Sequential Action Selection for Budgeted Localization in Robots

Nassim AKLIL, Benoit GIRARD and Mehdi KHAMASSI
Institute of Intelligent Systems and Robotics - ISIR
University Pierre and Marie Curie - UPMC
Paris, France

Ludovic DENOYER
Laboratory of Informatics of Paris 6 - LIP6
University Pierre and Marie Curie - UPMC
Paris, France

***Abstract*—Recent years have seen a fast growth in the number of applications of Machine Learning algorithms from Computer Science to Robotics. Nevertheless, while most such attempts were successful in maximizing robot performance after a long learning phase, to our knowledge none of them explicitly takes into account the budget in the algorithm evaluation: e.g. budget limitation on the learning duration or on the maximum number of possible actions by the robot. In this paper we introduce an algorithm for robot spatial localization based on image classification using a sequential budgeted learning framework. This aims to allow the learning of policies under an explicit budget. In this case our model uses a constraint on the number of actions that can be used by the robot. We apply this algorithm to a localization problem on a simulated environment. Our approach enables to reduce the problem to a classification task under budget constraint. The model has been compared, on the one hand, to simple neural networks for the classification part and, on the other hand, to different techniques of policy selection. The results show that the model can effectively learn an efficient policy (i.e. alternating between sensor measurement and movement to get additional information in different positions) in order to optimize its localization performance under each tested fixed budget.**

I. INTRODUCTION

Spatial localization is one of the most challenging problems in Robotics. The main problem consists in taking a spatial decision in the environment in order to localize itself on a map using the different sensors that are available to the robot. The processing of these data is generally difficult because of their multimodality. The problem is made even more difficult by the mutual dependency of the localization and mapping steps: in order to localize itself, the robot needs to recognize cues and features which characterize a particular place and which have previously been perceived and stored. Conversely, to build a reliable map and correctly situate features within it, the robot needs to be able to localize itself relative to these features [1].

While several mapless robot navigation solutions exist [2], the problem of robotic localization has been classically and widely studied using the Self Localization and Mapping framework (SLAM, [3]; [4]; [5]), which proposes to simultaneously realize the localization and mapping steps. While SLAM methods may have difficulties during long navigation experiments – facing the loop closure problem where the robot needs to reset its estimations when recognizing a previously visited place, or having difficulties satisfying the hypothesis of a static world on which SLAM is anchored (see [6] for discussion) –, SLAM methods can produce robust and efficient localization when no limit is set on the amount of data and sensors which can be processed by the robot. However, while SLAM works both with lasers and cameras. Moreover, SLAM is not concerned with action selection, and thus cannot tell how information gathering for the localization process should be integrated within the global policy of the robot to maximize a given reward function.

Machine Learning research has recently come up with formal solutions to take into account an explicit budget for image recognition or data classification [7], [8], [9]. In particular, specific algorithms called Sequential Budgeted Learning algorithms are used in order to learn sequences and representations from limited amounts of data, which offers the possibility of adding an explicit budget to limit the model. One of the goals of these approaches is to limit the number of costly accesses to data to the minimum required for successful classification. One way to do that is to incorporate the decision to access or not to access data in the policy of the agent, so that it learns to timely access data among other possible actions.

The idea of data acquisition considered as an action is also at the core of the active sensing field, mainly developed in the 2000's. However, these techniques are limited by the fact that the systems learn action sequences having already learned the task representation. As shown in [10], the main technique used in the active sensing field is based on maximizing a weighted sum of rewards associated to a sequence of actions executed by a robot.

In this paper, we propose a model that makes a robot use as minimum data as possible to learn representations from the environment and to learn an optimal policy in order to accomplish a localization task. Hence, our problem is defined within a mapless navigation framework: the robot uses only the perceptions obtained via its sensors to take a spatial decision and is not based on a explicit map.

II. RELATED WORK

The mapless navigation problem has been widely investigated since the late 90s. Different techniques are used and can be divided into three main subsets: optical flow, appearance or object recognition based navigation [11]. The first category resumes the techniques that are based on the motion of all the surface elements from the visual world. The robot localizes itself using the velocity of the different images [12], [13], [14], [15], [16]. The second category describes the techniques that rely on memorizing the working environment: the idea is that in a way or another the robot stores images of the environment and then compares the received images in an online phase with the stored memory. The last category is based on objects and landmarks recognition in the environment.

