Intrinsic motivation mecanisms for incremental learning of visual saliency

Céline Craye

NNT : 2017SACLY006

THÈSE DE DOCTORAT
DE L'UNIVERSITÉ PARIS-SACLAY
PRÉPARÉE À L'ENSTA PARISTECH

École doctorale n°573
INTERFACES
Spécialité de doctorat: Informatique

par

MME CÉLINE CRAYE

Intrinsic motivation mechanisms for incremental learning of visual saliency

Thèse présentée et soutenue à Palaiseau, le 3 avril 2017.

Composition du Jury :

Prof.	SIMONE FRINTROP	Professor for Computer Vision Universität Hamburg	(Rapporteur)
Dr.	OLIVIER LE MEUR	Maitre de conférence Université de Rennes 1	(Rapporteur)
Dr.	PIERRE-YVES OUDEYER	Directeur de recherche INRIA Bordeaux	(Président du jury)
Dr.	SERGIO A. RODRIGUEZ F.	Maitre de conférence Université Paris-Sud	(Examinateur)
Prof.	DAVID FILLIAT	Directeur de recherche ENSTA Paristech	(Directeur de thèse)
Dr.	JEAN-FRANÇOIS GOUDOU	Ingénieur de recherche Thales SIX	(Co-encadrant de thèse)

Acknowledgements

La voie de la recheche est pour moi un projet de longue date. Je voulais déjà à l'âge de 14 ans devenir chercheuse en chimie. Je n'avais alors aucune idée de ce qu'était la vision par ordinateur ou l'intelligence artificielle, et il est assez fou de réaliser à quel point ces domaines ont évolué rapidement depuis cette époque. Je regarde aujourd'hui mes trois ans de thèse avec une grande fierté, ascenseur émotionnel permanent entre doutes et détermination, exaspération et passion, échecs et succès pour ne finalement garder que le meilleur. J'emprunte aujourd'hui une voie différente de celle que je m'étais fixée il y a des années, laissant de côté la recherche académique pour le monde industriel. Toutefois je garderai une vision éclairée sur la recherche, l'innovation, la technologie, et la transition entre l'académique et l'industrie qui ne manquera pas de me servir à l'avenir. Il est maintenant temps de remercier en bien moins de lignes qu'il n'en faudrait ceux qui ont, de manière délibérée ou non, directement ou indirectement, contribué à ce travail et à cette tranche de vie qui restera probablement l'une des meilleures que j'ai vécue.

Dans un premier temps, je voudrais remercier David qui a été un directeur de thèse parfait. Très à l'écoute, avec une vision claire et précise des problèmes, apte à répondre sur la plupart des questions techniques, et de proposer de très bonnes idées tout en laissant une grande liberté d'orientation. Il restera pour moi un modèle dans la manière d'encadrer des projets de recherche. Je sais qu'avoir un encadrant aussi impliqué est loin d'être courant et je mesure à quel point ma thèse aurait été différente sans lui.

Je remercie également Jean-François qui a représenté le pendant industriel de mon encadrement. Je n'ai pas forcément donné à mon travail toute la dimension bio-mimétique qu'il aurait souhaité, mais espère que mes recherches apporteront tout de même leur modeste contribution à l'édifice de ce très ambitieux projet. Par ailleurs je ne peux qu'être mille fois reconnaissante pour son accompagnement dans la découverte du métier d'ingénieur recherche chez Thales. Je lui dois à la fois mon recrutement en tant que doctorante et celui en tant que docteur, et il a su apporter à ma thèse la dimension entreprise qui m'a été et me sera précieuse pour la suite de ma carrière.

Je remercie mon jury de thèse, dont j'ai beaucoup apprécié la composition, car chacun des membres a eu une influence directe sur mes travaux. Bien entendu, Pierre-Yves Oudeyer qui a suivi depuis Bordeaux mes travaux, et dont les discussions, relectures et conseils m'ont beaucoup aidées. Il reste également pour moi un modèle pour la qualité technique et l'originalité de ses travaux de recherche, mais aussi dans la manière de les valoriser. Simone Frintrop que je n'ai eu le plaisir de rencontrer que durant ma soutenance, mais dont les publications scientifiques ont largement influencées, inspirées et guidées mes recherches. Sans cet apport, mes travaux auraient sans doute pris une autre direction. Olivier le Meur dont j'avais suivi une présentation lors du GdR Isis au tout début de ma thèse, et dont la qualité du travail m'avait marqué et également inspiré et éclairé. Enfin, Sergio Rodriguez, qui m'avait proposé lors de mes recherches de thèses un sujet très intéressant que j'ai bien failli accepter. J'ai finalement opté pour la proposition de Thales, mais notre collaboration n'est peut-être que partie remise.

Il est maintenant temps de remercier un peu plus précisément les deux entités dans lesquelles j'ai passé le plus clair de mon temps. Je ne remercierai évidemment pas l'ANRT, dont l'efficacité administrative m'a coûté sept mois de chômage dans un flou total avant de pouvoir commencer ma thèse. La suite fut heureusement bien plus réjouissante.

Dans un premier temps, je voudrais remercier les doctorants, post-docs, ingénieurs, chercheurs, et stagiaires de l'U2IS. Bien entendu, une pensée toute particulière pour Antoine avec qui j'ai partagé avec une constance presque rituelle de nombreuses « pauses café» à refaire le monde et débattre sur tout et n'importe quoi. À quoi aurait ressemblé ma thèse sans Yuxin, mon voisin de bureau excentrique, dont les habitudes n'ont pas manquées de m'interloquer, m'amuser, m'exaspérer, me toucher tout au long de ces trois ans. Amir, le workoholic acharné, probablement voué à une grande carrière scientifique. Adrien, le théoricien adepte de la méta-discussion. De manière plus générale, merci à ceux avec qui j'ai pu partager des soirées jeux ou encore des repas de groupe. Pauline et Gennaro, Clément P et Clément M ou Louise pour ne citer qu'eux. Je remercie également les membres de l'équipe Flowers que j'ai pu rencontrer à Bordeaux ou à l'autre bout du monde en conférence. Enfin ceux qui ont partagé mon bureau pour des périodes plus ou moins longues au cours de ces trois ans : Alexandre, Mathieu ou encore Timothée. Sur un plan plus pratique, je remercie Louis-Charles dont les travaux de segmentation m'ont beaucoup aidé pour ma propre recherche, Natalia dont le manuscrit m'a fortement inspiré en début de thèse, Panos, avec qui nous parviendrons peut être un jour à co-écrire un article, et enfin Mathieu Lefort, qui, malgré lui, m'a par deux fois aidé à voir et aller plus loin dans ma thèse. Sans lui, RL-IAC n'aurait probablement pas vu le jour.

Je tiens à présent à remercier mes collègues de Thales, qu'ils soient toujours présents à Palaiseau ou déjà partis vers de nouveaux horizons. Tout d'abord Philippe Mouttou puis Thierry Lamarque, ainsi que Catherine Simon puis Alain Marcuzzi pour avoir rendu ma thèse possible au sein de Thales. Merci à Louise Naud également pour avoir proposé ma candidature de thèse au sein du vision lab. Merci maintenant à ceux qui m'ont entouré au cours de ces trois années dans une ambiance très chaleureuse et amicale. Autour d'un café, d'un écran de PC, d'un robot, d'un escape game, d'une ou deux (ou plus !) bières, d'une démo qui marche bien, d'un pain au chocolat pour fêter quelque chose (et il y en a souvent !), devant, derrière ou sous une caméra, au fond d'un train[1] ou dans un vieux grenier[2]. Que chacun comprenant le sens de cette liste le prenne personnellement comme étant parti intégrante de ces remerciements. Un merci tout particulier pour Michael (celui avec un h) avec qui j'ai eu l'occasion de travailler davantage et avec grand plaisir pour monter les démos autour de BioVision. Enfin un remerciement particulièrement ému pour Luigi que j'ai encadré avec beaucoup d'enthousiasme, et dont les dernières semaines de stage ont été particulièrement captivantes.

Pour conclure, je tiens finalement à remercier mes amis. Clément et Louise pour les soirées jeux quasi-hebdomadaires et la pratique assidue du time's up, Kévin, Aurélien, Etienne et JA, mes amis d'enfance que j'ai pris plaisir à retrouver après mon exil brestois et canadien, les « pipos » de Télécom Bretagne avec un souvenir très ému du mariage inattendu de Perez, Julie, Alexis et mes parents, mais aussi ma belle-famille. Bien entendu, une grande partie de ces remerciements vont à Simon qui me suit (à moins que ce ne soit moi ?), m'accompagne, me soutient, me supporte depuis plus de sept ans. Je n'aurais pas pu imaginer meilleur retour à Paris après mes années d'étude au Canada. Malgré le climat défaitiste qui règne en France depuis quelques années, c'est en partant loin qu'on se rend compte que l'herbe n'est pas forcément plus verte dans le jardin du voisin, et qu'on peut y vivre plutôt bien !

[1] Comprenez le wagon bien entendu

[2] Le labo du fond, bien entendu

Contents

Chapter 1

Introduction

1.1 Context

The last few years have shown a strong renewed interest toward artificial intelligence, robotics and perceptual systems. The fast and impressive progresses in these fields suggest that exciting breakthroughs are to be announced in the years to come. Moreover, the biological inspiration is at the heart of this process, providing a rich pool of ideas and examples to truly understand and create intelligence.

This work is part of this framework, and investigates more precisely the problem of designing perceptual systems capable of autonomous behaviors, in a bio-inspired context.

1.1.1 Autonomous perceptual systems

An autonomous perceptual system is a system equipped with one or several sensors, able to behave or execute tasks with a high level of autonomy. This involves on the one hand using the available sensor data to make decisions, and, on the other hand avoiding as much as possible human assistance or intervention. Robots are a good examples of perceptual systems, but they are neither necessarily perceptive, nor autonomous. For example, factory robots designed to accomplish pre-defined sequences of actions are autonomous systems, but they are not perceptual one. On the contrary, drones equipped with cameras are perceptual systems, but they are most of the time remotely controlled by users; therefore, they are not autonomous. Building autonomous perceptual systems is one of the major challenges of this century. In the past decades, considerable advances have been achieved in that regard, but the conception of truly autonomous systems is still several decades ahead. This section presents a few present and future examples of autonomous perceptual systems, as well as the main challenges related to those technologies.

Present and future

Autonomous perceptual systems are already ubiquitous (See Figure 1.1. Among popular examples, one can think of the autonomous vacuum cleaner Roomba, that detects dust, avoids obstacles and autonomously finds trajectories to clean rooms. Intelligent personal assistants such as Siri, Google Now, or Cortana also lie in this category, using a microphone to analyze speech and answer, when possible, the user's request. The third generation "fire-and-forget" of missiles uses radars, inertial sensor and satellite guidance to autonomously adjust their trajectory towards a target. Lastly, Mars exploration rover Curiosity has used a large set of sensors to get a 3D mapping of the surface and navigate autonomously and safely on Mars' ground in spite of delayed and sometimes broken communications with earth.

(a) (b) (c) (d)

FIGURE 1.1: Today's autonomous perceptual systems: (a) Roomba vacuum cleaner (b) Intelligent Personal Assistants (c) Fire and forget missile (d) Mars rover

(a) (b) (c) (d)

FIGURE 1.2: Tomorrow's autonomous perceptual systems: (a) Google Car (b) UAV mail delivery (c) PTZ camera (d) Domestic robot

The future of autonomous systems involves a large variety of applications (see Figure 1.2) and will likely change our lifestyle by strongly enhancing human assistance or by accomplishing annoying or dangerous tasks. Systems of the next generation will operate in more complex environments, handle interactions with humans while coping with safety issues, and accomplish more diverse and more complex tasks. Those systems are heavily studied, and several promising prototypes have already shown good reliability on constrained environments. There is no doubt that many of those technologies will be part of our life in the next few decades.

Driverless cars is a first example of such systems and already has number of prototypes around the world. The most famous example is the Google Car, that has already shown high efficiency by driving millions of kilometers with a very small number of minor accidents. Autonomous UAVs also show promising potential applications, ranging from search and rescue missions in dangerous environment to mail delivery. Autonomous localization and navigation by UAVs raises a large number of challenges that are still far from being solved. In the context of urban surveillance, PTZ (pan-tilt-zoom) cameras could become fully autonomous so as to avoid human monitoring. This would for instance involve automatic detection and tracking of pedestrians or cars. Lastly, domestic robotics is also a promising field for helping or replacing humans in their everyday life chores. Today's domestic robots are specialized in a single task, but strong efforts are put to make them versatile and adapted to the user's need. Nevertheless, regarding the complexity of some tasks and human safety aspects, this area of research is still at its earliest stages.

Both academy and industries are supporting efforts to push forward advances

in this field, for example with challenges such as the DARPA Robotics Challenge [1], or the annual Imagenet for Large Scale Visual Recognition Challenge [2].

Challenges

Today's perceptual systems are often designed to perform a very specific task. However, to conceive more versatile, autonomous and interactive systems, significant challenges must be tackled. The ultimate goal of an artificial general intelligence (AGI) able to solve the Turing test [1] (try to fool a human while having a conversation with him) or the Coffee test [2] (go into an average American house and figure out how to make coffee) is still far from being achieved.

Those systems should efficiently operate in complex environments. Indoor environments are typically composed with many types of objects of various sizes and purposes. Strong object recognition and localization capacities are then required to efficiently interact with them, even when those objects are not stowed or piled up. The presence of humans in this environment is another major issue: understanding and predicting their behavior to avoid harming them, or just bothering them is of paramount importance. Outdoor environment are also challenging, as perception must be robust to various lighting, temperature and weather conditions. Dealing with rough or muggy ground surfaces also makes the problem harder to solve.

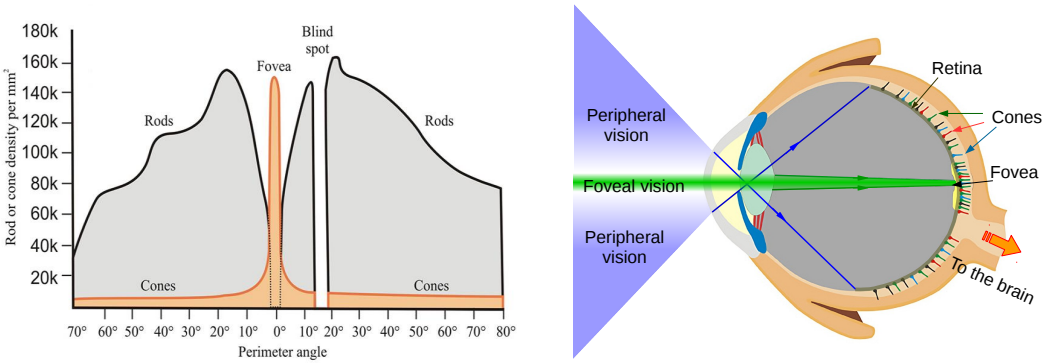
Another challenge consists in processing the raw collected data to make it usable for analysis and understanding of the environment. This processing should be both accurate and stable enough to handle an infinite variety of inputs. Major improvements in that regard have emerged with the use of deep neural networks on many types of input signal. Today's deep networks perform remarkably well on object or speech recognition, on extremely challenging types of data [3], [4]. However, integrating novel, unusual, or modified data to strengthen knowledge on the fly, is still far from being solved.

The process of actions and decisions is a critical aspect as well. Asimov's laws enunciated in the 40's described a set of rules a robot should respect. More recently, Google announced five key problems towards robot safety [5]. Among the main difficulties, decisions should be such that it enable a smart and safe displacement within the environment (for example, to avoid missing important information in the case of surveillance), consider contextual information and take decisions accordingly (for example, being quiet at night, consider moods and emotions of humans before interacting with them), and most importantly, avoid harmful situations, either for human beings or for itself.

1.1.2 Bio-inspired systems

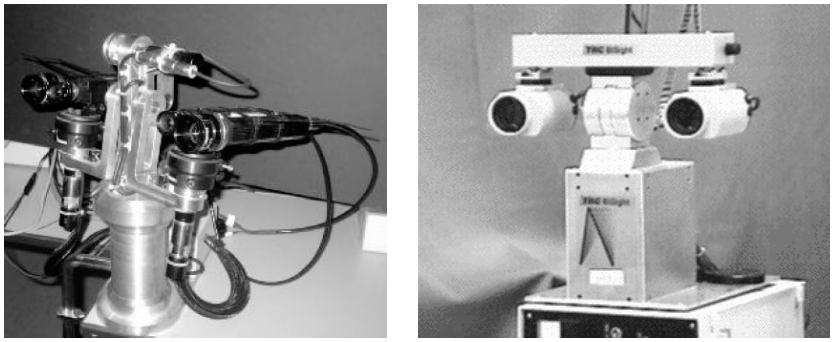
Living-beings have been seen for centuries as a wide source of inspiration for human technology. Although biology is not always a good example to produce artificial mechanisms (such as wheel or plane for instance), it contains a wide variety of concrete examples for thoughts and ideas (would we have planes if birds did not exist?). Unlike bio-mimetics, bio-inspiration takes biology as a pool of interesting problems rather than interesting solutions. In robotics, nature and especially humans represent a good baseline to achieve complex functions such as environment perception or learning.

[1] http://www.darpa.mil/program/darpa-robotics-challenge
[2] http://image-net.org/challenges/LSVRC



(a) (b)

FIGURE 1.3: Cones and rods distribution in the retina



(a) (b)

FIGURE 1.4: Example of robots using foveal vision (a) Gonccalves *et al.* [6], (b) Bjorkman *et al.* [7]

Human vision

The human vision system, ranging from the retinal photo-receptors to the deepest layers of the visual cortex able to identify objects, is an excellent example of biological system that has inspired number of applications related with computer vision - including the digital camera itself. So far, the performance of human vision as regards detection, recognition and contextual information integration remains outstanding and unequaled by machines. It is therefore very likely that a deeper understanding of its core mechanisms could push advances in computer vision forward.

A first interesting component of the human vision is the retina, which is the light-sensitive layer of tissue at the back of the eye. The retina is composed with photo-receptors (cones and rods) as well as ganglion cells that apply pre-processing on contrasts, illuminations or motions before transferring information to the visual cortex. An interesting feature of the retina lies in the density distribution of photo-receptors: cones are significantly (more that 30 times) denser at the center of the retina than at the periphery. The area having the denser population of cones is called the fovea and only covers 2° of the visual field. It is particularly important in the vision process, as the acuity is much higher in this region, and about 25% of the visual information projected to the brain comes from there. As opposed to the fovea, the periphery covers a much larger field of view with a much more modest resolution (See Figure 1.3).

