International Journal of Semantic Computing
Vol. 12, No. 1 (2018) 109–127
©World Scientific Publishing Company
DOI: 10.1142/S1793351X18400068

World Scientific
www.worldscientific.com

Sequential Action Selection and Active Sensing for Budgeted Localization in Robot Navigation

Nassim Aklil* and Benoît Girard

Institut des Systemes Intelligents et de la Robotique — ISIR
UMR 7222 - CNRS — UPMC
Paris 75005, France
**aklil@isir.upmc.fr*

Ludovic Denoyer

Laboratoire Informatique de Paris 6 — LIP6
UMR 7606 - CNRS - UPMC
Paris 75005, France

Mehdi Khamassi

Institut des Systemes Intelligents et de la Robotique - ISIR
UMR 7222 - CNRS - UPMC
Paris 75005, France

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. Our approach enables to reduce the problem to a classification task under budget constraint. We apply this algorithm to a localization problem in a simulated environment. We compare it first to simple neural networks for the classification part and second to different techniques of policy selection. The results show that the model can effectively learn an efficient active sensing 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. We also show that with this algorithm the simulated robot can transfer the learned policy as well as knowledge about which budget gives the best performance/budget ratio in a given environment to other environments with similar properties. We finally test the algorithm with real navigation data acquired in an indoor environment with the PR2 robot. Altogether, these results suggest a promising framework for enabling budgeted localization in robots and avoiding to make robots relearn everything from scratch in each new environment.

Keywords: Budgeted learning; deep learning; policy gradient; robotic navigation.

1. 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, [3–5]), 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) —, they 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 usually works with multiple sensors (lasers, RGB or RGBD cameras, whiskers, etc.), to our knowledge no method currently exists to autonomously learn which sensor is sufficient to localize in tasks where a limited budget does not permit to use all sensors *ad libitum*. 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 or minimize a given objective 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– 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 2000s. However, these techniques are limited by the fact that the systems learn action sequences after having already learned the task specificities, which leads to learn the task twice (task representation first and actions as a second step). 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 and minimizing the cost of each choses action. This approach is close to the one we describe in this paper in the way that we try to minimize an error function associated to a learned action sequence by constraining the model with an explicit budget instead of punishing each chosen action. Nevertheless, the sequential budgeted learning framework offers the advantage of simultaneously learning task representations and action selection.

In this paper, we propose a model that makes a robot use as minimal 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 an explicit map. We moreover show in simulation that this approach allows the robot to transfer the learned policy as well as knowledge about which budget gives the best performance/budget ratio in a given environment to other environments with similar properties. Hence the robot can generalize learned active sensing policies to new but similar environments, and is only required to learn the new classification task (i.e. localization) specific to each environment.

The paper is organized as following: the next section presents and discusses related work. Then, we describe the model, the learning algorithm and the experimental setup. Section 4 presents numerical experiments in simulation. Then Sec. 5 presents an application of the model to real navigation data collected within an indoor environment with the PR2 robot. The paper finishes by discussing the results and concluding.

2. 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 navigates by using the velocity of the different images [12–16]. 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 ([17]).

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, [18] performs indoor route construction by comparing the current image with the training data set, simply calculating a distance between them. In [19] the robot creates a sequence of images by storing the motion associated to each image. [20] uses 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 previous proposals since, in the training phase, we do not manually extract informations that are specifically describing the images, but directly use the image as is (in the simulated case).

The machine learning field has developed new ways of analyzing data by building models that learn autonomously how to interpret, describe and treat them. The recent deep learning state-of-the-art has many promising results on how data can be 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 proposed a sequential architecture, where the model learns representations at each time step using sequentially provided data, but the available amount of data is unlimited. A budgeted version of this model was recently proposed in [21]. It is specified in both articles that data are given between each transformation step. A similar architecture has been presented in [9], where the authors used an explicit budget (that stops the data acquisition after a certain number of steps), however the model does not acquire data at each time step but rather treats it as a classical supervised classification task.

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

3. Model

3.1. *Principles*

We propose a model applied to a localization task, which 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 [22], 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.