Our approach belongs to the second category: appearance-based navigation, usually consisting in two different phases. First, a training phase where the robot learns the places in the environment from the recorded images. Second, a navigation phase where the robot has to recognize the places by comparing them to the images stored during the training phase. In this context, [17] performs indoor route construction by comparing the current image with the training data set, simply calculating a distance between them. In [18] the robot creates a sequence of images by storing the motion associated to each image. [19] use a histogram representation for the images encountered in a training phase and during the inference they compare the new images to the training samples with a quadratic distance to localize the robot in its environment. Our case is slightly different from these papers since, in the training phase, we do not extract specific informations from the images (only the RGB description). The work described here can also be compared to the active sensing literature [10], whose main goal is to minimize the acquired data in order to complete a task.

On the other hand the machine learning field has developed different ways of analyzing data. The recent deep learning state-of-the-art has many promising results on how data are processed in order to make agents learn representations, policies or both. More specifically, algorithms in the budgeted learning field have been studied in order to make agents learn from limited amounts of data. In [7] the authors have a sequential architecture, where the model learns representations at each time step using sequentially given data, but the available amount of data is unlimited. A budgeted version of this model was recently proposed in [20]. In both papers, it is specified that data are given between each transformation step. A similar architecture has been presented in [9], where the authors used an explicit budget but with no observations (or data) between each step.

Our model proposes a version where we use both approaches described above. The model uses an explicit budget and observations are returned given the action that the agent performs.

III. Model

A. Principles

We propose a model applied to a localization task, where the model aims at learning which action to choose in a set of possible actions (movement or acquisition of new information) at each time step. The model is restricted by a budget B that limits the number of actions allowed in order to complete the task in a given environment. We aim at learning to alternate between movements and data acquisition in order to collect relevant information and thus to localize efficiently. Our algorithm relies on the Deep Reinforcement Learning paradigm i.e learning a neural network-based policy by using reinforcement learning techniques, more precisely by using policy gradient techniques [21], where the model will reinforce the sequence of actions that allowed it to successfully complete the task. However in our case, the policy learning is not driven by a reward signal but by a defined loss function Δ that computes the quality of the system resulting in a model different than classical RL approaches.

Model Description

Let us denote $\mathcal{X}$ the set of all the possible positions of the robot in a particular given environment. At the beginning of each episode, when $t = 1$, the robot will be at a particular unknown position denoted x, Then, by sequentially choosing actions a_t at each time step t in the set of all possible actions $\mathcal{A}$, the robot will either gather a new information by using one of its sensor, or move in the environment.At the end of the process, the robot will predict its position y. The quality of the prediction will be measured through a differentiable loss function $\Delta(x, y) \in \mathbb{R}^+$.

Let us denote o_t the observation acquired by choosing action a_t such that $o_t \in \mathbb{R}^{n_{a_t}}$, n_{a_t} being the size of the observation space corresponding to action a_t i.e the size of the acquired information if a_t is a sensor acquisition action, or 0 if a_t corresponds to a robot movement. Note that this assumption is different then the classical assumption of Reinforcement Learning where an agent receives at each time step an observation from the same observation space. The value of o_t is defined by the unknown probability $P(o_t|a_t, a_{t-1}, ..., a_1, x)$ which depends on the environment. We will denote $\pi(a_t|a_{t-1}, o_{t-1}, ..., a_1, o_1)$ the policy of the robot, i.e the probability of choosing action a_t knowing the previously acquired information $o_{t-1}, .., o_1$ and the previously chosen actions $a_{t-1}, ..., a_1$. The final decision function which will predict the robot position w.r.t acquired information will be denoted $f(a_t, o_t, a_{t-1}, o_{t-1}, ..., a_1, o_1, x)$.

