Exploring to learn visual saliency: The RL-IAC approach

Céline Craye, Timothée Lesort, David Filliat, Jean-François Goudou

Exploring to learn visual saliency: The RL-IAC approach

Céline Craye, Timothée Lesort, David Filliat, and Jean-François Goudou

***Abstract*—The problem of object localization and recognition on autonomous mobile robots is still an active topic. In this context, we tackle the problem of learning a model of visual saliency directly on a robot. This model, learned and improved on-the-fly during the robot's exploration provides an efficient tool for localizing relevant objects within their environment. The proposed approach includes two intertwined components. On the one hand, we describe a method for learning and incrementally updating a model of visual saliency from a depth-based object detector. This model of saliency can also be exploited to produce bounding box proposals around objects of interest. On the other hand, we investigate an autonomous exploration technique to efficiently learn such a saliency model. The proposed exploration, called *Reinforcement Learning-Intelligent Adaptive Curiosity* (RL-IAC) is able to drive the robot's exploration so that samples selected by the robot are likely to improve the current model of saliency. We then demonstrate that such a saliency model learned directly on a robot outperforms several state-of-the-art saliency techniques, and that RL-IAC can drastically decrease the required time for learning a reliable saliency model.**

***Index Terms*—Visual saliency, bounding box proposals, intrinsic motivation, intelligent adaptive curiosity, autonomous mobile robots, incremental learning, deep learning**

I. Introduction

In the scope of assistive robotics, where autonomous mobile robots assist and help humans with their everyday life tasks, the need for robots to efficiently analyze and understand their environment is critical. To this end, robots should have the capacity to efficiently find and identify objects they can interact with.

Object localization in cluttered environments is still a difficult problem. Today, deep learning-based methods provide efficient ways to localize and identify a large set of objects in a wide variety of complex configurations [40], but they generally require hours or days of offline training, high GPU resources, thousand to millions of training images, and are not really flexible to novelty. Furthermore, domestic mobile robots are meant to evolve essentially in indoor environments, interact with a limited amount of objects, for specific tasks and thus do not require such wide scope capacity. However, they should be able to adapt to novelty by quickly updating the representation of their environment. Learning to localize objects online and directly within the environment is then a very desirable property.

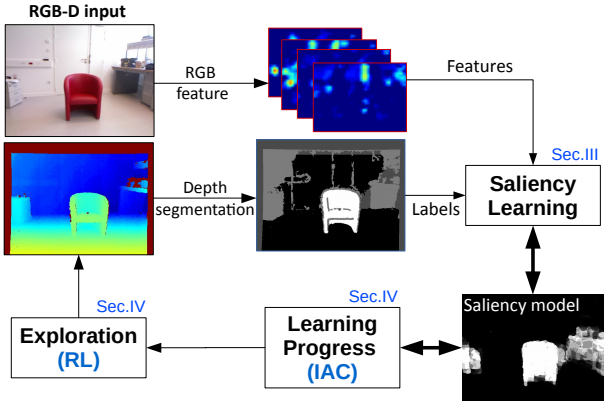


Fig. 1. Overview of our saliency learning and exploration approach.

Nevertheless, online learning must come with a methodical exploration of the environment. The displacement of the robot makes it possible to move to favorable observation conditions in order to improve recognition performances, but a critical point is to monitor this performance quality, and use this information to drive the robot accordingly.

In this article, we consider a mobile robot exploring its environment while building a model of visual saliency enhancing objects of interest. Based upon our previous work [19], we present a system (Fig. 1) able to:

- produce object-oriented visual saliency maps and propose object bounding boxes;
- learn this saliency model incrementally directly within the robot's environment;
- make the robot explore the environment autonomously and efficiently, by using reinforcement learning to estimate the interest of long term trajectories.

More precisely, the system is composed of two major components. The first one is a method for learning a visual saliency model incrementally, without any user supervision. This model can be exploited to enhance objects of interest in the environment, and used as a posterior to generate bounding boxes around salient objects. The second one is an algorithm based on the *Intelligent Adaptive Curiosity* that drives the robot in its environment, so that learning is done is an efficient

Céline Craye is a PhD research engineer at Thales SIX- Vision and Sensing laboratory. Email: celine.craye.eu@gmail.com

Timothée Lesort is a PhD student at Thales SIX and at ENSTA Paristech, and a member of the INRIA FLOWERS team. Email: timothee.lesort@ensta-paristech.fr

David Filliat is a professor at ENSTA Paristech, and a member of the INRIA FLOWERS team. Email: david.filliat@ensta-paristech.fr

Jean-Fraçois Goudou is a R&I project manager at Thales SIX - Vision and Sensing laboratory. Email: jf.goudou@thalesgroup.com

U2IS, ENSTA ParisTech, Inria FLOWERS team, 828 bd des Maréchaux, 91762 Palaiseau cedex France

Thales - SIX - Theresis - VisionLab 1, avenue Augustin Fresnel, 91767 Palaiseau, France

You can visit the project's repository at https://github.com/cececr/RL-IAC

and organized manner. We call this algorithm RL-IAC, for Reinforcement learning Intelligent adaptive curiosity. RL-IAC encapsulates the saliency learning technique and can be seen as a whole system for autonomous exploration and efficient learning. We demonstrate that our method for learning saliency online generates saliency maps that are more accurate than most state-of-the-art techniques in the robot's environment. In addition, the efficiency of RL-IAC for exploration is evaluated versus alternative environment exploration techniques.

The article is organized as follows: we present related work in Section II, Section III describes the method used to learn visual saliency incrementally, while Section IV explains the exploration strategy based on RL-IAC. We propose an experimental evaluation of our system in Section V, and finally provide concluding remarks and perspectives in Section VII.

II. Related work

As our system is based on two independent components, we consider separately the related work on saliency maps and object localization, and the one on the field of exploration on mobile robots. We last highlight our main contributions and positions towards state-of-the-art.

A. Saliency maps and object localization

To efficiently analyze visual inputs and interact with objects in cluttered environments, robots usually rely on a visual attention strategy. This mechanism turns the raw visual scene into selected and relevant information the robot should focus on, possibly involving zooming [48], foveal vision [7] or physical displacements [8], [39] of the robot. This concept has been widely studied and discussed [9], [34], [24], from biological and computer vision points of view. We restrict visual attention in this study to the localization of objects of interest.

Visual attention is strongly related with the concept of visual saliency, defined as a "subjective perceptual quality which makes some items in the world stand out from their neighbors and immediately grab our attention" [34]. The first computational models of visual attention were relying on saliency maps [35], representing the saliency of an image on a pixel-by-pixel basis. General convention is to associate a pixel intensity proportional to the pixel saliency.

Saliency maps can be either purely *bottom-up* [71], [21], [31], or refined by *top-down modulation* [28], [72], [23], [25]. Bottom-up saliency highlights stimuli that are intrinsically salient in their context, which may sometimes be sufficient for scene exploration [74]. However, top-down modulation, which highlights elements that are relevant for a specific task, is more meaningful for the problem of object detection in indoor environments. Saliency maps are either fixation-based [35], [21] or area-based [13], [25], [71]. Fixation-based approach is related with the probability of a human being to make a fixation at a given pixel position, while area-based approach consider salient elements (typically objects) as a whole area of the image. The latter approach is then closely related to object segmentation. In the context of a mobile robot in an indoor environment, our technique aims to build top-down, object-oriented models of saliency.

Saliency maps are most of the time based on RGB images only, but a few of them also integrate the depth component [55], [15] [61], [59], [33]. Another possible approach is to fuse both depth and RGB components [27] processed separately. Usually, depth is good at detecting objects and is particularly well-suited for indoor environments. These approaches typically use geometrical constrains such as symmetries [58], [20], convexity [54], or detecting elements placed on planar surfaces (such as floors or tabletops) [12], [2]. These approaches can detect objects much more accurately than using only the RGB component, but are limited by the sensor quality and geometrical constraints (reflectance, size or distance to the objects). Our approach uses the depth component to detect objects, and learns the visual aspect of these objects on the RGB image.

A particular kind of saliency that can also be learned from RGB-D is the co-saliency [16], [26], [67], [70]. The co-saliency is an unsupervised approach applicable on a set of images that highlight salient features common to several images. Consequently, Co-saliency methods need object images in different situation to detect them. However, our setting does not have this variability. Most objects only appear in one context and thus co-saliency methods would not highlight them.

Machine learning, and especially deep learning have also been used for the generation of saliency maps. The best performance reported on saliency benchmarks is so far based on Convolutional Neural Networks (CNN). Whether on stimuli-based benchmarks [30], [44] or object-based one [43], those methods have shown excellent results compared to traditional approaches. More interestingly, it was shown that CNN activation maps can be used as powerful objects detectors even when trained on a weakly-supervised basis [73], [50]. To obtain saliency based on deep learning, activation maps can be sampled from different CNN layers to produce multi-scale saliency maps [30], [44]. Moreover, these neural networks are also particularly efficient for image segmentation. Therefore, a CNN segmentation model can be used as a complement to a object detection CNN to improve the quality of the saliency maps [43]. CNN models are generally learned by supervision but they can also be trained for image saliency prediction through adversarial examples [53].

In the scope of object detection, the use of agnostic bounding boxes [1], [75] has become a very popular alternative to the traditional sliding window technique. Not only this type of bounding boxes significantly decreases the number of windows to evaluate, but it also provides much more accurate bounding boxes. Several CNN-based object-detection techniques, such as faster R-CNN [62], are based on agnostic bounding box proposals. Hosang *et al.* [29] have published in that regard an extensive review on detection proposals. We investigate in this work how our saliency model can be used as a quality measure for these kind of proposals. Other methods take advantage of CNN to propose objects segmentation based on multi-scale features map [56], [57].

B. Autonomous environment exploration

In robotics, the *exploration problem* is the one of maximizing knowledge over a working environment by means of a single or several robots. Autonomous exploration of the environment is then made possible by providing rules able to guide the robot's actions to reach specific goals in that regard. We here present three types of approaches that can be potentially combined to lead the robot's exploration.

1) Vision-based exploration:: Exploration may be guided by visual inputs and driven by a *visual attention mechanism* [35]. In this case a visual focus of interest is selected in the environment (from saliency maps computation, for example), and the actions performed by the robot aim to provide more information about the selected target. In a mobile robot scenario, actions are displacement in order to get a closer or better point of view of areas of interest [47], [39], [42], [8]. Exploration is therefore based on a pre-attentive stage, where potentially relevant targets are selected and uninformative areas are discarded, and an attentive stage after action, where more complex tasks (such as object recognition [60]) are performed on the targets to obtain more information about them.

2) Map-based exploration:: When autonomous exploration is made by a mobile robot, a 2-D or 3-D map of the environment is commonly used. Those maps can either be predefined before exploration [39], or incrementally updated [6] as the robot discovers new portions of the environment. The exploration problem in mobile robotics is often related with a problem of maximizing map under constrains. For example, minimizing displacement time [51] while visiting a certain number of areas by solving a traveling salesman problem, minimizing the number of views [36] with an art-gallery problem, or re-visiting previously observed areas based on potential uncertainty reduction [39] or information-gain [63]. Unlike visual attention strategies, exploration based on map coverage is often composed of a problem of displacement cost minimization.

3) Skill-based exploration:: When the robot aims to learn skills from its environment, actions can be oriented to the task of learning rather than that of pure exploration. This is then typically the case in reinforcement learning [5] (RL), where actions are taken to learn an optimal state-action policy rather than for gaining knowledge about the environment. *Q-learning* is a typical RL algorithm for this kind of exploration [46], [48]. In the scope of developmental robotics [69], intrinsic motivations are also used as a drive for robot's acquisition of skills through experience and exploration. *Intrinsic motivation*, defined as intrinsic reward (*i.e.* not related to an external goal, but to the acquisition of competences or knowledge) able to drive a behavior, is a possible approach for guiding exploration in that regard. For example, Huang *et al.* [32] have used *novelty* to guide visual exploration, while Chentanez *et al.* [14] have used the error of prediction of *salient events* 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 [49], [4], [65]. The *Intelligent Adaptive Curiosity* (or IAC) [52] is one of the most emblematic implementation of intrinsically motivated exploration using progress. Learning progress has also been exploited in a reinforcement-learning context, typically with artificial curiosity [38], [64], or to make exploration flexible to changes in the environment or wrong assumptions [46].