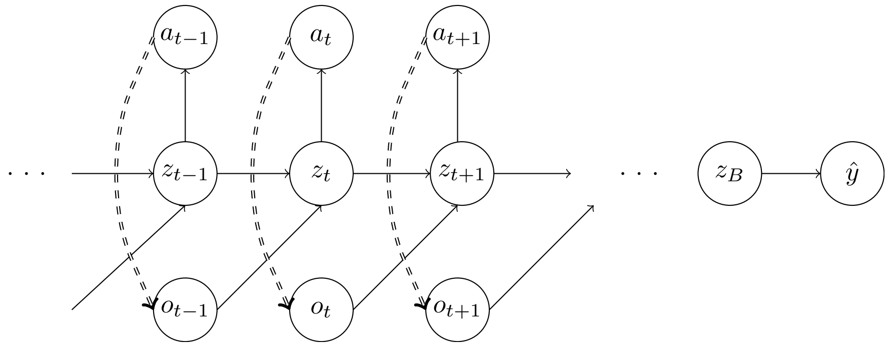

Fig. 1. Scheme of the model architecture. z_t is a latent state of the model. From z_t an action a_t is sampled. The sampled action returns an observation o_t that is used to compute the next z_{t+1} that is an aggregation of z_t and o_t. At the end of the given budget B and after computing the last z_B the model predicts the equivalent $\hat{y}$.

3.2. *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 $t = 0$, 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 from 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 $P(o_t|a_t, a_{t-1}, \ldots, a_1, x)$ which depends on the environment. We will denote $\pi(a_t|a_{t-1}, o_{t-1}, \ldots, a_1, o_1)$ the policy of the robot, i.e. the probability of choosing action a_t knowing the previously acquired information $o_{t-1}, \ldots, o_1$ and the previously chosen actions $a_{t-1}, \ldots, a_1$. The final decision function which will predict the robot position w.r.t acquired information will be denoted $f(a_t, o_t, a_{t-1}, o_{t-1}, \ldots, a_1, o_1, x)$ (Fig. 1).

3.3. *Learning algorithm*

Let us denote $(x_1, \ldots, x_m)$ 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.[a] 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

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

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

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

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

and its gradient can be written as

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

We can expand this gradient such as

$$\begin{aligned}
\nabla_{\gamma,\theta} L(\gamma, \theta) &= \int \nabla_{\gamma,\theta}(P(T|x))\Delta(f(x, T), y))P(x, y)\, dTdxdy & (5)\\
&\quad + \int P(T|x)\nabla_{\gamma,\theta}\Delta(f(x, T), y))P(x, y)\, dTdxdy & (6)\\
&= \int \frac{P(T|x)}{P(T|x)} \nabla_{\gamma,\theta}(P(T|x))\Delta(f(x, T), y))P(x, y)\, dTdxdy & (7)\\
&\quad + \int P(T|x)\nabla_{\gamma,\theta}\Delta(f(x, T), y))P(x, y)\, dTdxdy & (8)\\
&= \int P(T|x)\nabla_{\gamma,\theta}(logP(T|x))\Delta(f(x, T), y))P(x, y)\, dTdxdy & (9)\\
&\quad + \int P(T|x)\nabla_{\gamma,\theta}\Delta(f(x, T), y))P(x, y)\, dTdxdy. & (10)
\end{aligned}$$

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):

$$\nabla_{\pi,f} L(\pi, f) \approx \frac{1}{n}\sum_{i=1}^{n} \left[\frac{1}{M}\sum_{k=1}^{M} \nabla_\pi(\log P(T|x_i))\Delta(f(x_i, T), y_i) + \nabla_f \Delta(f(x_i, T), y_i)\right]. \tag{11}$$

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

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

$$\nabla \log P(T|x_i) = \nabla \sum_{t=1}^{B} \log P(a_t|a_{t-1}, o_{t-1}, \ldots, a_1, o_1). \tag{12}$$

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 $b = E_{x,T,y}[\Delta(f(x, T), y)]$ that can be estimated from the training set.

3.4. *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

$$h_a(z_t, o_t) = \tanh(W_z z_t + W_a o_t), \tag{13}$$

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

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

Algorithm 1 Inference

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

[b]The source code in Torch7 can be found at: https://github.com/nassim5/torch_ws/tree/master/recurrentSequentialActionSelection.

4. Numerical Experiments

We tested the capacity of this model in two different setups: the first in a simplified simulated environment, and the second on data recorded with a PR2 robot navigating indoor within a local experimental room. We ran three different experiments in total: (1) the first experiment in simulation serves as proof-of-concept that the algorithm enables an agent to autonomously learn a policy (e.g. turn, check camera information, check laser information) within a fixed budget (that limits the number of allowed actions) while providing the agent with sufficient information to localize itself within the environment. Preliminary results of this first experiment have been presented at the IEEE Robotic Computing 2017 Conference [23]. The second experiment is performed again in simulation in order to test the ability of the algorithm to transfer acquired knowledge from an environment to another. The idea is that the algorithm can learn a general active sensing policy for different families of environments (e.g. large versus narrow environment, or environments with a small versus large number of obstacles) and then reuse this policy for other environments from the same family while only learning the classification (i.e. localization) task specifically in each new environment. Finally, the third experiment is performed with real data collected with the PR2 robot in order to test offline the ability of the algorithm to autonomously learn to neglect unreliable sensors and to exploit reliable ones in a given environment.

