On the Use of Intrinsic Motivation for Visual Saliency Learning

Céline Craye, David Filliat, Jean-François Goudou

On the Use of Intrinsic Motivation for Visual Saliency Learning

Céline Craye*†, David Filliat* and Jean-François Goudou†
*ENSTA Paristech - INRIA FLOWERS team,
Unité informatique et Ingénierie des Systèmes
828 boulevard des Maréchaux, 91762 Palaiseau, France
†Thales - SIX - Theresis - Vision & Sensing
1, avenue Augustin Fresnel, 91767 Palaiseau, France
Email: celine.craye@thalesgroup.com, celine.craye@ensta-paristech.fr

***Abstract*—The use of intrinsic motivation for the task of learning sensori-motor properties has received a lot of attention over the last few years, but only little work has been provided toward using intrinsic motivation for the task of learning visual signals. In this paper, we propose to apply the main ideas of the *Intelligent Adaptive Curiosity* (IAC) for the task of visual saliency learning. We here present RL-IAC, an adapted version of IAC that uses reinforcement learning to deal with time consuming displacements while actively learning saliency based on local learning progress. We also introduce the use of a *backward evaluation* to deal with a learner that is shared between several regions. We demonstrate the good performance of RL-IAC compared to other exploration techniques, and we discuss the performance of other intrinsic motivation sources instead of learning progress in our problem.**

I. Introduction

In the context of developmental learning, robots need to explore their environment methodically to gain knowledge in a limited amount of time. In that regard, intrinsic motivation (i.e. behavior driven by a reward system that is not related to an external goal, but to the acquisition of competences or knowledge) is an efficient drive for exploration. In this paper, we address the problem of a mobile robot trying to learn visual saliency of its environment from RGB-D images. The robot is guided by intrinsic motivation, while dealing with the cost of moving across the environment.

In robotics, intrinsic motivation has proven very efficient in numerous applications. For example, Huang *et al.* [12] have used *novelty* to guide visual exploration, while Chentanez *et al.* [6] have used the error of prediction of *salient event* to speed up a classical reinforcement learning approach. To overcome limitations related to novelty or error in unlearnable situations, intrinsic motivation based on *progress* has been proposed [2], [20], [21]. In these approaches, the use of progress as the main source of intrinsic motivation makes the robot focus on cases that are neither too easy nor too hard, so that progresses are constantly made and no time is wasted in unlearnable situations. Those approach nevertheless rely on an empirical local evaluation of the progress, which may reveal difficult to obtain in high dimensional spaces such as visual signals. The progress is then evaluated on higher level tasks such as object recognition [19] or learning abstraction libraries [14]. Progress has also been exploited in a reinforcement learning context, to provide a robust exploration, flexible to non stationary environment or wrong assumptions [16].

Visual exploration of the environment by mobile robots is generally associated with a way to localize areas of interest on which the robot should focus on. This localization mechanism is typically driven by visual saliency maps [10], [18] or, if depth information is available, geometrical segmentation [1], [5]. In the first case, saliency maps [13], [23] highlight in the input image areas that are 'interesting' in their context. 'Interestingness' covers a broad variety of cases and can either be related to a specific task [10] or based on how much a stimulus differs from its neighborhood [13]. In the second case, indoor object segmentation based on depth information usually relies on finding planar surfaces and objects lying on them. Those methods can accurately detect objects on tables or floor, but are limited by the sensor quality and geometrical constraints (size or distance to the objects). So far, saliency maps are mostly used as a black box and are not learned (although sometimes refined) directly during the exploration of a particular environment. Yet, learning visual attention directly on the robots would make sense in a developmental perspective. It was found that human visual attention significantly varies at different ages [11], but also between cultures [3]. This means that visual saliency is learned and modulated by our environment.

In a previous work [8], we described a method able to learn online a model of visual saliency. We demonstrated its efficiency compared to state-of-the-art techniques. We also presented some preliminary results of an exploration mechanism based on the *Intelligent Adaptive Curiosity* (IAC) [20]. A that time, we successfully applied IAC in a semi-simulated setup without considering the time spent by the robot in displacements. Based on these first results, we here propose RL-IAC (for *Reinforcement Learning-IAC*) that provides a way to get a trade-off between the time lost in displacements across the environment, and the acquisition of relevant visual samples allowing a faster and better learning. We also propose a *backward evaluation* of the progress to overcome biases introduced by a learner shared between several regions.

II. Saliency learning

In [8], we presented an algorithm able to incrementally learn and generate saliency maps for the task of detecting objects in indoor environments. The method is designed to work with an RGB-D sensor and uses the depth map as a learning signal, a feature extractor on the RGB image, and a classifier that predicts and generates saliency maps. More precisely, the algorithm uses the procedure depicted in Figure 1 each time an RGB-D input is acquired:

The RGB image is sent to a feature extractor that produces feature maps. A set of 39 features is extracted for each pixel of the image and encodes the state of the pixel in its neighborhood. A more detailed description of the feature extraction can be found in [7].