The fovea is strongly involved in the process of recognition. Conversely, the

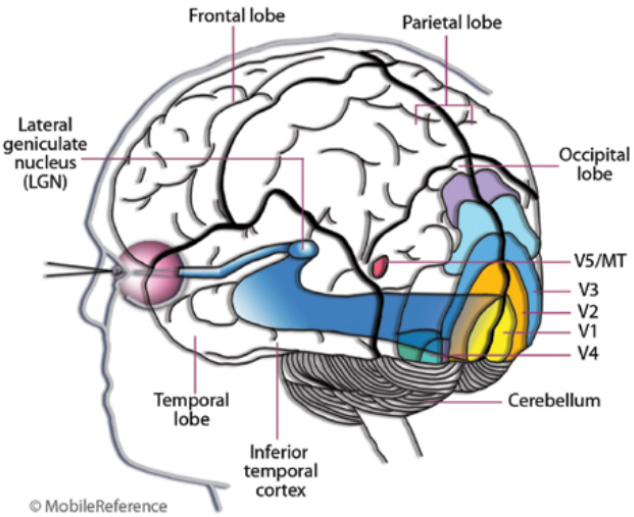


FIGURE 1.5: Main areas and interconnections of the visual cortex. From Gilbert and Li [10]

periphery monitors with a very rough and fast process the potentially interesting (or *salient*) elements of the visual field. As a result, the information from the periphery of the retina is processed by the brain as a pre-attentive analysis of the visual field, to detect targets where fovea should be directed to, for more attentive processes (such as recognition). Such mechanisms have been widely imitated in robotics, typically by using pairs of cameras with different focal distances (See Figure 1.4). The pre-attentive step is processed on the cameras with a wide field of view and produces *saliency maps* to suggest interesting targets. Cameras with a sharp field of view are then oriented toward those targets to obtain images at a much higher resolution.

The major processing of the visual inputs takes place in the visual cortex. This one is composed with interconnected neurons, and can be divided into areas (See Figure 1.5). Those areas are also connected, and process the information at several layers of abstraction, starting from basic edges and orientations (V1) to simple shapes (V4) and finally objects (IT). This architecture of interconnected neurons has strongly inspired artificial systems able to learn representations from raw inputs (HTM [8], convolutional neural networks [9]).

Developmental learning

A natural idea when designing fully autonomous systems and robots is to make them gain knowledge as they evolve in their environment instead of just programming them for the task they are meant to. This approach should enable a better behavior and understanding of their environment. This approach also makes it possible to integrate new skills depending on the needs of the user, and to remain flexible to any change in the environment. Designing robots able to learn by themselves is not a new idea. In 1950, Turing [1] already suggested to create robots able to learn like children, hoping that using such learning mechanisms would be easier than reproducing an adult human intelligence from scratch.

Developmental robotics takes inspiration from learning and development of humans and animals, to conceive robots able to adapt to their environment and acquire skills of increasing complexity over time [11]. Examples of developmental

FIGURE 1.6: (a) A typical study case in developmental robotics: the infant developmental milestones. (b) A developmental robot: the ICub [12]

robotics setups are displayed in Figure 1.6. Of particular interest are the constraints that make learning skills or knowledge easier. Indeed, the extremely high number of things that can be learned during a lifetime makes it impossible to explore in a fully open-ended manner, without any constraint. Considering vision, the infant's visual system in the few days following their birth is surprisingly poor: the child only perceives very close objects or faces. As infants grow, their recognition capacity extends to farther objects as the visual system gains maturity. More than a limitation due to the immature visual system, this development also makes learning constrained at the beginning, before more complex skills such as visual attention can emerge.

A major area of study in developmental robotics is the one of intrinsic motivation [13]. This mechanism makes an autonomous agent select actions or activities where learning is possible, without any short-term goal. More specifically, some approaches implement curiosity mechanisms based on learning progress to learn how to perform tasks [14]. The main idea is to make robots focus on tasks where learning progress is maximal at a given time. This strategy allows a robot to learn continuously, by focusing first on simple things, before investigating more complex tasks.

1.2 Goals and contributions

1.2.1 Problem definition and areas of investigation

To better understand the mechanisms lying behind autonomous perceptual systems, let us consider Figure 1.7. Autonomous perceptual systems are composed with a physical and a mental layer. The physical (hardware) layer is used as an interface between mental (software) layer of the system and the environment: The actuator is controlled by a decision and performs an action that influences the environment. In addition, the state of the environment is reflected by perceptual data accessible by sensors. It is then turned into appropriate representations to construct a mental set of knowledge and skills.

As a general problem, we would like to investigate how to design the mental components of the system. This raises two major questions:

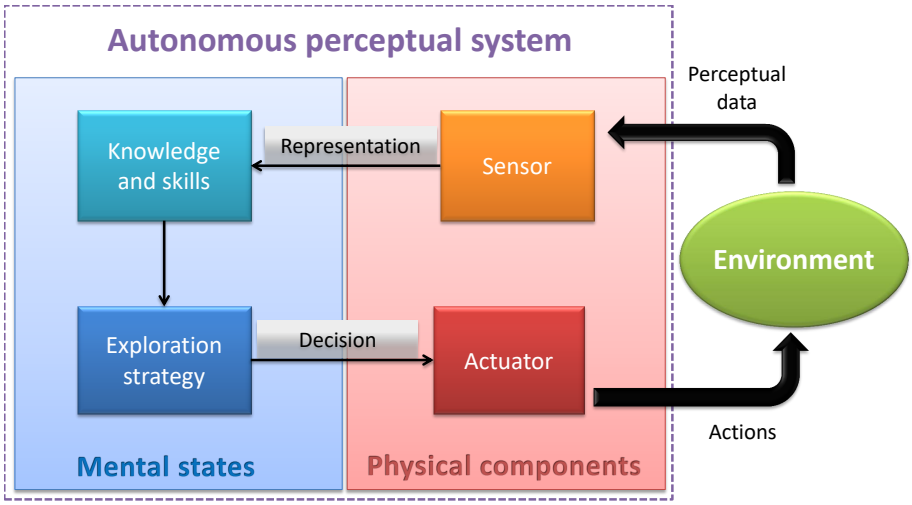


FIGURE 1.7: General problem

- How to convert perceptual data to usable skills and knowledge?
- What kind of exploration strategies can be applied to methodically explore the environment in order to acquire new skills and knowledge?

In this work, the problem is presented in a more restrictive scenario by considering the following elements:

- autonomous perceptual systems are limited to active vision systems, able to autonomously control the viewpoint of their camera sensors;
- the skill that should be learned is a visual attention mechanism able to localize salient objects in the surrounding environment;
- we use a developmental learning approach based on intrinsic motivation to provide a methodical and lifelong exploration and learning strategy.

Figure 1.8 represents our case study. To solve this problem, we have identified three main areas of research described in the following sections.

Object localization

To efficiently localize objects in their environment, we propose to develop an approach able to generate *saliency maps* given visual inputs.

Visual saliency was defined by Laurent Itti as [15] "the distinct subjective perceptual quality which makes some items in the world stand out from their neighbors and immediately grab our attention.". Salient items can either be objects, shapes or any visual stimuli. The only common point is that our fovea is naturally attracted by these items rather than the neighborhood.

A *saliency map* could be simply defined as a mapping of the input image, where the intensity of each pixel represent the saliency of the input pixel (See example in Figure 1.9). They are typically obtained by applying and combining filters on the input image. This filtering approach is consistent with the biological pre-attentive process in the periphery of the visual field selecting salient items before eye saccades.

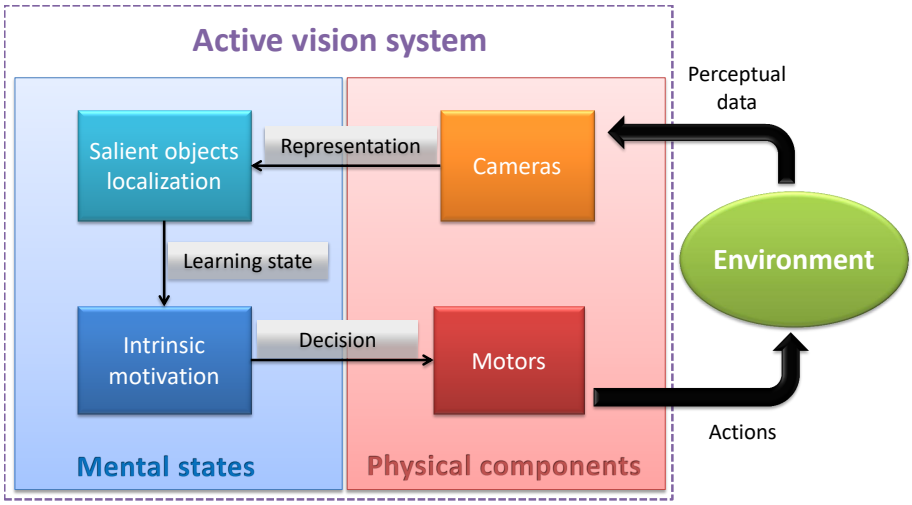


FIGURE 1.8: The problem in our case

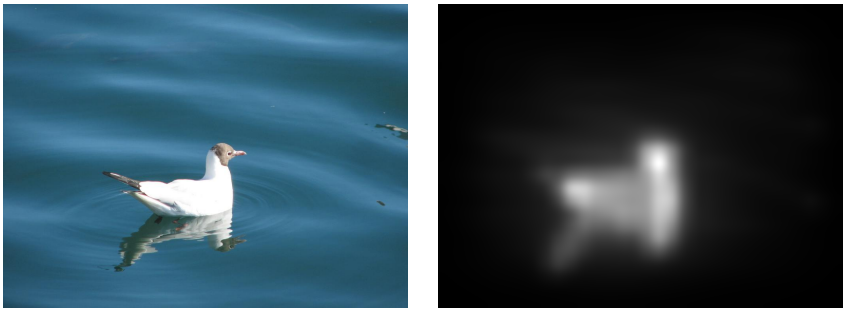


FIGURE 1.9: An example of RGB input and associated saliency map (image from [16] with GBVS saliency map [17])

The link between saliency maps and object localization is not immediate though. First, salient items are not always objects. Then, some objects may not be salient in their context. In this work, we are interested in a type of saliency oriented for a specific task. Instead of being attracted by naturally salient items (bottom-up saliency), we would like to enhance areas that are relevant for a task, in a specific environment (top-down saliency). Based on this definition, our saliency maps are specialized for objects having specific constraints (for instance, being on the floor or on tables, belonging to a database), what we produce is an intensity map representing the likelihood of a pixel to belong to these objects. Another aspect is the one of grouping salient pixels to determines the boundaries of the object, that is not taken into account by the saliency map itself.

Online learning

The core mechanism of this work lies in a way to lean saliency models on-the-fly, directly during the exploration of an environment. Instead of providing the system with a model consistent with all types of environments and not able to integrate additional knowledge, we would like the system to construct its own representations, that would be adapted to both the environment and tasks, and be flexible enough to continuously integrate additional knowledge.

Learning and updating models online raises a certain number of constraints. First, the system has to learn by itself without receiving any feedback from users.

Second, the model update process should be fast enough and should not increase too much in time, so that the system can continuously add knowledge to its representations and rapidly make inferences based on this update. Lastly, the model should remain stable and robust in time.

To create a saliency model for object detection, we suggest to use restrictive object detectors, working for examples on objects that are close enough, clearly separated from the background, or that require a sufficiently high resolution. Those restrictive detectors are then used as a teaching signal to feed a saliency model, that should progressively be able to generalize this knowledge to predict the position of those objects when they are farther away, partially occluded, or in cluttered environments.

Intrinsically motivated exploration

In our approach, we take advantage of the action capacities of the active vision system to improve its own representation of the world. As online learning is used to integrate additional knowledge on-the-fly, it does not directly provide a strategy to drive the system and select the best action to take to improve the model. Without any exploration strategy, the system cannot learn autonomously, and without an efficient exploration strategy, there is no warranty that the integrated knowledge will not destroy the present representation, or that learning can be done in a reasonable time.

We then propose to drive the exploration by intrinsic motivation in order to learn autonomously, by actively seeking for missing information given the current state of knowledge. This approach is used to create saliency models adapted to the environment and to the capacity of the system, while making sure that the model is able to adapt to any evolution in the environment. In addition, intrinsic motivation provides a way to learn in an organized manner, adapted to a lifelong learning scenario. This approach ensures efficiency by avoiding catastrophic forgetting, and provides a faster learning than a pure random exploration of the environment.

1.2.2 Contributions

Figure 1.8, divides the mental states in two components that are salient objects localization and intrinsic motivation. Although intertwined in the final system, the two problems are orthogonal and can be treated and applied separately. Our main contributions are then related with those two aspects and could be summarized by the following points.

An incremental approach for learning visual saliency

As a first contribution, we propose a mechanism able to learn saliency models in an incremental way. The literature in the field of visual attention already contains a large number of techniques to generate saliency maps, but these approaches are usually designed to be used as black boxes for other tasks and are not learned (although sometimes refined) during the robot's exploration.

Our system is working in two modes. On the one hand, a learning mode, integrating on the fly the visual inputs perceived by the robot into a model of visual saliency. On the other hand, an inference mode, using the learned model to generate saliency maps. The produced saliency maps are then adapted to the environment

of the robot, and remain flexible to future model updates. In addition, our implementation is modular and flexible to the integration of new components.

Aside from this main mechanism, we have developed and improved an existing technique for segmenting objects on planar surfaces [18]. This technique is not essential in our work, but is used in the process of saliency learning.

An intrinsically motivated environment exploration with IAC and RL-IAC

Our second contribution is the adaptation of the IAC algorithm [13] in the case of saliency learning in general, and saliency learning on a mobile robot. The IAC (for *Intelligent Adaptive Curiosity*) mechanism is a drive able to guide the robot in the exploration of the environment in a lifelong learning scenario. This is one of the most famous implementation of an intrinsically motivated exploration drive. In addition, the method is generic enough to be applicable in many scenarios.

We then adapted for the first time IAC to the task of saliency learning. This involved a number of design choices and modifications to make the algorithm consistent and efficient in that regard.

Moreover, we found that IAC was not integrating the time for processing action in the decision scheme. For a mobile robot, this factor was too important to be omitted. We therefore created a new algorithm, RL-IAC (for *Reinforcement Learning-Intelligent Adaptive Curiosity*) to overcome this limitation and ensure an efficient exploration strategy for mobile robots.

A generic approach tested on several platforms and scenarios

Our methods aims to be generic and applicable to a large number of active vision systems. We therefore conducted our experiments on two types of robotics platforms, as well as on publicly available datasets.

The first robotic platform, called *BioVision*, was developed at Thales. This platform consists in an anthropomorphic head constituted with foveal and contextual cameras. During the experiments, BioVision is placed on a table, but has the ability to control the orientation of the foveal and contextual cameras to explore its environment.

The second platform is a mobile robot equipped with a Kinect camera and a laser range finder. The laser range finder provides the robot with a localization capability, and is used in the construction of the mapping of the environment surrounding the robot. This robot is then able to explore the environment by displacements across a room or a building, while keeping a track of its trajectory and actual position.

Our last contribution is then the construction, recording and preparation of several datasets to qualitatively and quantitatively evaluate the performance of our systems. In addition, to be more easily compared with other techniques, we also used publicly available datasets.

Publications and patents

This work has led to several publications and one patent that are listed below:

National conferences:

[19] Craye, C., Filliat, D. & Goudou, J.F. (2015). Apprentissage incrémental de la saillance visuelle pour des applications robotique. Accepted for oral presentation.

International conferences:

[20] Craye, C., Filliat, D. & Goudou, J.F. (2015).Exploration Strategies for Incremental Learning of Object-Based Visual Saliency. *ICDL-EPIROB-The fifth joint IEEE International Conference on Development and Learning and on Epigenetic Robotics, At Providence, RI, USA*. Accepted for oral presentation (±25% acceptance rate).

[21] Craye, C., Filliat, D. & Goudou, J.F. (2016).Environment Exploration for Object-Based Visual Saliency Learning. *Robotics and Automaton (ICRA), 2016 IEEE International Conference on, At Stockholm, Sweden*. Accepted for interactive session (33% acceptance rate)

[22] Craye, C., Filliat, D. & Goudou, J.F. (2016). RL-IAC: An autonomous exploration policy for online saliency learning on a mobile robot. *In 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Accepted for oral presentation (8% acceptance rate)

[23] Craye, C., Filliat, D. & Goudou, J.F. (2016). On the Use of Intrisic Motivation for Visual Saliency Learning. *ICDL-EPIROB-The sixth joint IEEE International Conference on Development and Learning and on Epigenetic Robotics, At Cergy Pontoise, France*. Accepted for oral presentation (~25% acceptance rate).

Journals:

Craye, C., Filliat, D. & Goudou, J.F. (2017). Exploring to learn visual saliency: The RL-IAC approach. *To be submitted in the International Journal of Robotics Research*

Patents:

Goudou, J.F., Craye, C. & Fagno, M. (2016). Procédé et système de reconnaissance d'objets par analyse de signaux d'image numérique d'une scène. THALES SA (THALES). EP3070643 - 2016-09-21.

1.3 Overview of the thesis

This manuscript is divided in two parts, tackling on the one hand the incremental learning of visual saliency, and, on the other hand, the intrinsically motivated exploration of the environment. Each part of the manuscript contains a literature review, a technical description of the proposed approach, and an experimental evaluation. Following this introduction, a chapter presents the proposed experimental setups that are to be used in the two parts of the manuscript. Lastly, a concluding chapters provides a discussion and some perspectives to this work.

Chapter 2 - Experimental setup This chapter details the two robotics platforms used to carry out our experiments. We provide technical details on both platforms as well as the required software to run our implementations. Additional information on the publicly available datasets used in this work are also reported. Lastly, we explain the datasets recording, construction and annotations procedures used for evaluation.

Part I - Incremental learning of visual saliency

Chapter 3 - Visual attention, object localization and learning This bibliographic chapter covers the state-of-the-art of the fields related to our proposed approach. We more specifically study the literature on visual attention, object localization in computer vision and machine learning techniques. Although very different, those approaches are intertwined and necessary to fully explain the background of our technique.

Chapter 4 - Proposed approach

We propose in this chapter a detailed description of the algorithms and software components developed to learn a model of saliency on-the-fly during exploration. With this approach, the robot is able to generate saliency maps specialized for the environment the robot is exploring, for the task the robot is asked to accomplish, while remaining flexible to any change or novelty in the environment.

Chapter 5 - Experimental results

We propose in this chapter to evaluate the performance, behaviors and limitations of the incremental saliency learning approach. To this end, we rely on four datasets, and conduct a set of experiment to evaluate different aspects of the method. In a first part, we study the incremental and self-supervised aspect of the method. Then, we study the performance of our approach once a stable model of saliency has been learned. Our evaluations suggests that we outperform state-of-the-art methods when evaluated in the environment the model was trained for.