Learning Algorithm

Let us denote $(x_1, ..., x_m)$ the set of training positions i.e the m robot positions that will be used during training. Let us denote B the maximum number of actions allowed to the robot[1]. The learning objective is to find both the policy π^* and the prediction function f^* that minimize the prediction error:

$$\pi^*, f^* = \arg\min_{\pi, f} L(\pi, f) \tag{1}$$

where

$$L(\pi, f) = \mathbb{E}_\pi[\Delta(f(a_B, o_B, ..., a_1, o_1, x), y)] \tag{2}$$

where the trajectories $a_B, o_B, ..., a_1, o_1$ are sampled following π. The minimization of this objective will be made by using policy gradient techniques proposed in [9].

[1] We consider that B is fixed, the extension of this model to variable number of steps being the object of a future research.

Let us denote T a trajectory, where $T = a_B, o_B, ..., a_1, o_1$, the previous objective function can be rewritten as:

$$L(\pi, f) = \int (P(T|x)\Delta(f(x,T), y)dTdxdy \quad (3)$$

and its gradient can be written as :

$$\nabla_{\pi,f} L(\pi, f) = \int \nabla_{\pi,f}(P(T|x)\Delta(f(x,T), y))P(x,y)dTdxdy \quad (4)$$

This gradient can be estimated by using Monte Carlo sampling techniques over the set of training positions where M is the number of trajectories (total of action sequences):

$$\nabla_{\pi,f} L(\pi, f) \approx$$
$$\frac{1}{n}\sum_{i=1}^{n}\left[\frac{1}{M}\sum_{k=1}^{M}\nabla_\pi(\log P(T|x_i))\Delta(f(x_i,T), y_i) + \nabla_f \Delta(f(x_i,T), y_i)\right]$$

The gradient is then composed by two terms. The first one aims at correcting trajectories by penalizing the trajectories with high loss and the second one is the gradient for the prediction part. Note that:

$$\nabla \log P(T|x_i) = \nabla \sum_{t=1}^{B} \log P(a_t|a_{t-1}, o_{t-1}, ..., a_1, o_1) \quad (5)$$

This estimation of the gradient can have a high variance that has been corrected by replacing $\Delta(f(x_i,T), y_i)$ with $\Delta(f(x_i,T), y_i) - b$ where $b = E_{x,T,y}\left[\Delta(f(x,T), y)\right]$ that can be estimated from the training set.

Recurrent Neural Network-based Policy

From these definitions we have: (i) to model $\pi(a_t, a_{t-1}, o_{t-1}, a_1, o_1)$ and (ii) to model $f(a_t, o_t, a_{t-1}, o_{t-1}, a_1, o_1)$. In these two cases, we have to aggregate the information gathered by the robot i.e the o_ts. This will be handled by using a classical recurrent neural network mechanism: at each time step, the current trajectory will be captured through a latent vector z_t in a latent space $\mathbb{R}^N$ where N is the dimension of this space.

We denote $h_a(z, o) : \mathbb{R}^N \times \mathbb{R}^{n_a} \rightarrow \mathbb{R}^N$ the aggregation function associated with action a and which computes the latent vector z_{t+1} from z_t using the information o_t collected at time t by choosing action a. Moreover, we denote $g(a, z_t)$ the function that computes the probability of each possible action from z_t. Given these two functions, the inference algorithm can be written as Algorithm 1 . In this algorithm, at each time step t, $g(a|z_t)$ is computed from the latent state z_t and the action a_t is sampled from $g(a|z_t)$. An observation o_t associated to a_t is then returned that will be used to compute the next latent state z_{t+1} such that:

$$h_a(z_t, o_t) = tanh(W_z z_t + W_a o_t) \quad (6)$$

Where W_z and W_a are matrices associated to z and o.

The resulting gradient can be computed by using back-propagation techniques[2].

[2]The source code in Torch7 will be made available in case of acceptance of the article

Algorithm 1 Inference

```
1: for t ← 1, B do
2:     p = g(a/z_t)              ▷ Compute the action distribution
3:     Apply action a_t
4:     Acquire observation o_t
5:     z_{t+1} ← h_{a_t}(z_t, o_t)   ▷ Compute next latent vector
6: end for
7: ŷ ← f(z_B)                    ▷ Prediction
```