On the other hand, the depth map is processed in order to produce a segmentation mask (See the method in [5]). The segmentation algorithm detects planar surfaces and highlights elements of the scene that are lying on those surfaces. Elements highlighted this way are considered as being salient. This algorithm accurately detects salient objects (high precision), but it also misses a lot of them (low recall). Because the depth map is noisy, some pixels are not associated with a depth value. Similarly, some elements of the depth map could not be formally identified as salient. The segmentation then generates a mask with three states (see colors in Figure 1): *salient* (white) for elements that were clearly detected as salient, *not salient* (black) for the main plane, walls, and elements that are for sure not salient, and *unknown* (gray) for the others.

Last, a classifier is constantly updated based on the feature map and the segmentation mask to learn a model of saliency. For each pixel labelled *salient* or *not salient*, we associate the corresponding 39 dimension feature vector and send it to the classifier to update the model. The classifier is a modified version of a Random Forest, adapted to be re-trained online. For more details, please refer to [8].

As new RGB-D images are acquired and processed, the model of saliency is updated and improved. To generate a saliency map, we extract the RGB features from a given image, and we send them to the classifier that produces a score between 0 and 1 estimating how likely to be salient a pixel is (See the saliency result in Figure 1).

To sum up with, segmentation provides on the one hand a partial but accurate estimation of the saliency in an image: because of depth missing data (shadows, far distance) or because the algorithm could not clearly determine the saliency of an object, some pixels are not associated with a *salient* or *not salient* label. On the other hand, saliency maps are estimated based on the RGB image only, and the saliency is obtained for each pixel of the image. It is therefore able to generalize from the partial information provided by the segmentation mask. A few samples provided in Figure 2 illustrate this generalization capability.

This method of incremental learning can be coupled with an exploration strategy so that input samples are actively selected. This way, learning should be faster and of better quality. Based on the evolution of the learning performance of the saliency, we can compare the efficiency of the exploration strategies.

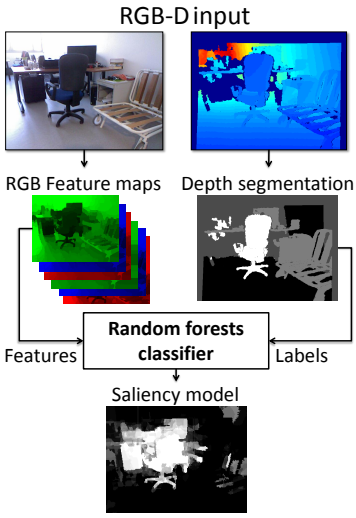


Fig. 1. Flowchart of the saliency learning approach

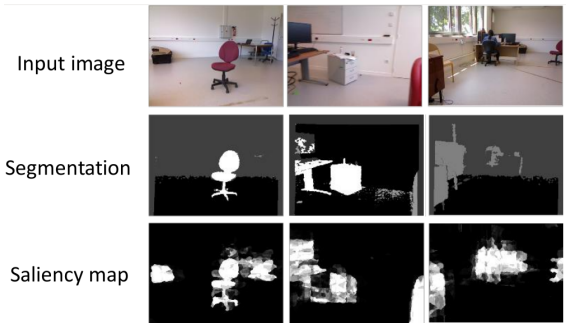


Fig. 2. Generalization capability of the saliency learning approach: the segmentation contains gray pixels that correspond to missing data, either because the surface is reflective (computer screen), too far, or because the algorithm could not determine the state of the pixel. On the other hand, the saliency map produces an estimate on the whole image and detects objects that segmentation could not find.

III. From IAC to RL-IAC

To better understand the similarity and differences of our approach with traditional IAC application, let us take a practical case in which IAC has been successfully applied. Take a robotic arm and a camera and processing unit able to localize the position of the hand in the camera frame. Suppose that IAC is used to learn to predict the position of the hand in the camera given a motor command of the joints of the arm. In other words, after a training phase, the outcome of the algorithm would be a forward model $X \rightarrow Y$ between joint motor commands of the arm X and the 2D position of the hand in the camera frame Y.

IAC is composed with 3 main components:

- a learner that learns a mapping $X \rightarrow Y$;
- a way to divide the exploration into *regions*;
- a meta-learner that monitors the learning evolution in each *region* and estimates progresses.

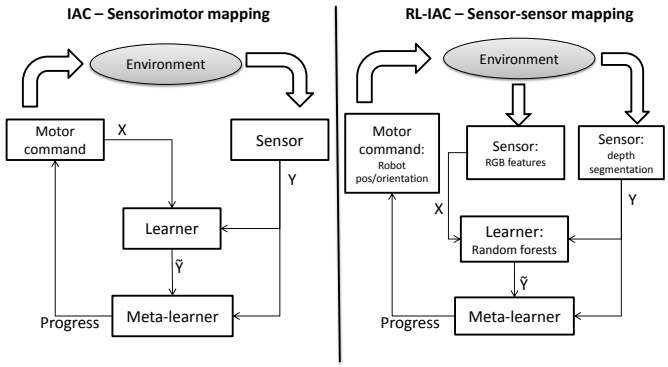


Fig. 3. IAC vs RL-IAC. Unlike IAC, our approach uses motor commands to learn a mapping between two sensor inputs. The motor is therefore an indirect way to provide new samples.

The notion of *region* is an essential element of the algorithm. The core idea of IAC is to get a local estimation of the progress in order to select actions that are likely to improve the model. In the first versions of IAC [20], this local estimation was possible by creating regions in the X space, by keeping a history of the samples received in each region, and by training a dedicated meta-learner in each region to predict the error that the learner was likely to make in this region. Regions were created and splitted incrementally as new samples were obtained, based on the local density of samples.

