* the intrinsic and extrinsic parameters, here the transformation between camera frame and robot frame, are known. Moreover, as we also assume the odometry is perfect, which means that the transform between world frame and camera frame is known. As shown in Figure 2, since the height and orientation of the camera is known and fixed, we can determine the corresponding free cells of the 2D grid map. We denote this projection function as $P$ and the bird-view map as $M_t$ at time

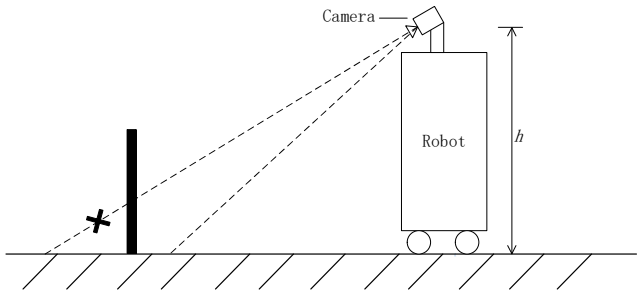


**Figure 2: Sketch of free space projection.**

step $t$, *i.e.* $M_t = P(F_t)$. It is noteworthy that this projection only fits the free space, *i.e.* floor, because any object which is higher than floor will be projected to the wrong location in this way, *e.g.* the black obstacle shown in Figure 2. This is the reason why we only project free space rather than all object classes.

We maintain two egocentric maps in this module. One is binary, *i.e.* the value of a cell of the map is 1 if it's free or 0 if not. This binary egocentric map (BEM) is the explicit output of our method. On the other hand, we also have an accumulative egocentric map (AEM) whose cells are accumulating when a new projection hits them. The AEM is generated as the input for the action decision network. We do not normalize the AEM, as it implicitly encodes the visited times of areas, which could be very useful for the robot to decide its next actions. In addition, we note that the robot's trajectory also includes important information: the places that have been visited. Hence, when a robot arrives at a new position, we increase the cells of the AEM which the robot stands on by a large value (20.0 in this paper). Besides indicating the trajectory of the robot, this operation also enhances the map of free space, since obviously the places a robot passed though are traversable. On the other hand, a place that the robot tried to reach but failed to do so is occupied, and thus we subtract the corresponding cells of the AEM by a large value (here it is also 20.0). This operation helps avoiding collision to the obstacles at the beginning of the training.

We denote AEM and BEM by $A_t$ and $B_t$ at time step $t$. At time step $t$, if the robot takes action $a_t$ according to $A_{t-1}$, AEM and BEM should be updated by a transform $T_t$, which can be calculated from

$a_t$. Then $A_t$ can be updated as

$$A_t = T_t(A_{t-1}) + M_t. \quad (1)$$

As BEM is a binary map, we apply a threshold to binarize $M_t$. To reduce false positives, which have large impact on the map quality due to incorrect projections, we set the threshold to 0.8. So $B_t$ is updated as

$$B_t = T(B_{t-1}) \vee (M_t > 0.8) \quad (2)$$

The size of AEM and BEM, theoretically, should be as large as possible to ensure the robot stays in the map and to avoid information loss. However, due to the constraints on computation resources in practice and the physical size of the area to be explored, we set a maximum size of AEM and BEM according to those factors.

## 4.3 Action Decision

To generate the full map, a critical step is to make action decision for navigation. A good navigation strategy should always guide the robot to unexplored areas and avoid collisions on the way. To achieve this, we adopt a neural network with fully connected layers to predict the next movement based on the map inputs. As this is a typical sequence decision problem, we utilize an actor-critic framework to learn the action decision network and to fine-tune the rest of the pipeline. We will focus on two key elements of the reinforcement learning, the reward and state design, in this section.

**Reward Design.** The goal of the exploration is to explore an unknown environment as quickly as possible. As such, we design a reward function with the following three components. The first component of our reward is the area of free space that the robot identifies given the time spent. This reward allows us to guide RL training at each step as it can be computed densely during the exploration. Given a fixed time budget, it also enables us to parallelize the training of several robots in a distributed manner due to the fixed length of each episode. On the 2D grid map, the reward can be easily computed as the additional number of free grids of the map. In addition, we punish the false positive of free space segmentation by giving a negative reward in that case. This negative reward is quantified by the number of pixels of the input image which are labeled as free incorrectly. Furthermore, to make the robot learn to avoid collisions, which obviously have a negative impact for exploration, a small negative reward will be given if a collision happens.

