Paper review: Where are the keys? Learning object-centric navigation policies on semantic maps with graph convolutional networks

Niko Sünderhauf

Patricio Cerda Mardini
IA Lab - Cognitive Robotics
2020

Contents

Introduction - About the paper

- Pre-print
- Presented at IEEE RAS International 2019 Summer School
- Research done at Queensland University of Technology, Australia

Introduction - Problem

- Where to look for objects within larger context?
  e.g. "Please get the milk"
- Humans intuitively know where to search
- Probabilistic underlying process: items are not randomly placed, but near a small set of other similar objects
- Application? Smart domestic service robotics: indoors, household-like environments

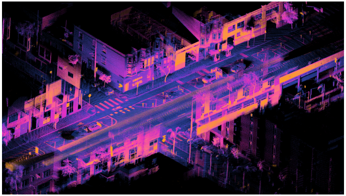
Introduction - Contribution

- They train an agent via Reinforcement Learning, using Graph Convolutional Networks (GCNs)
- Operate at graph-based map model of the environment
  - Nodes: robot poses *or* **static** object landmarks
  - Edges: within range for interaction
- Agent learns to find non-static objects on the map, even if not seen during training
- Agent generalizes over different graphs
- Fast convergence, and sped-up if pre-trained on proxy task

Contents

Concepts - Semantic SLAM

SLAM: Simultaneous Localization and Mapping



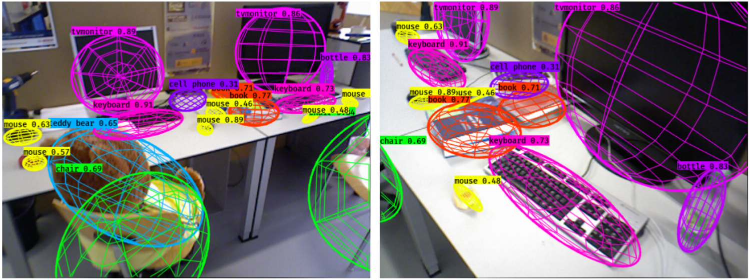
SLAM in action

Semantic? → rich environment representations, with objects as central unit

1. SLAM++
2. QuadricSLAM
3. Fusion++

These semantically rich graphs (with pose and object nodes) will be the starting point

Concepts - QuadricSLAM



QuadricSLAM - Sünderhauf et al

Objects as landmarks to estimate camera pose.

Uses DL-based low-dimensional visual descriptors.

Concepts - Goal directed navigation

Objective: policy that enables robot to find specific target in environment

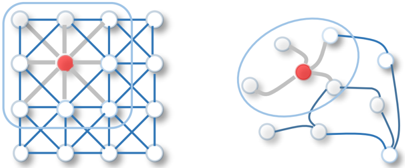
1. Object Goal Task Taxonomy (more on this later)
2. Visual navigation: from raw pixels, features can be low-level or high-level
3. Literature on building implicit map representations

Concepts - Graph Neural Networks

GNNs operate over graph data structures

A graph G is a pair (V, E) where:

- V is a set of vertices
- E is a set of edges



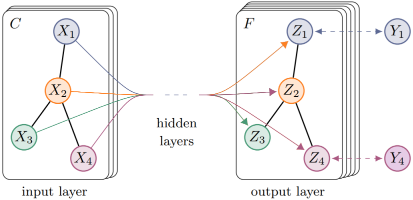
2D convolution vs graph convolution

Concepts - Graph Neural Networks

There are 4 families of GNNs:

1. Convolutional GNNs $\rightarrow$ used in this work
2. Recurrent GNNs
3. Spatial-temporal GNNs
4. Graph Auto-encoders

Concepts - Graph Convolutional Network (GCN)



Graph Convolutional Network

Concepts - REINFORCE algorithm

function REINFORCE
  Initialise θ arbitrarily
  for each episode $\{s_1, a_1, r_2, ..., s_{T-1}, a_{T-1}, r_T\} \sim \pi_\theta$ **do**
    for $t = 1$ to $T - 1$ **do**
      $\theta \leftarrow \theta + \alpha \nabla_\theta \log \pi_\theta(s_t, a_t) v_t$
    end for
  end for
  return θ
end function

$$\Delta\theta_t = \alpha \nabla_\theta log \pi_\theta(s_t, a_t) v_t$$

$$v_t = Q^{\pi_\theta}(s_t, a_t)$$

Contents

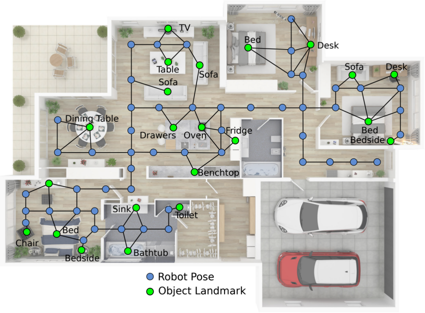
Problem

Definition (Task)

Given an indoor environment and its graph, find a given non-static "target object" by navigating to it

