Multiple Subgoals-guided Hierarchical Learning in Robot Navigation
Luo, Sha; Schomaker, Lambert

Published in:
2022 IEEE International Conference on Robotics and Biomimetics, ROBIO 2022

DOI:
10.1109/ROBIO55434.2022.10011912

IMPORTANT NOTE: You are advised to consult the publisher's version (publisher's PDF) if you wish to cite from it. Please check the document version below.

Document Version
Publisher's PDF, also known as Version of record

Publication date:
2023

Link to publication in University of Groningen/UMCG research database

Citation for published version (APA):

Copyright
Other than for strictly personal use, it is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license (like Creative Commons).

The publication may also be distributed here under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license. More information can be found on the University of Groningen website: https://www.rug.nl/library/open-access/self-archiving-pure/taverne-amendment.

Take-down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Downloaded from the University of Groningen/UMCG research database (Pure): http://www.rug.nl/research/portal. For technical reasons the number of authors shown on this cover page is limited to 10 maximum.

Download date: 17-12-2023
Multiple Subgoals-guided Hierarchical Learning in Robot Navigation

Sha Luo and Lambert Schomaker

Abstract—Solving obstacle-clustered robotic navigation tasks via model-free reinforcement learning (RL) is challenging due to the extended decision horizon and sparse rewards. Previous work has demonstrated efficient learning with single subgoal-conditioned hierarchical approaches. The subgoal is the action from the high-level policy and it operates on the low-level module, which could invoke sub-optimal policy when the selected subgoal is suboptimal. This work introduces multiple subgoals-guided navigation (MSGN) which consists of a high-level multiple subgoals Planner and a low-level goal-conditioned RL Controller. By passing multiple subgoals to the low-level agent, MSGN could alleviate the suboptimal subgoal problem by transferring the subgoal selection process to the RL agent. At the same time, multiple subgoals could help the goal-conditioned RL agent better explore and understand the environment and task. We tested our method on the Safety Gym suite. The results verified that MSGN could achieve a higher success rate and lower collision cost compared to baselines.

I. INTRODUCTION

Although Deep Reinforcement Learning [1] has achieved remarkable successes in many sequential decision making fields, such as playing games [2]–[3], recommendation systems [4], human-robot interaction [5], manipulation [6] and locomotion [7]. Applications of RL in robotic navigation that involves extended planning and obstacle avoidance are still challenging. This situation attributes to the difficulties in exploration near obstacles and the lack of informative rewards over a long decision horizon.

Hierarchical learning methods have been widely developed to split the long decision making problem into several subtasks to tackle the challenges in obstacle-clustered robotic navigation tasks. One of the most acknowledged methods is hierarchical reinforcement learning (HRL) [8]–[9]. HRL decomposes a RL problem into multiple levels of hierarchies, and each hierarchy corresponds to a subproblem and been solved by an Markov Decision Process [10]–[11]. However, HRL learns an end-to-end off-policy, which shares the same state space across layers and imposes disadvantages when generalizes to unseen tasks. Furthermore, end-to-end HRL also limits the modularity in transferring different functional policies across agents, environments and tasks [11].

In order to improve the generalizability and flexibility of policies in robotic navigation, a more explicit hierarchical decomposition method has emerged. Instead of coupling multiple RL layers into the hierarchy, each layer can be decoupled with different forms of methods [12]–[13]. An example of this is a hierarchical system incorporating both supervised and unsupervised neural networks with RL [14]. It is also entirely possible to include planning and optimization methods as one of the layers. For instance, optimization-based motion planners on the top-level propose spaced subgoals and RL on the lower level to learn to reach [15]. These compositional structures provide a more flexible scheme for learning different tasks through subgoals.

The essence of hierarchical learning systems lies in the design of subgoals, which simplifies the long-horizon decision making procedure by introducing a coarser time scale policy that chooses the subgoals as the action at certain steps to instruct the low-level behaviour policy. However, most work focuses on single subgoal solution at each decision step [9]. In reality, when human beings plan intermediate subgoals, there are always multiple options to choose from. These options reflect how the agent understands the task and provides backup plans in unexpected situations.