C. Contributions

Our first contribution is a method that incrementally learns RGB saliency from a low quality depth-based object segmentation as the robot explores the environment. The produced saliency maps are therefore dedicated to the environment that was explored, but remain flexible to novelty. The model that is learned here is a top-down type of saliency that is related to the concept of objectness, and the model that is learned is used to produce object-oriented saliency maps. We further show that these saliency maps can be used to refine object box proposals. Unlike most saliency techniques based on learning, ours is self-supervised, so that the robot is able to learn without any human annotation or assistance. The main mechanism consists in a transfer learning method between a restricted depth-based segmentation technique and the RGB frame.

Our second contribution is the RL-IAC (*Reinforcement Learning Intelligent Adaptive Curiosity*) algorithm that provides an autonomous exploration strategy. RL-IAC encapsulates the online saliency learning technique, and uses it as a core component to drive the robot's exploration. RL-IAC is a hybrid approach: on the one hand, it uses the idea of the IAC algorithm [52] that drives exploration towards learning progress. On the other hand, it uses a map-based exploration that tries to minimize the time spent in displacements. For that, we add a reinforcement-learning module to the traditional IAC approach. The RL module is constantly retrained during the exploration, so that the robot is attracted by regions with high learning progress while trying to minimize its displacements.

This paper is an extension and a synthesis of several previous publications. In [18], the approach is tested on a specific robotic platform: A biomimetic head equipped with a foveal vision system where saliency is learned on the peripheral field of view, through object recognition on the fovea. Here, the implementation is designed for mobile robots equipped with an RGB-D sensor, using depth maps segmentation rather than direct object recognition for learning saliency. Moreover, in [18] the exploration of the surrounding scene is done with an adapted implementation of the IAC algorithm rather than RL-IAC, and the evaluation is conducted on the biomimetics platform mainly. In a second publications [19], early work on the RL-IAC method is presented and evaluated on a mobile robot. We here provide more technical details and further experimental results. Results are also updated withe CNN-based feature extractor, and a method to automatically and incrementally obtain a navigation graph required by the RL-IAC algorithm.

III. Saliency learning and object proposals

This section describes the component that learns visual saliency and produces object proposals. Figure 2 presents the general block architecture along with the corresponding

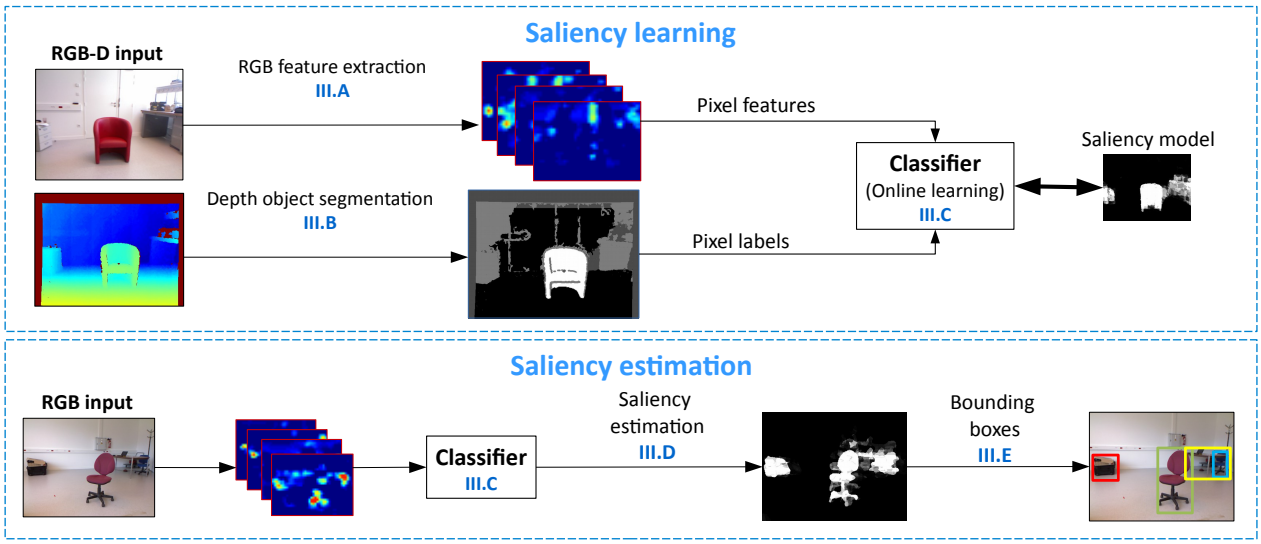


Fig. 2. General mechanism of the saliency learning component. The RGB stream is used to generate CNN-based features, while the depth stream is sent to a segmentation algorithm. The segmentation result is used as a learning signal to train a classifier. The resulting trained model actually predicts the saliency of a given image, without using the depth segmentation component.

section for each block. In a learning stage, the system extracts RGB features (see Section III-A) and learns the visual (RGB) aspect of salient elements within their context using a depth-based object segmentation as a supervision signal (see Section III-B) that is only used in the learning phase. Learning is performed by a classifier (Section III-C) that produces and constantly updates a saliency model. When available, the saliency model is exploited to generate environment specific saliency maps using the RGB image only (Section III-D), and these saliency maps can be used to generate boxes that isolate objects of interest (Section III-E).

A. RGB Feature extraction

As in our previous work [18], feature extraction in the color image is based on convolutional neural networks following the approach of Zhou *et al.* [73] and is fully independent from the classification step.

We use the publicly available trained model [73] and perform feature extraction at the level of the *class activation mapping*, or CAM layer (called *CAM-CONV* in the network, see [18] for details). Because of striding and pooling in the network, the output feature maps have a resolution that is 16 times lower than the input image. To overcome this loss of resolution, we present in Section III-D a method to reconstruct saliency maps at the original scale.

B. Depth-based object segmentation

The segmentation procedure has been designed for RGB-D cameras such as Kinect or Asus Xtion, and mainly works for indoor environments to detect objects lying on planar surfaces (typically tables or floor), with a diagonal size between 10 and 180 centimeters. The method is a modified version of the depth-based object segmentation of Caron *et al.* [12] and is used to produce a partial but accurate estimation of the

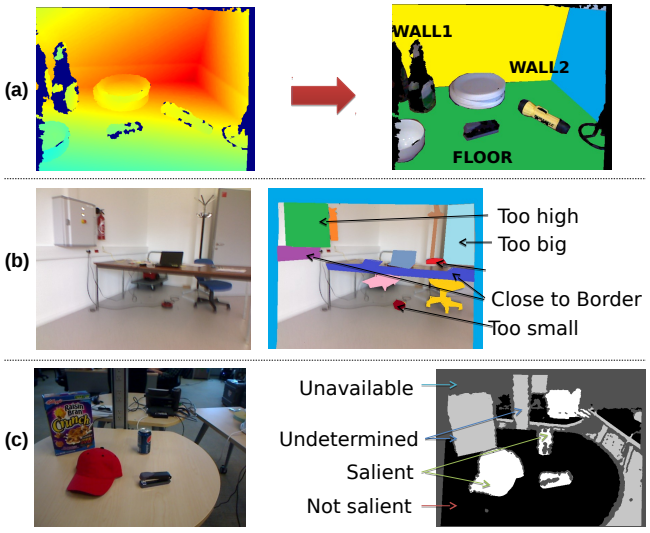


Fig. 3. Main segmentation components (corresponding to section III-B in Figure 2). (a) Floor and wall removal. (b) Object candidates filtering based on geometrical criteria. (c) Construction of the segmentation mask

salient objects in the scene. Processing is based on the depth-map only. Figure 3 illustrates the main components of this approach.

As a first step, the depth map is turned into a point cloud, and the algorithm detects the major plane (most likely the floor, or a tabletop) of the cloud based on a RANSAC algorithm [22]. The major plane is then tracked in the following frames and during the whole sequence to make sure that the same surface is used during the whole experiment and to monitor false detections.

Given this major plane, potential walls are detected and filtered out: they are detected by finding large planar surfaces perpendicular to the major plane. The remaining points of the cloud are likely to be part of salient objects, but could as

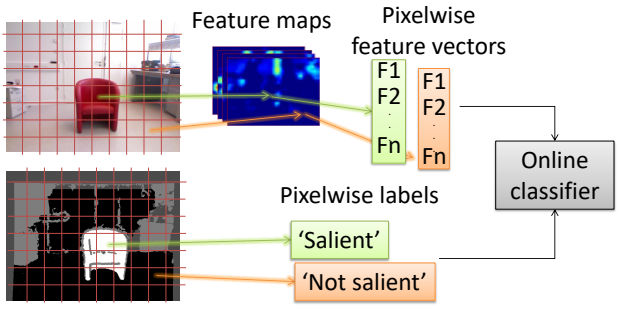


Fig. 4. Pixelwise data formatting for online saliency learning.

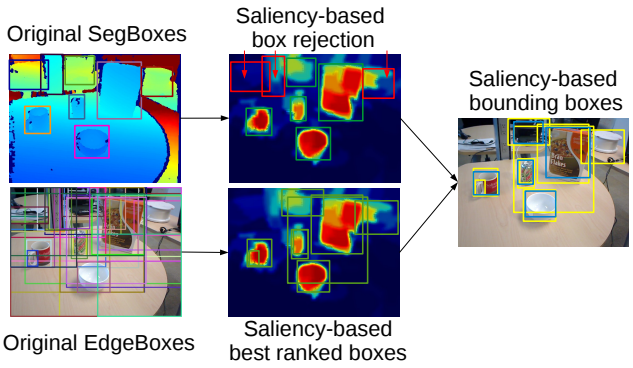


Fig. 5. Bounding box proposals biased by our saliency map

well be small portions of walls, poles, or any other artifact that is irrelevant for a robot. Remaining points are grouped in blobs to form object candidates. We then remove candidates that are either too small, too large, or too far from the floor. Last, to avoid false detection from large objects cut by the border of the frame, only candidates having no contact with the border of the field of view are kept and considered as salient objects. Whether discarded or not, all of the object candidates are associated with a bounding box to generate the SegBoxes (see Section III-E).

To convert this segmentation procedure into a 2-D segmentation mask compatible with saliency learning, we project the whole point cloud back to the image frame and attribute a label value to each corresponding pixel. Figure 3, row (c) illustrates an example of such segmentation mask. In this case, pixels having no corresponding point cloud, because of reflectance (for example, the plastic bottle in (a)), shadows (at the border of objects), or visible points too far from the kinect are labeled "*unavailable*" (dark gray). Points of the cloud that either belong to the major plane or to a wall are labeled "*not salient*" (black). Points of the cloud belonging to a cluster detected as a salient object is labeled "*salient*" (white). Last, all remaining points are categorized as "*undetermined*" (light gray), as the algorithm was not able to determine their actual state of saliency. In particular, candidates that are too close to a border could be either walls or objects (see portion of wall and table of row (b) that are both close to a border of the image).

C. Online learning

As in our previous work [18], learning uses an online classifier that is continuously updated based on the RGB-D observations, turned into a set of labels and features: the segmentation mask is first resized to the same size as the feature maps as labels and the 1024 features associated to each pixel are collected and turned into a feature vector so that each pixel is attributed a features-label sample (see Figure 4). To make sure that the dataset has as few false detections as possible, only pixels having a label "*salient*" or "*not salient*" in the segmentation mask are selected to feed the saliency model. The classifier used in our implementation is an incremental version of random forest, as described in [18].

D. Saliency estimation

Saliency maps are generated by applying the classifier to RGB images and increasing the resolution of the result to the resolution of the original RGB image by using a superpixel segmentation of the image (see [18] for details). Although less accurate than depth segmentation, saliency maps provide an estimation of the saliency for each pixel of the image, as opposed to the partial saliency prediction of the segmenter. The classifier is also able to generalize saliency even when the segmenter fails at predicting it.