To apply IAC in a classical case, the following procedure is used: The robot takes an action (moving its arm with the input motor command X) in a given region R_i. It receives a sensor feedback about the consequence of the action (the 2D position Y of the hand of the robot in the frame of a camera), while the learner tries to predict the sensor feedback $\widetilde{Y}$ based on the X input and previously seen samples. The learner is then updated based on the (X, Y) sample, whereas the prediction error $\|Y - \widetilde{Y}\|$ is added to the error history of the meta-learner region R_i. Last, the progress in region R_i is re-evaluated based on the error history over the last few samples. The next action to be taken from the robot is then randomly chosen in the region that has the highest learning progress.

This procedure can be directly applied for the learning of a visual saliency model, but a special care must be considered for a few aspects, discussed in the following sections.

A. Role of the motor commands

The mapping learned in the original IAC scenario is between a motor command and a sensor input. In our case, IAC would be used to learn to predict the saliency of a pixel given the corresponding RGB feature vector. The mapping $X \rightarrow Y$ would then be between a feature vector X extracted from an RGB image, and the state of saliency (*salient* or *not salient*) of this pixel, the learner being the random forest classifier. As a result, our saliency algorithm learns a mapping between a sensor input X (the RGB-features in Figure 3) and an other sensor input Y (the depth segmentation in Figure 3). Motor commands in our case are used to move the robot across the environment to a new position and orientation. We do not use motor command directly as new samples for our learner, but as a way to receive new incoming samples (from the RGB-D input), by showing a different point of view of the environment. Once learning is finished, the model is then completely independent from any motor command.

B. Regions definition

In the first versions of IAC [20], the regions are obtained by splitting the input (X) space of the mapping which is also the motor space (See Figure 3). In our case, the X space of the mapping is the 39 dimensions space of the RGB-features which is not well-suited to define regions. Indeed, to get a specific input signal X, the robot has to move to a point of view containing this signal. However, the correspondence between points of view and visual input is not known, thus making this space inappropriate to select samples. In more recent implementation of IAC [2], [19], the algorithm considers regions in a space of *problems* that are accessible to the robot. We follow that idea by considering the space of positions and orientations accessible by the robot. In this case, a problem would correspond to learn saliency within a specific portion of the environment. A region is then a subset of positions (x, y, θ) that the robot can reach. Thus, the robot is in region R_i at time t if its current position $(x(t), y(t), \theta(t)) \in R_i$. An action in this space is a displacement of the robot to a given position (x', y', θ'). In our current implementation, the geometry of the robot's environment is supposed to be known, and the regions are defined arbitrarily and manually, prior to the experiment.

C. Behavior in dynamic environments

IAC is often criticized as not being adapted to dynamic environments. As our model is designed with regions that are manually designed in an *a priori* known environment, the implementation might look even more restrictive. However, the term environment might refer to the space in which regions are defined, or the space where the learner is trying to make predictions (which are actually the same in the first implementation of IAC). In our case, regions are defined from a map of the place where the robot is (a building, for example). In most cases, this map is not likely to change significantly in time (objects in the building may move, but walls probably won't), thus making the regions defined stable in time. On the other hand, visual signals (the space of the learner in our case) are much more likely to change in time, because salient objects are moved or illumination is changed for example. As explained earlier, the visual model is shared between regions and completely independent from any position in the map. If a salient object is moved somewhere else in the room, it will still look roughly the same in the new position, and is therefore likely to be estimated salient by the learner, whatever its new position. In that regard, our version of the algorithm is compatible with dynamic environments.

D. Shared learner and backward progress estimation

Saliency shows common properties in different areas of the environment (for example, walls are often uniform and white),

it is then more interesting to have a common learner shared by different regions. The drawback is that taking samples in a region may influence the learning in other regions. To avoid catastrophic forgetting, or poor estimation of the learning evolution in the case where a region is not visited frequently enough, we define a way to monitor learning progress based on previously seen examples (the *backward progress estimation*).

A forward evaluation of the progress in a given region R_i (i.e. the standard evaluation used in [20] and [8]) would be done using the following procedure: after t_i-1 RGB-D frames acquired in R_i, we would calculate the segmentation mask O_{t_i} and the estimated saliency E_{t_i} of the new t_ith frame, and the corresponding prediction error $Err_i(t_i)$ would be added to the meta-learner error history of R_i:

$$Err_i(t_i) = 1 - F_1(O_{t_i}, E_{t_i}) \tag{1}$$

where $F_1(.,.)$ is the F_1 score[1] obtained from pixels of O_{t_i} labelled *salient* and *not salient* only. Then, an estimation of the learning progress in R_i, would be obtained by a linear regression of the error history Err_i over the last τ samples ($\tau = 10$ in our case):

$$\begin{pmatrix} Err_i(t_i - \tau) \\ \vdots \\ Err_i(t_i) \end{pmatrix} = \beta_i(t_i) \times \begin{pmatrix} t_i - \tau \\ \vdots \\ t_i \end{pmatrix} + \begin{pmatrix} \epsilon(t_i - \tau) \\ \vdots \\ \epsilon(t_i) \end{pmatrix} \tag{2}$$