Motivated by this phenomenon, we are curious about the performance of hierarchical navigation under multiple subgoals guidance. Instead of proposing one subgoal, multiple subgoals are proposed from a higher level policy. For example, Xing [16] introduced a hierarchical learning method with a high-level Manager predicting multiple subgoals to speed up the exploration. Each subgoal corresponds to an option to control the environment. However, it uses multiple actor-critic architectures to generate multiple subgoals. Each architecture corresponds to a subgoal, which restricts the flexibility of the number of subgoals. Furthermore, the algorithm was highly tailored for Atari’s Montezuma’s revenge environments. Vivek et al. [17] designed a method called "many-goals", in which the agent generated many goals in the form of raw pixel observations. They verified that training RL with multiple goals and then using them as pre-training for auxiliary tasks can improve the performance of 49 Atari games. However, these algorithms do not consider taking advantage of many goals by exploiting these goals simultaneously to improve exploration efficiency.

To achieve fast exploration in obstacle-clustered navigation tasks, we devise a hierarchical learning method called multiple subgoals-guided navigation (MSGN). It consists of a high-level subgoal Planner and a low-level behaviour Controller, in which the Planner predicts multiple subgoals at a coarse time step invoked by certain conditions. The multiple subgoals provide richer information than single subgoal methods. Each subgoal corresponds to an option for the Controller policy to fulfil the task. The Controller focuses on learning the skill of reaching subgoals with the ultimate goal in mind. Any successful reach to one of the subgoals will be regarded as a subgoal completion, and the agent will be rewarded for choosing the correct actions. The Planner...
and the Controller cooperate in a synergistic way to finish the navigate task.

In summary our main contributions include:

- Propose a novel hierarchical framework that decomposes a long-horizon obstacle-clustered navigation task into a high-level multiple subgoals Planner and a low-level behaviour Controller.
- Propose a new concept of multiple subgoals-guided navigation. Compared to single subgoal methods, multiple subgoals provide richer information to lower level policies, which could help with the understanding and exploration.
- Provide empirical evidences to prove that multiple subgoals-guided navigation performs better than corresponding single subgoal-guided navigation and end-to-end valina RL-based navigation.

II. MULTIPLE SUBGOALS-GUIDED NAVIGATION

Considering the navigation task in obstacle-clustered environments, we propose a multiple subgoals-guided navigation, dubbed MSGN, to explore the effectiveness of multiple subgoals in subgoal-guided navigation. MSGN is a hierarchical learning method composed of two levels. At the higher level is a supervised learning-based Planner. The Planner works at a courser time scale and operates abstractly, mapping image observations to multiple subgoals for low-level navigation. The low-level Controller learns the reaching skills to navigate around obstacles under subgoals’ guidance. We explain the two modules in details in this section.

A. Planner: subgoals generation

1) Egocentric input transformation: In navigation, subgoals generation heavily depends on the robot’s movements and goal locations. So it is natural for the planner to focus more on geometrical relationships rather than the environment’s layout. There is a method called GOSELO: Goal-Directed Obstacle and Self-Location [18] which verified that a goal-directed map representation could help mobile robot’s navigation. Here, we customize our input based on GOSELO. The core concept is to transform the obstacle map with crop, rotate and rescale steps to locate the goal above the robot. So, the state representation is egocentric and explicitly correlated with the robot’s movement in the goal-reaching task instead of the plain representation of the environment.

The original representation of GOSELO contains two sets of maps, and each set has three different scale maps, resulting in six maps in total. One of the base maps is the obstacle map constructed from a bird’s eye viewpoint. The map is represented by binary values, in which one means the location of the pixel is obstacle-free and zero means the location of the pixel is obstacle-free. Another base map records the agent’s history locations from the start to the current state. This map uses integer values to represent the times of visiting a location. In our experiments, we eliminate the history images and use the obstacle related maps as the input for speed training as the transformed obstacle maps already implicate the locations of the robot.

The transformation basically includes two steps:

- **Rotation and translation:** The obstacle base map is rotated to locate the goal $\mathcal{G}$ on the top of the current location $\mathcal{P}$. Then the map is translated to locate point $\mathcal{M}$ at the centre of the maps, in which point $\mathcal{M}$ is the centre on line $\mathcal{GP}$.
- **Crop:** The resulting map is cropped into three different sizes. The widths of the squares are experimentally selected based on the field of the environment: $(L + 4, 2L, 4L)$, where $L$ denotes the pixel length of the line $\mathcal{GP}$. These cropped obstacle maps are then rescaled into a same-sized image with 3 channels ($W \times H \times 3$).