Part II - Intrinsically motivated exploration

Chapter 6 - Environment exploration on an autonomous mobile robot

This chapter provides an overview of the exploration problem in autonomous robotics under two different fields of robotics that are mobile robotics and developmental robotics. In mobile robotics, exploration is typically seen as a problem of path planning across the environment by maximizing different costs, while in developmental robotics, the stress is put on the learning and development of skills and knowledge based on motivations. In spite of highlighting two different ways for tackling the problem of exploration, we also demonstrate a strong overlap between those fields.

Chapter 7 - Proposed approach

In the scope of an autonomous, life-long exploration and learning, the robot must be equipped with an exploration strategy able to drive the robot's displacements. This one should allow the robot to collect interesting examples, while avoiding problems such as catastrophic forgetting. The IAC algorithm is a possible option for obtaining such a behavior. In the meanwhile, exploration should be organized and efficient, so that the time spent traveling across the environment to reach goals is acceptable. The proposed algorithm, RL-IAC, is a possible way of tackling this issue. We then describe in this chapter the technical details of the implementation of IAC and RL-IAC for the task of saliency learning.

Chapter 8 - Experimental results We propose in this chapter a numerical and qualitative evaluation of the approach proposed in the previous chapter. We then exhibit the behavior, advantages and drawbacks of both IAC and RL-IAC for the task of saliency learning. For IAC, the stress is put on the behavior of the robot during exploration, and its ability to methodically explore the environment. For RL-IAC, we emphasize the influence of some parameters of the systems and demonstrate the efficiency of the approach versus other exploration techniques.

Chapter 9 - Discussion and perspectives This concluding chapter wraps up the work presented in this manuscript and provides a discussion on the main components of this work. We then propose future work directions and perspectives that would be worth investigating in this context.

Chapter 2

Experimental setup

2.1 Introduction

This chapter details the methodology applied to conduct our experiments. We first present the available robotic platforms and additional software components. Then, we describe the procedure used to produce annotated datasets and to simulate the displacement of the robot in semi-simulated environments.

Indeed, our methods aims to be generic and applicable to a large number of active vision systems, as long as it is possible to control the viewpoint of the visual input. We essentially conducted our experiments on two types of robotics platforms, one relying on the foveal vision principle, the other one being a mobile robot. To further enhance and validate our results, some publicly available datasets were used to make our methods more easily compared with state-of-the-art.

We also relied on a semi-simulated framework to deeply investigate the performance of exploration strategies available to the robot. With this simulation procedure, we could produce thorough and repeatable evaluations.

2.2 Robotics platforms and software

We present in this section a technical description of the three robotics platforms used in the scope of this work. We also mention required libraries and software to make our work usable on those platforms.

2.2.1 Robots

BioVision v1

The *BioVision platform* was developed in the scope of a bio-mimetics project at Thales, aiming to study and exhibit some aspects of the human vision. *BioVision* consists in a bio-mimetic head imitating the human vision system to develop new strategies for learning, recognition, or tracking. The platform first takes into account the properties of a human eye by relying on both foveal and contextual fields of view. Then, the visual cortex is modeled by deep convolutional neural networks performing object recognition.

The first version of the platform is an anthropomorphic robotic head presented in Figure 2.1. This head is composed with a Smartek vision camera [1] having a large field of view of 54° (the *peripheral camera* or *contextual camera*), and two Tokina cameras [2] with a very narrow field of view of 2° (the *foveal cameras*). The combination

[1] https://smartek.vision/products-services/cameras/29x29-usb3-vision
[2] www.tokinalens.com/tokina/products

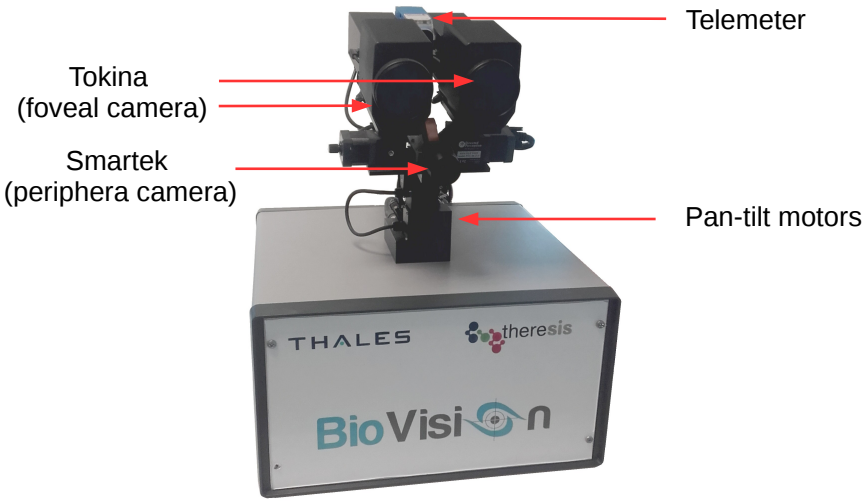


FIGURE 2.1: The *BioVision1* platform

of those cameras offers a binocular vision on the narrow fields as well as a foveal-like vision by having a much higher resolution at the center of the field of view. The vergence of the two foveal cameras is done automatically based on a telemeter measuring the distance from the object at the center of the visual field. The head is also composed with pan and tilt rotation axes[3] to pilot the orientation of the cameras. In this setup, the three cameras move along the same pan-tilt angles and are oriented the same way depending on the pan-tilt position.

The object recognition system described in Section 2.2.2 was trained from images recorded on this setup.

BioVision v2

The second version of the head aims to be lighter and faster, so that it can be mounted and operate on a mobile robot. The *BioVision2 platform* is presented in Figure 2.2. The foveal cameras are replaced by a single EXG50 Baumer camera[4] with a narrow field of view of 5°. An Optotune EL-10-30TC liquid lens[5] is used on this camera to adjust the focus within a few milliseconds. An RGB-D Asus Xtion Pro live camera[6] plays the role of the peripheral camera with a field of view of 57° horizontally. The pan-tilt motors are now piezoelectric ones[7], and the setup is around 2kg so that it can be easily mounted and powered on a mobile robot.

The results presented in this manuscript are obtained on the second version, without using the TurtleBot motion capacities. In our experimental setups, the pan-tilt and camera components of the robot are simply placed on a table. Actions performed by the robot are then *eye saccades*, used to direct the foveal camera to relevant portions of the visual field, in order to get a higher resolution and a better representation.

[3] Flir PTU E-46, http://www.flir.com/mcs/view/?id=63554

[4] http://www.baumer.com/us-en/products/identification-image-processing/industrial-cameras/

[5] http://www.optotune.com/technology/focus-tunable-lenses

[6] https://www.asus.com/3D-Sensor/Xtion_PRO_LIVE/

[7] Pan: ServoCity DDP125. Tilt: ServoCity DDP500

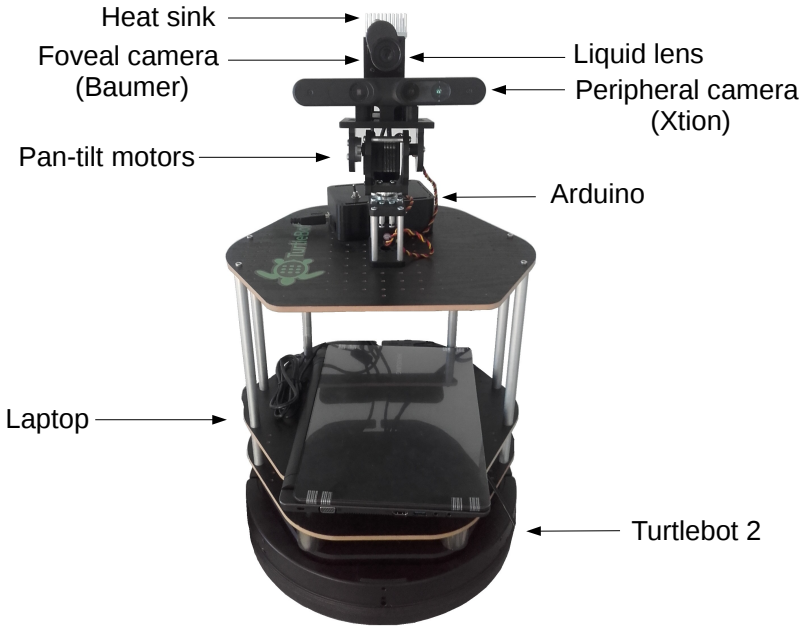


FIGURE 2.2: The *BioVision2* platform mounted on a Turtlebot

Mobile robot

The second platform is a mobile robot Pioneer 3-DX[8] as presented in Figure 2.3. It can move at a speed of 1.6 meters per second. Moreover, the robot is equipped with a Kinect RGB-D camera[9]. The Kinect is mounted on the robot about 1 meter from the ground. It faces the front of the robot and is slightly tilted downward. The depth component of the Kinect does not provide reliable measurements if an obstacle is closer than 0.8 meters or farther than 4 meters. To overcome the limitations of the Kinect in obstacle detection, an Hokuyo UTM-30LX laser range finder[10], having an accuracy of a few centimeters at 10 meters is also mounted on the robot. The wide field of view of 270° makes it well-suited to precisely scan the surroundings of the robot.

In this setup, the actions performed by the robot are displacements through the building. The displacements modify the perception of the robot by showing areas of the environment at different points of view.

2.2.2 Software

Object recognition

For the experiments carried out on the *BioVision* platform, we relied on an object classification system applied to images of the foveal camera.

The object classifier is based on convolutional neural networks, and more precisely, an Alexnet [4] architecture pre-trained on the ImageNet dataset [24]. We replaced the last layer of the network to identify 8 different categories of objects, and a last class called "*other*", containing any other types of elements.

[8] http://www.mobilerobots.com/ResearchRobots/PioneerP3DX.aspx
[9] http://msdn.microsoft.com/en- us/library/jj131033.aspx
[10] https://www.hokuyo- aut.jp/02sensor/07scanner/utm_30lx.html

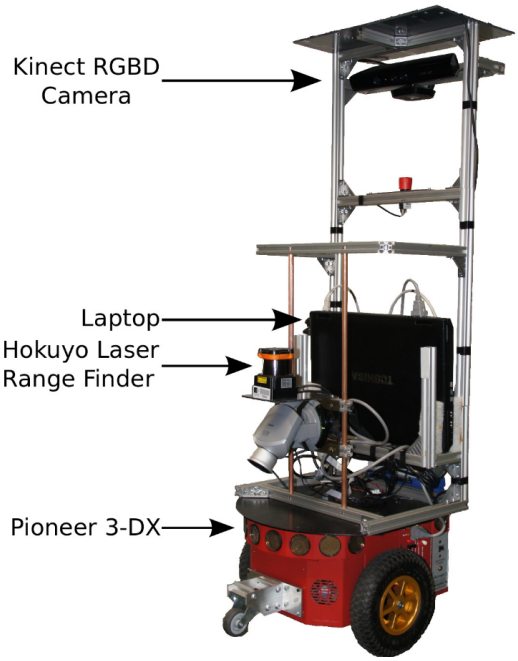


FIGURE 2.3: The *ENSTA* mobile robot platform

To train the network, we constituted a dataset of objects captured from the foveal camera of *BioVision1*. In total, 8 different objects were placed on a white table and recorded at thousands of different points of view (see Figure 2.4). The additional class "*other*" was collected similarly with various kinds of background surrounding the objects. In total, around 10 000 images were collected this way.

The network was finally fine-tuned with the collected dataset, and an accuracy of 99 % was measured on the validation set. Note that the dataset construction and training was done months before the experiments presented in this work, recorded on another platform than the one used later on (*BioVision2*). Therefore, this object classifier is not seen as a part of our system but as an additional component.

FIGURE 2.4: Sample images from the training dataset of the 9-classes object classifier

Dataset	BioVision	ENSTA	RGB-D scenes	Ciptadi
Point of view	BioVision 2	Pioneer 3DX	Human	Willow Garage PR2
Image type	RGB contextual/foveal	RGB-D	RGB-D	RGB-D
Publicly available	No	No	Yes	Yes
Nb. of sequences	10	7	8	n/a
Nb. of frames	2000	4000	1500	80
Nb. annotated frames	150	100	100	80
Annotations	mask	mask + boxes	mask + boxes	mask

TABLE 2.1: Dataset features summary

Simultaneous localization and mapping

Our work strongly relies on the position of the robot within the environment. For *BioVision*, the robot remains on a table and the only useful position is the orientation of the pan and tilt axes. However, an efficient algorithm is required to determine the position and orientation of the mobile robots within the environment.

To this end, we rely on a *Simultaneous Localization And Mapping* (SLAM) algorithm called Hector mapping [25]. Hector mapping is part of the ROS library and is easily interfaced with the Hokuyo laser range finder and odometry measurement of our Pioneer robot. The algorithm is able to fuse those two input cues to provide a 2D pose estimation of the robot, while building a map of the surrounding obstacles. The implementation provides a reliable estimation of the position in most indoor scenarios and is widely used by the mobile robotics community.

Operating system and additional libraries

Our implementation is written in C++ and has been tested on Ubuntu 14.04 with an Intel Core i3-3240, CPU at 3.4GHz quadcore processor. The implementation requires OpenCV [26], PCL [27] and Caffe [28]. Wrapper nodes have also been implemented to make our work compatible with the ROS framework [29].

Biovision2 requires ROS Indigo for pan-tilt controls and input acquisition. Similarly, the Pioneer robot is fully interfaced with the ROS Hydro framework for controls and acquisitions.

2.3 Datasets and simulations

To efficiently evaluate our systems, we have used publicly available datasets, and created our own from the available robotics platforms. We describe in this section the four datasets used in our evaluation, including data collection and annotation. Table 2.1 summarizes the main components of each dataset.

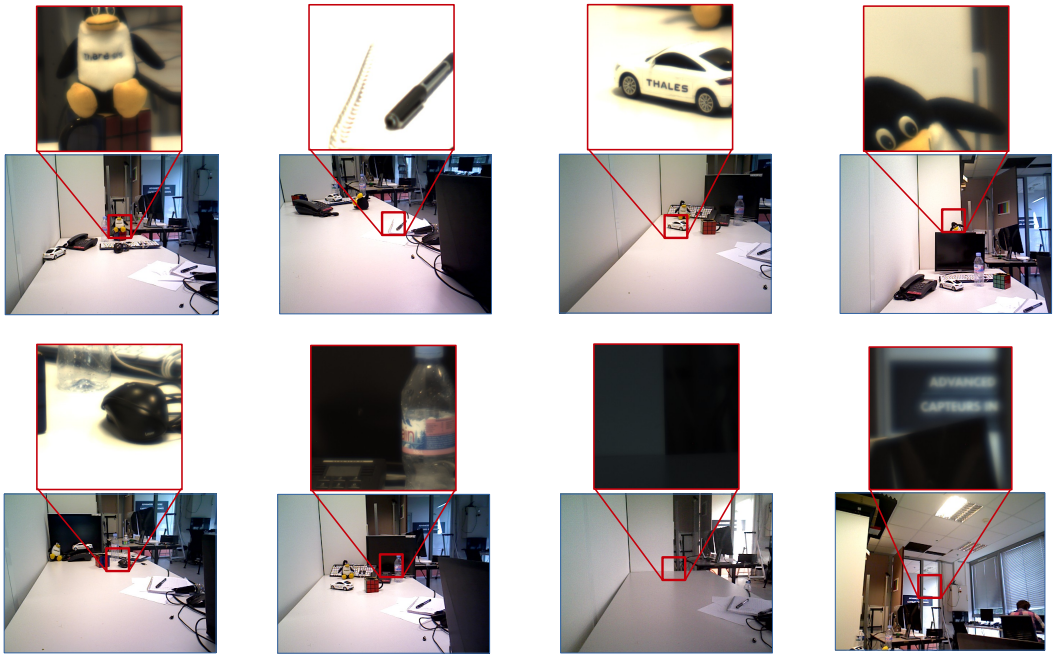


FIGURE 2.5: Sample images from the *BioVision* dataset

2.3.1 BioVision dataset

Our first dataset, denoted as the *BioVision dataset*, is recorded on the *BioVision2* robot.

Dataset recording

During the sequence acquisition, *BioVision* was put at the extremity of a two meters-length table, looking mainly at objects at the other end. The table and adjacent wall are all white, so that objects on the table are likely to be naturally salient in this local context. However, the background of the room is largely visible by *BioVision*, and contains a lot of distractors.

To make the experiment consistent with the object classifier described in Section 2.2.2, we put the same list of objects on the table, sometimes with additional distractors (such as pens).

Our dataset is composed with 10 sequences recorded at different times of the day. For each sequence, we placed an arbitrary configuration of objects on the table, either known by the object classifier or not. We then let *BioVision* observe the environment for about three minutes, meaning that the robot was randomly selecting a pan-tilt position, reaching this position, acquiring both contextual and foveal images at this position, and selecting a new one. In total, around 2000 *observations* (an observation is defined here as a pair of foveal and contextual views) were collected this way. Figure 2.5 presents a few samples of contextual and foveal images of the dataset.

Annotations

To evaluate our results from this dataset, we constituted a training and a testing set. The training set is composed with the observations taken from 7 of the sequences, and the testing set is composed with the 3 last sequences.

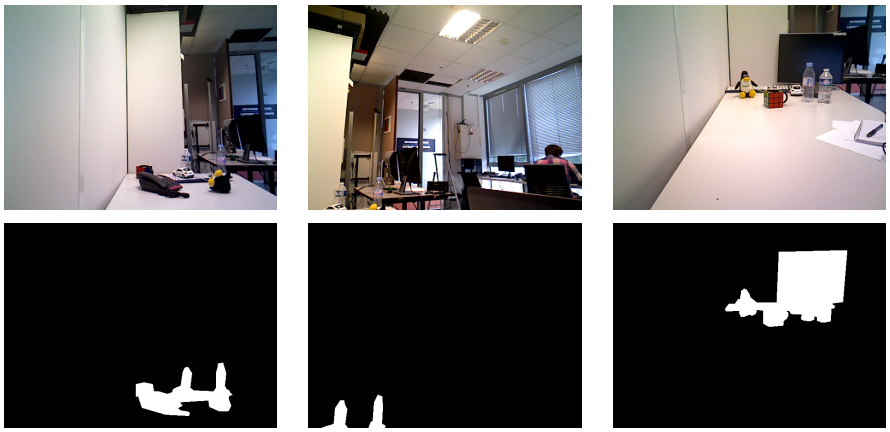


FIGURE 2.6: Contextual images (top) and associated mask (bottom)

In those three sequences, we randomly selected 150 samples that we annotated. To do so, we manually segmented the contextual images of the selected samples, to create ground truth masks of salient and non salient elements. Figure 2.6 shows an example of contextual images and associated ground truth masks.

2.3.2 ENSTA dataset

The second dataset, denoted as the *ENSTA dataset*, was collected from the pioneer 3DX mobile robot. The robot is able to record RGB-D streams from the Kinect camera mounted on the robot.