with $\epsilon(t)$ the residual error. The learning progress LP_i in R_i is defined as the derivative of the learning curve (or the opposite of the error rate) in that region:

$$LP_i(t_i) = -\beta_i(t_i) \tag{3}$$

With the forward evaluation, each time a new sample is acquired in region R_i, the only learning error that is updated is the one of R_i. However, when a single learner is shared between several regions, adding a sample in a given region may impact the learning error in other ones. In this configuration, forward estimation is then inaccurate, as learning error is not updated in regions where no samples have been acquired. The proposed *backward evaluation* relies on a history of features and labels in each region. At each step and for each region, we measure the updated error based on this history rather than on the last observed sample. Formally, after the acquisition of $k-1$ RGB-D frames, the shared learner has been updated with corresponding features and labels and produces a model L_{k-1}. Among the $k-1$ frames, only $t_i < k$ are in R_i. Based on those frames, we define the observations history $\mathcal{O}_{t_i} = \{O_1, O_2, ...O_{t_i}\}$ from depth segmentation and the features history $\mathcal{F}_{t_i} = \{F_1, F_2, ...F_{t_i}\}$ from the feature maps of each frame. To obtain the *backward evaluation* of the error $Err_j^{back}(k)$ in each R_j after the acquisition of the kth frame, we first update the learner to obtain the updated model L_k. We then estimate saliency on the history $\mathcal{E}_{t_j}^{L_k} = \{E_1^{L_k}, E_2^{L_k}, ...E_{t_j}^{L_k}\}$, obtained with the current saliency model L_k applied to the feature history $\mathcal{F}_{t_j}$. $Err_j^{back}(k)$ is then obtained by:

$$Err_j^{back}(k) = 1 - F_1(\mathcal{O}_{t_j}, \mathcal{E}_{t_j}^{L_k}) \tag{4}$$

This way, the evaluation of the error in each region is updated each time a new model of the learner is obtained. The procedure to obtain the learning progress based on the error history is then exactly the same as in the forward evaluation. For practical reasons, we also limit the size of $\mathcal{O}_{t_i}$ and $\mathcal{F}_{t_i}$ by keeping at most 10 frames in memory from R_i, and by randomly replacing them when new inputs are available.

E. Cost for actions and policy

In IAC, the cost for taking actions is not considered. In that case, a greedy policy exploring the regions with highest learning progress is enough. For a mobile robot moving in a large environment (e.g. a building), the displacements between two regions can be extremely time consuming, making the greedy policy inefficient. We therefore propose to extend the IAC policy with a RL module that estimates the best trade-off policy between progresses and displacement. The idea is to simulate future displacements of the robot, and to use Q-learning [22] to determine the policy that optimizes progress (or reward). The next displacement is taken by following this policy, the progress in regions is updated, and a new policy is re-estimated.

The problem is modelled as a navigation graph where states are the regions in which progress is evaluated. Adjacent regions are connected by edges in the graph, and the cost for moving to an adjacent region is defined by the time a robot would take to move between the centroids of the two regions. See Figure 4 for an example of navigation graph. To move on this graph, the robot can take a displacement action among $M = \{up, down, left, right, stay\}$. If so, the robot selects the adjacent upper, lower, left, right, or current region. If the taken action is not allowed by the navigation graph, $stay$ is taken by default. After selecting the target region R_j, a new position (x', y', θ') within R_j is randomly chosen, and the robot moves to that location. Instead of moving, the robot can also take the action $learn$. In this case, the robot grabs an RGB-D input at its current location, processes it and updates the saliency learner as well as the meta-learners.

Suppose that at time t the robot is in region R_i. Each region R_j of the environment is then associated with an estimated learning progress $LP_j(t)$. To decide the next action to take, we run a batch of 1000 simulated episodes for a horizon of time $N = 3600s$ (*i.e.* from t to $t + N$) in which the robot takes (virtual) action in the environment and collects reward based on learning progress in the region. The time spent for displacement depends on the distance between centroids of two regions, whereas the time spent for updating the saliency model is set as 1s (which is the average measured time to update the random forests during the experiments). For each episode, the initial state is always the region where the robot is at time t, $R_i(t)$. As the batch of episodes is a simulation,

[1] $F_1 = \frac{2tp}{2tp+fp+fn}$, where tp, fp and fn are the true positives, false positives and false negatives. We use the F_1 score as our error metrics for Err_i, because *not salient* pixels are representing more than 80% of the samples, making accuracy inappropriate for error estimation.