The image transformation process is illustrated in Fig. 1. The resulting three channels are shown on the right side of the figure. The dashed squares with magenta, blue and black colours represent crop widths $L + 4, 2L, 4L$ respectively.

2) Multiple subgoals prediction: The benefits of modeling the multimodality of subgoals include: a better understanding of the environment and the task, flexibility in changing situations, and avoiding manually selecting a potential sub-optimal subgoal. However, existing approaches in subgoal-guided navigation are mostly limited to one next subgoal prediction. In this work, we explore the method of predicting multiple next subgoals with an Evolving-Winners-Takes-All (EWTA) [19] loss function and use this loss function to group features into clusters iteratively.

A conceptual overview of the multi-subgoal prediction diagram is shown in Fig. 2. The network’s input $x$ is the customized GOSELO-formatted images with the goal transformed to above the robot $x = (I_1, I_2, I_3)$. Given $x$, the goal of the network is to predict multiple subgoals at a fixed travel distance $d$ in the future. The training data contains a set of images $(x)$ and ground-truth future subgoals $(y = (loc_1, loc_2))$: $D = (x_1, y_1), ... (x_N, y_N)$, with $N$ representing the dataset size. The network is composed of a three-layer Convolutional Neural Networks (CNNs). The first two layers are followed with a dropout layer with a probability of 0.5, and the last CNN layer followed with a dropout layer with 0.2 as the probability. Then an average pooling layer is added and followed by two fully connected layers.

We use the loss function EWTA for training. The Planner maps state and goal to subgoals under policy $\pi_g$: $sg = \pi_g (|g, s|)$. Each output $sg_k$ is regarded as a subgoal and we calculate the Euclidean distance $dis_k$ between the predicted subgoal and the ground-truth $y$ as: $dis_k = \|sg_k - y\|$. We define $Dis = [dis_1, ..., dis_K]$ to gather the Euclidean distances from all of the subgoals to the ground truth. Then, the loss function denoted as:

$$L = \sum_{k=1}^{K} w_k \times dis_k$$

where $w_k$ is a binary value used to select the top $n$ winners for the update, which equals 1 when $dis_k$ is sorted among the first $n$ shortest distance in $Dis$, otherwise $w_k$ equals 0. We use $K$ to represent the number of subgoals. We start with
$n = K$, and then gradually decrease $n$ until $n = 1$. Instead of optimizing one output one time with the risk of getting stuck in equilibrium (i.e. a predicted subgoal is attracted by multiple ground truths), EWTAP pairs multiple subgoals with the ground truths at each update. This way, it alleviates the convergence problem and reduces the number of un-updated subgoals.

### B. Controller: subgoals-guided navigation

A goal-conditioned RL algorithm is used as the base algorithm for the controller. The objective of the algorithm is to maximize the expected discounted sum of reward $R = \mathbb{E}_\pi[\sum_{t=0}^\infty \gamma^t r(s_t, a_t)]$, where $\pi = \pi_{\theta}(a_t|s_t, g, s_{gt})$ with $g, s_{gt}$ representing the goal and subgoals at time step $t$ respectively.

The training scheme works on different time scales. For generality, we assume each episode has maximum $T$ steps, and the Planner is triggered to predict the subgoals under certain conditions $\mathcal{C}$. $\mathcal{C}$ includes: the robot travelled $H$ steps since the start or since the last reach of subgoal; one of the subgoals has been successfully accessed. A subgoal is reached when the distance between the robot and the subgoal is less than $\epsilon$. Given multiple subgoals, the Controller interacts with the environment with maximum $H$ steps before the Planner predicts the next new subgoals. Formally, we define:

$$\text{Planner} : s_{gt} = \begin{cases} \pi_{\theta}(|g, s_t) & \text{if } \mathcal{C} \text{ meet,} \\ s_{gt-1} & \text{otherwise.} \end{cases}$$

Controller : $a_t \sim \pi_{\phi}(|s_t, g, s_{gt}), \quad 0 \leq t < T \quad (2)$

where $s_{gt}$ represents the subgoals, $s_t$ and $a_t$ denote the state and action respectively at the time step $t$. We illustrated the algorithm MSGN in Algo. 1.