E. Object bounding box proposals

The saliency map provides an indication of the interestingness of a given pixel, but does not say much about objects. In order to localize objects in an image, an additional step is then necessary to group salient pixels into object candidates, represented in Figure 5. To this end, we use two types of bounding box proposals, and we select or reject each of them based on a score related to saliency. The first bounding boxes are obtained by the EdgeBoxes [75] algorithm: we compute for a given RGB input the 100 most likely EdgeBoxes, and their associated edge score (h_b^{in} in [75]). The second type of bounding boxes are obtained with the segmentation result (called SegBoxes here for simplicity): the segmentation algorithm produces, on top of salient objects, some additional object candidates.during segmentation, pixels of the depth map are clustered in order to create object candidates (See Section III-B). Among these candidates, some are labeled "*salient*", some should be salient but are labeled "*undetermined*", and some are just artifacts. We then define Segboxes as the bounding boxes around all objects candidates proposed by the output segmentation mask. of the segmentation mask.

For both EdgeBoxes and SegBoxes, we associate each box B with a score related to saliency (called here the *saliency consistency score*, or SCscore), representing the ratio of salient pixels in the box:

$$SCscore(B) = \frac{1}{w_B \times h_B} \sum_{i,j \in B} S(i,j) \qquad (1)$$

where $S(i,j)$ is the saliency of the pixel at (i,j), obtained from the saliency map and w_B and h_B are the width and height of B. The highest the score is for a given box, the most

likely it is to contain a salient object. For the EdgeBoxes, the SCscore is multiplied by h_b^{in}. This way, small boxes found within a salient object might be rejected if the h_b^{in} score is low enough. Last we filter out Segboxes and Edgeboxes with a final score below a certain threshold and keep the remaining ones. In our dataset, we found $SCscore = 0.2$ for SegBoxes and $SCscore \times h_b^{in} = 0.01$ to be good trade-off thresholds between false alarm and false rejection rates.

IV. RL-IAC

Our exploration strategy, that we call *Reinforcement Learning-IAC* (or RL-IAC) is based on the IAC (*Intelligent Adaptive Curiosity*) algorithm. In IAC (such as used in [18]), the robot focuses on areas where learning is neither trivial nor impossible. This way, learning is faster as no time is wasted in useless or unlearnable areas, and better as mainly relevant samples are selected. Nevertheless, the original version of this algorithm does not consider the case where actions (displacements in our case) have a non negligible time. To make it compatible with our application, we couple IAC with some navigation information to find a right balance between learning and displacement.

Further explanations about the differences and similarities of our approach with traditional IAC applications have been described in a previous work [17]. We focus here on describing the mechanisms of RL-IAC as a whole.

Figure 6 presents the general architecture of RL-IAC, using the notations and vocabulary proposed by Oudeyer *et al.* [52]. The module strongly depends on the saliency learning module and takes as an input the feature maps X and the object segmentation Y to feed a learner (called classifier in Section III-C) that is constantly updated. Based on the available model, the learner produces a saliency map, which is also an estimate $\widetilde{Y}$ of segmentation Y. In the meanwhile, as the robot navigates in its environment, the space is cut into regions based on a 2-D map, and a navigation graph is updated to model the connexity between regions (Section IV-A). By comparing Y and $\widetilde{Y}$, the meta learner produces a local estimation of the error in each region, and derives a local progress measure L (Section IV-B). Last, a reinforcement learning module (Section IV-C) uses the navigation graph as the set of states and actions, and attributes to each state the reward L of the corresponding region. The RL module is re-trained at each step to provide the robot with a displacement policy. After each displacement, a new RGB-D input is acquired and the modules are updated. The bounding box proposal module (section III-E) was left out of the diagram as it was not exploited in the RL-IAC approach.

A. Regions and navigation graph

1) Role in RL-IAC: One of the essential component of RL-IAC is to locally monitor how good the learner is at predicting saliency. This local estimation is obtained by creating statistics on samples collected within the same region. In our case, regions are defined as portions of a 2-D map of the environment, produced by a SLAM algorithm. The (x, y) position of the mobile robot on the map then determines the region that is currently being explored.

In addition, displacement in the environment is made possible by a navigation graph, representing the different regions, their relative positions and connexities, and the distance between two neighbor regions centroids. This navigation graph is also used by the reinforcement learning module to decide the next displacement of the robot.

2) Algorithm:: We propose a method that incrementally splits the space into regions and produces the associated navigation graph. The method relies on an occupancy grid (the 2-D map) of the environment and the visible field of view of the RGB-D sensor. Figure 7 presents an example of regions and navigation graph obtained this way. On this figure, visible areas are colored depending on the region they belong to. Walls and obstacles are represented by gray pixels. Lastly, circles with region index correspond to the region centroid, and edges represent the available displacements of the robot. Each region has at most 4 neighbor regions.

Initially, the occupancy grid has a pre-allocated size, where each pixel state is "*unexplored*". This map is first divided in proto-regions based on a regular grid of arbitrary size of 5 meters-length. This way, each position of the occupancy grid is associated with a proto-region. The regions determined in our algorithm are subdivisions of those proto-regions.

As soon as a new robot observation is available, we update the occupancy grid based on the RGB-D field of view: we transpose the point cloud obtained from the depth map in the occupancy grid frame. Points belonging to the floor are marked as "*free*" on the occupancy grid, and other visible points are marked as "*occupied*". Let us denote V the list of visible points marked as "*free*" on the map. Let us consider $\{P_i\}_{i=1..N}$ the N proto-regions having an overlap with V. Each P_i is then the list of all pixels contained in the square delimiting the proto-region. For a given P_i, let us define $\{R_j\}_{j=1..M}$ the pixels of the M regions contained in P_i. Regions are now updated using the following procedure for each P_i (Please refer to Figure 8 for an illustration of each case):

- If $V \cap P_i \cap \{R_j\}_{j=1..M} = \emptyset$, create a new region R_{M+1} constituted with all pixels of $V \cap P_i$;
- If $V \cap P_i \cap \{R_j\}_{j=1..M} \neq \emptyset$ and V is overlapping a single region R_j, update R_j with V. $R_j \longleftarrow R_j \cup V \cap P_i$
- If $V \cap P_i \cap \{R_j\}_{j=1..M} \neq \emptyset$ and V is overlapping a set of $L \geq 2$ regions $\{R_k\}_{k\in 1..L}$ merge all those regions by creating a region $R_{M+1} = V \cap P_i \cup \{R_k\}_{k\in 1..M}$. Then, empty all R_k.

The nodes of the navigation graph correspond to the positions of the regions centroids (if not empty). Edges are determined based on the connexity between regions and their neighbours. In other words, if a region has a common border with another region, they are connected by an edge. Note that regions within the same proto-region do not have any connexity, otherwise they would have been merged. As a results, edges are necessarily between a region of proto-region P_i, and a region of the upper, lower, left or right proto-region P_j. We also attribute to each edge a weight representing the distance between the two centroids.

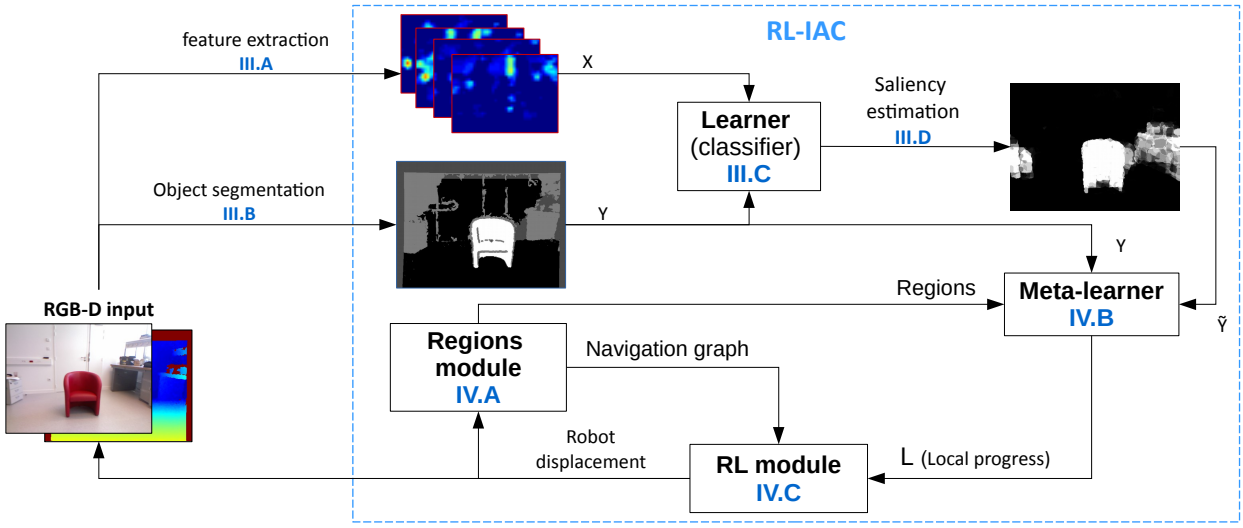


Fig. 6. General achitecture of the RL-IAC module. The saliency learning module is improved by using a feedback loop considering the learning quality (through meta learning) in order to guide the robot's displacements.

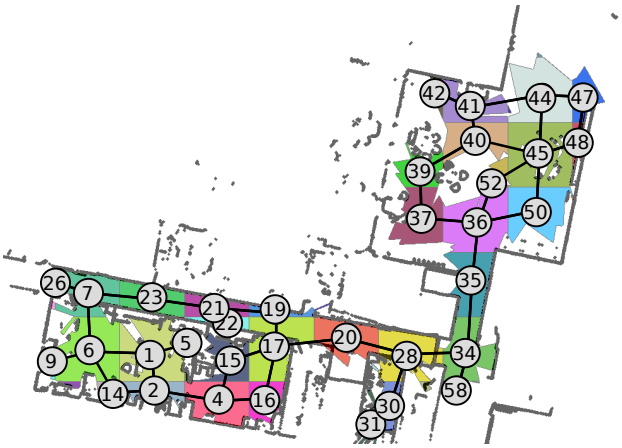


Fig. 7. Regions and navigation graph obtained after the exploration of the ENSTA building. Note that the bounding box proposal is independent from the RL-IAC process, so that it is not displayed here.

Fig. 8. Update procedure. the robot is in proto-region P_2 but the field of view V covers proto-regions P_1 to P_4. In P_2, no regions were previously defined, so a new region is created (region 6). In P_3, region 4 already existed but has no overlap with V: a new region is created (region 7). Region 1 in P_1 overlaps V, so the region is extended. Lastly, V overlaps two regions in P_4, so that regions 2 and 3 are merged to create region 5. Last configuration, when a wall is splitting a proto-region, two separate regions are naturally created on each side (as they are not connected components). Thus, region 0 and 1 of P_1 are separated by a wall and cannot be merged together.

B. Meta-learner

The meta learner is similar to the one used in [18] and aims at monitoring the local error made by the learner and derive a local estimate of the learning progress. The local estimation is made possible by grouping samples collected in the same regions and making statistics within each region. Recall that the robot is in region R_i at time t if its current position $(x(t), y(t))$ falls within R_i's boundaries.

Error estimation is done by comparing a segmentation mask (the learning signal) with the corresponding predicted saliency map (the estimated signal). More precisely, from an RGB-D input, we extract features and compute a segmentation mask. We then consider the observation set O by keeping only *salient* and *not salient* pixels from the segmentation mask. We estimate the saliency response for each of these pixels. These responses (between 0 and 1) are then binarized with a threshold of 0.5 to obtain the estimation set E. Last, we compute the estimated learning error Err of a particular frame based on Equation 2:

$$Err = 1 - F_1(O, E) \tag{2}$$

where $F_1(.,.)$ is the F_1 score [1].

From the history of errors in each region R_i, a linear regression is used and the slope of the resulting model is used as the learning progress $LP_i(n)$ in this region (see [18] for details).

C. RL-based displacement policy

In previous applications of IAC, the time required to reach a region A when in a region B is not considered. In this scenario, a greedy policy exploring the regions with highest learning progress is enough. For a mobile robot moving in a

[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, because *not salient* pixels are representing more than 80% of the samples, making accuracy inappropriate for error estimation.

large environment (e.g. a building), the displacements between two regions can be extremely time consuming, making the greedy policy inefficient.