**State Design.** As mentioned before, AEM is the input of the action decision module. However, AEM is very large, most information is useless for the current action decision, so as to utilize the AEM efficiently, we crop and scale the map around the center. The cropped AEM with the steps of 1 and 3, respectively, have the same size: $257 \times 257$. Then we concatenate these two AEMs, so finally the size is $257 \times 257 \times 2$. The AEM is simple but very powerful, because it not only functions as a free space map, but also contains information about the trajectory of the robot and occupied space. Moreover, we tried adding image features extracted by the segmentation network along with features extracted from AEM, but did not find any performance improvement. This indicates that the AEM is sufficient as a state in this task.

# 5. EXPERIMENTAL RESULTS

**Implementation Details:** The semantic segmentation network is based on DeepLab [3] using the ResNet-50 [8]. We do not employ any label refinement module such as Conditional Random Field. The action decision network is a ResNet-18.

The RGB image size is $257 \times 257 \times 3$. Unlike many deep RL approaches, we only utilize the current frame as the input rather than stack several historical frames. One grid of AEM or BEM represents a $5cm \times 5cm$ square on the ground floor.

Our model is trained asynchronously with 4 parallel GPU workers using TensorFlow [1]. We use A3C [19] as the training protocol of the actor-critic network, and train the full model in an end-to-end fashion. We use ADAM [10] to optimize our loss function with an initial learning rate of 2e-5. The loss function comprises the policy loss of the actor and value loss of the critic in the actor-critic network, regularized by the cross entropy loss of the semantic segmentation. We balance them with scale factors. The episode length is fixed to 200 in the experiments, and the number of local steps is 20, *i.e.* a single reward directly affects the values of 20 preceding state action pairs. We set the discount factor of rewards to 0.99.

There are 6 areas in the S3DIS dataset, but the floors of *area2* is not in one level, so we exclude it and conduct experiments on the other 5 areas. *area1*, *area3*, *area4* and *area6* are used as training dataset, and *area5* is used as evaluation dataset. During training, we run several simulated robots on one GPU worker in parallel, and distribute them evenly over the four areas. We believe this kind of training reduces the bias of a batch of data which may impede gradient descend.

## 5.1 Comparison to Baseline

To our best knowledge, there are no similar learning-based exploration methods like ours, so as to evaluate our approach, we implement two kinds of baselines using RGB image and depth image as input, respectively. Both of two baselines simply use the current frame as the state of the actor-critic network, inside which is ResNet-50 followed by LSTM. With regard to free space segmentation, it is very easy for the depth image, but difficult for RGB image. So as to make the RGB-based baseline more competitive, we utilize the ground truth as free space segmentation result for it. Similarly, a third baseline method is also using the ground truth segmentation to crate the map but a random action policy. We called the three baselines “RGB+LSTM", “depth+LSTM" and “Random". Also, the range of available commercial depth cameras is 8m at most (Kinect 2), so as to show the performance of a real depth camera, our fourth baseline is “depth+LSTM" where the range of depth camera is limited to 8m, named “depth+LSTM (8m)".

Obviously, it will be more difficult for the robot to start in a room than in a corridor, because there can be much more stuff inside a room impeding the robot's movement. To compare the performance of different algorithms, we separate the evaluation tasks by the place where the robot starts, which consists of 3037 different starting points in corridors and 2073 in rooms. Each method runs for one episode (200 steps) by starting at all points, and the final score is the mean of that. Regarding evaluation metrics, the score are the free areas that are labeled correctly subtracting the occupancy areas that are labeled as free falsely. The experiment results are listed in Table 1. The performance is only worse than that of “depth+LSTM", which actually cannot be achieved with an available depth sensor. In comparison to “RGB+LSTM", which uses the same sensor, our method improves the area of explored free space by 49.7% and 51.3% when starting in rooms and corridors, respectively.