**Algorithm 1 Multiple Subgoals-guided Navigation (MSGN)**

**Training Planner policy $\pi_\theta$**

**Initialize Controller policy $\pi_\phi$**

**Initialize maximum episodes $E$ and steps $T$ in an episode**

**for** $e$ in $1, ..., E$ **do**

**Initialize initial state and goal** : $s_1 \sim \rho_0$ and goal $g_e \sim \rho_g$;

**Initialize subgoals $s_{g0} \sim \pi_\theta(|g_e, s_1)$**;

**Initialize steps in subgoal window** : $h = 0$

**for** $t$ in $1, ..., T$ **do**

Sample action $a_t$ using goal-conditioned Controller policy $\pi_\phi(|s_t, g_e, s_{gt-1})$;

Execute $a_t$ and obtain next state $s_{t+1}$;

If $\mathcal{C}$ then

Update subgoals $s_{gt} \sim \pi_\theta(|g_e, s_t)$;

Reset $h = 0$;

else

$s_{gt} = s_{gt-1}$;

$h = h + 1$;

Update $\pi_\phi$;

**III. EXPERIMENTAL RESULTS**

In this section, we evaluate the proposed MSGN to answer the following questions:

- How do the subgoal distance $d$ and the number of subgoals $K$ affect the Planner and the navigation performance?
- Does our method MSGN improves the navigation performance compared to the vanilla RL algorithm?
- Does multiple subgoals scheme outperforms single subgoal method?

**A. Environment setup**

We evaluate the method on a customized Mujoco environment: safety-gym [20], in which a point robot is selected as
the agent. The task for the robot is to reach a goal while avoiding obstacles. We illustrate the environment and task in Fig. 3 where all of the four sub-figures are recorded from a bird’s eye viewpoint. We considered two different environments in the experiments, one with hazards as the obstacles and another with pillars as the obstacles. Hazards are dangerous areas the agent should avoid. They are non-physical, and the agent can get across them but with a penalty, as we can see from Fig. 3b. Pillars are immobile rigid obstacles in the environment that should not be touched, and the agent can not come across, as shown in Fig. 3d.

![Fig. 3: Experimental environments. The red point is the robot, the green circle is the goal to be reached. We used two different obstacles: hazards and pillars in the environment as shown as blue circles and light purple pillars respectively. (a) environment with hazards as the obstacles; (b) a collision situation in the hazards environment; (c) environment with pillars as the obstacles; (d) a collision situation in the pillars environment.](image)

The details of the states, actions and rewards in the environment are explained as follows:

- **States**: The agent’s state is composed of three parts: standard robot sensors, robot status and environmental perceptions. The standard robot sensors include an accelerometer, a velocimeter, a gyroscope and a magnetometer; each sensor dimension has a dimension of three. The robot status is represented by the robot centroid in the world frame. Same as the robot sensors, each has a dimension of three. Environment perceptions contain two robot-centric lidar observations: a goal lidar and an obstacle lidar, and each lidar observes ten directions for a full 360 view. An additional subgoal lidar will be added to the state in the hierarchical framework. So, the state will be a dimension of 35 and 45 for pure RL policy and hierarchical policy respectively.

- **Actions**: We used robot point for the experiments. The robot has two actuators, one for turning left or right, another for moving forward or backward. So the dimension of the actions is two; each action represents the amount of torque applied on the actuator.

- **Rewards**: The reward is formed with three components: a goal-reaching stimulus term, a punishment for collision and an encouragement term for subgoal reaching. The reward function is:

\[
f(s_t, a_t) = \alpha_1 * R_{\text{collision}} + \alpha_2 * R_{\text{goal}} + \alpha_3 * R_{\text{subgoal}},
\]

in which \(R_{\text{subgoal}}\) only existed in hierarchical policy learning settings, and \(\alpha_1, \alpha_2, \alpha_3\) are the weights for different reward terms.

- **Terminal conditions**: See the definition of \(C\) in II-B.

In the experiments, the robot, goal and obstacles are randomly scattered in the environment at the beginning of each episode, given the constraints that they do not overlap. The field size is \((W \times H)\), where \(W = H = 2m\). The maximum steps in an episode is set to \(1k\). In addition, the tolerance for a successful goal-reaching is set to \(0.2m\).