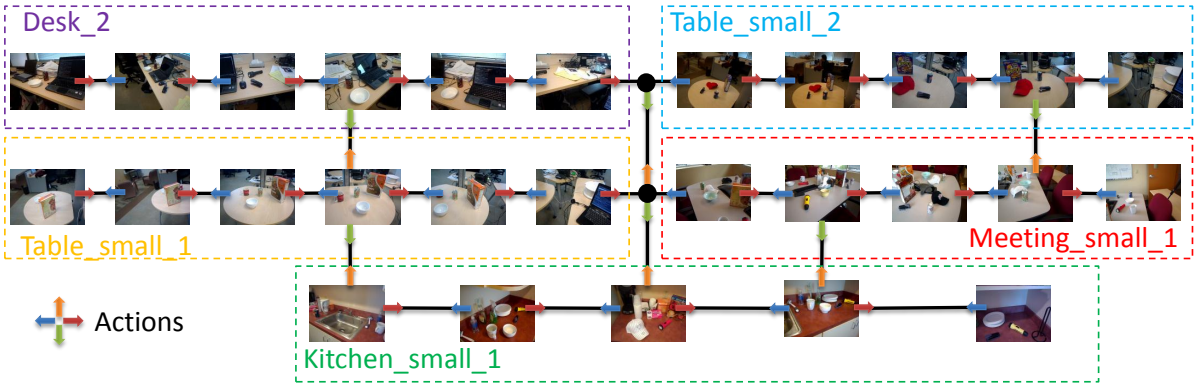


Fig. 4. An example of navigation graph. Here, the environment of the robot is an artificial building in which each room (dashed lines) contains a set of images from a video sequence. Each region contains a subset of the video sequence and is here represented by a single image. Transitions between regions are arbitrarily defined and are made possible by the different actions represented by arrows. Black dots between rooms represent the centroid of regions in a corridor.

the robot is not physically moving when taking actions in M and not grabbing frames and updating model for action $learn$. Instead, the robot just receives reward associated with taking virtual actions in a given region. As no model update is done during this simulation, the reward based on learning progress in a given region remains constant, and the 1000 episodes are run in a few milliseconds only. The reward received when taking a virtual action a in region R_j is as follows:

$$r(R_j, a) = \begin{cases} LP_j(t) \text{ if } a = learn \\ 0 \text{ otherwise} \end{cases} \quad (5)$$

Based on the simulated episodes, we determine an optimal policy by updating a Q-matrix according to the following rule

$$Q(s_k, a_k) = r(s_k, a_k) + \gamma \max_{a' \in M \cup \{learn\}} Q(s_{k+1}, a') \quad (6)$$

with γ the discount factor (0.1 in our implementation), s_k the region where the robot is after k (virtual) actions, a_k the action to take next, and s_{k+1} the region after taking action a_k. Once all the episodes have been simulated, we select the next (not virtual) action a_t taken by the robot such that

$$a_t = \underset{a' \in M \cup \{learn\}}{\text{Argmax}} \; (Q(R_i, a')) \quad (7)$$

After this action is taken, the learning progress is re-estimated and a new batch of episodes is simulated to train a new Q matrix and decide the next action to take.

Note that each Q-learning policy is obtained by considering the reward as constant in time. This assumption is wrong in practice, as each new $learn$ action influences the learning progress (and therefore, the reward) that would eventually decrease to 0 when the learner cannot be any better. However, the assumption is accurate enough to estimate the next action to take. As the Q matrix is re-estimated before each new action, this approximation does not introduce a significant bias. Moreover, to force the robot to quickly get a first estimation of the progress in each region, we forced the progress in a given region to be very high as long as less than three samples were collected in that region. This additional constraint has the same effect as the R-MAX [4] exploration policy. Last, we used an epsilon-greedy strategy to move to a random region 10% of the time.

IV. Experimental results

A. Setup

For evaluation, we used a publicly available dataset called *RGB-D scenes dataset* [15] and designed an artificial building to simulate the robot's displacements. The *RGB-D scenes dataset* consists of 8 video sequences of indoor scenes of everyday-life objects lying on tables. To evaluate the exploration policy, we create a navigation graph similar to the one of Figure 4. The navigation graph represents a building in which each room contains a sequence (5 in total) of the dataset. Each room is divided into 5 to 6 regions, and some regions are connected to other rooms (as if there were doors between rooms). A corridor in which no salient elements can be found is also available to move between rooms. Regions in the corridor are represented by black dots in Figure 4. To simulate the displacement of the robot and the acquisition of new frames, we associate a set of frames to each region of the navigation graph. A SLAM algorithm [9] is used to obtain the (x, y, θ) positions of each frame and divide the sequences in regions based on their (x, y, θ) positions (Figure 5). When the exploration policy sends a command to move the robot to a region R_i, we randomly select a frame that belongs to R_i, and calculate the time the robot would take to physically move to this position. We also took 10 frames per region and manually created a ground truth for each of them to create an evaluation set. More precisely, the ground truth of a frame is a mask on which objects lying on planar surfaces (i.e. salient objects) are manually cropped. Those frames were removed from the dataset and used for evaluation only.

B. Exploration strategies

To demonstrate the efficiency of RL-IAC versus other exploration strategies, we run simulations in which the robot is exploring the building based on several exploration rules. Each

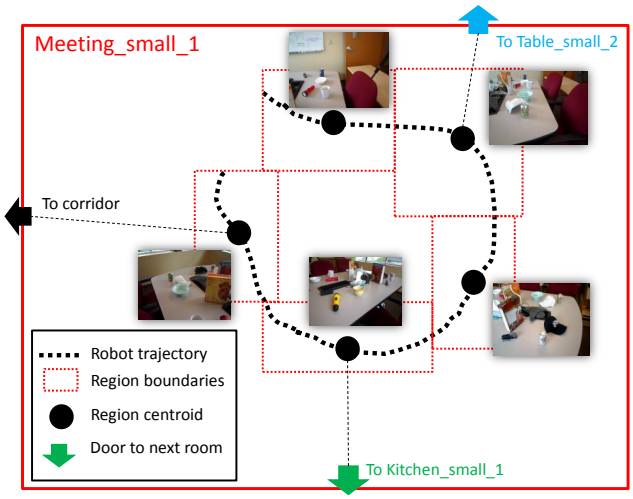


Fig. 5. Example of region definition based on the robot trajectory. The regions are then connected to each other by the navigation graph.