4.1. *Simulated case*

We first validate the algorithm in a set of experiments using a simulated 2D environment. This environment corresponds to a 50×50 grid where each position can be empty or occupied by a colored wall. We consider an agent moving within 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 (i.e. to localize itself within the environment) using a limited number B of actions. Hence, this corresponds to an active sensing task where an explicit budget is taken into account for the classification.

An example of such a maze is given in Fig. 2. 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 receives 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 phase, 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 $\{0, \pi/2, \pi, 3\pi/2\}$.

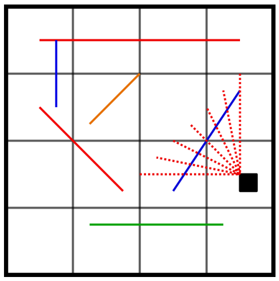

Fig. 2. Simulated data. The black square represents the robot. The dotted lines represent the range of the camera. The plain colored lines represent randomly placed walls. The horizontal and vertical lines represents the spatial discretization.

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

- The image classification baseline corresponds to a classical classification model (i.e. multilayer perceptron) where $1, 2, 3$ or 4 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 the case where the agent can freely choose which action to apply, and where our model learns the optimal active sensing policy in parallel to learning the classification task for the tested environment.

We explored the values of the different parameters with a grid search paradigm:

- The learning rate for both the policy learning and the representation learning: $\{0.0001, 0.001, 0.01, 0.1\}$
- Size of latent space z: $\{10, 50, 100\}$
- The sensor resolution: $\{10, 30, 50\}$

For the *Forced policy* and the *Learned policy*, we have explored two variants, i.e. *recurrent* and *nonrecurrent*. In the first case, the h_a function is reused at every time step while in the second case, one uses an 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 1 have been averaged over 5 different runs.

First, the quality of the classification model improves when the number of acquired images increases. This confirms that providing more information to the agent helps him to compute a better localization. Moreover, when using the *Forced policy*, the model is able to achieve 75.2% when collecting 3 images ($B = 5$) and thus to increase its performance by 16.4 points (corresponding to a 28% increase) with

Table 1. Results for simulated experiment. The results represent the performance on test set (in percentage).

Data acquisition		Budget 1	3	5	7
Image classification		44.4	52.8	56.7	58.1
Forced policy	Recurrent	58.8	70.2	75.2	76.1
	Nonrecurrent	55.8	60.8	61.2	65.3
Free policy	Recurrent	43.6	71.6	75.2	79.2
	Nonrecurrent	46.8	69.4	70.4	73.2

respect to using only 1 image. The *Learned policy* model is able to achieve a 75.2% 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. We also note that the model improves by 31.6 points (a 72% increase) between $B = 1$ and $B = 5$ and only improves it by 4 points from $B = 5$ to $B = 7$ (a 5% increase). 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 *nonrecurrent* versions, allowing a better generalization. In Table 2, we see that for $B = 1$ and $B = 3$ the model reaches the best policy 5 times out of 5 simulated runs. In the case of $B = 1$, the model always asks for an image in order to examine the surroundings. In the case

Table 2. Individual performances in a Learned Policy case (in percentage) with the recurrent model. The table is classified from best performance to worst in respect to each budget with the equivalent learned policy.

Budget	Run	Policy	Performance
1	1	Image	46.4
	2	Image	45.3
	3	Image	43.3
	4	Image	42.1
	5	Image	41.3
3	1	Image-Right Turn-Image	75.2
	2	Image-Right Turn-Image	73.8
	3	Image-Right Turn-Image	70.5
	4	Image-Left Turn-Image	69.8
	5	Image-Right Turn-Image	68.8
5	1	Image-Left Turn-Image-Left Turn-Image	78.8
	2	Image-Left Turn-Image-Left Turn-Image	77.2
	3	Left Turn-Image-Right Turn-Right Turn-Image	74.8
	4	Image-Right Turn-Image-Image-Image	73.7
	5	Image-Image-Image-Right Turn- Image	71.6
7	1	Right Turn-Image-Right Turn-Image-Right Turn-Image-Image	80.7
	2	Image-Image-Left Turn-Left Turn-Image-Left Turn-Image	80.6
	3	Image-Image-Right Turn-Image-Image-Right Turn-Image	79.7
	4	Image-Image-Left Turn-Image-Left Turn-Image-Image	79.4
	5	Image-Image-Left Turn-Image-Left Turn-Image-Image	75.5

of $B = 3$, the agent learns to alternate between the data acquisition and the exploration actions. However, in the case of $B = 5$ and $B = 7$, we see that the agent tends to take redundant actions by repeating the data acquisition step (without moving, thus acquiring twice the same image) and/or actions related to exploration (not acquiring images after movements).