The coefficients in (3) are designed as follows: \(\alpha_2 = 1\) for all the situations; \(\alpha_1 = 0.05\) for SAC hazards environment and \(\alpha_1 = 0.08\) for PPO hazards environment, \(\alpha_1 = 1\) for SAC pillars and PPO pillars environments; in pillars environments, \(R_{\text{collision}}\) is the number contact between the agent and the pillars; in hazards environments, \(R_{\text{collision}} = ss - dd\), where \(ss\) is the size of the hazard and \(dd\) is the distance between the agent centroid and the hazard centroid when the agent enters the hazard; \(\alpha_3 = 0.5\), \(R_{\text{subgoal}} = 0.2\) for a successful subgoal reaching; \(R_{\text{goal}} = 1\) for a successful final goal reaching.

![Fig. 4: (a) Evaluation loss on different number of subgoals. (b) Evaluation loss on different distances between the agent and subgoals. (c) Change of \(n\) in the training process. (d) Learning rate schedule in the training process.](image)

**B. Parameters design**

We conducted an ablation study to analyze how different numbers of subgoals affect the subgoals prediction performance in the Planner. The selected numbers are \(K = 1, 2, 4, 6, 10, 20\), and the resulted corresponding prediction losses are \(0.165, 0.107, 0.077, 0.066, 0.056, 0.045\). From the results, we observe that more subgoals leads to predictions with higher accuracy. This trend is also verified from the evaluation loss along the training process, as shown in Fig. 4a. However, increasing the number of subgoals will also slow down the training process because of the increased computations on subgoals’ backpropagations. So, we choose \(K = 20\) as the number of subgoals in our experiments.

The distance between the current robot’s location and the subgoal also plays an important role. We conducted an experiment to compare the effects of different subgoal distances \(d = [0.25, 0.50, 0.75, 1.0]m\) and received prediction losses of \(0.058, 0.045, 0.044, 0.067\) respectively. The results show...
that the Planner achieved the highest prediction accuracy with subgoal distance $d = 0.75m$. However, the subgoal functions as an information bridge to help exploration, and the prediction accuracy is not the only factor that decides the high performance. The best goal navigation behaviour is not necessarily bound with the most accurate subgoal. The small error difference in $0.5m$ and $0.75m$ may not affect the final performance much. Hence, we evaluate the navigation performance with the subgoal distance of $0.5m$ and $0.75m$ on the low-level RL to further analyze the correlations. The state-of-art RL algorithm SAC is implemented in an environment with eight hazards. Since we use $H = 50$ trial steps for a distance of $0.5m$, we use $H = 75$ trial steps for a subgoal distance of $0.75m$. We evaluate the learned policies under three randomly selected seeds and get average rewards over $100$ episodes; the average rewards for $0.5m$ and $0.75m$ are $2.177$ and $2.010$ respectively (the larger, the better). The cost for $0.5m$ and $0.75m$ are $11.63$ and $13.767$ respectively (the smaller, the better). The episode lengths are $155$ and $193.33$ steps respectively (the smaller, the better). The results show that the subgoal distance of $0.5m$ works better than $0.75m$ in the final navigation task. The evaluation loss is also plotted in Fig. 4b. The value of $n$ in top-n subgoals that used for updating the parameters, and the learning rate $lr$ during training process are recorded and illustrated in Fig. 4c and Fig. 4d respectively.

C. Performance comparison

In order to evaluate the performance of multiple subgoals-guided navigation, we test MSGN in two different Safety Gym environments with hazards and pillars as the obstacles. For the low-level Controller, the state-of-the-art RL algorithms PPO and SAC are used to learn the short-distant subgoals navigation skills. The baselines include vanilla PPO [21], SAC [22] [23] algorithms and single subgoal-guided navigation (SSGN) methods. SSGN is directly derived from MSGN. The difference between MSGN and SSGN is that MSGN uses all of the subgoals predicted by the high-level Planner, while SSGN selects the best subgoal among all of the predicted subgoals.

PPO’s architecture and parameters are strictly the same as the implementation in Spinningup [24]. The difference is that we train the policy with 500 episodes, and each episode has a maximum of 1000 steps based on the task’s complexity. For SAC, the network architecture and parameters are the same as used in the Safety Gym benchmark [20].