Fig. 6. Evolution of the error rate in each room in terms of number of frames observed.

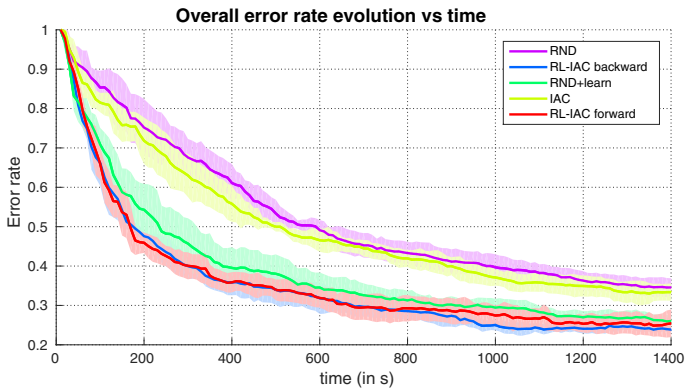


Fig. 7. Overall error rate and variance evolution versus simulated time for a few exploration strategies.

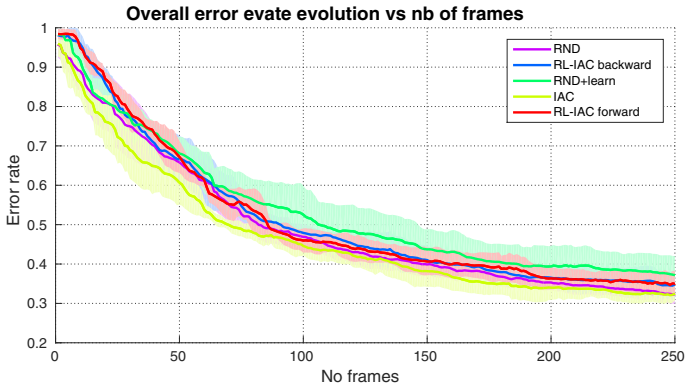


Fig. 8. Evolution of the error rate in terms of number of frames observed. This time, RL-IAC is not the approach that performs best.

exploration strategy was tested 10 times and the average and standard deviation error rate were used to produce the results. To make sure that saliency is correctly learned in each room, we used a dedicated learner in each room and corridor. Each learner is then shared between the regions of the corresponding rooms, but is independent from learners in the other rooms.

To evaluate the performance of the system, we follow the evolution of the *overall error rate*: after each action, we apply the available model of saliency on the evaluation set. We then compare the ground truth of each of these frames with the estimated saliency map, using the formula of equation 4 (based on the F_1 score). The *overall error rate* of the system after each action is then defined as the prediction error averaged over the evaluation set. Note that the *overall error rate* is an extrinsic metrics used to evaluate the performance of the system. It then differs from the *region error rate*, the intrinsic metrics (based on segmentation rather than ground truth) used to get an estimate of the error in each region in Section III-D. As a preliminary experiment to demonstrate the need for an efficient exploration strategy, the error rate evolution in each room as a function of the number of frame taken in this room is displayed in Figure 6. As rooms have significantly different learning progress, following an exploration strategy that takes into account the learning progress should be more efficient than randomly selecting rooms. Figure 7 shows the evolution of the *overall error rate* in time on both environments, for 4 exploration strategies:

- RND: Random selection of a region and a position in it. Find the shortest path in the navigation graph to reach this position. Move to that position. Once arrived, take action *learn* that updates the saliency learner and meta-learner.
- IAC: Same as RND, except that the region selected is the one with highest learning progress.
- RND+learn: Same as RND, except that during displacement, the action *learn* is taken once in each visited region, rather than just in the region of destination.
- RL-IAC with forward error estimation, as described in Section III-D.
- RL-IAC with backward error estimation, as described in

Section III-D.

According to Figure 7, the random exploration strategy (RND) is the one that provides the slowest learning. IAC is doing slightly better, but it remains much slower than RL-IAC. The main reason for such a gap is that both RND and IAC are wasting a lot of time in displacement. Adding learning during displacement (RND+learn) significantly speeds up learning, while using RL-IAC provides the fastest learning. Last, the backward evaluation of the progress slightly improves the result over the classical forward evaluation.

To further validate the need to consider displacement time in the exploration strategy, we display the results of the experiment differently. This time, we plot the error rate versus the number of frames observed rather than time (*i.e.* without considering the displacement of the robot). This representation is the one used in [8], where exploration based on IAC was found to be more efficient than a random exploration. Based on results in Figure 8, IAC is, as expected, the most efficient approach with this representation. As RL-IAC is designed to find a compromise between learning and displacements, the progress between two consecutive observations is not as good as the one obtained with IAC. Therefore, in Figure 8 where displacement time is not represented, RL-IAC is not doing much better than random exploration.