This first set of results shows that the model can learn an efficient policy for localization provided that a sufficient budget is available. Moreover, tests with increasing budgets show that the model tends to converge to a maximum performance of classification in a given environment, as increasing from $B = 5$ to $B = 7$ does not improve it much. This suggests that finding a compromise between the performance and the budget may be useful in order to find an optimal budget by environment. Furthermore, the model could be useful for real-world robotics if it could learn a policy and an optimal budget transferable to other environments with similar properties (e.g. similar size, similar number of obstacles), rather than having to exhaustively relearn a policy under different budgets in each new environment xperienced by a robot. This is what we explore in the next section.

4.2. *Transfer learning in the simulated case*

In the previous experiments, a virtual agent controlled with our algorithm had to learn both an efficient active sensing policy (e.g. acquire image with camera, then turn left, then acquire another image with camera, etc.) and a classification task (i.e. localize itself) in a simulated environment with obstacles. Moreover, we have tested this process exhaustively for different budgets. However, a more efficient and generalizable process would consist in having the agent learn a single general active sensing policy for a whole family of environments with similar properties (i.e. similar sizes, similar number of obstacles), and only learn the specific classification task for each environment. Moreover, our approach could be promising for robotics if we could automatically learn the optimal budget — optimal in the sense of maximizing a certain cost-benefit function — for a family of environments without having to exhaustively explore all possible budgets when experiencing a new environment from the same family.

In order to test the ability of the model to learn an effective and general active sensing policy, we developed here an experimental protocol where an agent learns a policy in an environment A, then transfers the learned policy in an environment A' of same family, i.e. same size than A and same number of obstacles however rearranged. The model learns the policy and the classification task in the environment A but only the classification task in the environment A', while re-using the policy learned in A.

We created four different mazes in order to test this protocol: A and A' of size 50×50 px, B and B' of size 20×20 px (see Fig. 3). A and B are the "Learning environments", A' and B' are the "Testing environments". The actual protocol is equivalent to applying a "free policy" learning in the learning environments and a

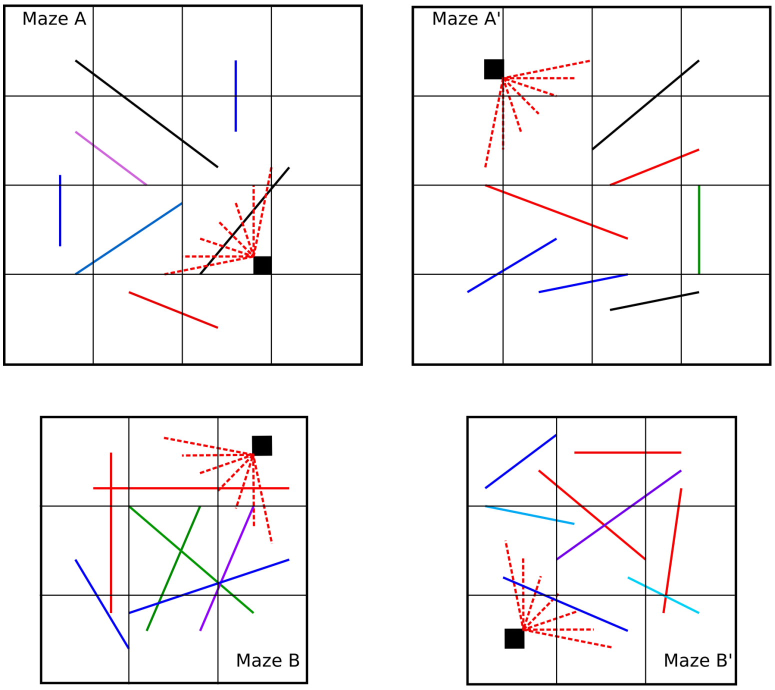

Fig. 3. Example of different mazes used for the transfer learning task. Maze A and Maze B are the learning environments and Maze A′ and Maze B′ are the testing environments.

"forced policy" learning in the testing environments, with the best policy learned in the previous environments.