Dataset recording

To build the dataset, we manually controlled the robot in the *ENSTA* building in order to visit corridors, a laboratory, a hall and offices. The sequence was recorded so as to avoid as much as possible the presence of humans in the images. This way, we recorded a 15 minutes-length video sequence at 20Hz in which the robot was moving at the average speed of 0.5m/s. During the sequence, the robot recorded a large variety of views and lightning conditions across the building and different rooms. Figure 2.7 presents an overview of the robot trajectory during the sequence as well as a few recorded viewpoints in each room.

We also collected at different times of the year 6 smaller sequences of the same area to make sure that illumination and object position in the rooms were not the same. These additional sequences were used for evaluation only.

Annotations

We separated again our dataset into a training and an evaluation set. The training set was not annotated, but we selected a subset of 100 images to constitute the evaluation set.

We annotated the frames with a similar procedure than for the *BioVision dataset*, except that the ground truth masks were drawn so that chairs, desks or tables constituted the salient elements.

Lastly, we annotate the same 100 images in terms of bounding boxes around salient objects. Figure 2.8 shows a subset of images annotated with ground truth masks and bounding boxes annotated this way.

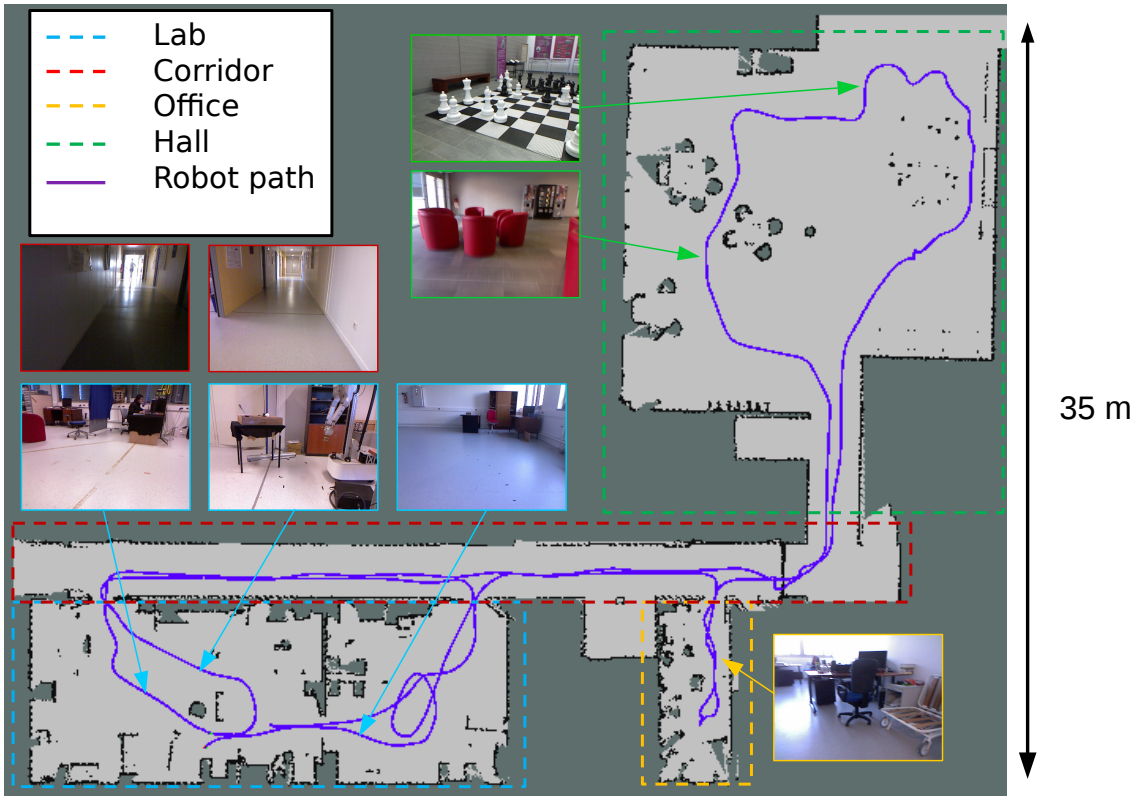


FIGURE 2.7: Map of the trajectory and views recorded by the robot in the *ENSTA dataset*.

(a) (b) (c)

FIGURE 2.8: Samples of the *ENSTA dataset* and with associated ground truths. (a) RGB image, (b) manually annotated ground truth mask, (c) manually annotated bounding boxes

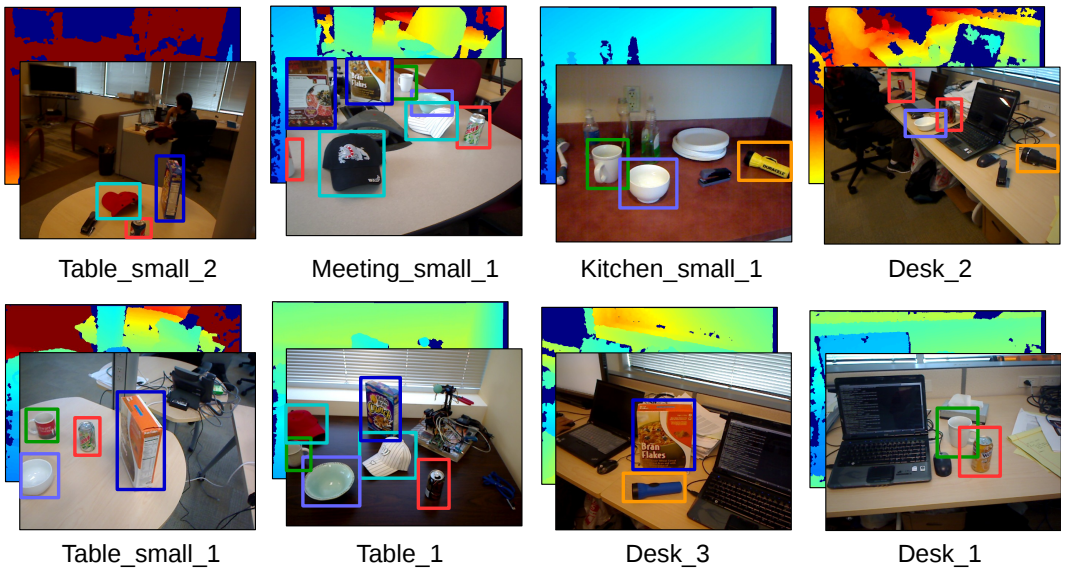


FIGURE 2.9: RGB-D input and annotation samples for each sequence of the *RGB-D scenes dataset*

2.3.3 Washington RGB-D scenes dataset

The third dataset is constructed from a publicly available dataset called the *RGB-D scenes dataset*, that is part of the Washington dataset [30].

Original dataset

This dataset is composed with 8 video sequences of indoor scenes containing everyday-life objects on tabletop surfaces. Sequences are recorded with an RGB-D sensor and contain the RGB and depth map separately. Figure 2.9 provides a sample RGB-D input for each sequence and the associated annotations.

The recording sensor is held by a human user during the sequence. Therefore, the point of view is concentrated on the tabletop containing objects. Each sequence is roughly one minute length, and the point of view at which objects are observed is changing as the user moves in the room. The overall dataset contains around 1500 samples.

Each sequence is annotated by bounding boxes around objects registered in the database (cereal boxes, bowls, caps, *etc*...). However, some objects of the sequence are distractors (staplers, computers ...). They are objects that would be salient according to our definition, but they are not annotated by bounding boxes.

Annotations

The annotations provided by the original dataset are not exactly compatible with the way we would like to exploit them, so that we had to create our own once again.

Similar to the *ENSTA dataset*, we construct our own set of annotations by selecting a subset of 100 images from all of these scenes and manually construct ground truth masks and bounding boxes around objects on the tabletops (including the ones considered as distractors in the original dataset). These images were removed from the training set and kept for evaluation only. Figure 2.10 provides a subset of such annotations.

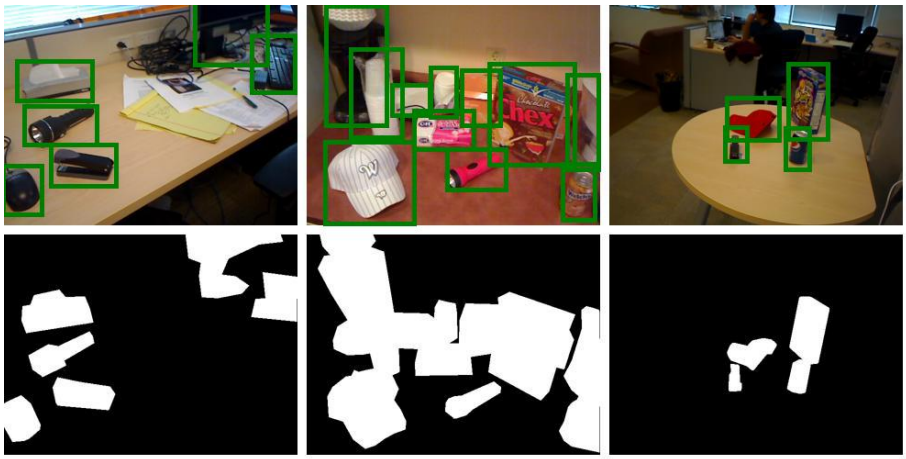


FIGURE 2.10: Ground truth masks and bounding boxes used in our experiments

FIGURE 2.11: Sample images and annotations from the *Ciptadi dataset* [31]

2.3.4 Ciptadi dataset

Our last dataset is again a publicly available one, called the *Ciptadi dataset* [31]. This one was design to evaluate RGB-D saliency algorithms in indoor environments. We found this dataset particularly interesting as perfectly well-suited for our experimentation and evaluation, but also because this is one of the rare dataset designed for evaluating RGB-D-based saliency. This dataset shows some similarities with the *RGB-D scenes dataset*, while containing objects, rooms and viewpoints that *RGB-D scenes* does not.

The *Ciptadi dataset* is composed with 80 images of indoor scenes, recorded in the Georgia Tech Aware Home by a Kinect camera mounted on a Willow Garage PR2 robot. Each image contains a ground truth mask that was manually annotated by a human user (see sample images in Figure 2.11). With this dataset, we then have an external and objective set of annotations to evaluate our method.

2.3.5 Simulated environments

Motivations

A strong limitation when using robots, especially in scenarios involving active 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 also rely on simulations that can be run in parallel, without any particular user monitoring. This way, a large number of experimental results can be obtained, and a better evaluation and analysis is carried out.

Methodology

When required by the experiments, our method is then evaluated by constructing semi-simulated environments in which the robot can virtually navigate. We use the term semi-simulated, as real images taken from the aforementioned datasets are used, but actions taken by the robot are simulated.

More precisely, to make the robot virtually move in its environment, we add in our datasets a localization information, that associates a position with each available observation of the dataset (either an RGB-D frame, or foveal-contextual pair). These positions typically correspond to the positions estimated by a SLAM algorithm, or the pan and tilt orientations of the *BioVision* head. Then, by taking virtual actions, the robot reaches one of the available positions, and we estimate the time that should have been taken by the robot to reach this position in the real environment it has been recorded. Suppose for example that the robot is in position p_1 and virtually reaches position p_2. The estimated time t_{p_1,p_2} for performing a displacement between p_1 and p_2 would be obtained by

$$t_{p_1,p_2} = V \sqrt{\sum_{i=1}^{N} (p_2(i) - p_1(i))^2} \tag{2.1}$$

N being the number of dimensions defining the robot's position and V being the average speed of the robot measured during the sequence recording.

As a result, when the robot takes an action in our simulated environment, we just determine and keep track of the time to reach this position. We then consider that the robot has successfully reached the target, and process the associated observation as if the robot had really moved to that position.

Chapter 8 explains in deeper details the procedure for simulating displacement time in the scope of the robot's exploration.

2.4 Conclusion

In conclusion, we presented in this chapter the robotics platforms available to carry out our experiments, and the required hardware and software to construct and evaluate our techniques.

Four datasets that were used in our experimentation, either constructed from robot sequences recorded in our laboratories, or publicly available. We give a description of each dataset as well as their available annotations. For more clarity, the annotation process is to be further discussed in Chapter 5, after the incremental saliency learning method has been presented.

Lastly, we discussed the need to construct simulated environments to efficiently evaluate our approach, and provide an overview of the way we simulated the displacements of the robot. This simulation aspect is to be further explained in Chapter 8 after the intrinsically motivated exploration has been presented.

Part I

Incremental learning of visual saliency

Chapter 3

Visual attention, object localization and learning

3.1 Introduction

This chapter is a literature review on the background fields our saliency learning technique is based upon. Although covering very different aspects of the problem, we aim to present the literature so as to make links between them.

The main background of this work is the one of *visual attention* from biological, computer vision and robotics points of view. We intend to describe the main aspects of visual attention as well as the biological inspiration in the artificial systems. We then precisely describe the literature regarding saliency maps and evaluation means. Lastly, we provide a list of applications and robotics platforms relying on visual attention.

Another field, not directly related to visual attention, but extremely active in computer vision is the one of *object localization* in natural images. We go through the literature on this field for two reasons. First, our system is built on tools developed in this field, such as superpixels, bounding box proposals, or binary segmentation. Second, the concept of saliency maps and object localization are largely intertwined, especially in the setup where we carry out experiments. Lastly, as the goal of our approach is mostly to learn a model able to generate saliency maps, we review approaches that learn models of visual attention, and we describe the main features and challenges of our learning constraints. This approach then provides the context of our work under a machine learning perspective.

3.2 Visual attention

Because our retina is composed with photo-receptors that are not homogeneously distributed, our brain needs to use a mechanism of selection aiming to prioritize input stimuli, and to examine them in a selective and sequential way. This attentive selection mechanism is probably due to evolution, in order to handle the high amount of information constantly surrounding us.

3.2.1 Generalities

Several literature review on the topic of visual attention define the main keywords and concepts. The work of Itti and Koch [32] is a pioneer in the field of computational visual attention by defining the main ideas of a visual attention system, while insisting on the importance of eye saccades. Henderson's review [33] is more focused on the gaze control at a computational and cortical level. Frintrop et al.

[34] have produced a literature review on visual attention by comparing computational and cognitive points of view. Lastly, the recent survey of Borji et al.[35] covers pretty exhaustively a large panel of methods and aspects of computational visual attention. In this section, we define a few important concepts and recalls regarding visual attention.

Attention, saliency and gaze

First, let us make a distinction between the three terms *visual attention*, *visual saliency* and *gaze control* that are sometimes mismatched. Visual attention is the most general term and represents the whole mechanism responsible for deciding where to look at.

Saliency is more related to the scene and to visual stimuli that are likely to catch the attention. In computer vision, saliency is usually represented by a saliency map. A *saliency map* is a grayscale (or heatmap) image where the intensity (or heat) depends on the interest of each pixel of the original image. In this document, we may represent saliency maps in both ways depending on the case. They have exactly the same meaning though.

Gaze is directly related to eye motion and represents the orientation of the eyes. Saliency is then related to external stimuli whereas gaze is intrinsic. The visual attention systems then coordinates both aspects so that gaze is directed towards salient area of the visual space.

Overt and covert attention

Directing attention towards a region of interest is usually associated with an eye movement, also called a *saccade* (represented as black lines in Figure 3.1). Once attention target has been reached, the eye stays focused for some time to analyze the corresponding visual input. This is called a *fixation* (represented as yellow dots in Figure 3.1). Lastly, the sequence of fixations and saccades necessary to examine a visual scene is called the *scanpath*.

Overt attention is then defined as the visual attention process that directs attention towards areas of interest for a deeper examination. This was first studied by Yarbus [36] in the 60's. However, we are still able to examine elements at the periphery without moving the eyes. We call this phenomenon *covert attention*. Covert attention usually comes before overt attention as it makes a pre-selection of interesting targets and potentially moves the gaze toward these items. Sometimes, covert attention is enough and does not require a visual focus of attention, for example for simple manipulation tasks. Conversely, it may happen that eye movements are not related to spontaneous covert attention, typically in the case of learned saccades, where the frequency of saccades is much higher than the cognitive demand for covert attention.

Bottom-up and top-down saliency

A key concept in visual attention is the difference between *bottom-up* and *top-down* visual attention. Bottom-up visual attention is related to visual stimuli of the environment. The gaze is directed towards the item that produces the highest stimuli at the retinal or visual cortex level. This occurs when observing a scene with no specific goal: our attention is then simply guided by surrounding elements.

FIGURE 3.1: An example of saccades and fixation sequences (from [37])

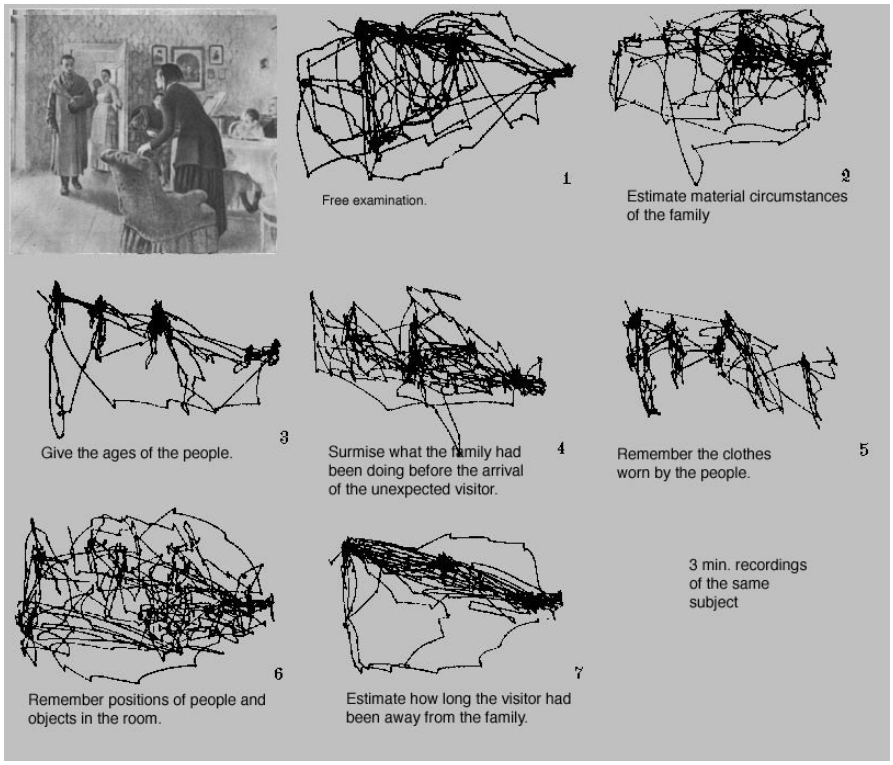


FIGURE 3.2: Yarbus experiment (image from [36])

On the contrary, top-down attention depends on a goal and has been highlighted by Yarbus experiments [36] (see Figure 3.2). In this experiment, several subjects were asked to look at an image to answer a specific question (estimate the age of the people, remember their position in the room, *etc*...). Yarbus discovered that the subjects' gaze was drastically different depending on the question, suggesting that attention was depending on the task of the user. Top-down attention is both more demanding and more specialized than bottom-up attention, as it is used in a more conscious and voluntary perspective. It then requires higher cognitive loads.