We therefore represent the regions by the navigation graph described in Section IV-A that encodes the relative distances between regions and the possible displacements the robot can make to reach a region. We use this navigation graph to model states, actions and rewards of a reinforcement learning problem (we use Q-Learning [68]). The reward is the learning progress in each region, and a batch of simulations is run on this setup to determine the policy that optimizes progress while minimizing displacements. After simulations, the robot moves to another region by following this policy.

To describe how the module work, let us consider that the robot is in a region R_i at time t. As the navigation graph and learning progress is constantly evolving, the procedure is repeated before each robot's displacement.

1) States:: The states are the node of the navigation graph. In other words, the regions centroids.

2) Actions:: Actions are represented by displacements on the graph. Each region is connected by edges with neighbor regions in one of the four (above, below, left and right) adjacent proto-regions. As a result, the robot selects one among actions "*up*", "*down*", "*left*" or "*right*", depending on the graph edges, and moves to the corresponding neighbor region.

3) Reward:: A reward r is associated with each region R_j. We take for each region the last calculated learning progress. Thus, at time t, if R_j contains $n_j(t)$ observations, we define reward as:

$$r(R_j) = LP_j(n_j(t)) \tag{3}$$

LP_j being the learning progress, as defined in [18].

4) Simulation setup:: We simulate a batch of 1000 episodes where a virtual robot moves in the navigation graph. For each episode, the initial state is R_i (the actual state of the robot). During the episode, reward is collected right after taking an action and arriving in a given region. The episode stops when the traveled distance exceeds $N = 1km$ [2].

5) State-Action policy:: During the batch of episodes, a state-action matrix is updated according to the following rule

$$Q(s_k, a_k) = r(s_k) + \gamma \underset{a' \in A}{\text{Max}} Q(s_{k+1}, a') \tag{4}$$

with γ the discount factor (0.9 in our implementation), s_k the region where the robot is after k (virtual) actions, a_k the action to take next, A the set of all possible actions, and s_{k+1} the region after taking action a_k. During the whole simulation, the virtual robot follows an ϵ-greedy policy ($\epsilon = 0.1$) to take the next action.

6) Robot displacement:: Once all the episodes have been simulated, we use the Q-matrix to select the next (not virtual) region to visit. For that, we select the next action a_t that should be taken by the robot according to Equation 5:

$$a_t = \underset{a' \in A}{\text{Argmax}}(Q(R_i, a')) \tag{5}$$

[2]This distance is obtained by cumulating the edge weights visited by the robot during the simulation.

10% of the time, the policy is not followed and the action is selected randomly among all available actions.

We now consider the region R_j connected to R_i and accessible from action a_t. A position (x_j, y_j) in R_j is randomly selected and constitutes the next target to reach. The robot then moves to this position, updates the navigation graph, grabs a new RGB-D input, updates the learner and meta-learner, and determines by Q-learning the next position to reach.

Note that each Q-learning policy is obtained by considering the navigation graph and the reward as constant during the whole simulation. This assumption is not representative of the real world, as each displacement influences both the regions and the learning progress (that would eventually decrease to 0 when the learner cannot be any better). However, the assumption is accurate enough to suggest a displacement. 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 force the progress in a given region to be very high as long as less than three samples are collected in that region. This additional constraint has the same effect as the R-MAX [10] exploration policy.

V. Experimental results

We evaluate in this section the saliency learning and the RL-IAC exploration strategy separately, on datasets constructed slightly differently. Experiments were carried out on both publicly available datasets and datasets recorded on a mobile robot in our laboratory.

A. Saliency maps and object proposal

In this Section, we use three different datasets containing RGB-D images and ground truths, to evaluate both our saliency maps and the bounding box proposals methods.

The first dataset (denoted here as the *ENSTA dataset*) was collected from a pioneer 3DX robot, with a Kinect RGB-D camera mounted at 1 meter from the ground and tilted slightly downward. The robot was equipped with a laser range finder, and a SLAM algorithm (Hector mapping [37]) was used to simultaneously localize the robot in its environment and produce the occupancy grid. To build the dataset, we manually controlled the robot in an office building in order to visit corridors, laboratory, hall and offices. Within this environment, the robot was typically observing chairs, desks, or boxes. We recorded a 15 minutes length video sequence at 5Hz with the robot moving at a 0.5m/s average speed, in which a large variety of views and lightning conditions were captured (See Figure 9). In total, around 4000 RGB-D images were collected this way, associated with the position in the occupancy grid where they were taken.

The second dataset was constructed from the publicly available *Washington dataset*, and more specifically, from the *RGB-D scenes dataset* [41]. This dataset is composed of 8 video sequences of indoor scenes with everyday-life objects placed on tabletops. In total, around 1500 RGB-D frames are available in this dataset along with bounding boxes around objects.

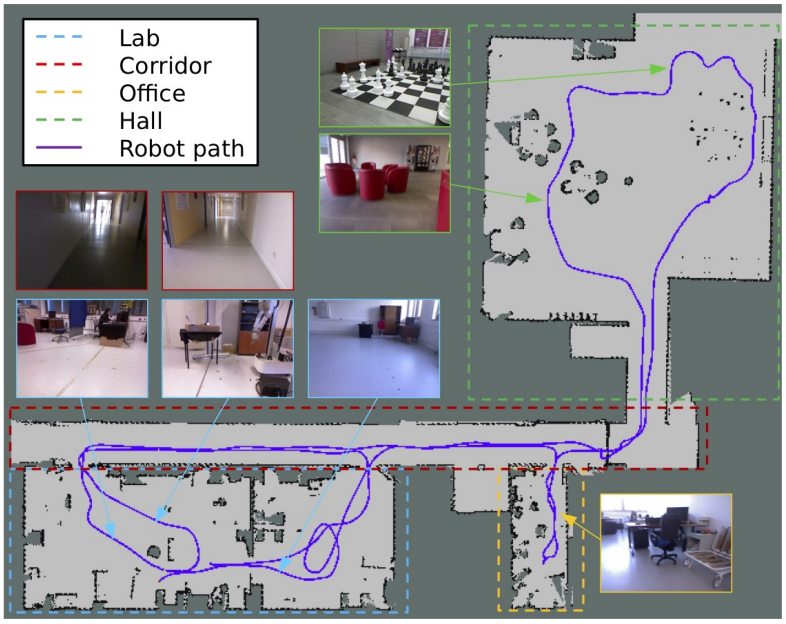


Fig. 9. Map of the ENSTA building recorded in the *ENSTA dataset*.

However, these annotations are not well-suited for object-based saliency evaluation, as they consist in bounding boxes around a limited list of objects (other are just considered as distractors). For saliency, we rather need pixelwise annotations for every object of the scene.

The third dataset is the *Ciptadi dataset* [15], designed for RGB-D saliency evaluation. This dataset was recorded on a mobile robot in a house, and contains everyday-life objects and scenes. In total 80 RGB-D frames are available, with a pixelwise saliency annotation. This dataset is used for two purposes: first, to confirm that our saliency technique is able to generalize from a dataset to another one. Second, to validate the performance of our method with annotations are not produced by our team, thus enhancing the reliability of our experiments.

To evaluate the performance of our saliency technique, we manually labeled 100 randomly chosen images from the *ENSTA dataset* and *RGB-D scenes*. Annotations were done to be consistent with our definition of saliency. As a recall, we defined saliency objects as elements standing on planar surfaces (either floor or table), that can be detected by our segmentation technique. Note that this can be contradictory with bottom-up saliency, e.g. red plugs on a white wall will not be salient and white furniture in front of a white wall will be salient. Annotations are such that we have a ground truth masks and a list of bounding boxes around salient objects. Those frames were removed from the dataset and used for evaluation only.

To evaluate the saliency model, we analyze the final performance reached by the classifier when all samples of the training set are used. We denote in this section our incremental saliency learning approach and produced saliency maps as ISL.

1) Depth segmentation vs. RGB saliency: We first demonstrate the capacity of the saliency model to generalize what was learned from the depth segmentation, and produce a reliable estimation of the saliency on the whole image. This generalization is made possible by two factors: first salient objects often show common visual properties. The classifier is able to find those properties and is therefore able, to a certain extent, to find salient objects that were not detected by the object detector. Second, the datasets are such that the same objects are visible at different point of views. This way, if the object detector fails at identifying an object for a given frame, this object may be detected for other point of views. The classifier then extrapolates those point of views and is able to retrieve undetected salient objects.

Figure 10, second row, shows a set of segmentation results. Recall that for a segmentation mask, black and white pixels represent "*not salient*" and "*salient*" portions of the image. Grey pixels of the segmentation mask represent areas where the algorithm could not clearly determine the state of saliency. Except cases where the segmentation fails (sample 6) because of a bad plane estimation, the segmentation mask produces a pretty reliable saliency segmentation. However, this segmentation is only partial because of the many undetermined areas, thus making the incremental learning of saliency (third row) useful to recover missing data. For example, the segmentation algorithm is such that nothing salient can be detected further than 4 meters away (samples 4, 5), but saliency estimation is applied on the whole image and the generalization capability of the classifier makes it possible to detect salient objects further than four meters. Second, reflective surfaces are often hard to detect by the Kinect sensor (computer screen on sample 3, black plastic on sample 2). However, the aspect of salient reflective objects can be partially learned and fully retrieved based on the RGB data only. Third, the segmentation algorithm is very restrictive and is often not able to detect salient elements if they are in contact with a border of the image (samples 1, 2,5) or badly clustered by the segmentation (mobile container of sample 3 mixed with the floor). Con-

versely, the saliency algorithm provides an estimate of saliency even if the object is partially cut, occluded, or captured with a poor image quality.

2) Saliency map accuracy: To demonstrate the accuracy of our saliency model, we selected three publicly available saliency algorithms and computed the ROC curves for each method on each of the three datasets. First, **SALGAN** [53] is among the most accurate RGB saliency methods according to the *MIT saliency benchmark* [11]. Second, we use the the **DSS** algorithm [43] that produces an object-oriented kind of saliency and is one of the best performing method on the object-based ECSSD benchmark [66]. Third, we compare our method with saliency maps produced with the **CAM** [73] model. This model is trained to detect objects among the 1000 classes of ILSVRC. For a fair comparison, we disabled classes that were not present in the images of our datasets (i.e. their output score were systematically set to 0), so that the produced saliency maps were responsive to relevant objects only. In addition, the maps produced by the CAM approach have the same low resolution than our model. We therefore apply the superpixels approach presented in Section III-D to increase the resolution of these maps.

To evaluate our feature extractor versus the one proposed in previous work [19], we generate saliency maps from both the CNN-based feature extractor (denoted as **ISL** here), and the former feature extractor (denoted as **ISL-Make3D**). Last we evaluate the performance of the **segmentation**. As saliency is not estimated on the whole image in this case, we replace pixels labeled as "*unknown*" or "*unavailable*" by a "*not salient*" label. For the Ciptadi dataset, Ciptadi *et al.* have proposed a set of saliency maps for comparison on this dataset. We then present Ciptadi's results on this dataset only.

The results of the ROC-based evaluation are reported in Figure 11 and suggest that ISL significantly outperforms the evaluated bottom-up and top-down techniques on both *ENSTA* and *RGB-D scenes* datasets. Although slightly below ISL, our technique trained with the Make3D features is still performing well and confirms this trend. These result were expected on those two datasets, because our model is trained from a learning signal that is close to the ground truth, in a specific environment. Surprisingly, DSS is performing quite poorly for an object-based state-of-the-art saliency. We believe that is is because DSS tends to enhance the most saliency element mostly. As the ECSSD mainly consist of images with a single or few salient objects, this feature is not penalized in this dataset. However, our datasets are more challenging and usually contain several salient objects at the same time, thus making DSS less efficient.

To demonstrate that the trained model is also usable in other kinds of environments, we use the Ciptadi dataset. The Ciptadi dataset has its own annotations on a pixelwise basis, so that results can be objectively compared with state-of-the-art. For this dataset, we obtain our results by training ISL on the whole RGB-D scenes dataset, and creating saliency maps with this model. This time again, ISL is the best performing approach, but the Make3D version has a very poor performance. This result is explained by the very good generalization capacity of the CNN types of features.

We also observe that all of the evaluated techniques outperform the depth segmentation. This is because the ROC curve is estimated on the whole image, while segmentation only returns a partial saliency estimation. As missing information is replaced by "*not salient*" labels, the produced ROC curve has a low true positive rate.