We recorded the average success rates, cost for collisions, and episode lengths during the training process. We compared them in four settings with different base RL algorithms and environments: PPO-Hazards, PPO-Pillars, SAC-Hazards, and SAC-Pillars. The measures in the four settings are depicted in Fig. 5. From Fig. 5a, we can see that vanilla PPO does not perform well in the Hazards environment. However, subgoals-guided methods: MSGN and SSGN play important roles in levering the performance to achieve higher success rates, lower cost and shorter length than vanilla PPO. This trend is also verified by the curves in SAC-Hazards setting in Fig. 5c, where the subgoals-guided methods perform better than the vanilla SAC. In the Pillars settings, PPO and SAC methods seem to have different learning styles. In PPO-Pillars, as shown in Fig. 5b, we can see that a greedy vanilla PPO policy could maximize the success rate with the sacrifice of having more collisions. As we can see, the vanilla PPO achieved high success rates but also with high costs. On the other hand, in SAC-Pillars, vanilla SAC was more conservative as it achieved a poor success rate but kept the collision number under control, as shown as the blue curve in Fig. 5d.
TABLE I: Performance comparison between different methods in SAC settings (standard deviation in the round brackets).

<table>
<thead>
<tr>
<th>Env</th>
<th>Base RL</th>
<th>Rewards</th>
<th>Cost</th>
<th>SR</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hazards</td>
<td>SAC</td>
<td>1.457 (0.348)</td>
<td>20.807 (5.05)</td>
<td>0.880 (0.026)</td>
<td>154.982 (13.902)</td>
</tr>
<tr>
<td></td>
<td>SAC-SSGN</td>
<td>2.200 (0.122)</td>
<td>11.703 (2.737)</td>
<td>0.983 (0.012)</td>
<td>131.152 (7.54)</td>
</tr>
<tr>
<td></td>
<td>SAC-MSGN</td>
<td>2.213 (0.006)</td>
<td>11.667 (0.351)</td>
<td>0.99 (0.00)</td>
<td>141.097 (6.399)</td>
</tr>
<tr>
<td>Pillars</td>
<td>SAC</td>
<td>-2.283 (1.277)</td>
<td>3.897 (1.102)</td>
<td>0.597 (0.124)</td>
<td>234.451 (19.478)</td>
</tr>
<tr>
<td></td>
<td>SAC-SSGN</td>
<td>1.403 (0.365)</td>
<td>1.343 (0.388)</td>
<td>0.947 (0.058)</td>
<td>154.529 (2.167)</td>
</tr>
<tr>
<td></td>
<td>SAC-MSGN</td>
<td>1.593 (0.327)</td>
<td>1.113 (0.172)</td>
<td>0.977 (0.015)</td>
<td>160.924 (17.753)</td>
</tr>
</tbody>
</table>

Based on the experimental results, we draw the following conclusions: MSGN and SSGN do outperform the vanilla RL algorithms; it is also clear that MSGN is superior to SSGN in PPO related settings as all of the criteria are showing better performance in MSGN. However, the latter conclusion is not easy to draw for the SAC settings.

Hence, we further tested the performance of the trained policies regarding the accumulated rewards, success rate, cost, and successful episode length in SAC settings. Each test had 100 episodes under four different randomly selected seeds. Finally, we summarize the results in Table I. The table shows that MSGN methods perform better than SSGN methods and vanilla SAC methods with higher accumulated rewards, lower costs, and higher success rates. However, the trajectory lengths of MSGNs are slightly longer than those in SSGNs, which can attribute to the necessary longer paths in successful obstacles-avoiding behaviours.

IV. CONCLUSIONS

We presented Multiple Subgoals-guided Navigation (MSGN), an approach for solving obstacle-clustered navigation tasks with high-dimensional state observations. MSGN can solve tasks that are difficult to solve with conventional vanilla goal-conditioned RL policies. This multiple subgoals concept holds the potential to make goal-conditioned RL methods in navigation more flexible and capable. Our MSGN takes the first step in this direction, though many crucial questions remain answered. For instance, the fixed-distanced subgoals are unnatural in real life navigation. Although the decoupling of Planner and Controller can improve the flexibility, it also causes reachability problems in subgoals because of the complex robot dynamics. Our future work direction will be the investigation of flexible distanced subgoals and subgoals reachability.

REFERENCES