Stimuli, space, and object-based saliency

The question of whether our attention is directed towards regions of the space, visual features, or directly objects is an open debate. Most studies on visual attention at neurobiologic level consider visual attention as being region-based, but other

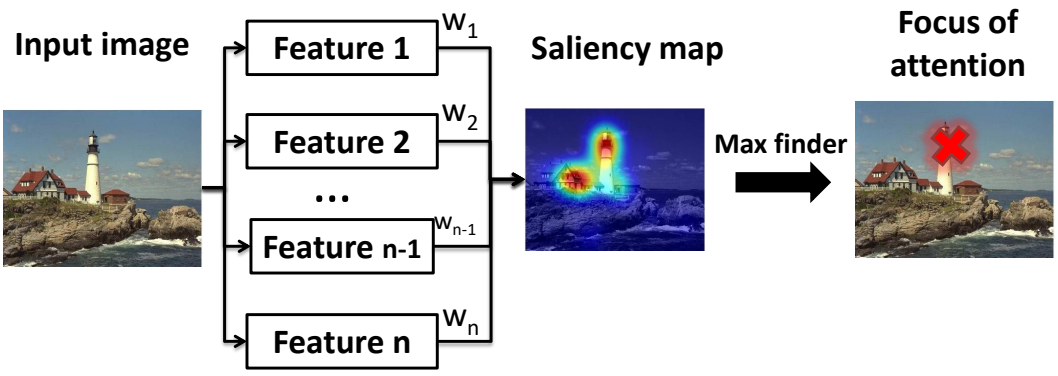


FIGURE 3.3: General architecture of most computational visual attention systems

works have also shown evidences of the other types of attention. Most research today agree that those theories are not incompatible, and that visual attention can, depending on the context, alternate between those models.

Computational modeling of visual attention

Dozens to hundreds of visual attention methods have been reported in the literature, for a large variety of applications. Borji and Itti [35] have identified more than 70 of them, without pretending to be fully exhaustive. However, most of these models use a similar core architecture, presented in Figure 3.3. Visual features are first extracted from the input image, then combined to produce a saliency map. We will describe some of these approaches in the next section. The global maximum of this map is then determined and visual attention is finally directed towards this area. The focus of attention can be expressed in many different ways depending on the system, either by cropping the area directly in the image, or by a saccade centering the gaze at the focus of attention.

In the case of a robotics application, the difference between visual attention and saliency is clear, as a new focus of attention usually results in a physical action (possibly a saccade). However, the distinction is much thinner when still images are considered.

3.2.2 Saliency maps in computer vision

In this section, we examine the different approaches for producing and evaluating saliency maps in computer vision.

Many definitions of saliency

Among the large number of methods for generating saliency maps, there is no single consensus to define what a salient item is and what should be enhanced in the input image. To make the classification easier, Borji *et al.* [35] proposed a taxonomy of the saliency models.

Taxonomy: The *cognitive models* [38], [39] try to define saliency based on our knowledge of the human visual system. Models are then developed by following the experimental findings on the human neurological responses to visual stimuli.

Other methods define saliency areas as being "unusual" and use mathematical properties of the image to enhance them. In this category, the *spectral models* rely for example on spectral residuals [40] or image signature [41], while *graphical models* such as the one of Harel *et al.* [17] have defined a saliency approach (GBVS) based on markov random fields and a definition of unusual pixels in their neighborhood. Other approaches defined as *information theoretic models* rather defined saliency as being "informative". For instance, Bruce *et al.* [42] have used a criterion based on Shannon self information to define saliency. Lastly the *pattern classification models* [43], [44] rely on machine learning technique to predict the position of salient items. Our own method of saliency could be classified in this last category.

Top-down and bottom-up saliency: Similar to visual attention, saliency can also be defined from a bottom-up or top-down point of view. Depending on the task and application, saliency maps may enhance items that are naturally salient in their context, or important for a given task. There is no single definition of what needs to be enhanced in a top-down context as the task is different for each application. However, in a robotics context and with no further precision, top-down saliency usually mean "object-oriented". We follow such a definition in our work, by defining salient elements as surrounding objects the robot can potentially interact with. The pure top-down approaches are not as common as bottom-up ones, and a few of them suggest to focus on combining bottom-up and top-down cues [45], [46].

Stimuli: Saliency maps are also classified in terms of stimuli. Depending on the purpose they are used for, they can enhance positions where subjects are likely to make a fixation [32], or enhance a whole object or a whole area [47]. The notion of scale is also important depending on what is supposed to be salient. For example, depending on how big a subject is in an image, saliency can either be on the whole subject [48], or on areas constituting the subject [35], [49]. Our work considers saliency at the scale of an objects and enhances the whole area of the image containing those objects.

Input: One could also think of defining saliency based on the nature of the input stream. Although a large number of saliency approach consider only the RGB component of still images, a natural extension is to integrate temporal or spatio-temporal components. In that regard, the optical flow has been used by Le Meur *et al.* [50]. When using an RGB-D camera, the depth integration is also a natural extension, either using an early [51] or late fusion approach [52] between RGB and depth components. In the RGB-D approaches, the geometrical nature of depth signal is better suited to detect salient objects rather than salient stimuli. Lastly, saliency can also be detected with event-driven cameras, at a much higher frequency than any RGB-based algorithm [53]. Our saliency maps aim to be generated from RGB images only, but we also rely on geometrical information from depth maps to learn our model of saliency.

Saccadic models: Another family of techniques proposes to study saccadic models rather than just saliency maps. Instead of considering the only spatial component of visual attention, saccadic models also study the temporal aspect of exploration by predicting plausible visual scanpaths rather than only salient locations [54], [55]. These kind of approaches both rely on saliency models to predict fixations, but are also able to produce saliency maps based on the simulated scanpaths [56]. The scanpath-based saliency maps are potentially more accurate, as additional stochastic components related to fixation transitions are integrated in the model (for exaple, vertical saccades are much more likely than oblique ones) [57], [58].

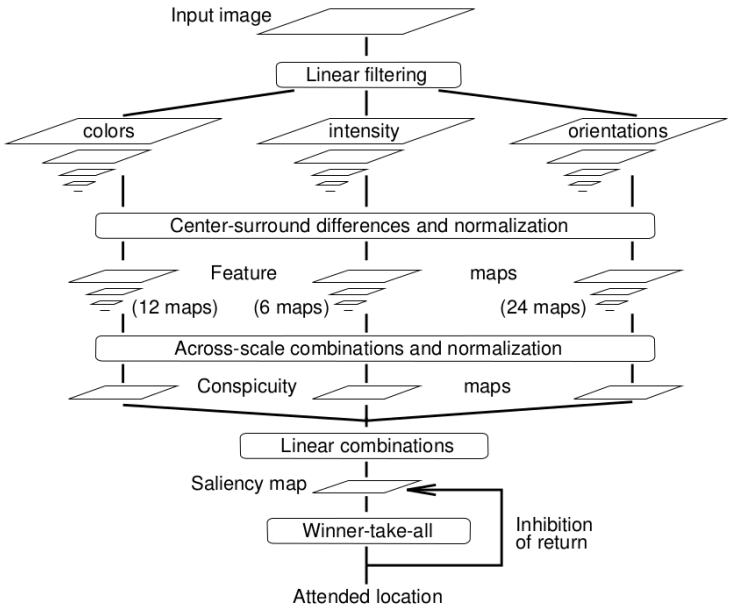


FIGURE 3.4: General architecture of Itti and Koch visual attention model (image from [38])

Computing a saliency map

The most famous model of saliency is probably the one of Itti and Koch [38] developed at the end of the 90's. Although many updates and improvements have been added to this method (for example, by adapting the scales of receptive fields [46], or by integrating face detectors and weights related to human fixations [59]), this one has the advantage of being fast, simple, and based upon biologically plausible mechanisms inspired by retinal and cortical processing of the visual information. Figure 3.4 presents the general approach for this method. Here, visual features are color, contrast and orientation-based. They are combined together by center-surround differences at different scales. This model is inspired by the receptive fields in the retina and visual cortex. A total of 42 maps are then generated and combined to create the saliency map. The basic idea of feature map extraction followed by a feature combination has inspired a large number of approaches in the literature.

Feature extraction: Saliency maps are usually obtained from simple visual features, applied on the whole input image. Most of these features are designed to be applied on RGB images, but some are also based on motion in the case of videos streams [50], [60], or exploit the depth component of RGB-D sensors [31], [51]. Features may be based upon center-surround differences [38], [46], integrating sometimes spatio-temporal components such as optical flow [17], [50]. Classical computer vision filters have also been proposed to produce saliency maps, such as HOGs[1] [61] , wavelets [62], SIFT[2] [63] or gist [64]. Features can also be based on thresholds in the color space [65], [66], statistical descriptors such as covariance [67] or Shannon information [42]. Similarly, spectral methods use transformation spaces

[1]Histograms of oriented gradients
[2]Scale Invariant Feature Transform

of the image such as DFT[3], DCT[4] to obtain visual features [40], [41]. In the last few years, a new process of feature extraction has been proposed based on deep convolutional neural network filers. These kind of features have also been successfully applied to the generation of saliency maps [68], [69]. We propose in our work three types of visual features that are center surround-based for the first two techniques, and deep feature-based for the third one.

Feature combination: Once extracted, the visual features are combined to generate a saliency map. By default, a simple summation or max-pooling approach can be enough [38], but smarter strategies are commonly used to improve the performances. Some approaches use statistical approaches to build meta descriptors in order to get a better feature combination scheme. Thus, Erdem *et al.* [67] have used a set of descriptors based on region covariance from basic features in order to improve saliency prediction. Bayesian strategies that aim to integrate prior and posterior knowledge to improve the feature maps combinations have also been proposed in the literature. For instance, Klein *et al.* [70] have used Itti and Koch center-surround saliency maps to compute saliency estimators from local statistics. In the perspective of optimizing saliency to get results as close as possible to human saliency, Zhao and Koch [59] have attributed optimal weights to each feature based on statistics obtained from human fixations on a large number of images. Li *et al.* [71] have proposed an improvement for basic saliency maps based on missed salient areas retrieval and false distractors removal using co-occurrence patterns. Lastly, pattern recognition models combine the extracted features by the mean of a classifier, thus creating a non-linear combination scheme [44], [72], [73]. This classifier approach is the one used in our technique.

Top-down modulation: Feature combination is also used to differentiate bottom-up and top-down approaches. As bottom-up approaches are used for an open-ended exploration of the visual environment, they do not require a combination approach able to bias features against each other. When using top-down visual attention to specialize the attention towards a task of a certain type of stimuli, the features have this time to be combined accordingly. The dedicated term for this kind of approach is the *top-down modulation*. Such modulation can be obtained by exciting visual features related to the objects of interest [74], by inhibiting irrelevant regions [75], [76], or by mixing those two approaches [45], [59] with learning.

Top-down cues: To further improve the performance of the saliency, many approaches integrate additional cues that are nothing more than object detectors. This integration may make approaches better responsive to human shapes, by integrating for example detectors for face [59], [77], skin [78] or cars [79]. Although not being in the philosophy of pure bottom-up approaches, integrating these cues is still consistent with the idea that visual saliency is partly learned from what we commonly see interact with in our everyday life.

Post-processing: The notion of stimuli-base and object-based visual saliency also influences the way saliency is obtained. Usually, a post-processing stage is enough to turn a stimuli-based saliency map into an object-based one. For stimuli-based, Gaussian blur are often applied to better fit the statistical distribution of the fixations [42], [67]. Object-based saliency is rather obtained by looking at homogeneous colors in contrast with their environment [65], [66], or combining saliency with segmentation techniques such as superpixels [80] or mean-shift [48]. Figure 3.5 shows a comparison of the same saliency method using the stimuli-based and

[3]Discrete Fourier Transform
[4]Discrete Cosine Transform

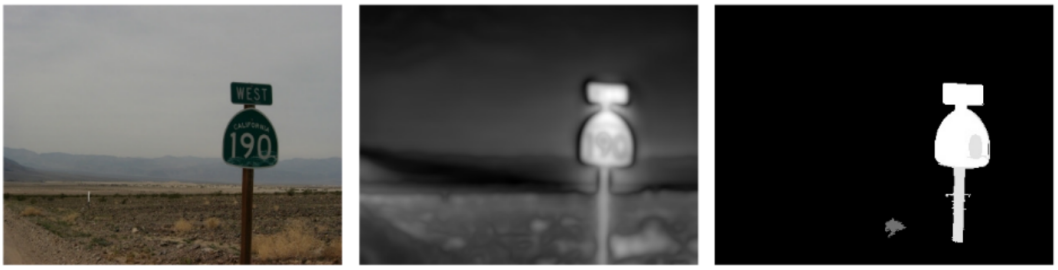


FIGURE 3.5: Comparison of stimuli-based and object-based post-processing on the same saliency method (images from Frintrop *et al.* [48])

object-based post-processing. Note that object-based saliency is tightly bounded with the notion of object detection or object segmentation, as the output saliency provides a rough estimation of object boundaries. In that regard, our method lies in the category of object-based saliency maps, and we propose a post-processing step to recover the shape of the objects when the produced saliency map is not clear enough.

Connectionist methods: An alternative to the filter-combination scheme is the fully-connectionist approach. Saliency is learned in this context using neural networks techniques and annotated datasets. The connectionist approach proposes to perform filtering and feature combination of an image within the same network, thus following a more biologically plausible scheme. Nevertheless, early approaches of connectionist models [81] were found to be less efficient and harder to design than the filter-based ones. This trend has been inverted with the recent breakthroughs in deep learning, who paved the way for a new paradigm about how to generate saliency maps. Similar to many other fields of computer vision, *deep convolutional networks* (or *CNNs*) have reported the best accuracy on many saliency benchmarks [43], [82]. Two types of supervising signals can be used to train such classifiers: first, on saliency datasets, annotated either in terms of fixations [43], [82], or segmented objects [80]. Second, a simple set of object detectors can suffice to train a descent saliency enhancer [68], [83]. This approach is particularly interesting, as they can be learned on a weakly supervised basis, using as a training signal only picture of salient objects. The localization aspect is then learned for free. However, this approach has two counterparts: first, the model is specialized for objects in the dataset rather than learning a pure bottom-up saliency. Second, the method does not detect the object, but rather what make the object discriminant from other elements. The limit between saliency and object localization is very thin in this type of approach. We do not propose in our work a fully connectionnist approach, but the influence of this new paradigm is clearly present in the way we design our technique.

Evaluation and datasets

Given the abundance of existing saliency approaches, dedicated datasets have been created for benchmark and comparison. The proposed datasets usually take human annotation and behaviors as ground truth. Saliency maps are then computed from those datasets and compared with ground truth based on metrics. Although most of these datasets propose an evaluation on RGB still images only, some are also based on video sequences [39], [84], or RGB-D inputs [31], [51]. Figure 3.6 shows an overview of the kind of images and ground truth found in the saliency benchmarks.

FIGURE 3.6: Sample image and ground truth from datasets. On the left, ground truth based on human fixations [85]. In the middle, manually segmented image [86]. On the right, dataset avoiding the center bias [31].

In general one should distinguish approaches trying to mimic human saliency models, and models for detecting salient objects. The former aims to determine where humans are going to look at whereas the latter seeks to extract the most salient objects, which are defined by well-defined boundaries. The goal of these models and the way we evaluate their performance are dramatically different.

Human fixations-based datasets: A first category of benchmark proposes to predict human fixations. The ground truth is then constructed with gaze trackers: several human subjects are asked to observe the same visual input for a certain time. The eye trajectory of each subject is then recorded and statistics are computed in order to determine the most frequent fixations and scanpaths. Popular datasets of this category include the *MIT300* [87], the *CAT2000* [85], or the *VIU dataset* [88].

Manually segmented datasets: A second category uses manual annotations from a few users to crop or draw boundaries around important. Among popular datasets of this type, the Achanta's salient object dataset [86], the MSRA [89], or the PASCAL-S [90]. Depending on the benchmark, a single or several regions are isolated this way. This type of datasets may also be used for segmentation tasks, or for object detection. We are mainly interesting in this type of annotations, more consistent with our definition of saliency.

Center bias: Most datasets are biased by the position of salient objects. As images constituting the datasets are most of the time taken by humans, the most salient element tends to be at the center of the image (which corresponds to the center of attention of the viewer). In that regard, considering a simple Gaussian distribution at the center of the image is a good saliency baseline that may outperform much more complex saliency techniques [91]. To limit this bias, some datasets are constituted from free view-points inputs, where the most salient element does not have to be at the center, or may not even be on the image at all [31], [42]. The datasets used in our experiments are mostly taken from a robotic platform, and are not biased by the center bias.

Top-down datasets: Most of the proposed benchmarks are conceived and labeled for a bottom-up saliency evaluation, probably because a better consensus exists on the definition of bottom-up saliency. Nevertheless, a top-down evaluation is still possible on datasets specialized on a particular type of stimuli, such as the *FiFA dataset* containing lots of faces [77], but to evaluate saliency on a specific task (which is our case), a dedicated dataset often needs to be created or adapted.

Common evaluation metrics: To compare an estimated saliency map with a ground truth, several metrics can be used (for more details about their exact computation, please refer to Riche *et al.* [92] or Borji *et al.* [93]). If the saliency map is seen as a probability distribution of eye fixations, metrics such as Kullback-Leibler divergence (DL), Histogram Intersection (HI) or the Earth Mover's Distance (EMD) are adequate, as they measure the distance between two distributions. When comparing the statistical relationship between estimate and ground truth, the linear correlation coefficients (CC) or the normalized scanpath saliency (NSS) are advised. Lastly, the comparison can be seen as a binary classification problem (salient and not salient) and is then evaluated in terms of false positive and false negative rates. The most common measure in this case is the area under curve (AUC) that measures the area under the ROC (Receiver Operating Characteristic) curve.

The AUC is probably the best-suited metrics for our evaluation and is the one used in our experiments. Further details on the computation of this metrics are provided in Chapter 5. The AUC has the nice property of being invariant by any monotonically increasing transformation. However, the result is represented in terms of true and false positives, which does not penalize false negatives in the estimated saliency.

3.2.3 Visual attention in robotics

Unlike classical computer vision, where the point of view is often chosen by a human operator and most likely centered at the object of interest, the viewpoint in robotics depends on the robot's position and movements. Is is therefore not - at least at first - centered at the interesting elements of the environment. Providing the robot with a good visual attention strategy is therefore critical for a good analysis of the environment. Aloimonos and Westelius' reference books constitute a good introductions to this field [94], [95].