Intuitively, when the robot starts in rooms, it should get out to explore more space, so finding the way out of rooms is quite an important ability the robot need to have. In Figure 3, we show three representative maps and the trajectories generated by our method. The figures illustrate that our algorithm empowers the robot to find the way even in deep positions of rooms. Of course, our method is

**Table 1: Average Performance of Different Methods**

| Method | Area of Explored Free Space ($m^2$) | |
|---|---|---|
| | Start in rooms | Start in corridors |
| Random Action | 57.34 | 102.80 |
| RGB+LSTM | 102.55 | 118.51 |
| depth+LSTM (8m) | 137.40 | 161.17 |
| Ours | 153.49 | 179.27 |
| depth+LSTM | 178.85 | 197.85 |

**Table 2: Performance of Ablated Versions of our Method**

| Method | Area of Explored Free Space ($m^2$) | |
|---|---|---|
| | Start in rooms | Start in corridors |
| Projecting Features | 18.33 | 55.83 |
| CNN Projection | 84.40 | 102.65 |

not perfect, there are some bad cases in which the robot gets stuck in rooms, as shown in Figure 4. To analysis the reason resulting in such bad cases, the perhaps one is that the rooms are very messy, occupying by various stuff which makes it more difficult to understand the current scenes and move.

## 5.2 Ablation Study

We also studied ablated versions of our proposed method, the performance of whom is listed in Table 2.

*Projecting Features.* There is concern that whether encoding more semantic features can improve the performance. To study that, we replace free space with semantic features, as illustrated in Figure 5.

*CNN Projection.* As described in Section 4.2, our method utilizes the prior knowledge of the robot setup, which makes map projection quite efficient. We would like to know that whether this projection procedure can be learned using CNN, which will make the calibration of the camera unnecessary.

"Projecting Features" is hard to train, even diverged at the end of training. We try to, but cannot get comparable performance. By comparison, "CNN Projection" performs a little better than "Random Action", but still falls behind the proposed method large.

## 5.3 Noisy Localization Experiments

We assume perfect odometry in our work, but for a real robot operating in real scenarios, perfect odometry is quite hard to obtain. So as to validate the robustness to localization noise, we add random noise to the robot's pose and find out the effect to our method. Obviously, the direct effect of localization noise to our method is the egocentric map. If localization is noisy, the map will be bend, affecting the exploration actions.

We assume that the noise is with normal distribution, *i.e.* its probability density is

$$f(|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

where $\mu$ is the mean and $\sigma$ is the standard deviation.

We do not re-train the network with noisy localization, but just add noise to test experiments. Moreover, the noise is added every step, so it will accumulate gradually, and may become quite large at the end of an episode.

Translation noise and orientation noise are both tested, and also different noise levels. We reduce the number of the different starting points, due to time constraints. Thus the resulting performance is slightly different to Table 1. In the translation noise experiments, the number is 1270, including points in rooms and corridors. In the orientation noise experiments, the number is 300, also including points in rooms and corridors. The starting points are selected randomly and not changed for different noise levels experiments.

The experiment results are listed in Table 3. Inevitably, the performance drops after adding noise, however, the decrease is within an acceptable range, and our method with localization noise still outperforms the other baselines. In comparison, the impact to episodes