C. Is progress the best source of intrinsic motivation for visual signals?

Our problem is similar to the strategic student learning problem, which considers the case where the agent has to select topics to learn in a limited amount of time. Lopez *et al.* [17] have shown that if the learning error curves are monotonic sub-modular, then exploring regions where learning progress is the highest is quasi-optimal. In practice, there is no warranty about these hypothesis, so that exploration based on learning progress are just heuristics. We are then interested in comparing exploration based on learning progress with other sources of intrinsic motivation. On the other hand, using learning progress makes the robot focus on areas where learning is actually possible and avoids loosing time in unlearnable or trivial ones. This makes learning progress even more efficient when learnable areas represent a small portion of the environment. As a result, a bigger gap should be observed between learning progress and other strategies in environments that are essentially unlearnable.

We evaluate the use of other sources of intrinsic motivation (namely *error*, *novelty* and *uncertainty*) in the RL-IAC backward procedure. More precisely, we follow the procedure described in Section III, except that we replace the learning progress $LP_i(t)$ in Equation 5 by

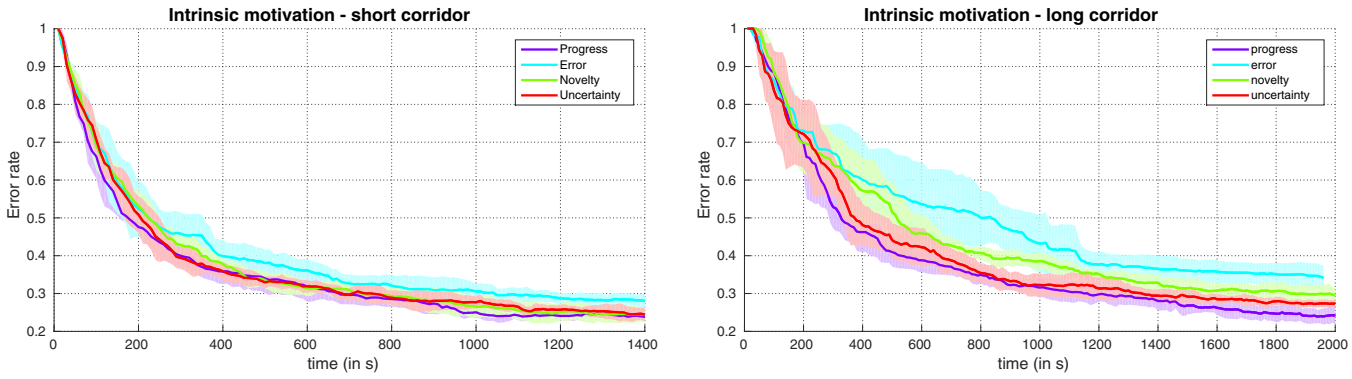
- $Err_i^{back}(t)$ (see Equation 4) for *error*. The agent is rewarded by exploring regions with high learning error.
- $Nov_i(t) = \frac{1}{t_i}$, t_i the number of frames observed in region R_i for *novelty*. The agent is rewarded by exploring regions having the fewest number of observations.
- $Unc_i(t) = \sum_{j=1}^{t_i} -|E_j^{L_t} - 0.5|$ for *uncertainty*, where $E_j^{L_t}$ are saliency maps of $\mathcal{E}_{t_i}^{L_t}$, described in Section III-D. Uncertainty measures the confidence of the learner in the estimation and is further explained in [7]. The agent is rewarded by exploring regions producing fuzzy saliency maps.

We compare the performance of RL-IAC with the aforementioned intrinsic motivations on the setup described above. However, we use two different navigation graphs. The first one is the one represented in Figure 4, where the corridor is only composed of two regions (*short corridor*). The second one is very similar, except that the corridor between rooms is much longer and represent almost half of the regions in the whole building (*long corridor*) with nothing salient inside. In this configuration, the classifier does not learn anything in the corridor, and the error rate is flat all along the experiment. The comparison is displayed in Figure 9. In both configurations, progress seems to be the best source of intrinsic motivation, but the difference is much more significant in the *long corridor* configuration. In this configuration, the *error* fails as the absence of positive samples keeps the error rate very high in the whole corridor[2]. *novelty* spends as much time in the corridor as in the rooms, thus making learning much slower. *Uncertainty* is the only one able to avoid the corridor, as, in the absence of positive samples in the model, any input sample returns '0', thus making output saliency maps not fuzzy at all.

V. Conclusion and future work

In this paper, we have presented an adapted version of the *intelligent adaptive curiosity* for the task of visual saliency learning in a building. The resulting algorithm, called RL-IAC takes into account the cost for displacing the robot to different areas of the environment, as well as a new way to estimate the progress for the case of a learner shared by several regions. The main differences between sensori-motor learning and visual learning are here considered and might hopefully be useful to understand how to adapt IAC to other panel of problems. Our results show that RL-IAC enables a much faster learning than a random exploration, and even exploration based on IAC. In addition, the use of progress as the main source of intrinsic motivation is discussed. The conclusion is that progress makes a much bigger difference in environments where a large portion of the environment is trivial to learn. In any cases, the use of uncertainty as intrinsic motivation also provides promising results. In a future work, we would like to investigate incremental alternatives to navigation graphs and predefined regions. Indeed, the use of navigation graphs is restrictive as it forces an operator to manually determine regions and the robot's trajectories. We could also apply this framework with other definitions of saliency. Instead of a generic object segmentation, we could for example use objects detectors and specialize our saliency to find those objects within their environments.