IV. Experiments

We have made a set of preliminary experiments using a simulated 2D environment. This environment corresponds to a 50×50 grid where each position corresponds to an empty one or to a colored wall. We consider an agent moving in this environment with 2 possible move actions *(turn left, turn right)*, and one acquisition action where the robot can acquire a 1D image through a virtual camera. This image corresponds to a partial observation of what is in front of the robot. The goal of the agent is to predict its position using B actions. An example of such a maze is given in Figure 1. Note that the position prediction problem has been casted in a 4×4 classification problem as it is illustrated on the figure. When a_t is a movement action, the agent receive an empty observation o_t while when a_t is an image acquisition action, the agent receives a vector of values corresponding to the RGB-pixels in front of the robot. Note that when choosing to turn right or left, the robot changes its angle by $\pi/4$. For the training, we have sampled 5000 training positions (and 2500 validation positions to tune the parameters of the model) and the quality of the model has been evaluated in term of accuracy on 2500 different testing positions. A position is characterized by the coordinates of the robot in the maze and its orientation, sampled in the set of $\{0, \pi/2, \pi, 3\pi/2\}$.

We have compared our approach with 2 different baselines, and for different values of B:

- The image classification baseline corresponds to a classical classification model (i.e multilayer perceptron) where $1, 2$ or 3 images are acquired by turning the agent on the left at each time step. The collected images are then concatenated and given to the classification model
- The *Forced policy* model is a model where π has been manually chosen in order to alternate between image acquisition and *turn left* action. The resulting trajectory for $B = 5$ is thus $(image, turn, image, turn, image)$.
- The *Learned policy* corresponds to our model learning where the agent can freely choose which action to apply.

For the *Forced policy* and the *Learned policy*, we have explored two variants i.e *recurrent* and *non recurrent*. In the first case, the h_a function is reused at every time step while in the second case, one uses on h_a^t function at each time step resulting in a model with more parameters to learn. the parameters (size of the latent space, learning rate, etc...) have been chosen by cross-validation. The results presented in Table I have been averaged over 3 different runs.

	Budget		
Type	1	3	5
Image classification	53.9	56.6	59.8
Forced policy			
Recurrent	49.9	67.4	75.9
Non Recurrent	55.8	60.8	61.2
Learned policy			
Recurrent	52.9	61	70
Non Recurrent	46.8	69.4	60.04

TABLE I
RESULTS FOR SIMULATED CASE. RESULTS REPRESENT PERFORMANCE ON TEST SET (IN PERCENTAGE)

First, one can see that the quality of the classification model improves when the number of acquired images increases. It confirms that providing more information to the agent helps him to compute a better localization than a single image. Moreover, when using the *Forced policy*, the model is able to achieve 75.9% when collecting 3 images ($B = 5$) and thus to increase its performance by 50% w.r.t using only 1 image. The *Learned policy* model is able to achieve a 70% accuracy on the same task showing that the agent has learned a relevant policy and has been able to discover how to move and when to acquire information. Note that the *recurrent* versions of the two models give a better performance since they need to estimate a smaller number of parameters than the *non recurrent* versions allowing a better generalization

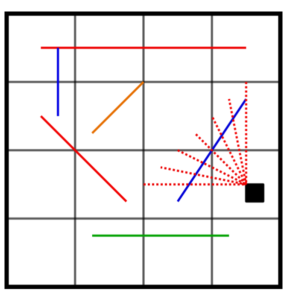

Fig. 1. Simulated data. The black square represents the robot. The dotted lines represent the range of the camera. The plain colored lines represent randomly placed walls. The horizontal and vertical lines represents the spatial discretization.

V. CONCLUSION

We have introduced in this paper a new learning model where an agent can decide when to acquire information for a given localization task. It corresponds to an original problem where the information acquisition has a cost which is different to the classical paradigm where information is gathered at each time step. We have proposed a set of preliminary experiments showing the interest of this approach. Future research directions include the evaluation of this model on a real robotic task where the robot has multiple sensors, each sensor being associated to a particular cost. Moreover, we plan to investigate an extension of this model where the number of steps is not fixed, and where the agent can decide when to stop the sequential acquisition process.

ACKNOWLEDGMENT

This work has been partially supported by the Agence Nationale de la Recherche and Labex SMART (ANR-11-LABX-65 Online Budgeted Learning Project).

REFERENCES

[1] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: Part I," *IEEE Robotics and Automation Magazine*, vol. 13, no. 2, pp. 99–108, 2006.