In Section 3.2.2, we presented in details how to obtain a saliency map. We explain here how to exploit them in a visual attention setup. As presented in Figure 3.3, the saliency map is typically used to determine the most salient regions. The basic visual attention scheme follows a *winner-takes-all* strategy by determining the global maximum of this map, and focusing attention on this particular region. Then, in case of a static scene, the system should keep exploring and avoid as much as possible looking always in the same direction. For this reason, an *inhibition of return* is usually associated with the system. The inhibition of return has biological foundations [96] and simply consists in inhibiting the salient region that has just been the focus of attention for a certain period of time. This way, attention is naturally disengaged after examination, and the next most salient region is necessarily a different one.

Applications using visual attention

In most cases, when a robotics system is equipped with a visual sensor, a strategy of visual attention must be applied. We describe in this section the main applications

relying on visual attention. For a more extensive review, please refer to Frintrop *et al.* [34] or Bengum *et al.* [97].

Finding areas of interest: First, visual attention and especially saliency have a wide variety of application when a need to localize regions of interest is required. Indeed, instead of examining the input stream with the same attention everywhere, a pre-selection of good candidates can significantly reduce the computational load. This principle is, for example, used for image segmentation, when segmentation seeds are required [98]. One may also think of image matching, when trying to find correspondences to associate images of the same area at different time (for example in SLAM[5] or loop closure detection systems [99], [100]). In these approaches, salient regions are associated with descriptors or landmarks, able to provide a meaningful and exploitable representation of the region while limiting the area of research. Similarly, saliency is widely used in the tasks of navigation and semantic mapping, where the goal is to update a 3D or 2D grids containing objects of the environment [18], [101]–[103].

Object recognition: Object discovery or object recognition, especially for indoor robotics is also a very natural and useful application of visual attention. Early biologically inspired object recognition systems such as HMAX [104] already integrated visual attention in the object recognition process, in order to select, modulate or suppress the most significant cues. Approaches combining both attention and recognition have known a renewed interest with CNNs able to simultaneously localize and identify object [83], [105]. A more sophisticated approach based on recurrent neural networks also proposed a sequential attention scheme to learn to localize, count and classify objects of a scene [106]. More generally, the integration of a visual attention scheme for object recognition may be used at two levels: first, for isolating object candidates in the environment, followed by a displacement of the robot to improve the perception of the target candidate [45], [107]. Second, for isolating areas within the object that may be relevant for its identification [49], [108].

Guiding the robot's actions: Lastly, attention systems may be used to guide the robot's action. The selective aspect of attention is particularly well-suited to perform a single task at a time. The natural action associated with visual attention is the displacement of the vision sensor towards the focus of interest. This action can either be a saccade [107], [109] or a displacement [102], [103], [110]. In the scope of visual tracking, active vision system can perform smooth pursuit to systematically keep the target at the center of the field of view [111], [112]. Rather than moving the sensor to modify the perception of the robot, the attention may also be used as target to reach for robot grasping [113], [114] or in robot navigation [115], [116]. Finally, the field of human-robots interaction is also needing attention to create a context and common objects of interest at which both humans and robots should focus on [117], [118].

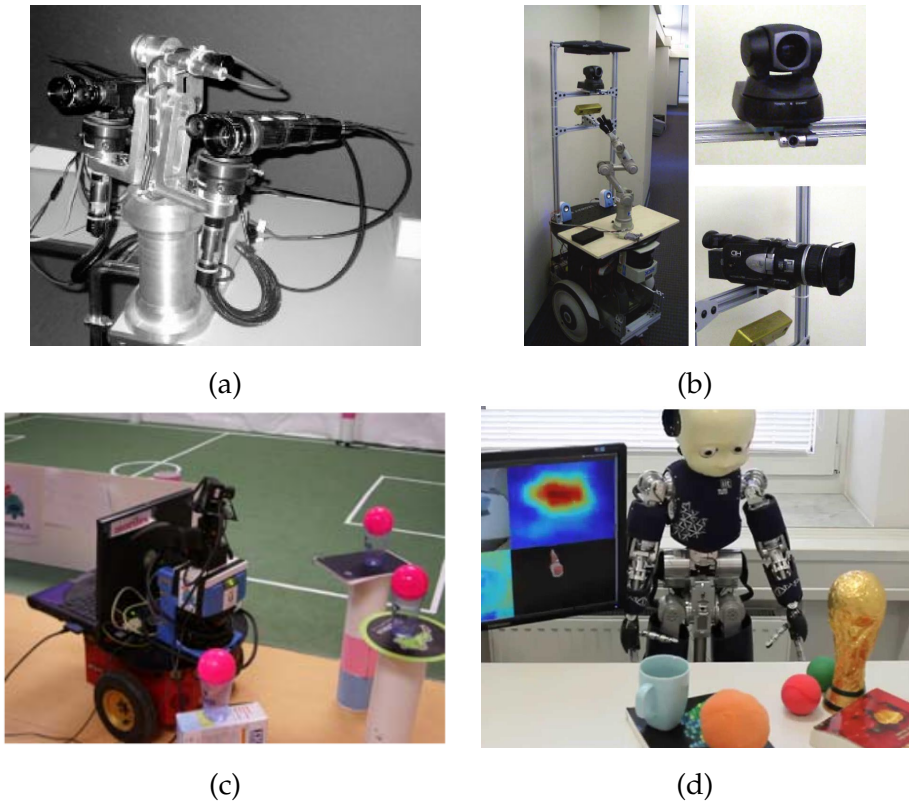
In our research, visual attention is not associated with a concrete application, but we exploit the mechanisms of saccade and mobile robot displacements to improve our model of saliency.

Indoor visual attention platforms

In this section, we mainly consider indoor robotics systems as it is the most related to our research.

Among indoor robotics platforms using visual attention, a distinction can be made between mobile robots able to move across their environment [102], [103],

[5] Simultaneous Localization And Mapping



(a) (b)

(c) (d)

FIGURE 3.7: Visual attention setups (a) foveal vision [107] (b) PTZ [103] (c) mobile robot [102] (d) humanoid ICub [119]

and static robots able to change their point of view with pan and tilt rotation axis of the camera. In this case, PTZ cameras [109] or a set of cameras [45], [107] are typically used to be able to modify the perception of some elements of the environment depending on their interest. Humanoid robots equipped with neck motion capacities, but placed in a human-robot interaction setup also lie in this category [117], [119]. Figure 3.7 presents a few of these setups.

Foveated system: In *foveated systems*, peripheral cameras are integrated to localize objects of interest, and foveal cameras are used to get a high resolution representation of the target. Our first robotic platform *BioVision*, is typically classified among this type of systems. Bjorkman, Kragik *et al.* [7], [45], [107], [113], [120] have worked on a system of four cameras (two foveal, two peripheral). Not only were the pairs of cameras alternating between foveal and peripheral visions, but also estimating the depth of the environment from stereoscopy. Other teams have simply used PTZ cameras to have an easier setup based on a single camera, while alternative foveal and peripheral view points. Minut *et al.* [109] have used this kind of setup to jointly learn to localize and identify an object, while Kragic *et al.* [110] have used a descriptor to adjust the zoom level and enable a better recognition. Gould *et al.* [103] as well as Canas *et al.* [102] have mounted a PTZ camera on a mobile platform able to explore their environment by moving in a room and identifying simple objects.

Mobile robots: When the robot moves in its environment, mechanisms similar to saccades and fixations are used. The shift of attention is associated with a displacement of the robot, either to put the object at the center of the visual frame [110], or to simply get closer and have a better representation [102], [103]. However, two

main differences have to be noted as compared to a non-mobile robot. First, the displacement modifies both the resolution of the object and the orientation at which the object is seen. This way, the object can be represented and identified based on a set of images rather than on a single one [121], [122]. Second, the time for moving the robot to a new position may have to be considered, as the distance to the target can be significantly different, and the required time to reach it is roughly proportional to this distance [22], [123]. Both of these problems are related to our research when considering experiments on our mobile robot platform.

Interactive perception: A last category could be the one of robots able to modify their perception of objects by manipulation and interaction. The field of interactive perception then proposes to use the robot's actions to directly interact with the object. When using a robotic arm, interactions such as pushing can help object segmentation given the concept of "what-moves-together-belongs-togeter" [124]. More complex interactions may be used to learn object appearance, either with robotics arms able to grasp and manipulate objects [113], [117], [125], possibly with additional tactile sensor [126]. The presence of a human user assisting in the learning task is common in this case, either to place objects in the robot's hand [125], [126], or to interact with objects instead of the robot [127], [128]. The advantages of interactive learning versus simple robot displacements is the ability to add proprioceptive inputs in the perception of the objects. The robot then not only learns the visual appearance of objects, but also their physical properties.

Saliency in robotics

Among all saliency methods, only part of them are well-suited for robotics setups. First, methods taking advantage of the center-bias are no longer efficient when the viewpoint is chosen by a robot. Then, the saliency map should be light and fast enough to compute to be reactive to the robot's displacement while dealing with limited computational resources. Lastly, it should be adapted to the sensor and potentially exploit all the components they can offer (for example, exploiting depth if an RGB-D sensor is available).

Objectness and top-down saliency: In robotics, we have shown that saliency is often tightly bounded with objects, and therefore the concept of *objectness* and object localization. In that regard, pure bottom-up attention is not always well-suited for robotics applications. Indeed, objects are not always intrinsically salient, and conversely, some elements of the environment can show salient property while being irrelevant for the robot's task. Top-down modulation is then a good approach to ensure the relevance of salient targets. This can be done by exploiting the context in which the object is supposed to be [109], [129], [130], or by enhancing features relative to the object [46], [74], inhibiting irrelevant ones [75], [76], or both [45], [59]

Proto-objects: Related to the idea of visual attention targeted at objects, many robotics systems rely on saliency maps able to provide an estimation of the boundaries of the object [47], [131], [132], sometimes referenced as *proto-objects*. As mentioned in Section 3.2.2, some saliency maps directly provide this feature. It is therefore easier to focus the attention at the center of the object, using an appropriate scale.

Exploration-exploitation trade-off: Although top-town attention is better-suited to localize and identify objects, it is sometimes useful to alternate between bottom-up and top-down in order to keep a good general exploration of the environment [133], [134]. The alternance between the two may be obtained by a static

(a) (b) (c)

FIGURE 3.8: Examples of segmentation approaches (a) over segmentation by SLIC superpixels [138]. (b) foreground-background segmentation. (c) semantic segmentation [139]

weighted fusion [135], more complex schemes driven by temporal differential equations [45] or just an alternation between exploration and exploitation modes [136].

3.3 Localizing objects in an image

We demonstrated that finding object candidates and top-down saliency were often tightly coupled in a robotics context, and that object detection can be obtained with saliency maps. We see object localization in this section at a wider point of view, and we investigate the alternative approaches to detect and localize objects in their environment from a still input image. An interesting review mentioning recent deep learning approaches in visual understanding problems has been written by Guo *et al.* [137]

3.3.1 Segmentation-based approaches

Image segmentation can be define by the process of splitting an image into several segments, or groups of pixels. Most of the time, the segments are grouped in a meaningful way and should help for further processing or for image understanding. Figure 3.8 illustrates the three types of segmentation that are further described in this section: *over-segmentation*, *background-foreground segmentation*, and *semantic segmentation*.

Over-segmentation

In over-segmentation, the image is separated into an arbitrary number of segments. These segments do not have a particular semantic meaning. The only criterion to gather pixels are based on image properties such as color, texture, or orientation. Pixels within the same over-segmented region should also be gathered in terms of position, so that regions are spatially consistent. Over-segmentation is usually useful as a pre-processing step to form meaningful clusters, that should be combined later on. These approaches are usually highly tunable depending on the user's requirement (size of superpixels, edge smoothness, *etc*...).

Most popular over-segmentation approaches consists in dividing the image into superpixels. A first type of superpixel uses seeds and grows regions around them. The SLIC superpixels [138] for example produces a set of segments from a regular distribution of seeds in the image. These seeds are used as an initialization for a

K-means clustering considering both pixel position and color. Similarly, the SEEDs superpixels [140] are obtained from a regular grid, by incrementally exchanging pixels between regions to adjust boundaries. This approach is particularly interesting as it is both fast and efficient. A second family of superpixels is based on graphs [141], [142], that are incrementally cut or clip the graph vertices (typically the pixels) based on objective functions. Graph-based approaches usually propose regions of more variable size and shape.

Over-segmentation is also applicable to point cloud or RGB-D images. The geometrical curvatures, normal orientations, convexity and symmetries of the point cloud are then extracted to generate super-voxels [143], [144]. 3D segmentation is also a complementary information that can be added to traditional RGB over-segmentation, either following an early [145] or late [52] fusion approach.

Superpixels are used in our work for two different reasons. First, for a feature extraction process, second, to recover the shape of the salient objects when the saliency map has a weak resolution. We use in both cases the SLIC superpixels [138]

Foreground-background segmentation

When doing foreground-background segmentation, the goal is to separate each pixel of the image into background and foreground. The problem can then be seen as a binary classification problem. The definition of background and foreground is unclear and depends on the context it is used for. If the task is to segment objects in general, we can consider foreground as being objects and background as being everything else. This notion of foreground and background has a lot in common with the definition of saliency, foreground items being most of the time considered as relevant elements.

When considering video, moving objects may be seen as forground elements. Traditional background learning and subtraction [146] may suffice if the sensor is set at a fixed position. Otherwise, other approaches exploit motion and tracking to find the boundaries of the target [147], [148].

Similar to superpixels, traditional approaches were using seeds and Markov random fields to optimally segregate foreground and background based on color, texture or contours [149], [150]. Those methods are sometimes trained on a set of representative images to learn the optimal parameter values [151]. Some other approaches also exploit the prior knowledge that the object of interest is surrounded by the background and rely on a logpolar transform as the key of the segmentation [152], [153].

Other methods [154], [155] start with an over-segmentation approach and progressively group superpixels based on compactness, convexity, curvature or symmetries to reconstruct foreground objects. The notion of symmetry to detect objects is particularly exploited with RGB-D sensors and tend to be efficient even in case of partial occlusions [156], [157].

For indoor environments, and when a depth cue is available, a popular separation technique exploits the fact that objects of interest are generally lying on planar surfaces [18], [52], [130], [156]. Planar surfaces are either tabletops or floor, or any other horizontal support. The main idea is to first detect this plane (typically using a RANSAC algorithm [158]) to discard the points belonging to this plane, and to use geometrical constraints to detect objects lying on it. A common approach is to only keep elements contained in the enclosing convex hull of the detected plane [130]. When the major plane is the floor, other important structures such as walls and ceiling must be detected and removed as well, as proposed by Caron *et*

al. [18]. Many RGB-D datasets are composed with objects on tabletops, either well separated [30] or in a more cluttered scenario [157]. These types of approaches are performing well on these datasets, as long as a large portion of the main plane is visible on the image.

In our work, the produced saliency maps could amount to a sort of soft background-foreground segmentation, where foreground elements would be the objects. Depth-based segmentation is also at the heart of our saliency learning procedure as we use a modified version of the work of Caron *et al.* [18] to generate a learning signal.

Semantic segmentation

Semantic Segmentation aims to label each pixel with a class of objects (such as car, pedestrians, dogs, ...) and non-objects (for example, water, sky, or road). It is therefore a multi-class classification problem. In recent years, challenging datasets have been proposed to evaluate semantic segmentation accuracy. The most famous one is probably the PASCAL VOC 2007 dataset for RGB [159] and the NYU dataset for RGB-D [160].

Before deep learning approaches, the best performing techniques were based on hand designed features applied on a set of pixels to generate patches. Those patches were locally describing the image and were sent to a classifier, typically a random forest or a boosted cascade, to predict the class probability of the center of the patch. These types of approaches were providing pretty noisy probabilities and were typically refined with conditional random fields [161]. Other techniques tried to predict every pixels of the patch rather than just the center [162], or used an over-segmentation technique to group pixels [163]. Classification refinement was typically able to improve the classification in large areas like sky, but did not really improve small clusters. Lastly, Gupta *et al.* [164] achieved good performance by detecting boundaries and applying hierarchical clustering as a pre-processing step to segmentation.

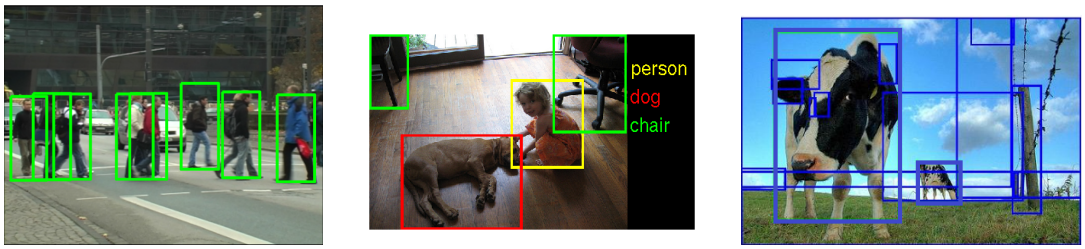
Similar to many other fields of computer vision, a major breakthrough occurred in the field of semantic segmentation through CNN-based techniques. Early deep learning approaches were based on blocks identification, but provided a segmentation mask at a poor resolution [165]. More recent approaches such as SegNet [166] are following the auto-encoders philosophy to design fully convolutional networks able to retrieve a segmentation mask at the original scale. This approach is sometimes coupled with a post processing using recurrent neural networks [167].

An interesting advantage when using CNN for semantic segmentation is the possibility to train a network in a weakly supervised basis. Papandreou *et al.* [168] typically demonstrated that state-of-the-art performance could be obtained from a small number of fully supervised data and a large number of weakly supervised data such as bounding boxes or raw image with a single label.

Semantic segmentation is not directly used in our work, but the methods for retrieving the original resolution and the weak supervision aspects are interesting items that have inspired some of our choices.

3.3.2 Object recognition and bounding box proposals

Another way to consider object localization is to think in terms of instances rather than at the pixel level. When doing object localization in this case, the output of the system is typically one or several bounding boxes, providing information about the number of instances, and a rough localization and size of them in the image. The



(a) (b) (c)

FIGURE 3.9: (a) Object (here, pedestrians) detection. (b) Object recognition (from ImageNet challenge [24]). (c) Example of class-agnostic bounding boxes (with EdgeBoxes [169])

boundaries are not precisely detected, but the information is easier to interpret at a semantic level.

Before going any further, a reminder on the terminology in this field is presented. Figure 3.9 provides an illustration of the three discussed categories of object localization that are *class-agnostic bounding box proposals*, *object recognition*, and *object detection*.