In figure 12, a visual comparison between the saliency maps is presented. Samples 1 and 2 are from the Ciptadi dataset. For these samples, the results provided by ISL do not look as neat as in the other datasets. This is because ISL is used in an environment it was not trained for. Regarding the performance of ISL for the other samples, the superpixel reconstruction approach makes it possible to retrieve shapes of salient objects (in spite of the low-resolution feature maps produced by the output of the CNN). When applied to the CAM saliency map, the superpixel reconstruction does not provide such good results. This might be because the produced saliency is much more diffuse (as a comparison, the CAM algorithm is displayed samples 1 and 2 without superpixel reconstruction). As explained earlier, we notice that DSS tends to enhance only a few salient objects or portion of objects, thus leading to a lot of false negatives in the evaluation. Second, ISL is learned from a segmentation derived from a depth map. This way, salient and not salient elements are determined from geometrical criteria rather than from RGB textures. As a results, ISL avoids the detection of distractors such as windows or trees outside (sample 5), or red power outlet (samples 4, 6), that are visually salient but irrelevant for an indoor mobile robot. Lastly, it enhances elements that are not naturally salient (mobile container on sample 6) but consistent with our definition. The saliency maps produced by the Make3D features have a better capacity for retrieving fine details of objects than ISL (samples 2, 6), as this type of feature extraction does not decrease the original resolution. However, ISL based on deep features have a much better generalization capability. Considering samples 1 and 2, where training was done on another dataset, or the chessboard of sample 6, where salient elements (the pawns) are of the same color than the ground (black and white squares), ISL clearly provides a much better saliency estimation.

3) Bounding box proposals: We now demonstrate that ISL can be used to produce relevant bounding boxes around objects. To this end, we run the EdgeBoxes [75] algorithm for each frame of the *ENSTA* and *RGB-D scenes* evaluation sets and keep the 100 best ranked bounding boxes along with their h_b^{in} scores. These boxes are used as a reference to evaluate our method. Then, these EdgeBoxes are re-ranked based on the saliency map to make boxes containing salient pixels better ranked than others. For that, we rank each of these boxes according to the SCscore defined in Section III-E. To demonstrate the ability of our saliency map to produce relevant box proposals, we calculate an SCscore based on ISL saliency maps, and another one based on BMS [71] saliency maps (denoted as **EB+ISL** and **EB+BMS** in Figure 13). Lastly, we generate the SegBoxes from the depth segmentation process, as described in Section III-E. We also produce an SCscore for each of them. We filter out Segboxes having a low SCscore (below 0.2 in our case). Those SegBoxes, obtained from depth

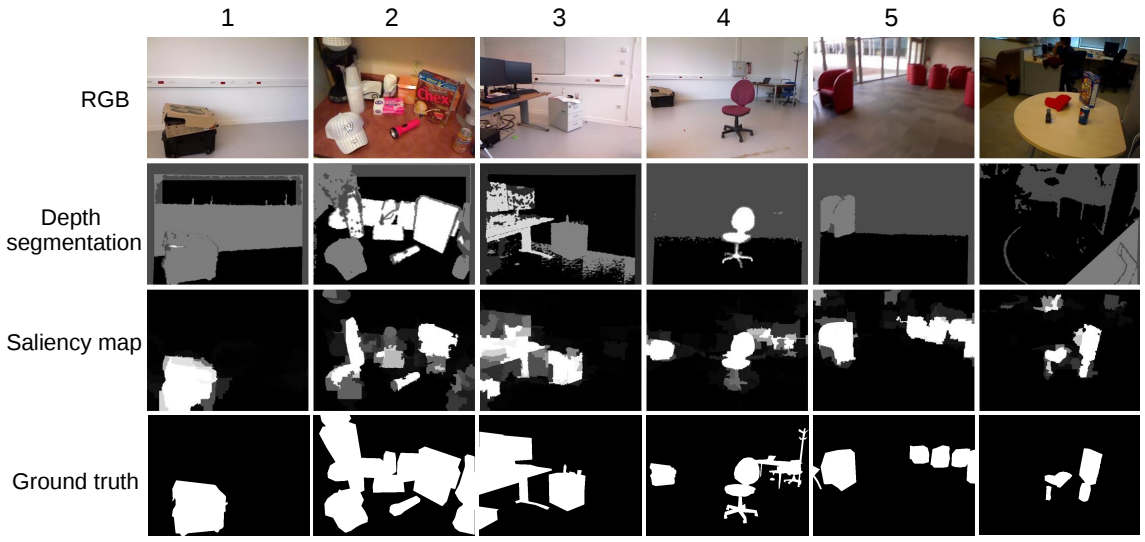


Fig. 10. Generalization capabilities of the classifier. Compared with the ground truth, the depth segmentation mask is reliable, but partial. Conversely, saliency maps provide an estimation for each pixel of the input image.

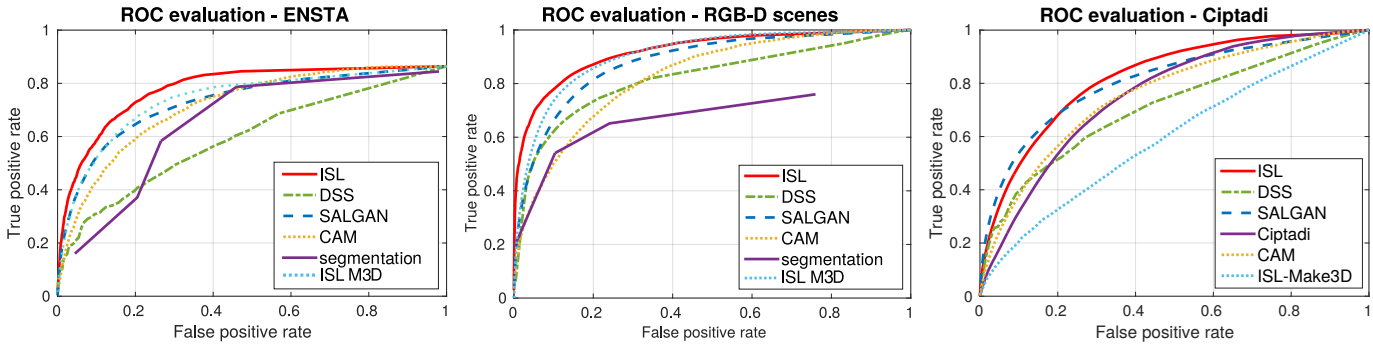


Fig. 11. ROC curves of several saliency approaches on three different datasets

segmentation are complementary to the RGB-based EdgeBoxes and allow the detection of additional relevant boxes. The remaining SegBoxes are reported as **SegBoxes** in Figure 13. In practice, a small number of SegBoxes are detected on each frame (between 0 and 7 on average). Lastly, we combine the re-ranked EdgeBoxes and the SegBoxes to produce a better set of box proposals. This approach is presented in Figure 13 and Figure 14 as **EB+ISL+SB**.

The evaluation metric is the detection rate versus the number of proposal, based on the *intersection over union* measure (IoU=0.5 here) to count the number of detections. This measure is used by Zitnick *et al.* [75] to evaluate their performance over state of the art approaches. To obtain the detection rate for N proposals, we consider the N best ranked box proposals and measure the proportion of boxes in the ground truth that have an IoU score over 0.5 with at least one of the proposals.

Numerical results are reported in Figure 13 for both *ENSTA* and *RGB-D scenes* datasets. As expected, the use of ISL maps to improve the EdgeBoxes ranking allows a much better detection rate on both datasets. Moreover, using a bottom-up saliency map such as BMS instead of ISL does not show significant improvements on both datasets. The SegBoxes usually propose relevant candidates, possibly not detected by the EdgeBoxes. Because they are complementary to the EdgeBoxes, combining the two approaches significantly improve the detection rate on both datasets. However, the number of proposals is low (between 0 and 7 most of the time), and they do not cover the entire image as they are produced from the depth segmentation.

Figure 14 shows sample results of the top 5 EdgeBoxes (column **EdgeBoxes**), top 5 EbdgeBoxes re-ranked by the SCscore with ISL (displayed in column **EB+ISL+SB**, blue boxes), and Segboxes (same column, yellow boxes). The SegBoxes almost always provide relevant boxes, but many objects are also missed this way, either because they are too far to be segmented (sample 3), or because segmentation failed (sample 4). In this case, the remaining objects locations are recovered by the EdgeBoxes. Again, the use of ISL to rank the EdgeBoxes favors boxes that surround salient elements while removing distractors such as windows (sample 3). Lastly, it is possible to cope with frames that do not contain any salient object (sample 2) by filtering boxes with an SCscore below a certain threshold (0.01 in our case).

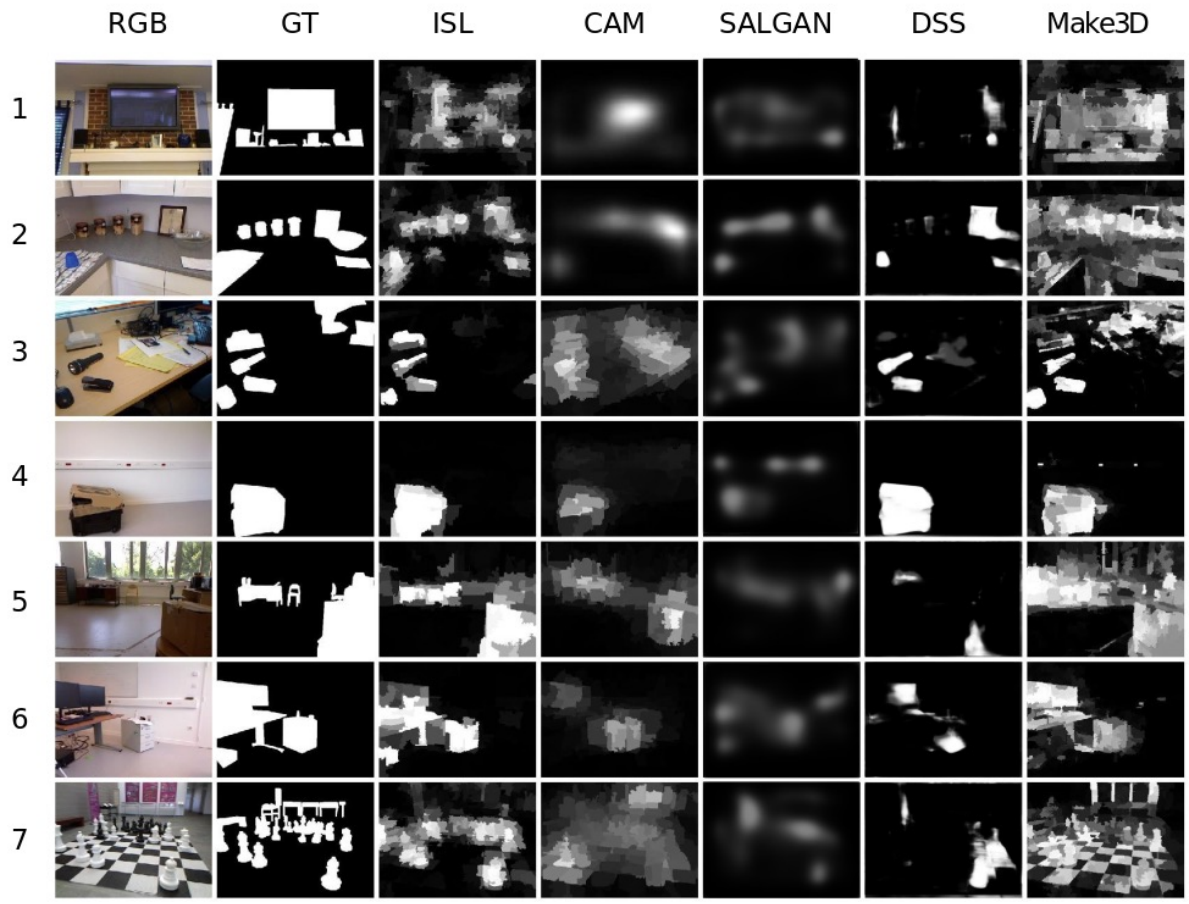


Fig. 12. Comparison of several state of the art approaches with ISL, on Ciptadi (sample 1 and 2), RGB-D scenes (sample 3) and ENSTA (samples 4, 5, 6 and 7) datasets