[2] S. Chen, Y. Li, and N. M. Kwok, *Active vision in robotic systems: A survey of recent developments*, 2011, vol. 30, no. 11.

[3] P. Moutarlier and R. Chatila, "An Experimental System for Incremental Environment Modelling by an Autonomous Mobile Robot," *Experimental Robotics I*, vol. 1560, pp. 327–346, 1989.

[4] R. C. Smith and P. Cheeseman, "On the representation and estimation of spatial uncertainty," *International Journal of Robotics Research*, vol. 30, no. 9, pp. 2907–2926, 1986.

[5] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "FastSLAM: A factored solution to the simultaneous localization and mapping problem," *Proc. of 8th National Conference on Artificial Intelligence/14th Conference on Innovative Applications of Artificial Intelligence*, no. 2, pp. 593–598.

[6] A. Angeli, D. Filliat, S. Doncieux, and J. A. Meyer, "Fast and incremental method for loop-closure detection using bags of visual words," *IEEE Transactions on Robotics*, vol. 24, no. 5, pp. 1027–1037, 2008.

[7] W. Zaremba, I. Sutskever, and O. Vinyals, "Recurrent Neural Network Regularization," *Icrl*, vol. 27, no. 2013, pp. 1–8, 2014.

[8] G. Dulac-Arnold, L. Denoyer, and P. Gallinari, "Text Classification: A Sequential Reading Approach," *Advances in Information Retrieval*, vol. 6611, no. 2, pp. 411–423, 2011.

[9] L. Denoyer and P. Gallinari, "Deep Sequential Neural Network," pp. 1–9, 2014.

[10] L. Mihaylova, T. Lefebvre, H. Bryunincks, and J. De Schutter, "A Comparison of Decision Making Criteria and Optimization Methods for Active Robotic Sensing." *5th International Conference, NMA*, pp. 316–324, 2003.

[11] G. N. DeSouza and A. C. Kak, "Vision for Mobile Robot Navigation : A Survey Keyfeatures," *Pattern Analysis and Machine Intelligence*, vol. 24, no. 2, pp. 237–267, 2002.

[12] A. Bernardino and J. Santos-Victor, "Visual behaviours for binocular tracking," *Proceedings Second EUROMICRO Workshop on Advanced Mobile Robots*, pp. 2–7, 1997.

[13] A. P. Duchon and W. H. Warren, "Robot navigation from a Gibsonian viewpoint," *EEE International Conference on Systems, Man, and Cybernetics. Humans, Information and Technology.*, pp. 2272–2277, 1994.

[14] K. Souhila and a. Karim, "Optical flow based robot obstacle avoidance," *International Journal of Advanced Robotic Systems*, no. 1, pp. 13–16.

[15] M. Guzel and R. Bicker, "Vision Based Obstacle Avoidance Techniques," *Recent Advances in Mobile Robotics*, pp. 83–108.

[16] B. Herissé, T. Hamel, R. Mahony, and F.-X. Russotto, "Landing a VTOL Unmanned Aerial Vehicle on a Moving Platform Using Optical Flow," *IEEE Transactions on Robotics*, vol. 28, no. 1, pp. 77–89, 2012.

[17] Y. Matsumoto, K. Sakai, M. Inaba, and H. Inoue, "View-based approach to robot navigation," in *Proceedings. 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000) (Cat. No.00CH37113)*, vol. 3, no. 5, 2000, pp. 1702–1708.

[18] P. Gaussier, C. Joulain, S. Zrehen, J. P. Banquet, and A. Revel, "Visual navigation in an open environment without map," *International Conference on Intelligent Robots and Systems*, pp. 545–550, 1997.

[19] H. Zhou and S. Shigeyuki, "Learning bayesian network structure from environment and sensor planning for mobile robot localization," *IEEE Conference on Multisensor Fusion and Integration for Intelligent Systems*, 2003.

[20] A. Graves, "Adaptive Computation Time for Recurrent Neural Networks," *arXiv*, pp. 1–19, 2016.

[21] R. S. Sutton, D. Mcallester, S. Singh, and Y. Mansour, "Policy Gradient Methods for Reinforcement Learning with Function Approximation," *Advances in Neural Information Processing Systems 12*, pp. 1057–1063, 1999.