Environment A is more difficult to learn than B (compare the first two rows of Table 3), the policies learned in the learning environments give good results, when applied in the testing ones (Table 3, third and fifth rows). This suggests that the model learns policies that are very efficient and can be transferred to other environments without re-learning the policies all over again at each environment. Moreover, we tested the ability of the learned policy to be validated in

Table 3. Transfer learning performances (in percentages) between environments of same or different nature.

	Budget			
Type env	1	3	5	7
Train env (A)	46.4	75.2	78.8	80.7
Train env (B)	81.5	89.5	93.6	96.3
Train env (A) $\rightarrow$ Test env (A')	51.6	56	69.1	75
Train env (B) $\rightarrow$ Test env (A')	56.4	57.5	61.7	66.6
Train env (B) $\rightarrow$ Test env (B')	72.2	75.7	90.7	96.4
Train env (A) $\rightarrow$ Test env (B')	74.6	81.4	89.2	95.2

cross-environments, i.e. that the policies achieved in the learning environment A and B have been respectively transferred to the testing environments B' and A'. When the policy learned from the environment A (50×50 px) is transferred to B' (20×20px), the performances are similar to those obtained after training in B, which is not the case in the reverse situation, where performance at high budgets (5 or 7) are clearly lower (Table 3, fourth and sixth rows). This is due to the fact that in environments more difficult to learn, like A, the model is forced to learn more efficient policies, which is not the case in simpler environments B, where the model learns redundant policies that are efficient enough. This suggests that having a high budget in a simple environment may lead to policies that do not bring significantly more information than policies corresponding to smaller budgets. Thus under strong budget constraint, one may want to find the smallest possible budget bringing enough information for classification. We will define this as an *optimal budget* hereafter, in the sense that such a budget maximizes a certain cost-benefit function.

In order to quantify how much each tested increase in budget contributed to a more or less important improvement of the classification performance, we define a function RG which computes the relative gain in performance from using a budget $B = i$ compared to the preceding one ($B = i - 2$) according to

$$\mathrm{RG}(i) = \frac{(\mathrm{Per}\, f(B=i) - \mathrm{Per}\, f(B=i-2))}{(1 - \mathrm{Per}\, f(B=i-2))}. \tag{14}$$

While the RG function provides us with a quantitative evaluation of the relative efficiency of learning under different tested budgets, it does not explicitly penalize high budgets. Alternatively, one may prefer to divide the performance by a certain function of the budget, so that high budgets are evaluated as relatively less advantageous than low budgets, unless they really lead to a strong improvement in classification performance.

The choice of a function which penalizes the model as a function of the budget is somehow arbitrary. Ideally, we would like a function as general as possible, so that the human experimenter remains free to either accept high budgets for a given performance x if the priority is put on maximizing performance, or to reject high budgets for the same performance x if the priority is put on limiting the budget. As proof-of-concept, we thus define the following simple cost-benefit function CB which computes a compromise between the performance and the budget:

$$\mathrm{CB}(i) = \frac{\mathrm{Per}\, f(B=i)}{\sqrt[n]{(B=i)}}, \tag{15}$$

where n is a parameter ($n \in \mathbb{N}^+$) determining the budget pressure. The higher n, the more one is ready to accept a higher budget for a small improvement of performance.

Figure 4 illustrates the results obtained when applying these two criteria with a budget pressure parameter $n = 3$ on the data simulated in the training environments

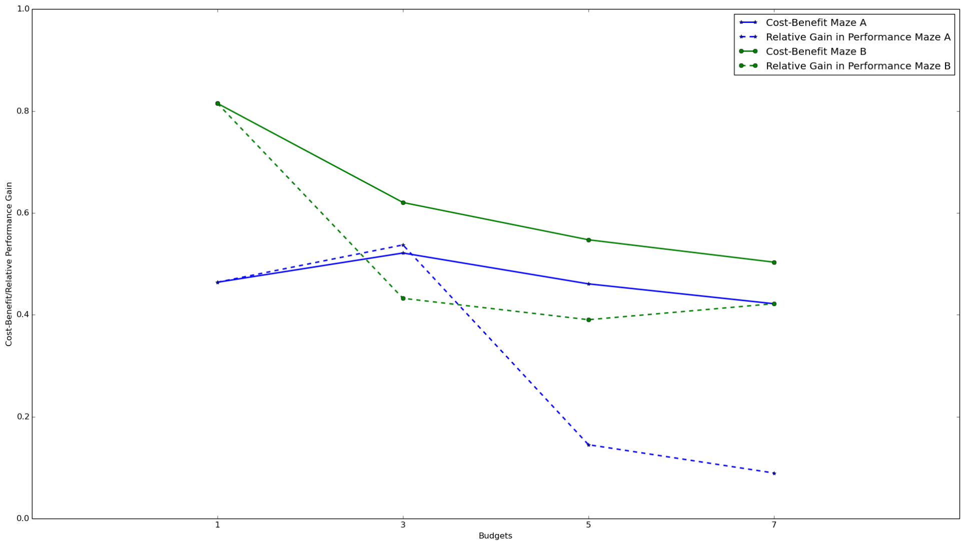

Fig. 4. Cost-Benefit/Relative performance gain: blue lines show the cost-benefit/relative performance gain in maze A and green lines in maze B. Thick lines are the Cost-benefit (15) and dotted lines are the relative performance gain (14).