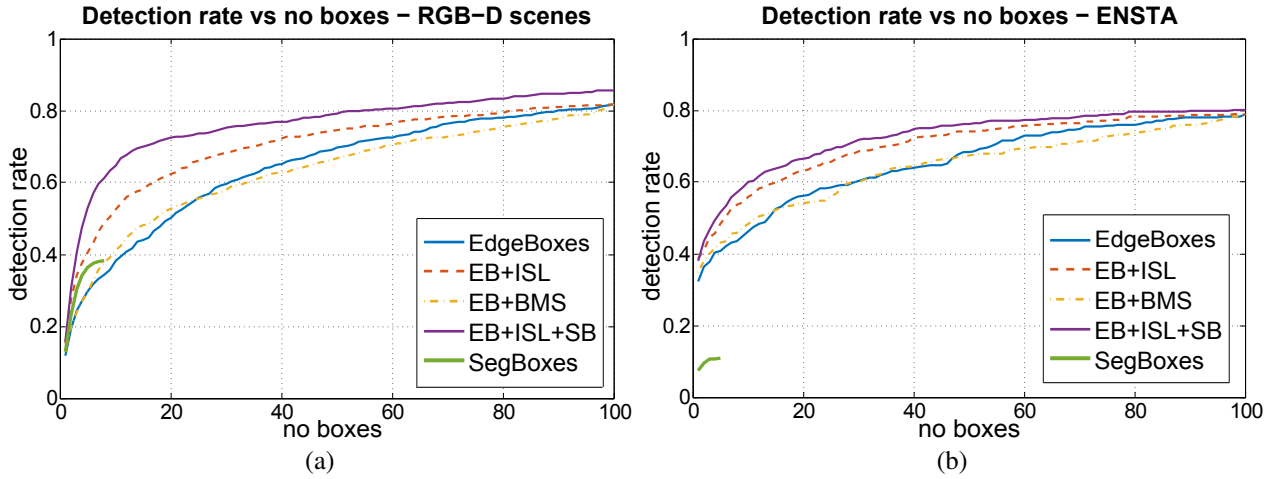


Fig. 13. Detection rate vs number of boxes proposal comparison on the *RGB-D scenes* dataset (a) and ENSTA dataset (b).

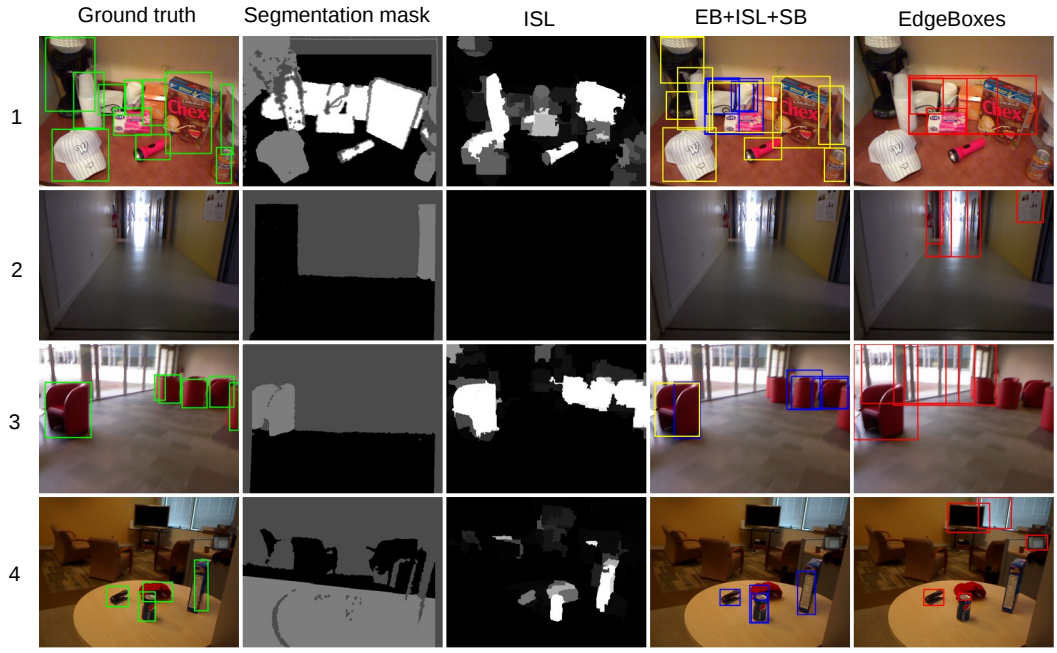


Fig. 14. Sample results of bounding box proposals versus EdgeBoxes

B. RL-IAC

1) Experimental setups: A strong limitation when using robots, especially in scenarios involving online learning, is the reproducibility of the experiments. If a single experiment requires the robot to explore a building for hours, the total number of possible trials is rather limited, and the efficiency of a method may be hard to analyze. For that reason, we rely on semi-simulated setups to run a large number of experiments in parallel without any particular user monitoring. The semi-simulated setups are created from the recorded sequences of the *ENSTA* and *RGB-D scenes datasets*. We call them semi-simulated as saliency is learned from real images taken from these sequences, but actions taken by the robot are simulated.

The first setup is constructed from the *ENSTA dataset*. From this sequence, we build before our experiments a navigation graph based on the technique described in Section IV-A (although map building and saliency learning could be run simultaneously). The navigation graph used in all our experiments is the one represented in Figure 7. Proto-regions were arbitrarily defined to be of 5 meters length. We then considered the positions of each observation recorded during the sequence, and we associated each of them to a region. When selecting a position in a given region, we select one among all associated frames, we consider the position of this frame, and we simulate the displacement of the robot to reach it. Once attained, we use this observation to update our model. To get an overview of the incremental map building and the simulated robot displacement, a video is available on the project's webpage[3].

The two other datasets consist in artificial buildings constructed from the *RGB-D scene dataset*. Each of the eight video sequences of the dataset is recorded in a single particular room (kitchen, office, meeting room), so that our artificial building contains rooms (one for each sequence) divided into 5 to 6 regions, with some regions connected to other rooms (as if there were doors and corridors between rooms). We created two different building configurations, illustrated by Figure 15. The first artificial building, denoted as the *short corridor building*, is composed of five of the video sequences, and contains a short corridor of three regions to switch between rooms. The second one, denoted as the *long corridor building*, is composed of the eight video sequences and contains three long corridors. To construct the navigation graph in each room, we cut each of the sequence into five or six sub-sequences of equal length, and we created an arbitrary trajectory to travel across the sequences. We also limited the number of connections per region to four. This corresponds to the four possible actions the robot can take, namely "*up*", "*down*", "*left*" and "*right*".

To simulate the displacements of the robot, we considered the following sequence of steps:

1) The robot grabs an RGB-D frame, extracts features and segmentation, and updates the meta-learner;
2) the robot determines the next region to visit given learning progress and Q-learning training;
3) the robot determines the next position to reach in this region;
4) the robot moves to this position;
5) while moving, the robot updates the learner based on the previous RGB-D frame;
6) before taking the next RGB-D frame, the robot waits for the displacement and the learner update to be both finished.

In our experiments, this sequence was repeated 3000 times. Each estimated error rate was timestamped with the simu-

[3] https://github.com/cececr/RL-IAC

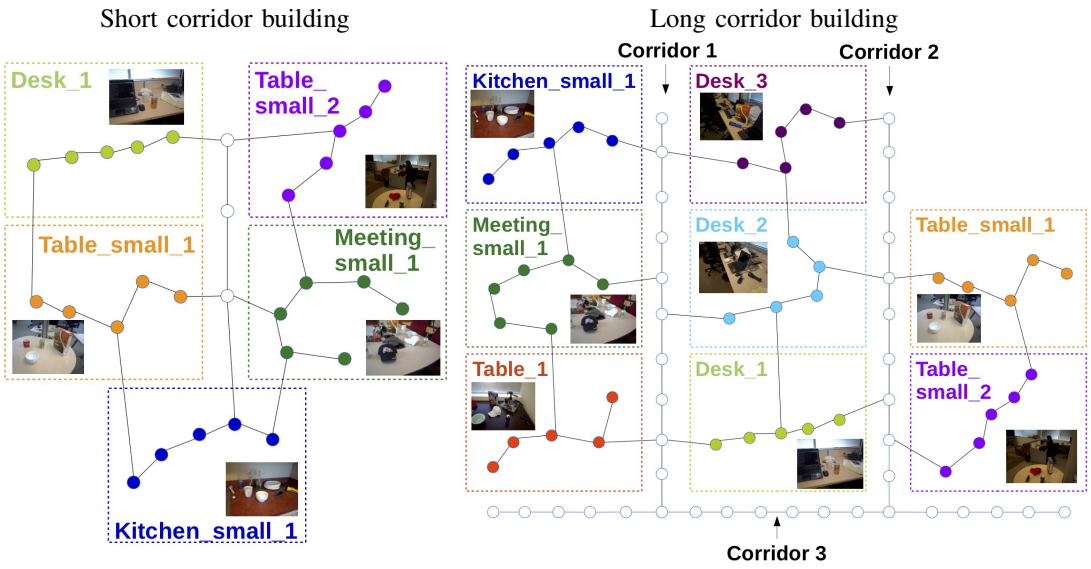


Fig. 15. *short corridor* and *long corridor* artificial buildings created from the *RGB-D scene dataset*

Processing	Min time	Max time
Meta-learner update	100 ms	150 ms
Q-learning training	250 ms	300 ms
Learner update	23 ms	13.5 s
Robot displacement	0 ms	22 s

TABLE I
MIN AND MAX PROCESSING TIME FOR THE MAIN STEPS OF RL-IAC

lated time starting at the beginning of the experiment. This timestamps was then used to plot our results. To obtain the simulated time, we measured for each iteration the time spent by the system to compute steps 1 to 3 (not simulated), and we added the longest step between steps 4 and 5, as they are supposed to be run in parallel and wait for the other to be finished. Table I provides additional measurements to get a better overview of the execution time for each step.

To simulate the robot's displacement, we considered an average speed of 0.5 m/s, and we measure the time for a robot to reach a certain point by considering the euclidean distance to this point and a constant speed of 0.5 m/s. The maximum displacement time is then bounded by the distance between two adjacent regions. To get a rough idea of the simulated time for a single experiment, the five steps of an iteration take on average 10 seconds. Given the 3000 successive iterations, an experiment then lasts for 8 hours.

2) Evolution of the saliency: We first look at the evolution of the saliency quality during incremental learning. Figure 16 first shows a qualitative example of the evolution of the saliency at a given point of view, while the sequence is used in chronological order for training the classifier. We can observe the generalization capability because even before the seat was observed (in frame 400), the classifier is already able to recognize it as a salient element, because it has already learned a partial model of the background. For a better visualization of this evolution, a video is also available online on the project's webpage.

3) Exploration efficiency: To demonstrate the benefits of exploring the environment using RL-IAC, we now compare the evolution of the saliency with different exploration strategies on the three datasets. In a previous publication [19], we demonstrated that RL-IAC was outperforming the exploration based on IAC (described in [18]) that behave similarly to the random exploration in this particular context. We here investigate other types of explorations.

In mobile robotics navigation, the goal is generally to have a good coverage of the environment to explore so as to get an accurate mapping. Our goal is not to make a mapping of the environment, but using an exploration based on an extensive and efficient coverage of the environment is a good baseline to compare with. For this reason, our first exploration strategy consists in determining an exploration pattern covering the whole regions of the environment, and repeating this pattern until the end of the experiment. In Osswald *et al.*, pre-defined map and navigation graph is used as a prior for exploration. An efficient exploration route is obtained by solving a traveling salesman problem (TSP) in their map, based on the Concorde software [3]. Similarly, we then used Concorde with our regions configurations, to find an optimal map coverage that is to be used in our experiments.