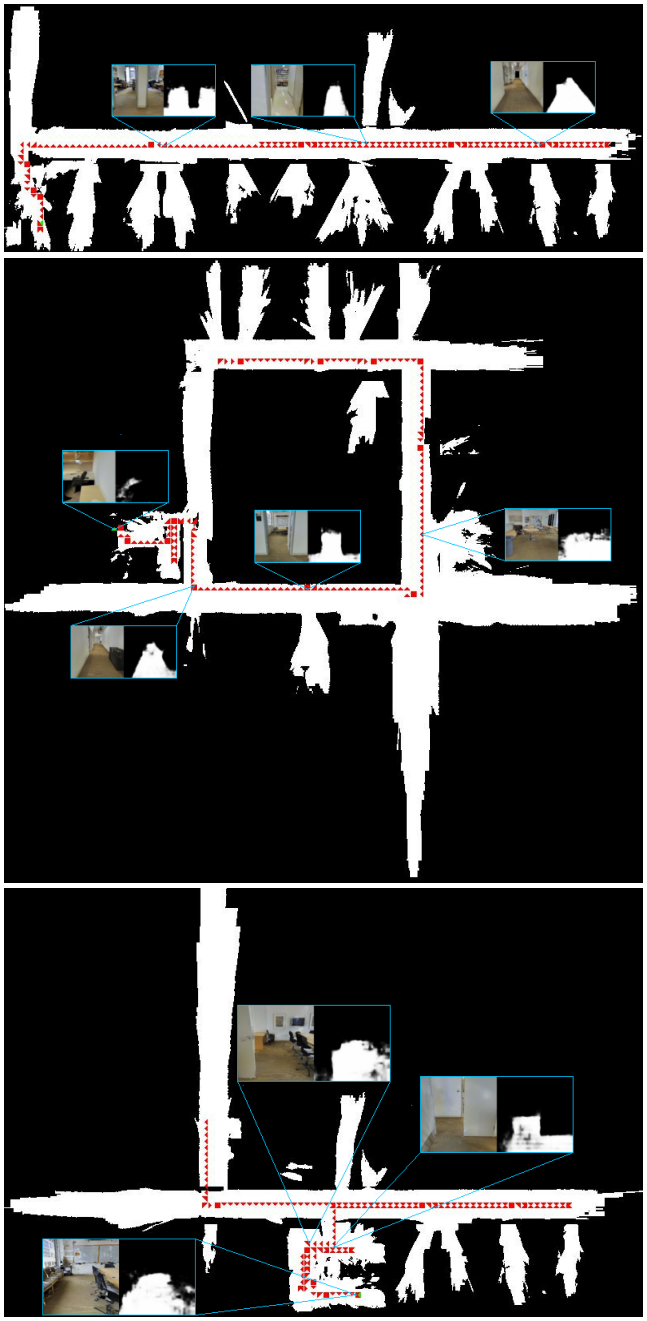

**Figure 3: Three representative maps generated by our method. White regions show free space, and red triangles form the trajectories of the robot. Each triangles denotes one position and orientation of the robot, and the longest edge opens towards the orientation of the robot. The green triangles are starting points. We also embed some obtained first-person images and corresponding free space images segmented by our method.**

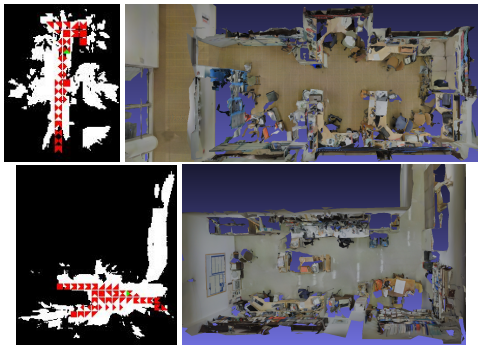

**Figure 4: Two cases in which our algorithm performs bad because it doesn't exit the room. Left figures are the maps and trajectories whose format is described in Figure 3, and the figures on the right are the bird-view of corresponding scenarios of the maps on the left.**

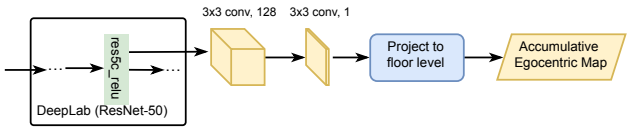


**Figure 5: The diagram of semantic features projection.**

starting in rooms is larger than that starting in corridors. This situation is predictable. When the robot is at a room, it needs a more precise map to help itself avoid collisions and to find the way out of the room. However, in corridors, the free space is wider, and the robot usually follows corridors easily.

These experiments indicate that our method is robust to reasonable localization noise. This result is not unexpected. The free space segmentation is not perfect, so there is always noise in the egocentric map, our network owns the power to extract useful information from noisy map.

**Table 3: Average Performance of Proposed Method under Different Noise Level. In left column, TN is abbreviation of 'Translation Noise', ON is abbreviation of 'Orientation Noise'. Then is mean ($\mu$) of noise, followed by standard deviation($\sigma$).**

| Noise Level | Area of Explored Free Space ($m^2$) | |
|---|---|---|
| | Start in rooms | Start in corridors |
| TN: 0cm, 0cm | 148.88 | 181.90 |
| TN: 0cm, 3cm | 143.97 | 182.44 |
| TN: 0cm, 5cm | 143.95 | 180.34 |
| TN: 2cm, 3cm | 142.96 | 180.13 |
| TN: 2cm, 5cm | 140.55 | 175.96 |
| | | |
| ON: 0.0°, 0.0° | 146.04 | 176.23 |
| ON: 0.0°, 0.5° | 137.36 | 172.46 |
| ON: 0.2°, 0.5° | 131.32 | 171.20 |
| ON: 0.0°, 0.5° TN: 0cm, 2cm | 133.08 | 172.20 |
| ON: 0.2°, 0.5° TN: 0cm, 2cm | 128.97 | 170.25 |