A and B: the best cost-benefit ratio and relative gain both correspond to a policy with a budget of 3 in large environments (A) and with a budget of 1 in smaller (B). This can be used as a criterion to decide *a priori* which budget is the most useful in environments of same nature without re-learning the task from the beginning.

To summarize, in the second simulation experiment we have studied how different types of information could be transferred from one environment to another. We have found that an active sensing policy learned in an environment from a size family (A or B) could be transferred to another environment from the same size family (here A' and B', respectively). Moreover, we found that a policy learned in a large environment could be successfully transferred to a smaller one. Finally, we found that different families of environments have a different optimal budget which maximizes some cost-benefit function. These properties would enable a navigating robot to save time by re-using previously learned active policies and optimal budgets when experiencing new environments recognized as belonging to the same family.

4.3. *Real data case*

The experiment to test the model described above has been extended to a real world data case, where the robot tries to localize itself within a real environment. The protocol is very similar to the one described in the simulated case.

A PR2 robot (Fig. 5) has been used for data acquisition. We used the wide stereo camera available in order to extract the images that will feed the model with information about the environment.

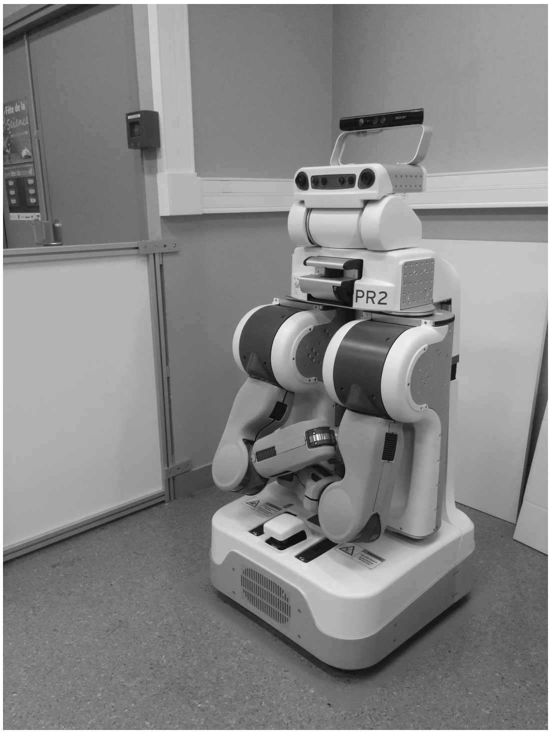

Fig. 5. The PR2 robot, used for data extraction in a real-world indoor environment.

The problem being kept as a classification one, we needed to create classes for the model to learn from. For that purpose, the environment has been divided into a fixed number of 16 "cells". Each cell is a square of $4\,\mathrm{m}^2$ of surface ($2\,\mathrm{m} \times 2\,\mathrm{m}$). Within each cell we randomly selected 40 coordinates, from which data was acquired. At each coordinate, we acquired a whole panorama of images with the camera (an image every 45° for a 360° panorama). This leads us to a database of $16 \times 40 \times 8$ images which makes a total of 5120 images where each image is associated to a given cell (i.e. class), a set of coordinates and an orientation (example images on Fig. 6).

We ran a grid search algorithm on the same parameters that have been listed in Sec. 4.1 for the learning rates and the size of the latent space z.

The extracted images have been preprocessed with an already learned Convolutional Neural Network (CNN) developed in [24] called OverFeat, which provides image representations as vectors of 4096 dimensions. This preprocessing is used to generate meaningful representations that will be used as inputs for the model.

We compared the three different data acquisition methods as done in the simulated case (Sec. 4.1), i.e. concatenation of images, forced policy and free policy.