Sliding-window approaches

Until recently, the best performing approaches for object detection and recognition were based on sliding windows, where a light classifier was applied multiple times on a portion of the image (the window) at regular intervals all along the image. This approach then required a significant amount of inferences, considering about 10^4 windows per images, and potentially up to 10^7 if the task also required to detect the scale of the object. When considering an object recognition problem, each window had to be evaluated for each detector, thus multiplying the number of operation by the number of output classes. As a result, a very light object classifier was applied in these types of methods to provide a result in a reasonable time. Incidentally, the efficiency of the classifier was pretty limited. In addition and in spite of the large number of windows, they were not necessarily adapted to the size of the object to detect and then provided a very limited efficiency.

Among the most popular and best performing approaches relying on a sliding window, one can think of the Viola and Jones object detector [170]. The basic idea was to train a binary classifier to discriminate between the object to detect and distractors. The original image was first decomposed into wavelets, and classification was obtained though a boosted cascade classifier. A more recent and successful approach was the Deformable Part Model [171], where discriminative portions of objects were first detected and combined based on their relative positioning with each other.

Class-agnostic bounding box proposals

In recent years, other approaches have proposed to directly generate bounding boxes around potential objects of interest to limit the high amount of inference required by the sliding window approach. The class-agnostic bounding box proposals then aims to take advantage of the image shapes and colors to produce bounding boxes that may contain objects. They are not restricted to a single class but rather exploit the general properties of objects (colors, context, boundaries) to

generate bounding boxes. Again, these properties are strongly related to the notion of saliency. Among available techniques, two categories of approaches have emerged (See the review of Hosang *et al.* [172] for an extensive review on this field.

The first category is referred as the grouping methods. The most popular one probably being the *Selective Search* [173]. In these approaches, the goal is to generate segments that are likely to represent objects. The most naive approach would be to generate a bounding box out of each segment provided by a segmentation technique. Most of the time, these methods first apply an over-segmentation technique, and combine the segments based on shapes, convexity, appearance of boundaries. Several combination trials are performed, and related bounding boxes are produced accordingly. One of the advantages of these approaches is that they provide a segmentation proposal on top of bounding boxes.

The second one is the window scoring proposal method. In this category, *Objectness* [174] and *EdgeBoxes* [169] are the most famous techniques. In these approaches, a set of randomly sampled bounding boxes are generated, and a score is attributed to each of them based on shapes, contours, superpixel straddling and so on. Although faster, these approaches are known to be less accurate in localization.

Class-agnostic bounding boxes are now typically coupled with deep learning classifiers, faster R-CNN [175] probably being the most famous example. The first reason is that they significantly reduce the number of proposals over the sliding window, thus making efficient the use of a slower classifier. The second one is that proposals are most of the time more accurately centered and scaled than a sliding window approach would. This way the recognition is likely to be more efficient.

Several methods have been proposed to evaluate the efficiency of a box proposal. The standard one is probably the recall vs number of proposals, relying on an *intersection over union* measure (IoU) that defines how well a box matches the ground truth. As this method relies on an arbitrary threshold to determine the match, Hosang *et al.* [172] proposed to use the average Recall (AR) that uses an integral over a set of threshold and demonstrated that this approach was more objective and stable over datasets. Another approach is to use the detector responses around objects.

In our work, we propose a way to use saliency maps to provide relevant bounding boxes candidates around salient objects. Our technique is based on a posterior bias on the scores produced by the EdgeBoxes algorithm.

Multi-class object detection

Object detection and object recognition are two different classes of problems, usually solved in a similar way. Object recognition consists in the detection, localization and labeling of all objects of an image (or a restricted number of classes) with a labeled bounding box. In object detection, the purpose is similar except that only one class of object is considered. For example in pedestrian detection, the goal is to detect all pedestrians (and only pedestrians) with their bounding boxes. We describe in this section methods able to solve the more general problem of multi-class object detection.

Multi-class object detection is close to the problem of semantic segmentation. The techniques used in both fields sometimes even are the same and simply differ in the post-processing step. Common datasets such as PASCAL VOC [159] or Microsoft COCO [176] propose both annotations alternatives.

Today state-of-the-art's methods all use deep learning to identify objects, but two major axes are then used to locate them.

The first one is based on bounding box proposals. Each proposal is cropped and send to a classifier able to identify the class of the whole image, finally deciding if the proposal contains one of the objects to detect or not. The most representative work in this type of approaches is R-CNN or faster R-CNN [175]. As a critical aspect of these methods is the quality of the bounding box, approaches such as LocNet [177] try to refine the proposal to obtain a better identification.

The second one uses a single convolution pass on the whole image to simultaneously localize and identify the object. These approach, such as YOLO [105], are generally much faster and have more in common with semantic segmentation by designing fully convolutional networks able to solve both tasks [178]. Another interesting approach is to use labeled images as a weak supervision signal to turn an image classifier into an object detector [179].

Like semantic segmentation, we do not consider the problem of multi-class object detection in our method, but it could be a direct application of our saliency maps. In addition, the state-of-the art in this field is a source of inspiration in terms of methodology.

3.4 Learning visual saliency on a robot

This section describes the main requirements to learn visual saliency directly on a robot. After presenting existing methods related to visual attention and learning, we focus on three main components of the approach described in the next chapter. This section does not aim to be an extensive review in the fields of machine learning, but rather a spotlight on the mechanisms underlying our approach.

3.4.1 Robotics systems that learn visual attention

Learning is used in robotics for many types of applications. Specifically for saliency, we mentioned in Section 3.2.2 the pattern classification and the fully connexionist approaches. These approaches are however designed to be learned offline and not modified during the robot's exploration. Learning or refining visual attention directly during exploration is not so common. Nevertheless, we identified two cases in which this aspect is used.

Top-down refinement

The first one consists in refining the top-down modulation by finding appropriate weights to combine the extracted features. In VOCUS, Frintrop [46], [180] has proposed to learn weights to discriminate a specified target based on training images. Learning was done through a Region Of Interest (ROI) found in a manually specified rectangle, and each weight was determined based on the mean activation ratio with and without the ROI.

Rasolzadeh *et al.* [45] based their approach on the same idea of ROI, but proposed to find the weights by solving an optimization problem with the Levenberg-Marquardt algorithm. They further improved the method by integrating the context in the optimization. As the problem was not convex anymore, the weights were found by training a neural network. An additional idea in this work was to learn the top-down weights online by alternating between bottom-up and top-down attentions. Bottom-up attention served as an open exploration to select targets and discover objects. Once an object was discovered, a foveated segmentation was used

to create the ROI, and the top-down weights were optimized. To combine bottom-up and top-down attentions, a system of temporal differential equations was used to weight the importance of each component.

In the context of learning how to combine features, Borji *et al.* [44] have investigated the use of several classifiers to combine both bottom-up and top-down (in this case, object detectors) cues. They concluded that the AdaBoost classifier had the best prediction accuracy for this task. Nevertheless, this approach was trained offline, in a fully-supervised manner, based on eye fixation datasets.

Visual attention control policies

The second type of methods aims to learn eye movements displacements in order to better perceive the objects of interest. These approaches typically rely on reinforcement learning techniques.

Minut *et al.* [109] have presented a system based on Q-learning to determine the area the most likely to contain the object to identify. They then used the camera zoom and small saccades to precisely localize and identify the object.

Saccades can also be directed towards areas of interest within an object to improve its identification. Paletta *et al.* [49] proposed a framework based on Q-learning to learn the best scanpath within an object to ensure correct identification.

Lastly, the work of Borji *et al.* [181], inspired from McCallum's research [182], proposed to learn a top-down attentional model able to simultaneously discriminate objects in an environment (for example traffic signs, buildings) from their visual appearance, and learn the best sequence of actions to reach a goal (for examples, how to move the robot across the streets to reach a house).

3.4.2 Learning constraints of our setups

Learning visual signals on a robot can be done in many different ways, but we here focus on two aspects related to our setups. First, our robot starts exploration with no prior knowledge about the environment, collects data through observations and learn a representation on-the-fly as exploration goes. Therefore, exploration needs to be coupled with an *online learning* mechanism. Second, our approach does not include a human in the loop, so that learning has to be done without supervision. We talk here of *self-supervised learning* rather than unsupervised, that has a different meaning in the machine learning literature.

Online learning

Online learning, or *incremental learning* refers to a machine learning approach where data arrive in a sequential order, and where the model integrating those data is updated simultaneously with the reception of new data. Online learning is opposed to offline learning, or batch learning, where data is available from the beginning, and where the model is build in a single shot. Online learning is common in robotics to refine predictions and representations as new perceptual data is acquired by the robot.

Statistical methods: Methods doing predictions based on a statistical distribution of the data are well-suited for being updated incrementally. In particular, classification based on a k-nearest neighbor [183] comparison is trivially updated by adding the new sample to the representation dataset, although the search method may also require a KD-tree update to be faster. A major drawback of this approach

is that memory increases as new data arrives. Voting methods based on bags of visual words [184] also have a natural incremental version where statistics (counts) of visual words are updated on the fly.

Online random forests: Most offline machine learning algorithms also have their incremental version, and it is the case for random forests we have used in our work. To make a random forest incremental, the general approach is to update the splits of the tree by maintaining quality scores at every leaves, and to grow the tree incrementally [185], [186]. However, most online random forests approaches are memory consuming (the whole dataset is stored and re-used for updates), slow for a large dataset, and do not perform as well as in offline mode. A recent promising approach is the Mondrian forests [187] that determines new tree splits based on mondrian processes. We have developed in our work an alternative technique to update random forests faster than what offline training would.

Neural networks: Neural networks are from essence online classifiers. Although often used in batch mode and trained by feeding the network with the whole dataset several times (a whole dataset pass is called an *epoch*), samples are provided one by one to update the network. A very desirable property of these methods is that there is no need to keep the whole dataset for updates. Once presented to the network, the model keeps an internal memory of the sample. A major issue though is that the order in which examples are presented strongly influences the performance of the system. The earliest samples tend to be forgotten and the model is more accurate for recent examples. Moreover, presenting too many similar examples in a row may have catastrophic consequences in the learning quality (this phenomenon is called *catastrophic forgetting*). Online learning with a neural network must then be considered carefully to lead to successful results. In CNNs, the problem is the same. Although little work has been presented yet in that regard, the field is actively investigated as well [188].

Self-supervised learning

In most supervised learning techniques, a human contributes in the task of annotating data, which means provides a trustful signal that the algorithm should follow to build its predictive model. In the case where a robot does not use any human assistance, other approaches should be considered.

Alternatives to supervised learning: A first family of learning without human supervision is *unsupervised learning*, where classes or clusters are determined only by considering properties of the dataset distribution. Another approach is to learn from a partially or roughly annotated dataset, referred respectively as *semi-supervised* and *weakly supervised learning*. When the signal used for learning is a reward from the environment, following an action, the problem is called *reinforcement learning*.

Automatic or self-supervised annotation: Sometimes, there is a way to automatically annotate data, or to generate annotated samples without any supervision. We call this process *self-supervised learning*. In computer vision, a typical example of automatic annotation is the colorization of black and white films, since there is an unlimited amount of color films and that conversion from color to grayscale is trivial. Another example is the work proposed in Make3D [189] that aims to learn to predict the depth map of an RGB image. From RGB-D sensors or stereo cameras, a fully supervised correspondence is available between these two components. In robotics, sensory-motor correspondences may also be automatically obtained by performing an action from a motor command, and waiting for the sensor feedback

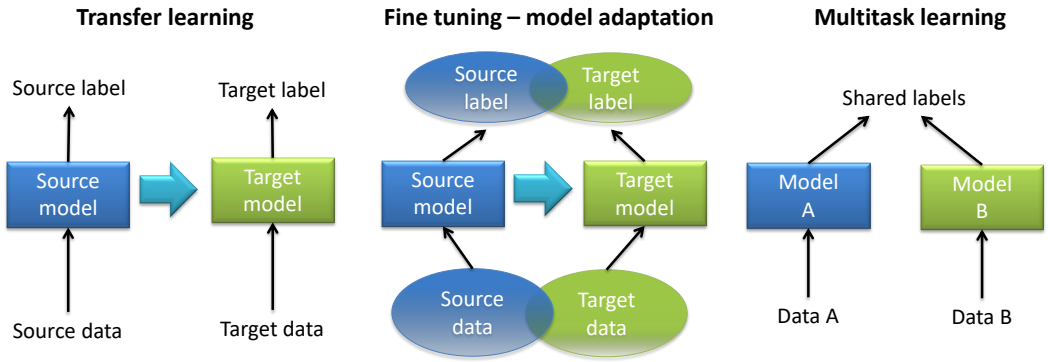


FIGURE 3.10: Two categories of transfer learning: domain adaptation and multi-task learning

that plays the role of the annotation [14], [190]. Our system is based on this principle of automatic annotation of the saliency, based on a correspondence between a depth-based segmentation and RGB images.

Transfer learning: *Transfer learning*, or *inductive transfer* [191] aims to transfer knowledge between a source and a target. The problem is typically formalized by a source/target data, a source/target model and a source/target label. The goal is then to take advantage of the source model to improve the target one (see Figure 3.10). This field gathers a lot of different methods, but two of them are of particular interest for our work.

The first one is the *domain adaptation*, or *fine-tuning*: in some cases, a large, fully annotated dataset is available for a source problem, and a limited amount of data (annotated or not) is available for target problem, having a lot in common with the source (in terms of data and/or labels). It is then possible to take advantage of the large dataset to train a robust model, and use the limited dataset just to optimize the learned model to the target task. Domain adaptation has been widely exploited in speech recognition to adjust a natural language trained on thousands of speaker to a single user [192], or in CNN for a specific task on which little annotation is available [193].

The second one is the *multi-task learning*, where problems A and B are solved at the same time by exploiting common properties between the tasks. Our learning method follows this type of approach, but solves the two tasks sequentially: In our case, a first object recognition task is learned offline, and the corresponding model is used to automatically annotate data for the task of saliency learning. This kind of approach is called a *knowledge transfer*, and is also used in deep learning framework [194].

Weakly-supervised learning: In a weakly supervised context, the data is poorly annotated, and the model is able to generalize over this signal to learn more than what was originally provided (see example of Figure 3.11 where the goal is to localize cats in images with cropped images of cat as the only learning signal). The core characteristics of the annotated data are then naturally extracted from the partial annotations. With such learning, it is then possible to generalize the learned model, and to apply it to a slightly more general task. Several examples of weak supervision have been presented in Sections 3.3.1 and 3.3.2 for the task of semantic segmentation or object recognition. We also rely on such weak supervision to generalize saliency over a partial learning signal.

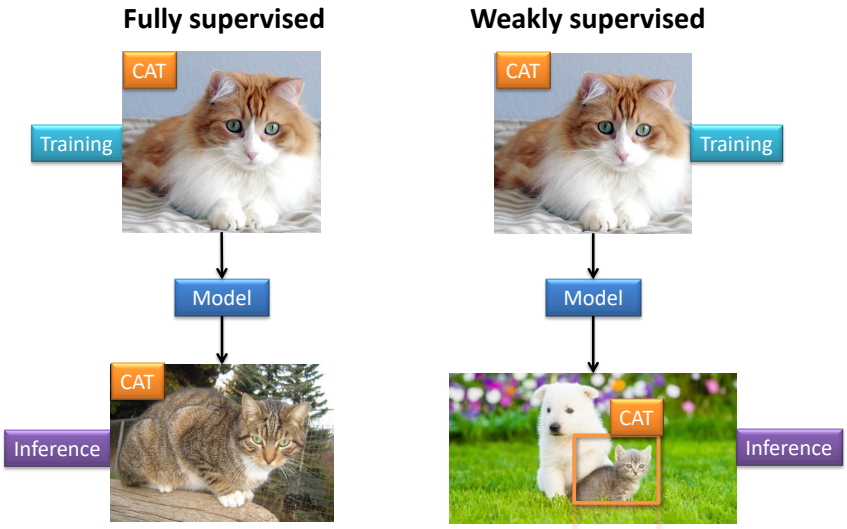


FIGURE 3.11: Fully versus weakly supervised learning

3.5 Conclusion

In this chapter, we have presented an overview of the background related to the approach to be proposed in the next chapter. Visual attention, object localization and machine learning are separate fields, but they all contributed to the construction of our methods. Moreover, we have shown that those fields were sometimes largely intertwined, and that our system precisely lies at their junction.

We aim to design a system able to learn visual saliency on a robot. Visual saliency is at the heart of many visual attention systems, and we have re-used a very classical scheme to construct our own technique. Then, as our robotics platform are designed for indoor environments, our saliency must be oriented for the task of localizing objects in their surrounding. We therefore investigate this field that is more related to computer vision. Lastly, as our approach involves learning on a robot, we have studied machine learning techniques able to satisfy the constraints of our setup.

The next chapter is dedicated to the technical description of our proposed approach and its connections with each of the aforementioned fields.

Chapter 4

Proposed Approach

4.1 Introduction

For a robot to efficiently identify the interesting elements of its environment, we base our approach on saliency maps. More specifically, we would like to learn these saliency maps on-the-fly during exploration. Thus, these maps will be specialized for the environment the robot is exploring, for the task the robot is asked to accomplish, while remaining flexible to any change or novelty in the environment.

This chapter describes the algorithms and software components developed to learn such a saliency model. We first position our approach versus the state-of-the-art provided in Chapter 3. We also provide a general overview of the approach before going in details through each component of the architecture.

The incremental saliency learning is made possible by three major items that are feature extraction, object detection and online classification. We provide for each of them a description of several possible methods, and discuss their advantages and drawbacks.

Lastly, we describe the pipeline to produce saliency maps having the desired properties, and we suggest a measure to exploit these maps as a mean to produce bounding box proposals around salient objects.

4.2 Overview of the method

4.2.1 Positioning versus state-of-the-art

Regarding visual attention

Our approach has a strong background on the field of visual attention. We intend to follow the traditional visual attention pipeline described in Section 3.2.1 consisting in feature extraction and feature combination. Moreover, one of the setups used for evaluation is a foveated platform, very similar to many existing systems (See Section 3.2.3).

However, our approach differs from most visual attention techniques for the following reasons:

- Instead of using saliency maps as a black box, plugged as is in a visual attention system, we propose two operating modes: a learning mode, and an exploitation mode.
- In the learning mode, we focus on how to build an efficient feature combination online, during the robot's exploration. This combination should be consistent with the robot's future tasks.

- In the learning mode, the focus of attention is not directed towards the most salient elements. We do not discuss it here, but Part II of the manuscript suggests alternative methods to learn saliency more efficiently.
- In the exploitation mode, the combination model that has been learned is exploited to produce saliency maps. From there, it is possible to follow a classical visual attention pipeline.

Our method is designed to learn top-down types of saliency maps, that are dedicated to object detection tasks. In addition, our saliency models are specialized for the environment they were trained in, an are very likely to fail if exploited in a different context.