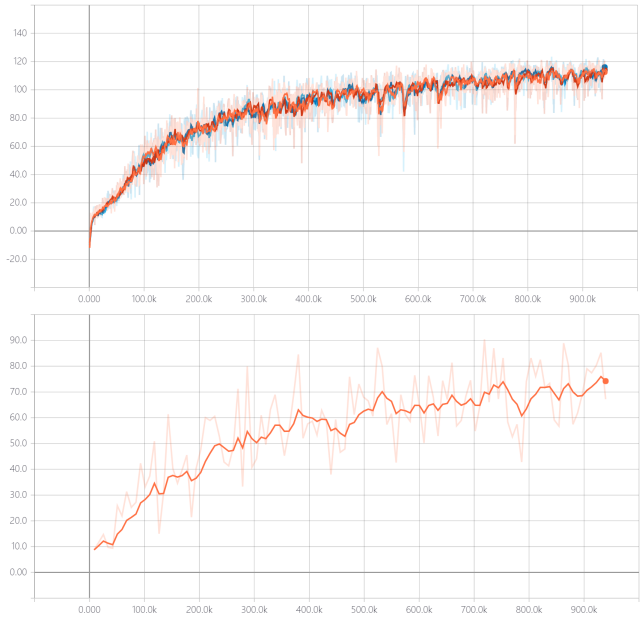

**Figure 6: The rewards Curve of training and test. The top figure is for training, which includes four curves, corresponding to four GPU workers. The bottom figure is for test. The x-axis is the overall steps of training and the y-axis is the reward. The transparent curves illustrate original data, while the bold curves are the smoothed version with the factor 0.85.**

## 5.4 Rewards Curve

We show the rewards curve of training, and testing results on validation dataset (area5) while training, see Figure 6. Please don't confuse the rewards and area of explored free space in Table 3, they are different. As clearly showed, we stop training a little bit early, the rewards are still growing up, so we may obtain slight better performance if we trained longer time.

# 6. CONCLUSION

This work presents an autonomous exploration and mapping algorithm for indoor scenes using visual inputs only. Our method formulates autonomous exploration and mapping as learning problem. Our network can be trained end-to-end, and we utilize deep reinforcement learning to train an exploration policy with additional guidance from semantic segmentation. Our network can generate exploration actions and the free space map, thus achieve the two core functions of exploration and mapping simultaneously. We conduct our experiments on real world datasets obtained by Matterport, which is very challenging because of various objects crowding in the scenarios. We even perform training and testing on different areas, and experiments show that our method outperforms vision based baseline approaches significantly and is also competitive to depth based methods, which obviously have big advantage with 3D information.

One main limitation of our method is that we assume perfect odometry, which is usually hard to achieve with a real robot in real world. But our explicit free space map makes it easy to cooperate with analytic mapping methods, *e.g.* visual SLAM, which may help solve the problem partially. So we would like to leave it as future work to deploy our algorithm on a real robot.

# 7. REFERENCES


[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. Tensorflow: a system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283, 2016.

[2] I. Armeni, O. Sener, A. R. Zamir, H. Jiang, I. Brilakis, M. Fischer, and S. Savarese. 3d semantic parsing of large-scale indoor spaces. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, 2016.

[3] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2018.

[4] F. Fraundorfer, L. Heng, D. Honegger, G. H. Lee, L. Meier, P. Tanskanen, and M. Pollefeys. Vision-based autonomous mapping and exploration using a quadrotor MAV. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4557–4564, Oct. 2012.

[5] H. H. González-Banos and J.-C. Latombe. Navigation strategies for exploring indoor environments. *The International Journal of Robotics Research*, 21(10-11):829–848, 2002.

[6] S. Gupta, J. Davidson, S. Levine, R. Sukthankar, and J. Malik. Cognitive Mapping and Planning for Visual Navigation. *arXiv:1702.03920 [cs]*, Feb. 2017. arXiv: 1702.03920.

[7] S. Gupta, D. Fouhey, S. Levine, and J. Malik. Unifying Map and Landmark Based Representations for Visual Navigation. Dec. 2017.

[8] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[9] H. J. Kim, M. I. Jordan, S. Sastry, and A. Y. Ng. Autonomous Helicopter Flight via Reinforcement Learning. In S. Thrun, L. K. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*, pages 799–806. MIT Press, 2004.