The conclusions that have been drawn from the simulated protocol are still valid in this case (Table 4). The bigger the budget, the better the performance. However,

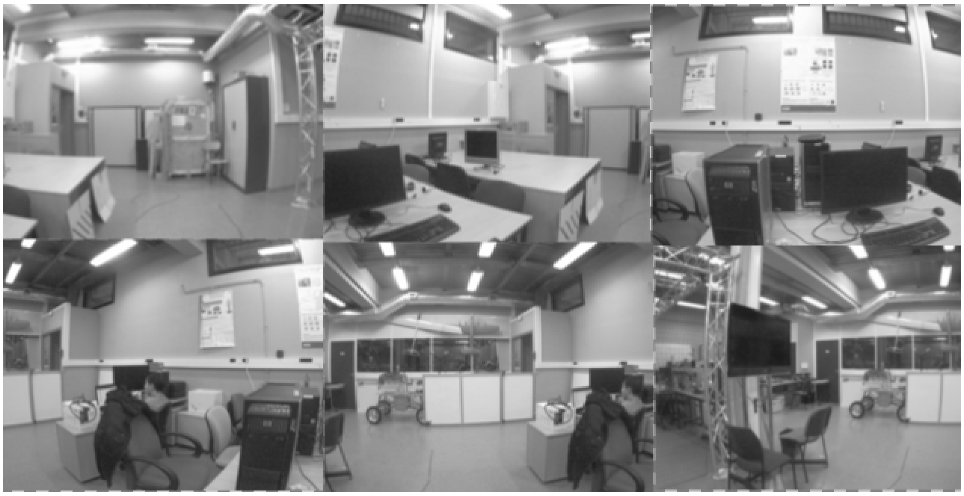

Fig. 6. Examples of data extracted with the wide stereo camera of the PR2 robot.

the average performance is not as satisfying as the performances observed in the simulated case: this is due to the fact that the model takes time to converge to the policies that alternate between data acquisition and exploration actions. This is due to several factors: the first one is the high dimensionality of the data, where an image is represented with a vector of 4096 features. This problem of dimensionality affects the convergence of the model. For example, in the case of $B = 3$, the model converges sometimes to a policy of { Image, Image, Image }, which is due to the reach of a local minimum that is hard to overcome with data of such large dimensions and a sample of data as small as 2560 training positions. Second, we faced a problem of over fitting, where in the training phase we observe high scores that can reach 99%. This is either due to the lack of data or to the fact that the image representations obtained with the OverFeat CNN are not accurate enough for these data. However, in Table 4 the performances in the free policy protocol are as high as the ones obtained in the forced policy case which reflects that the model tends to learn efficient policies, alternating between data acquisition and exploratory actions, equivalent to the ones forced by

Table 4. Results for real case experiment. The results represent the performance on test set (in percentage).

		Budget		
Data acquisition		1	3	5
Image classification		49	52.2	53
Forced policy	Recurrent	42	43	44.2
Free policy	Recurrent	43	46	47

the experimenter. The obtained results reflects the noncapacity of the model to generalize on these specific data, not its capacity at learning efficient policies.

In the following discussion, we will sketch a series of possible extensions of the model that could be attempted in future work to improve its performance with real data.

5. Conclusion and Discussion

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 used in the machine learning field, where information is gathered at each time step and in the robotic field, where the data are acquired without limitations. We have proposed a set of experiments in a simulated case showing the interest of this approach, and preliminary results in a real case, that require further investigation.

We also developed an experimental protocol that shows in simulation the capacity of the model to learn active sensing policies that can be generalized to environments with similar properties (e.g. size, number of obstacles). One particular interest of this approach for robotics is that it could enable robots to avoid relearning everything from scratch in each new encountered environment. Instead, the robot can re-use previously learned active sensing policies for localization which turned out to be efficient in environments with similar properties than the new environment.

Moreover we have tested explicit cost/benefit functions that can be used *a priori* as criteria to choose which budget is the most adequate to which environment in order to balance between the used budget and the gain in performance that each budget brings to the model. These functions constitute a simple way for the robot to generalize knowledge about which budget is sufficient in each type or family of environments, so that it does not need to re-explore exhaustively all possible budgets in each new environment. Although these functions are somehow arbitrarily defined, we have tried to make them as general as possible so that the human experimenter can parametrize the cost-benefit function depending on whether he is short on budget or not. Besides, both functions casted a different *optimal* budget for the two tested families of environments A and B, which suggests that these different environments can be robustly distinguished by different budget functions.