A few approaches rely on learning progress to guide exploration in a reinforcement learning context. In particular, Schmidhuber [64] or Lopes *et al.* [46] have used Q-learning to guide the robot's displacements in a context where the only reward is the learning progress. This kind of approach is very similar to RL-IAC, but differs at a critical point: while a single Q-learning is run during the experiment and directly decides the next action of the robot in Schmidhuber's approach, we define and solve a new problem with Q-learning after each robot's displacements. We then use the entire problem to find the next best displacement rather than following a policy from a partially trained Q-matrix. Our second approach to compare with is then following Schmidhuber's approach: instead of

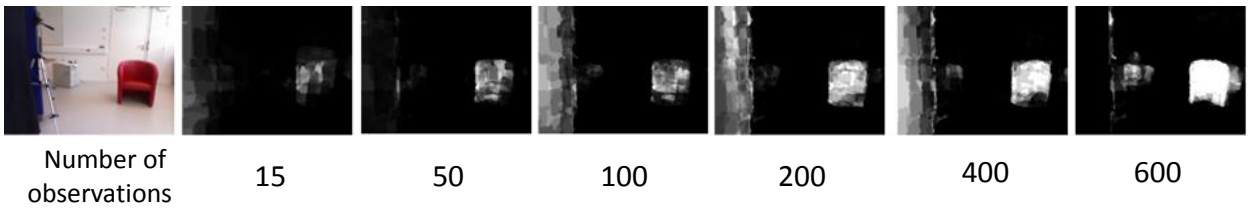
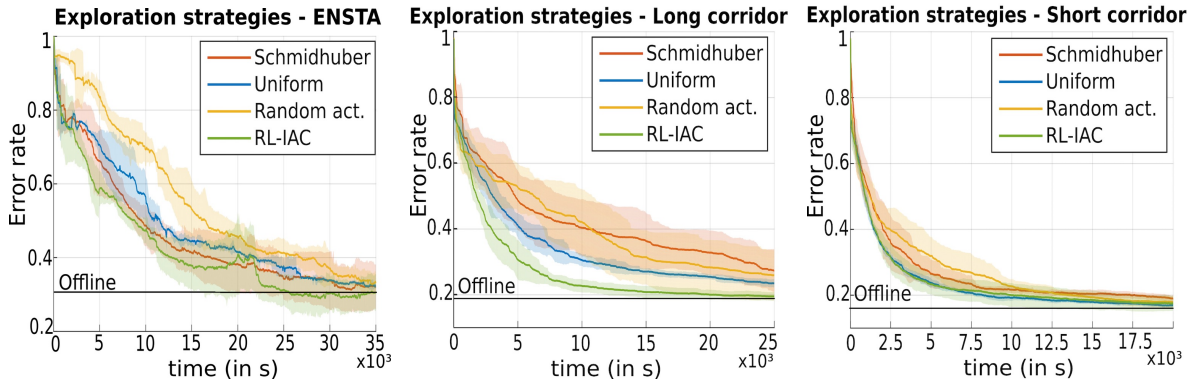


Fig. 16. Evolution of the saliency as number of observation increases. In this example, the frame were learned in chronological order, and the seat was observed for the first time only after 400 frames.



(a) (b) (c)

Fig. 17. Error rate evolution for several exploration strategies on three different environments

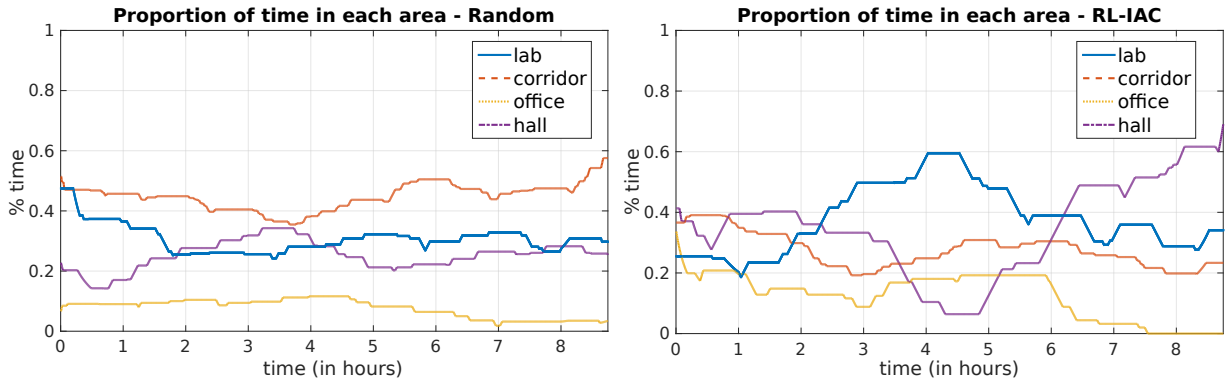
running virtual displacement simulations to train our Q-matrix, we run a single update of the Q-matrix after arriving in a given region. We also use an ϵ-greedy approach (50% random) to decide the next action to take.

Each exploration strategy was tested 10 times on each dataset and results are reported based on the average and variance over those experiments. The performance of the system was evaluated using the evolution of the *overall error rate* of the system: based on the reference frames on which a ground truth is available, we compare the estimated saliency map for all of these frames with the available ground truth. We then use the formula provided by Equation 2 on each frame and take the average error. 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.

Figure 17 shows the evolution of the *overall error rate* in time on both environments, for the 4 exploration strategies:

- **RL-IAC**: As described in Section IV. Selects the next region to visit from the Q-matrix, and the next position to reach in that region randomly.
- **Uniform**: We drive the exploration by a uniform coverage of the environment, from the sequence of regions determined with the TSP heuristic. This pattern is repeated until the end of the experiment. The next region to visit is determined from the sequence, and the next position to reach in that region is taken randomly.
- **Schmidhuber**: Similar to RL-IAC, except that the Q-matrix is updated after each observation rather than running a batch of simulations.
- **Random act.**: To get a worse case scenario, we select a random action to reach a region, and random position in that region.

On all datasets, RL-IAC is the method with the fastest decreasing error. The uniform exploration has a reasonable performance, even similar to RL-IAC in the short-corridor dataset. This can be explained by the fact that this setup only has a very small number of uninformative regions. RL-IAC, by evaluating progress, is precisely efficient at detecting such uninformative regions. This is even more visible in the *large corridor* experiment, where almost 50% of the regions are part of the corridors, which are typically uninformative. Schmidhuber has a varying performance depending on the dataset. We actually found this approach very sensitive to the parameters of the experiment, and performing well with very different parameters than RL-IAC. For example, to converge rapidly enough, a large percentage of random actions were necessary (typically 50%), while RL-IAC only works with 10% of random actions. Lastly, and as expected, the random action is providing the worse performance, sometimes close to Schmidhuber's approach. As a comparison, we also plot the error rate of the model trained offline (constant in time). The offline version performs roughly the same as the online one when enough samples have been acquired. However, the main difference is that the online version is flexible to changes in the environment, while the offline is not.



(a) (b)

Fig. 18. Time spend in different parts of the environment for (a) random position selection (b) RL-IAC

4) Time allocation in the environment: To get a better insight of the way exploration is done by the robot, we divide the building in 4 main areas, namely *lab*, *office*, *corridor* and *hall* (See Figure 9). In these areas the difficulty to learn saliency is not the same. For example, the corridor does not contain any salient element, whereas the hall is a very large room with many salient items and many distractors. We compare in Figure 18 the average percentage of time spent in each area when using RL-IAC and when using random (**Random act.**) exploration strategies. The graphs have been constructed by using a sliding widow of 500s over the whole experiment, and measuring for each window the number of frames obtained in each area. For random exploration, the time spent in each area is roughly the same all along the sequence and proportional to each area size. 40% to 50% of the time is spent in the corridor, whereas 10% is spent in the office. With RL-IAC, the time spent in the corridor (the least "interesting" area) oscillates between 30% and 20%, except at the beginning, and almost 20% is spent in the office. Moreover, the time spent in exploring each area is evolving in time: the time spent in the office finally decreases to 0%, because no progresses are made in there anymore. In the middle of the sequence, most of the time is spent exploring the lab, while most of the time is spent in the hall at the end of the exploration.

VI. Discussion

We took the assumption that salient elements are objects. This is of course a restrictive case, but this assumption holds in many indoor applications. Any definition of saliency could be used to replace the one chosen here, as soon as an appropriate learning signal can be used to learn this kind of saliency. For example, in [18], we propose another kind of learning signal and another kind of saliency, that is provided by a foveated platform.

Whatever the saliency definition is, our approach will depend on the quality of the learning signal. In the current work, this information is of good quality at short range, but very partial as it does not give information close to the image borders and at long range. It would be interesting to study what are the ideal characteristics of this learning signal, for example if a lower quality but more complete signal would be relevant, or if improving the current segmentation quality would lead to a noticeable final performance increase.

In this paper, the robot is only exploring its environment and thus takes all decision in order to improve its saliency model. Such situation would be rare in real-world scenarios, but our approach can be easily integrated into a robot that has other tasks to fulfill as there is no theoretical problems in mixing intrinsic and extrinsic motivations [14]. This would result in a robot opportunistically exploring to improve its saliency model, for example taking a route through a less known area while going to its charging station.

VII. Conclusion

In this article, we have presented a full architecture for learning to localize objects within a robot's exploration in an incremental and autonomous way. On the one hand, we described the main mechanisms for learning a model of visual saliency from a depth-based learning signal, and how to exploit this saliency model to general bounding boxes around salient objects of the scene. On the other hand, we investigated how the robot could methodically explore its environment to learn the saliency model faster and better. We proposed the RL-IAC approach to guide exploration in that regard, by finding the best compromise between robot's displacement and learning. We have carried out several experimentation to demonstrate the accuracy of our saliency maps as compared with other state-of-the-art approaches, and the efficiency of our exploration technique.

A critical aspect that should be consider in future work is the use of an end-to-end deep learning framework that would both produce saliency and bounding box proposals. We so far separate feature extraction, feature combination, and bounding boxes generation, but deep learning offers a way to integrate all these components at the same time. This could take the form of a fully convolutional network that would produce both saliency and boxes. This could for example resemble the SSD architecture [45]. Additionally, neural networks are by essence online classifiers, which may be better suited than the proposed method based on random forests. Although incremental learning with deep neural network is still at an

early stage, we could simplify the problem by only fine-tuning a small part of the network. Alternatively, various concise CNN models such as binary CNN could be used to increase the efficiency.

Two other possible directions would be worth investigating. First, running the incremental map building and RL-IAC at the same time. This way, the navigation graph would be constructed from scratch, without any prior environment exploration. Second, we would like to carry out experiments in a non simulated setup to have a fully operational system.

ACKNOWLEDGMENT

The authors would like to thank the INRIA Flowers team, and especially Pierre-Yves Oudeyer for the valuable help on the IAC aspect.

REFERENCES

[1] Bogdan Alexe, Thomas Deselaers, and Vittorio Ferrari. Measuring the objectness of image windows. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(11):2189–2202, 2012.

[2] 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.

[3] David Applegate, Ribert Bixby, Vasek Chvatal, and William Cook. Concorde tsp solver, 2006.

[4] 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.

[5] Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 1998.

[6] Stéphane Bazeille and David Filliat. Incremental topo-metric slam using vision and robot odometry. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 4067–4073. IEEE, 2011.

[7] Mårten Björkman and Danica Kragic. Active 3d scene segmentation and detection of unknown objects. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 3114–3120. IEEE, 2010.

[8] Ali Borji, Majid Nili Ahmadabadi, Babak Nadjar Araabi, and Mandana Hamidi. Online learning of task-driven object-based visual attention control. *Image and Vision Computing*, 28(7):1130–1145, 2010.

[9] Ali Borji and Laurent Itti. State-of-the-art in visual attention modeling. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(1):185–207, 2013.

[10] 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.

[11] Zoya Bylinskii, Tilke Judd, Frédo Durand, Aude Oliva, and Antonio Torralba. Mit saliency benchmark. http://saliency.mit.edu/.

[12] 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.

[13] Ming-Ming Cheng, Guo-Xin Zhang, Niloy J Mitra, Xiaolei Huang, and Shi-Min Hu. Global contrast based salient region detection. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 409–416. IEEE, 2011.

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

[15] Arridhana Ciptadi, Tucker Hermans, and James M Rehg. An in depth view of saliency. In *Eds: T. Burghardt, D. Damen, W. Mayol-Cuevas, M. Mirmehdi, In Proceedings of the British Machine Vision Conference (BMVC 2013)*, pages 9–13, 2013.