[10] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. Dec. 2014.

[11] N. Kohl and P. Stone. Policy gradient reinforcement learning for fast quadrupedal locomotion. In *2004 IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04*, volume 3, pages 2619–2624 Vol.3, Apr. 2004.

[12] E. Kolve, R. Mottaghi, D. Gordon, Y. Zhu, A. Gupta, and A. Farhadi. AI2-THOR: An Interactive 3D Environment for Visual AI. *arXiv*, 2017.

[13] A. K. Krishnan and K. M. Krishna. A visual exploration algorithm using semantic cues that constructs image based hybrid maps. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1316–1321, Oct. 2010.

[14] S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

[15] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International Journal of Robotics Research*, 37(4-5):421–436, 2018.

[16] S. Mahadevan and J. Connell. Automatic programming of behavior-based robots using reinforcement learning. *Artificial Intelligence*, 55(2):311–365, June 1992.

[17] J. Michels, A. Saxena, and A. Y. Ng. High Speed Obstacle Avoidance Using Monocular Vision and Reinforcement Learning. In *Proceedings of the 22Nd International Conference on Machine Learning*, ICML '05, pages 593–600, New York, NY, USA, 2005. ACM.

[18] P. Mirowski, R. Pascanu, F. Viola, H. Soyer, A. J. Ballard, A. Banino, M. Denil, R. Goroshin, L. Sifre, K. Kavukcuoglu, D. Kumaran, and R. Hadsell. Learning to Navigate in Complex Environments. *arXiv:1611.03673 [cs]*, Nov. 2016. arXiv: 1611.03673.

[19] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. pages 1928–1937, 2016.

[20] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, Feb. 2015.

[21] L. Murphy and P. Newman. Using incomplete online metric maps for topological exploration with the gap navigation tree. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 2792–2797. IEEE, 2008.

[22] P. Newman, M. Bosse, and J. Leonard. Autonomous feature-based exploration. In *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*, volume 1, pages 1234–1240 vol.1, Sept. 2003.

[23] I. Ohya, A. Kosaka, and A. Kak. Vision-based navigation by a mobile robot with obstacle avoidance using single-camera vision and ultrasonic sensing. *IEEE Transactions on Robotics and Automation*, 14(6):969–978, 1998.

[24] S. Oßwald, M. Bennewitz, W. Burgard, and C. Stachniss. Speeding-Up Robot Exploration by Exploiting Background Information. *IEEE Robotics and Automation Letters*, 1(2):716–723, July 2016.

[25] J. Peters and S. Schaal. Reinforcement learning of motor skills with policy gradients. *Neural networks*, 21(4):682–697, 2008.

[26] S. Ross, G. Gordon, and D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. pages 627–635, 2011.

[27] J. M. Santos, T. Krajník, and T. Duckett. Spatio-temporal exploration strategies for long-term autonomy of mobile robots. *Robotics and Autonomous Systems*, 88:116–126, Feb. 2017.

[28] D. Santosh, S. Achar, and C. V. Jawahar. Autonomous image-based exploration for mobile robot navigation. In *2008 IEEE International Conference on Robotics and Automation*, pages 2717–2722, May 2008.

[29] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. v. d. Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484, Jan. 2016.

[30] R. Sim and J. J. Little. Autonomous vision-based exploration and mapping using hybrid maps and Rao-Blackwellised particle filters. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2082–2089, Oct. 2006.

[31] S. Song, F. Yu, A. Zeng, A. X. Chang, M. Savva, and T. Funkhouser. Semantic scene completion from a single depth image. *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[32] J. Wettach and K. Berns. Dynamic Frontier Based Exploration with a Mobile Indoor Robot. In *ISR 2010 (41st International Symposium on Robotics) and ROBOTIK 2010 (6th German Conference on Robotics)*, pages 1–8, June 2010.

[33] F. H. Wullschleger, K. O. Arras, and S. J. Vestli. A flexible exploration framework for map building. In *(Eurobot '99) 1999 Third European Workshop on Advanced Mobile Robots, 1999*, pages 49–56, 1999.

[34] B. Yamauchi. A frontier-based approach for autonomous exploration. In *, 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation, 1997. CIRA'97., Proceedings*, pages 146–151, July 1997.

[35] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning. pages 3357–3364, 2017.