Assumptions

1. Environment previously mapped using semantic SLAM
2. Map is a graph
   - Pose nodes and object nodes
   - *(pose-pose)* edge: robot can navigate between both
   - *(pose-object)* edge: robot in range for interaction
3. Map objects are static



Example graph shows vertices and edges

Assumptions

4 Target objects are small, non-static, and not mapped

5 There are rules for which subset can they appear near to

[8] (*kitchen-table, benchtop, drawers, dining-table*) → *knife*. (*kitchen-table, benchtop, drawers, dining-table*) → *fork*. (*kitchen-table, benchtop, drawers, dining-table*) → *spoon*. (*kitchen-table, benchtop, drawers, dining-table*) → *bowl*. (*kitchen-table, benchtop, drawers, desk, dining-table*) → *cup*. (*kitchen-table, benchtop, drawers, dining-table*) → *glass*. (*kitchen-table, fridge*) → *milk*. (*kitchen-table, fridge, dining-table*) → *beer*. (*fridge*) → *apple*. (*fridge*) → *juice*. (*fridge*) → *oranges*. (*bed, sofa*) → *pillow*. (*bed, wardrobe, cabinet*) → *t-shirt*. (*bed, wardrobe, cabinet*) → *pants*. (*wardrobe, chair*) → *jacket*. (*wardrobe, cabinet*) → *socks*. (*bedside, desk, sofa, armchair*) → *glasses*. (*bedside, desk, sofa, armchair, TV*) → *keys*. (*bedside, shelf, sofa*) → *book*. (*sofa, armchair, TV*) → *remote*.

Target objects adjacency rules

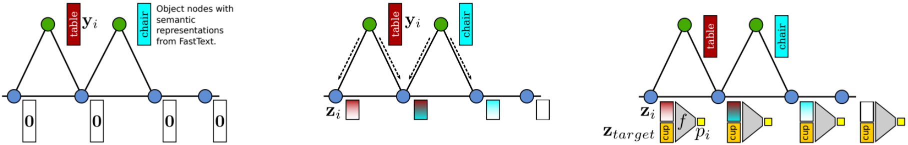
Assumptions

6. Underlying probabilistic process is hidden, unknown to robot
7. Positions are random per each episode
8. Policy is agnostic to target object[1]
9. π is a high-level planner: it proposes a node to visit
10. Robot can navigate to given goal pose using path-planning, motion control, obstacle avoidance, localization, etc.

[1] i.e. they do not train one π per target class

Contents

Approach



$$\pi(\mathcal{G}, c_{target}) = FC(GCN(Y))$$

- Single GCN layer
- Fully Connected block made up of 3 layers
- Trained with REINFORCE
- ReLU activations
- π provides distribution over vertices, conditioned on target
- Navigation goal selected sampling from π

Approach

$$\mathcal{G} = ((\mathcal{X} \cup \mathcal{L}), \mathcal{E})$$

- Poses $\mathcal{X}_i \in SE3: \ (x, y, z) + (\alpha, \beta, \gamma)$, initial feature vector $[0, ..., 0] \in \mathbb{R}^{300}$
- Landmarks $\mathcal{L}_j$: label $c_j \in \mathcal{C}^{map}$, FastText feature vector $y_j \in \mathbb{R}^{300}$
- Target labels $c_{target} \in \mathcal{C}^{targets}$
- $\mathcal{C}^{map} \cap \mathcal{C}^{targets} = \varnothing$
- $c_{target} \notin \mathcal{C}^{map}$

Approach

Aggregation operation:

$$Z = \sigma(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}Y\Theta)$$

Where:

- $\hat{A} = A + I$: adjacency matrix (with loops)
- Θ: GCN weights, $\in \mathbb{R}^{300x64}$
- Y: Node feature matrix, $\in \mathbb{R}^{Nx300}$
- $\hat{D}_{ii} = \sum_j A_{ij}$: the diagonal degree matrix of the graph

Approach

Logits for node i:

$$p_i = f_3(f_2(f_1([z_i; z_{target}])))$$

With:

$$z_{target} = \sigma(Y_{target} \cdot \Theta)$$

- $f_1 : \mathbb{R}^{128} \rightarrow \mathbb{R}^{64}$
- $f_2 : \mathbb{R}^{64} \rightarrow \mathbb{R}^{32}$
- $f_3 : \mathbb{R}^{32} \rightarrow \mathbb{R}^{1}$

Finally, π is the distribution denoted by every p_i. To set the navigation goal, we sample from π.

Some considerations

- Trick: manually set ($p_i = -100$) for landmark nodes to only sample pose nodes as goals
- Optimiser: ADAM(10^{-4})
- Task successful if target object is within first 10 navigation goals
- 20 different underlying probabilistic models are considered
- Environments are 100 % synthetically generated

Faster convergence

Pre-training on proxy task leads to better performance, as weights are initialized to better interpret semantic word representations:

- Does not require topology information
- In this case, objects in map can be from either $\mathcal{C}^{map}$ or $\mathcal{C}^{target}$
- Probabilistic model is different from all the ones in training
- Speeds-up learning process

Definition (Classification Task)

Given y_{target}, which pose nodes are connected to an instance of this class?