Future research for this model aims at testing the capacity of the transfer protocol to different navigation environments to which human experimenters classically assign different labels (e.g. open space, corridor, junction, indoor versus outdoor). Another future investigation would be the improvement of the real dataset: as seen here, the performances of the real data case are less conclusive than the simulated one. As discussed in Sec. 4.3, this is either due to the lack of data or to the fact that OverFeat does not return meaningful representations of the images. It would be useful to train a new CNN that learns new representations only on the images extracted from the robot by defining different sizes of the representation vector and to augment the dataset by dividing the images into patches. Another possible way of improving the performance with the real data could be to dynamically allocate the budget, so that the robot starts learning with a high budget and progressively tries to reduce the budget and see whether a smaller budget is acceptable or whether it dramatically impairs the performance. This somehow would be equivalent to having the robot learn when to stop acquiring new data and provide a classification. One potential advantage of this would be that an initialized high budget could allow a richer policy that the robot would later reduce, avoiding the robot from being stuck in local optima when starting from a small budget and trying to extend the policy, as we found in some cases.

Acknowledgments

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. Robot. Autom. Mag.* 13(2) (2006) 99–108.

[2] S. Chen, Y. Li and N. M. Kwok, Active vision in robotic systems: A survey of recent developments, *Int. J. Robotics Research* **30** (2011) 1343–1377.

[3] P. Moutarlier and R. Chatila, An experimental system for incremental environment modelling by an autonomous mobile robot, *Exp. Robot. I*, **1560** (1989) 327–346.

[4] R. C. Smith and P. Cheeseman, On the representation and estimation of spatial uncertainty, *Int. J. Robot. Res.* **30**(9) (1986) 2907–2926.

[5] M. Montemerlo, S. Thrun, D. Koller and B. Wegbreit, FastSLAM: A factored solution to the simultaneous localization and mapping problem, in *Proc. 8th National Conf. on Artificial Intelligence/14th Conf. on Innovative Applications of Artificial Intelligence*, Vol. 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 Trans. Robot.* **24**(5) (2008) 1027–1037.

[7] W. Zaremba, I. Sutskever and O. Vinyals, Recurrent neural network regularization, *ICLR*, 2015.

[8] G. Dulac-Arnold, L. Denoyer and P. Gallinari, Text classification: A sequential reading approach, *Adv. Inf. Retr.* **6611**(2) (2011) 411–423.

[9] L. Denoyer and P. Gallinari, Deep sequential neural network, *CoRR*, 2014, pp. 1–9.

[10] L. Mihaylova, T. Lefebvre, H. Bryuninckx and J. De Schutter, A comparison of decision making criteria and optimization methods for active robotic sensing, in *5th Int. Conf. NMA*, (2003) pp. 316–324.

[11] G. N. DeSouza and A. C. Kak, Vision for mobile robot navigation: A survey keyfeatures, *Pattern Analy. Mach. Intell.* **24**(2) (2002) 237–267.

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

[13] A. P. Duchon and W. H. Warren, Robot navigation from a Gibsonian viewpoint, in *IEEE Int. Conf. Systems, Man, and Cybernetics. Humans, Information and Technology*, 1994, pp. 2272–2277.

[14] K. Souhila and A. Karim, Optical flow based robot obstacle avoidance, *Int. J. Adv. Robot. Syst.* **1** (2007) 13–16.

[15] M. S. Guzel and R. Bicker, Vision based obstacle avoidance techniques, in *Recent Advances in Mobile Robotics, InTech*, 2011, 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 Trans. Robot.* **28**(1) (2012) 77–89.

[17] E. Royer, M. Lhuillier, M. Dhome and J.-M. Lavest, Monocular vision for mobile robot localization and autonomous navigation, *Int. J. Comput. Vis.* **74**(3) (2007) 237–260.

[18] Y. Matsumoto, K. Sakai, M. Inaba and H. Inoue, View-based approach to robot navigation, in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Vol. 3, 2000, pp. 1702–1708.

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

[20] H. Zhou and S. Shigeyuki, Learning bayesian network structure from environment and sensor planning for mobile robot localization, in *IEEE Conf. Multisensor Fusion and Integration for Intelligent Systems*, 2003, pp. 76–81.

[21] A. Graves, Adaptive Computation Time for Recurrent Neural Networks, *CoRR*, 2016.

[22] R. S. Sutton, D. Mcallester, S. Singh and Y. Mansour, Policy gradient methods for reinforcement learning with function approximation, *Adv. Neural Inf. Process. Syst.*, **12** (1999) 1057–1063.

[23] N. Aklil, B. Girard, M. Khamassi and L. Denoyer, Sequential action selection for budgeted localization in robots, in *IEEE Int. Conf. Robotic Computing*, 2017, pp. 97–100.

[24] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus and Y. LeCun, OverFeat: Integrated recognition, localization and detection using convolutional networks. arXiv: 1312.6229.