[2]When no positive samples are observed in a region, the F_1 score is either undefined or equals 0 (So that error equals 1). If undefined, we force the F_1 score to be 0



(a) (b)

Fig. 9. Comparison of several intrinsic motivations to explore the environment. (a) artificial building with a short corridor between rooms. (b) artificial building with a long corridor between rooms

REFERENCES

[1] Haider Ali, Faisal Shafait, Eirini Giannakidou, Athena Vakali, Nadia Figueroa, Theodoros Varvadoukas, and Nikolaos Mavridis. Contextual object category recognition for rgb-d scene labeling. *Robotics and Autonomous Systems*, 62(2):241–256, 2014.

[2] Adrien Baranès and P-Y Oudeyer. R-iac: Robust intrinsically motivated exploration and active learning. *Autonomous Mental Development, IEEE Transactions on*, 1(3):155–169, 2009.

[3] Aysecan Boduroglu, Priti Shah, and Richard E Nisbett. Cultural differences in allocation of attention in visual information processing. *Journal of Cross-Cultural Psychology*, 40(3):349–360, 2009.

[4] Ronen I Brafman and Moshe Tennenholtz. R-max-a general polynomial time algorithm for near-optimal reinforcement learning. *The Journal of Machine Learning Research*, 3:213–231, 2003.

[5] Louis-Charles Caron, David Filliat, and Alexander Gepperth. Neural network fusion of color, depth and location for object instance recognition on a mobile robot. In *Computer Vision-ECCV 2014 Workshops*, pages 791–805. Springer, 2014.

[6] Nuttapong Chentanez, Andrew G Barto, and Satinder P Singh. Intrinsically motivated reinforcement learning. In *Advances in neural information processing systems*, pages 1281–1288, 2004.

[7] Céline Craye, David Filliat, and Jean-François Goudou. Exploration strategies for incremental learning of object-based visual saliency. In *ICDL-EPIROB*, 2015.

[8] Celine Craye, David Filliat, and JF Goudou. Environment exploration for object-based visual saliency learning. In *Robotics and Automaton (ICRA), 2016 IEEE International Conference on*, pages 3140–3148, 2016.

[9] Jakob Engel, Thomas Schöps, and Daniel Cremers. Lsd-slam: Large-scale direct monocular slam. In *European Conference on Computer Vision*, pages 834–849. Springer, 2014.

[10] Simone Frintrop. *VOCUS: A visual attention system for object detection and goal-directed search*, volume 3899. Springer, 2006.

[11] MR Harter and L Anllo-Vento. Visual-spatial attention: preparation and selection in children and adults. *Electroencephalography and clinical neurophysiology. Supplement*, 42:183–194, 1990.

[12] Xiao Huang and John Weng. Novelty and reinforcement learning in the value system of developmental robots. 2002.

[13] Laurent Itti, Christof Koch, and Ernst Niebur. A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on pattern analysis and machine intelligence*, 20(11):1254–1259, 1998.

[14] Varan Kompella, Matthew Luciw, Marijn Stollenga, Leo Pape, and Jurgen Schmidhuber. Autonomous learning of abstractions using curiosity-driven modular incremental slow feature analysis. In *Development and Learning and Epigenetic Robotics (ICDL), 2012 IEEE International Conference on*, pages 1–8. IEEE, 2012.

[15] Kevin Lai, Liefeng Bo, Xiaofeng Ren, and Dieter Fox. A large-scale hierarchical multi-view rgb-d object dataset. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1817–1824. IEEE, 2011.

[16] Manuel Lopes, Tobias Lang, Marc Toussaint, and Pierre-Yves Oudeyer. Exploration in model-based reinforcement learning by empirically estimating learning progress. In *Advances in Neural Information Processing Systems*, pages 206–214, 2012.

[17] Manuel Lopes and P-Y Oudeyer. The strategic student approach for life-long exploration and learning. In *Development and Learning and Epigenetic Robotics (ICDL), 2012 IEEE International Conference on*, pages 1–8. IEEE, 2012.

[18] David Meger, Per-Erik Forssén, Kevin Lai, Scott Helmer, Sancho McCann, Tristram Southey, Matthew Baumann, James J Little, and David G Lowe. Curious george: An attentive semantic robot. *Robotics and Autonomous Systems*, 56(6):503–511, 2008.

[19] Sao Mai Nguyen, Serena Ivaldi, Natalia Lyubova, Alain Droniou, Damien Gerardeaux-Viret, David Filliat, Vincent Padois, Olivier Sigaud, and Pierre-Yves Oudeyer. Learning to recognize objects through curiosity-driven manipulation with the icub humanoid robot. In *Development and Learning and Epigenetic Robotics (ICDL), 2013 IEEE Third Joint International Conference on*, pages 1–8. IEEE, 2013.

[20] P-Y Oudeyer, Frédéric Kaplan, and Verena Vanessa Hafner. Intrinsic motivation systems for autonomous mental development. *Evolutionary Computation, IEEE Transactions on*, 11(2):265–286, 2007.

[21] Jürgen Schmidhuber. Formal theory of creativity, fun, and intrinsic motivation (1990–2010). *Autonomous Mental Development, IEEE Transactions on*, 2(3):230–247, 2010.

[22] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

[23] Jianming Zhang and Stan Sclaroff. Saliency detection: a boolean map approach. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 153–160. IEEE, 2013.