Contents

Experiments & results

- 200 agents in total, 1000 episodes each
- Baselines
    1. Random policy: pick navigation goal at random, never repeating
    2. Oracle policy: has access to the underlying probabilistic model, and picks node with maximum probability
- Metrics
    1. Success rate: considering 10 attempts
    2. Steps to target: only for successful episodes, how many attempts

Experiments & results

	Evaluate on Training Environment			
	random policy	no pre-training	with pre-training	oracle
success rate	0.33 ± 0.47	0.98 ± 0.13	0.99 ± 0.09	0.99 ± 0.09
steps to target	5.00 ± 2.70	1.41 ± 1.03	1.45 ± 1.09	1.66 ± 1.51

Table 1: results on seen environments

Evaluate on Unseen Environments			
random policy	no pre-training	with pre-training	oracle
0.26 ± 0.44	0.92 ± 0.28	0.96 ± 0.20	0.99 ± 0.12
5.10 ± 2.89	2.39 ± 2.07	2.02 ± 1.78	1.62 ± 1.49

Table 2: results on unseen environments

Experiments & results

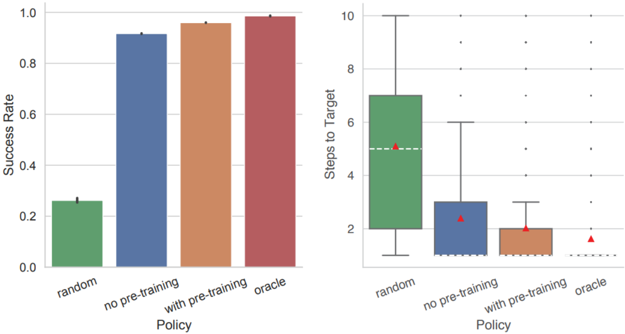


Fig. 3: Success rate (left) and distribution of steps to target (right) combined over all target classes for different policies.

Experiments & results

	random policy	no pre-training	with pre-training	oracle
success	0.25 ± 0.43	0.72 ± 0.45	0.76 ± 0.43	0.97 ± 0.17
steps	5.14 ± 2.99	3.16 ± 2.54	2.90 ± 2.44	1.89 ± 1.78

TABLE II: Results on unseen environments with *unseen* target objects.

Experiments & results

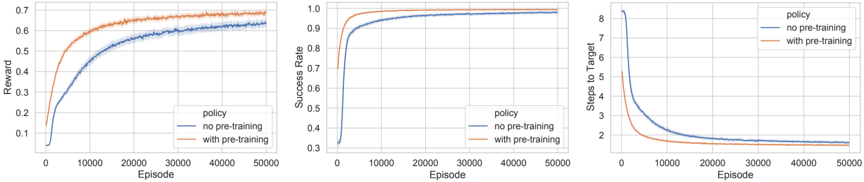


Fig. 4: Reward (left), success rate (centre), and steps to target (right) averaged over 200 training runs (10 randomly initialised networks $\times$ 20 environments with different probabilistic model). The shaded region around the line corresponds to the 90th percentile. When the policy network is initialised by the proxy task pre-training (explained in Section IV-F), it learns significantly faster, reaching the same level of performance after a fraction of the training episodes.

Experiments & results

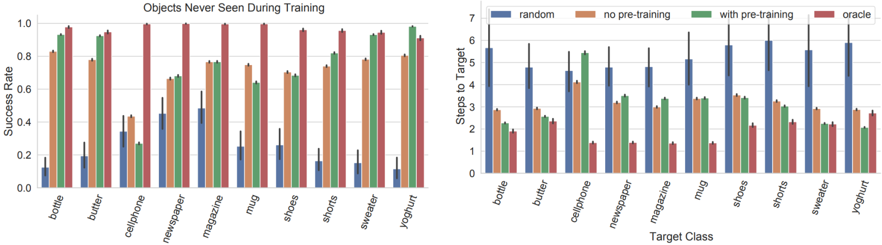


Fig. 6: Class-wise success rates (top) and steps to target (bottom) for different policies and target objects that were never seen during training. The learned policies generalise to these unseen objects as long as they are semantically close and behave similarly as objects the policy was trained on. The *cellphone* class is an exception here, since none of the original training classes is semantically close to it.

Commentary

- Using an inductive model, like Graph Attention Networks, would alleviate the performance drop in unseen environments. Also, attending in different magnitudes to each neighbor could be useful
- Maybe considering a "success@5" metric would tell interesting things
- Implement in a real robot!

Contents

Reference

[1] Sünderhauf, N. (2019). Where are the Keys? - Learning Object-Centric Navigation Policies on Semantic Maps with Graph Convolutional Networks. ArXiv, abs/1909.07376.

[2] Kipf, T., Welling, M. (2016). Semi-Supervised Classification with Graph Convolutional Networks. ArXiv, abs/1609.02907.

[3] Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y. (2017). Graph Attention Networks. ArXiv, abs/1710.10903.