[16] R. Cong, J. Lei, H. Fu, Q. Huang, X. Cao, and C. Hou. Co-saliency detection for rgbd images based on multi-constraint feature matching and cross label propagation. *IEEE Transactions on Image Processing*, 27(2):568–579, Feb 2018.

[17] Céline Craye, David Filliat, and Jean-François Goudou. On the use of intrinsic motivation for visual saliency learning. In *2016 Joint IEEE International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob)*. IEEE, 2016.

[18] Céline Craye, David Filliat, and Jean-François Goudou. Biovision: a biomimetics platform for intrinsically motivated visual saliency learning. *IEEE Transactions on Cognitive and Developmental Systems*, 2018.

[19] Celine Craye, David Filliat, and JF Goudou. Rl-iac: An exploration policy for online saliency learning on an autonomous mobile robot. In *Intelligent Robots and Systems (IROS), 2016 IEEE International Conference on*, 2016.

[20] Aleksandrs Ecins, Cornelia Fermüller, and Yiannis Aloimonos. Cluttered scene segmentation using the symmetry constraint. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 2271–2278. IEEE, 2016.

[21] Erkut Erdem and Aykut Erdem. Visual saliency estimation by nonlinearly integrating features using region covariances. *Journal of vision*, 13(4):11, 2013.

[22] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

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

[24] Simone Frintrop, Erich Rome, and Henrik I Christensen. Computational visual attention systems and their cognitive foundations: A survey. *ACM Transactions on Applied Perception (TAP)*, 7(1):6, 2010.

[25] Simone Frintrop, Thomas Werner, and Germán Martín García. Traditional saliency reloaded: A good old model in new shape. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 82–90, 2015.

[26] H. Fu, D. Xu, S. Lin, and J. Liu. Object-based rgbd image co-segmentation with mutex constraint. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4428–4436, June 2015.

[27] Germán M García, Ekaterina Potapova, Thomas Werner, Michael Zillich, Markus Vincze, and Simone Frintrop. Saliency-based object discovery on rgb-d data with a late-fusion approach. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1866–1873. IEEE, 2015.

[28] Fred H Hamker. The emergence of attention by population-based inference and its role in distributed processing and cognitive control of vision. *Computer Vision and Image Understanding*, 100(1):64–106, 2005.

[29] Jan Hosang, Rodrigo Benenson, Piotr Dollár, and Bernt Schiele. What makes for effective detection proposals? *IEEE transactions on pattern analysis and machine intelligence*, 38(4):814–830, 2016.

[30] Q. Hou, M.-M. Cheng, X.-W. Hu, A. Borji, Z. Tu, and P. Torr. Deeply supervised salient object detection with short connections. *ArXiv e-prints*, November 2016.

[31] Xiaodi Hou, Jonathan Harel, and Christof Koch. Image signature: Highlighting sparse salient regions. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(1):194–201, 2012.

[32] Xiao Huang and John Weng. Novelty and reinforcement learning in the value system of developmental robots. In *Second International Workshop on Epigenetic Robotics: Modeling Cognitive Development in Robotic Systems*. Lund University Cognitive Studies, 2002.

[33] Nevrez Imamoglu, Wataru Shimoda, Chi Zhang, Yuming Fang, Asako Kanezaki, Keiji Yanai, and Yoshifumi Nishida. An integration of bottom-up and top-down salient cues on RGB-D data: saliency from objectness versus non-objectness. *Signal, Image and Video Processing*, 12(2):307–314, 2018.

[34] Laurent Itti and Christof Koch. Computational modelling of visual attention. *Nature reviews neuroscience*, 2(3):194–203, 2001.

[35] 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.

[36] Islem Jebari, Stéphane Bazeille, and David Filliat. Combined vision and frontier-based exploration strategies for semantic mapping. In *Informatics in Control, Automation and Robotics*, pages 237–244. Springer, 2012.

[37] S. Kohlbrecher, J. Meyer, O. von Stryk, and U. Klingauf. A flexible and scalable slam system with full 3d motion estimation. In *Proc. IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*. IEEE, November 2011.

[38] 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.

[39] Danica Kragic. Object search and localization for an indoor mobile robot. *CIT. Journal of Computing and Information Technology*, 17(1):67–80, 2009.

[40] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[41] 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.

[42] Mikko Lauri and Risto Ritala. Stochastic control for maximizing mutual information in active sensing. In *IEEE Int. Conf. on Robotics and Automation (ICRA) Workshop on Robots in Homes and Industry*, 2014.

[43] G. Li and Y. Yu. Deep Contrast Learning for Salient Object Detection. *ArXiv e-prints*, March 2016.

[44] G. Li and Y. Yu. Visual Saliency Detection Based on Multiscale Deep CNN Features. *IEEE Transactions on Image Processing*, 25:5012–5024, November 2016.

[45] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[46] 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.

[47] Nikolaos A Massios, Robert B Fisher, et al. *A best next view selection algorithm incorporating a quality criterion*. Department of Artificial Intelligence, University of Edinburgh, 1998.

[48] Silviu Minut and Sridhar Mahadevan. A reinforcement learning model of selective visual attention. In *Proceedings of the fifth international conference on Autonomous agents*, pages 457–464. ACM, 2001.

[49] 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.

[50] Maxime Oquab, Léon Bottou, Ivan Laptev, and Josef Sivic. Is object localization for free?-weakly-supervised learning with convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 685–694, 2015.

[51] Stefan Oßwald, Maren Bennewitz, Wolfram Burgard, and Cyrill Stachniss. Speeding-up robot exploration by exploiting background information. *IEEE Robotics and Automation Letters*, 1(2):716–723, 2016.

[52] 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.

[53] Junting Pan, Cristian Canton, Kevin McGuinness, Noel E. O'Connor, Jordi Torres, Elisa Sayrol, and Xavier and Giro-i Nieto. Salgan: Visual saliency prediction with generative adversarial networks. In *arXiv*, January 2017.

[54] Jeremie Papon, Alexey Abramov, Markus Schoeler, and Florentin Worgotter. Voxel cloud connectivity segmentation-supervoxels for point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2027–2034, 2013.

[55] Houwen Peng, Bing Li, Weihua Xiong, Weiming Hu, and Rongrong Ji. Rgbd salient object detection: A benchmark and algorithms. In *Computer Vision–ECCV 2014*, pages 92–109. Springer, 2014.

[56] P. O. Pinheiro, R. Collobert, and P. Dollar. Learning to Segment Object Candidates. *ArXiv e-prints*, June 2015.

[57] P. O. Pinheiro, T.-Y. Lin, R. Collobert, and P. Dollàr. Learning to Refine Object Segments. *ArXiv e-prints*, March 2016.

[58] Ekaterina Potapova, Karthik M Varadarajan, Andreas Richtsfeld, Michael Zillich, and Markus Vincze. Attention-driven object detection and segmentation of cluttered table scenes using 2.5 d symmetry. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4946–4952. IEEE, 2014.

[59] L. Qu, S. He, J. Zhang, J. Tian, Y. Tang, and Q. Yang. Rgbd salient object detection via deep fusion. *IEEE Transactions on Image Processing*, 26(5):2274–2285, May 2017.

[60] Babak Rasolzadeh, Mårten Björkman, Kai Huebner, and Danica Kragic. An active vision system for detecting, fixating and manipulating objects in the real world. *The International Journal of Robotics Research*, 29(2-3):133–154, 2010.

[61] J. Ren, Xiaojin Gong, L. Yu, Wenhui Zhou, and M. Y. Yang. Exploiting global priors for rgb-d saliency detection. In *2015 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 25–32, June 2015.

[62] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, pages 91–99, 2015.

[63] Joo Machado Santos, Tom Krajnk, and Tom Duckett. Spatio-temporal exploration strategies for long-term autonomy of mobile robots. *Robotics and Autonomous Systems*, 2016.

[64] Jürgen Schmidhuber. Curious model-building control systems. In *Neural Networks, 1991. 1991 IEEE International Joint Conference on*, pages 1458–1463. IEEE, 1991.

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

[66] Jianping Shi, Qiong Yan, Li Xu, and Jiaya Jia. Hierarchical image saliency detection on extended cssd. *IEEE transactions on pattern analysis and machine intelligence*, 38(4):717–729, 2016.

[67] H. Song, Z. Liu, Y. Xie, L. Wu, and M. Huang. Rgbd co-saliency detection via bagging-based clustering. *IEEE Signal Processing Letters*, 23(12):1722–1726, Dec 2016.

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

[69] Juyang Weng, James McClelland, Alex Pentland, Olaf Sporns, Ida Stockman, Mriganka Sur, and Esther Thelen. Autonomous mental development by robots and animals. *Science*, 291(5504):599–600, 2001.

[70] X. Yao, J. Han, D. Zhang, and F. Nie. Revisiting co-saliency detection: A novel approach based on two-stage multi-view spectral rotation co-clustering. *IEEE Transactions on Image Processing*, 26(7):3196–3209, July 2017.

[71] 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.

[72] Qi Zhao and Christof Koch. Learning a saliency map using fixated locations in natural scenes. *Journal of vision*, 11(3):9, 2011.

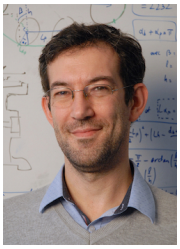
[73] B. Zhou, A. Khosla, Lapedriza. A., A. Oliva, and A. Torralba. Learning Deep Features for Discriminative Localization. *CVPR*, 2016.

[74] Jun-Yan Zhu, Jiajun Wu, Yichen Wei, Eric Chang, and Zhuowen Tu. Unsupervised object class discovery via saliency-guided multiple class learning. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3218–3225. IEEE, 2012.

[75] C Lawrence Zitnick and Piotr Dollár. Edge boxes: Locating object proposals from edges. In *Computer Vision–ECCV 2014*, pages 391–405. Springer, 2014.

Celine Craye Céline Craye received the diplome d'Ingénieur in telecommunications from the Ecole Nationale Supérieure des Télécommunications de Bretagne, France, in 2012. She received her M.A.Sc. degree from the Electrical and Computer Engineering Department at the University of Waterloo in 2013, and her Ph.D. from the University of Paris Saclay in 2017. She is now a research engineer at the Thales VisionLab, the Vision innovation laboratory for video-surveillance of the Thales Group. Her research interests are in computer vision, machine learning and developmental robotics.

Timothée Lesort Timothée Lesort received the diplome d'Ingénieur in electronics with robotics and learning algorithms major from the Ecole CPE Lyon, France, in 2017. He is currently a Ph.D. candidate at ENSTA Paristech in France at the UIIS laboratory. The Ph.D. is granted by the CIFRE program in partnership with Thales Vision Lab. His research interests are in computer vision, incremental deep learning and developmental robotics.



David Filliat David Filliat graduated from the Ecole Polytechnique in 1997 and obtained a PhD on bio-inspired robotics navigation from Paris VI university in 2001. After 4 years as an expert for the robotic programs in the French armament procurement agency, he is now professor at Ecole Nationale Supérieure de Techniques Avancées ParisTech. Head of the Robotics and Computer Vision team since 2006, he obtained the Habilitation á Diriger des Recherches en 2011. He is also a member of the ENSTA ParisTech INRIA FLOWERS team. His main research interest are perception, navigation and learning in the frame of the developmental approach for autonomous mobile robotics. http://www.ensta-paristech.fr/~filliat/

Jean-François Goudou Jean-François Goudou was born in Versailles, France, in 1979. He received the M.E. degree in applied mathematics from the Ecole Polytechnique, Palaiseau, France, in 2004, and the Ph.D. degrees in computer vision from Telecom ParisTech, Paris, France, in 2007. In 2008, he joined the Advanced studies department Theresis, from Thales, as a project manager for collaborative national and European projects. He is since 2009 in charge of the demonstrators of the VisionLab, the Vision innovation laboratory for video-surveillance of the Thales Group. He is also leading several research projects in the topics of video-surveillance and algorithmic evaluation. He is since 2016 deputy head of the VisionLab, in charge of academic co-operations and H2020 projects proposals and management. His research interest include the complete sensing chain from camera to processing, the human vision and bio-inspired processing and neural networks, including deep learning.