Regarding object localization

As discussed in Chapter 3, the multiple similarities between object-oriented saliency and object localization positions our approach at the junction of those two fields. Although our goal is to produce saliency maps, we also rely on object localization techniques for several reasons

- In a first setup, we rely on a background-foreground, depth-based, object-oriented segmentation technique to learn the saliency.
- In a second setup, we use an object classifier as a weak supervision signal to learn saliency maps, similar to recent approaches presented in Section 3.3.2
- We show that our saliency maps can be used to bias the output of existing agnostic bounding box proposals methods. The outcome is a set of bounding boxes that are much more likely to contain salient objects.

Regarding learning

Our method has a lot in common with the mechanism proposed in Frintrop [46] or in Rasolzadeh's work [45], where an ROI of the visual field was exploited to modulate top-down saliency. In Rasolzadeh's work, this modulation was done online, and an efficient trade-off was found between open exploration and samples seeking for top-down refinement. Our approach differs from these work for the following reasons:

- Top-down modulation is not obtained by a simple linear combination of the feature maps, but by a non linear classifier, able to capture more complex feature combinations.
- The method is self-supervised which means that salient elements of the visual fields are detected (when possible) and integrated to the model without any user supervision.

Many approaches propose to learn visual saliency offline, from a large database, the best performing results being obtained with CNN architectures [43]. However, learning saliency online with a classifier, without any initial database is far less common. We then investigate the online aspect of learning, while taking advantage of the CNN capacities.

In our approach, self supervision is made possible by a multi-task transfer learning run sequentially (called *knowledge transfer*). In our case, we learn offline, before

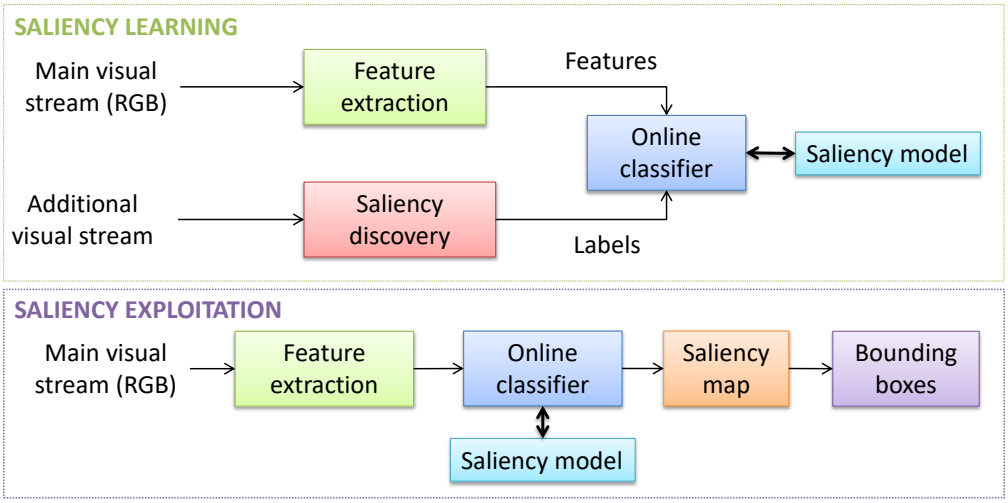


FIGURE 4.1: General architecture of our system

robot's exploration, an easy and restrictive task (for example, object recognition) that produces a reliable estimate (the class label). Then, we use the predicted output as a *learning signal* for a more complex tasks (for example, low-resolution object localization in a cluttered environment).

Lastly, the self-supervision provides a learning signal that is restrictive, thus making the problem weakly supervised. Our approach then aims to generalize over this weak learning signal.

4.2.2 General mechanisms

Our approach is designed so that the robot moves within the environment and receives image streams. Saliency is then learned progressively as image are acquired and processed.

Our approach is composed with three interconnected modules for learning, and works in a learning mode and exploitation mode. We here provide an overview of these module and the way they interact with each other (see Figure 4.1 to visualize the general architecture).

Saliency learning

During the learning phase, the robot explores the environment and progressively receives visual information that is processed and integrated to the model. The visual information is composed with two visual streams denoted as the *main visual stream* and *additional visual stream*. We also define as an *observation* a set of two synchronized frames from the visual and additional streams.

The core idea of the saliency learning is to continuously provide a set of *features* and *labels* (essentially two classes denoted here as "*salient*" and "*not salient*") that are send to an *online classifier*. The classifier then learns a saliency model that can be exploited to infer saliency maps. The classifier chosen in this study is a random forest designed to be updated incrementally (see Section 4.5.1).

Features are extracted from the *main visual stream* (RGB in our case) from which saliency is to be estimated. In Section 4.3, we propose thee feature extraction techniques to this end.

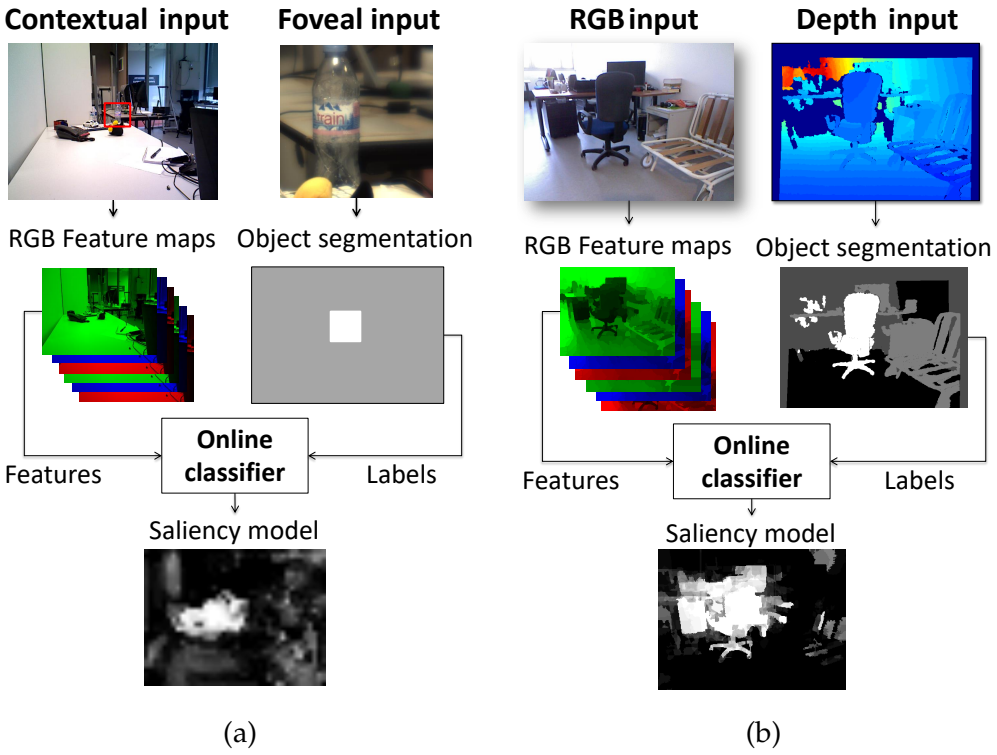


FIGURE 4.2: Saliency learning applied to two kinds of additional streams: (a) an RGB foveal input and (b) a depth input.

However, a more complex issue is to provide a reliable supervision signal without any human assistance. To this end, we rely on an *additional visual stream* from which a reliable saliency signal is obtained. This signal is then turned into labels and sent to the classifier. This saliency estimation must be trustful enough to be used as a learning signal, but can also be restrictive. The role of the classifier is then twofold: first, to transfer the saliency knowledge from the additional visual stream to the main visual stream. Second, to generalize the restrictive signal to produce saliency maps on the whole main visual stream.

Saliency exploitation

Once a saliency model is available, this one can be exploited to produce saliency maps in the specific environment it was trained for. In this configuration, we do not rely on the additional visual stream anymore. Features are extracted from the main visual stream, the same way it was done for learning, and these features are sent to the classifier to infer saliency according to the model (see Section 4.5.2).

Additionally, in Section 4.5.3, we show that the saliency map constructed this way can be used to produce bounding boxes around salient objects.

Two application scenarios

We describe in Section 4.4 two scenarios for which an additional visual stream can produce a restrictive saliency as a learning signal. These scenarios are represented in Figure 4.2.

The first scenario (Section 4.4.1) requires both a *contextual camera* (also called *peripheral camera*) and a *foveal camera*. The contextual camera plays the role of the

main visual stream, while the foveal is the additional one. Saliency is then detected based on an object recognizer trained for some specific objects on the foveal stream. Saliency is then determined by whether an object was identified in the fovea or not. Here, the saliency estimate is partial as salient objects are detected only when visible in the fovea.

The second scenario (Section 4.4.2) works mainly for indoor environments and considers salient objects as being on planar surfaces. The additional visual stream here is a depth map produced by a Kinect or similar sensors. The depth map is processed and produces a segmentation of the salient objects when detection was possible (when object was seen from an appropriate point of view). Saliency is then partial as salient objects are not always detected.

To make data formatting easier, the information produced by the additional streams are turned to what we call *segmentation masks*. A segmentation mask provides a pixel-wise mapping between the main visual stream and the saliency found in the additional stream.

4.3 Feature extraction

We want to use visual features based on the RGB components to construct saliency maps. Those features must be light enough to be computed in real time on the whole field of view, and representative enough to discriminate salient and non salient elements. Lastly, the features must be stable enough to avoid saliency misdetections, especially in areas of strong or sharp color variation.

We present here three different types of features that we used and evaluated. We systematically discuss the choice for such features, as well as their advantages and drawbacks.

4.3.1 Itti and Koch feature extractor

Motivation

A first natural idea to obtain relevant features for saliency is to look at the one commonly used in the literature. In that regard, many visual attention system rely on a saliency model created by Itti and Koch [38]. As it is flexible and modular, our first attempt was to use the filtering procedure proposed by the method as a feature extractor. Frintrop [46] has used a similar type of filtering and obtained better results. We did not investigate the performance of these, but expect similar performances.

Description

Itti and Koch have proposed a method for visual attention systems, including a saliency map computation and a way to consider this saliency map in the allocation procedure of the attention. We are more specifically interested here in the batch of filters applied on the input image to produce a set of feature maps, as presented in Figure 4.3.

These filters are based on Gabor filters [195] of various orientations at several scales.

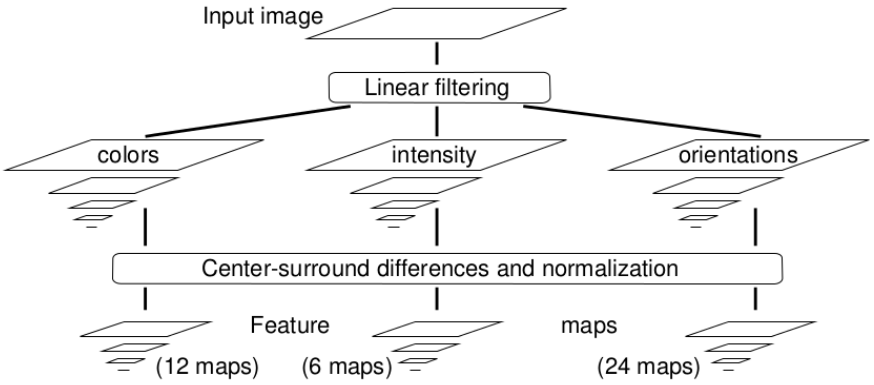


FIGURE 4.3: Feature extraction procedure from Itti and Koch [32]

A Gabor filter is defined by a scale σ and an orientation θ as:

$$\begin{cases} G(\sigma,\theta) = exp(-\frac{1}{2}\frac{x_\theta^2+y_\theta^2}{\sigma^2})cos(2\pi x_\theta) \\ x_\theta = x \times cos(\theta) + y \times sin(\theta) \\ y_\theta = -x \times sin(\theta) + y \times cos(\theta) \end{cases} \tag{4.1}$$

More precisely, each filter is obtained by the following equation:

$$O(\sigma_f, \sigma_c, \theta) = |G(\sigma_c, \theta) - G(\sigma_f, \theta)| \tag{4.2}$$

where $\theta \in \{0, \frac{\pi}{4}, \frac{\pi}{2}, \frac{3\pi}{4}\}$ is the orientation, $\sigma_f \in \{2, 3, 4\}$ is the scale of the central stimuli and $\sigma_c \in \{\sigma_f + 1, \sigma_f + 2\}$ the scale of the peripheral stimuli. In total, 24 filters are therefore calculated. Another type of filters represents the difference of intensity between the center and the periphery, obtained by equation 4.3:

$$I(c, f) = |I(c) - I(f)| \tag{4.3}$$

where I is a Gaussian pyramid created from a grayscale image. $f \in \{2, 3, 4\}$ and $c \in \{f + 1, f + 2\}$ represent the scales of this pyramid. In total, 6 filters of this type are computed. Lastly, we obtain the center-surround color opposition, from equations 4.4 et 4.5:

$$RG(c, f) = |(R(f) - G(f)) - (G(c) - R(c))| \tag{4.4}$$

$$BY(c, f) = |(B(f) - Y(f)) - (Y(c) - B(c))| \tag{4.5}$$

where R G B Y stand for the Gaussian pyramids obtained from the red, green, blue and yellow components of the image, at the same scales f and c used for the grayscale filters. In total, 12 filters are computed this way.

Pros and cons

These features are very common, many implentations are publicly available. They are also fast and easy to compute. However, the center-surround averages provide a poor discrimination of the saliency at the borders of an object.

4.3.2 Make3D-based feature extractor

Motivation

The problem considered in Make3D [189] is the one of predicting depth from a single RGB image. To this end, features are extracted from the RGB image for each pixel, and sent to a regressive classifier to predict the depth of each pixel independently. This classification phase is followed by a refinement stage, enhancing the overall depth map consistency with Markov Random Fields.

As explained in Section 3.4.2 of the previous chapter, the saliency learning approach is a sort of transfer learning from depth to RGB (at least for one of the two segmentation techniques), and saliency maps are obtained from features extracted for each pixels of the original image. Make3D then shares strong similarities with our technique and the type of signals we aim to manipulate. We then build our second feature extractor on the ideas proposed by Make3D.

Description

In Make3D, absolute and relative features are considered and extracted from the RGB image. Absolute features are sent to a classifier to estimate the absolute depth of a pixel, while relative ones are used to refine the global depth map. We base our technique on the absolute features. For each pixel, texture variation, gradients and color statistics are obtained from a patch surrounding the pixel (19 types of features in total). Then, to better describe the geometry of the surrounding, these features are also computed for neighbor patches (up, down, left, and right) at three different scales. In more recent work, Saxena *et al.* [196] also compute superpixels on the image and calculated a single descriptor per superpixel.

To extract our features, the first step consists in a superpixel extraction. We compute the SEEDS superpixels [140] on the whole image. The number of SEEDS to produce depends on the size of the input image, and on the average size of the objects on the image. Superpixels should then be small enough to oversegment the salient objects. In our experiments, we have used between 150 and 350 superpixels for images of size 640×480. Once extracted, we compute the mean RGB value in each of these superpixels.

Then, for a given pixel of the image, we associate a feature vector based on the mean superpixel value containing this pixel, and the mean superpixel values containing neighbor pixels at three different scales. More precisely, we obtain three features for the central pixel (red, green and blue components), 3×4 features for the four (up, down, left, right) superpixels located at 25 pixels of the central one, 3×4 for superpixels at 50 pixels, and finally 3×4 for superpixels at 100 pixels. This process is applied on each pixel of the image to produce in total 39 feature maps. To obtain features at the border of the frame, a padding is applied to the image, where the border pixel is simply replicated. Figure 4.4 shows an example of the feature extraction process for one pixel.

Using the superpixel mean value instead of the pixel value itself has several advantages: first, it makes the feature vector more robust to noise and sharp variations in the image. Second, it solves the problem of local averaging at the border of the objects: as pixels are grouped based on their consistency and oversegment objects, the color average is done within each segment. Thus, pixels at the border of an object will have the same representation as pixels at the center, as long as they belong to the same superpixel. Lastly, as features are representative of superpixels rather than a pixel, the descriptor is more robust to scale variations.

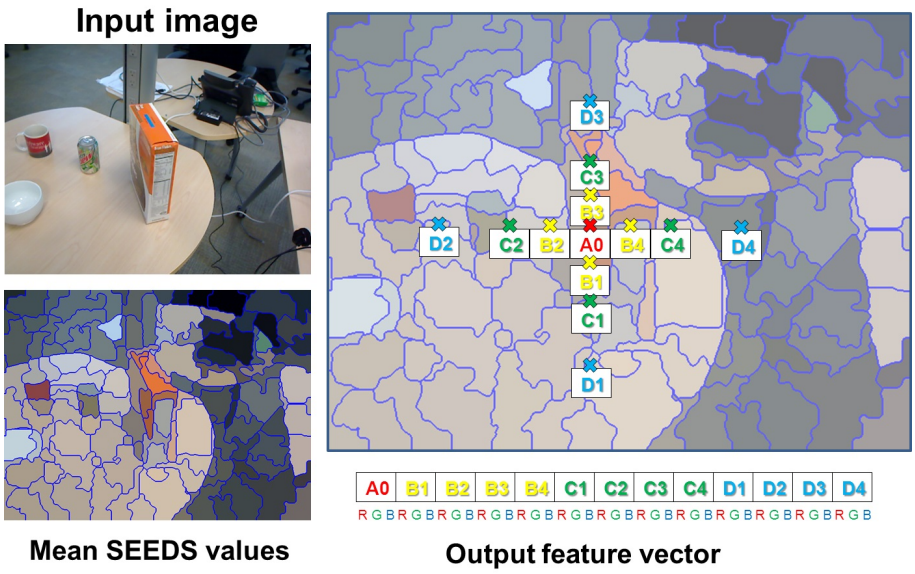


FIGURE 4.4: RGB-based feature extraction based on Make3D

We then base our saliency estimation on this set of features, pretty reduced compared to the one proposed in Make3D. However, we did not find much improvement in using the whole batch of features proposed by the method (using standard deviation types of features even decreased the performance), and the computation time was much higher.

Pros and cons

These features are conceptually very simple, easy to implement, and their extraction are pretty fast. Unlike Itti and Koch features, superpixels make an excellent distinction between objects and background, even at their periphery. As a drawback, they rely on RGB colors, thus making them very sensitive to illumination changes. In addition, the number of superpixels required by the algorithm depends on the average size of the objects of the scene, and has an impact on the feature quality.

4.3.3 Deep features

Motivations

Deep learning based on convolutional neural networks are now applied everywhere to solve computer vision problems. As discussed in Sections 3.3.1 and 3.3.2 of the previous chapter, saliency and object localization now have a bunch of techniques based on deep learning. A very nice property of such networks is their modularity, and their capacity to be re-used in many other types of computer vision problems by fine-tuning.

An end-to-end neural network architecture may be used for saliency learning, starting with convolutional layers and ending up with fully connected one. However, meta-parameters (learning rate, minibatch size, *etc*...) are hard to configure to allow an efficient incremental learning. A bad configuration could significantly deteriorate weights that were correctly learned for another problem.

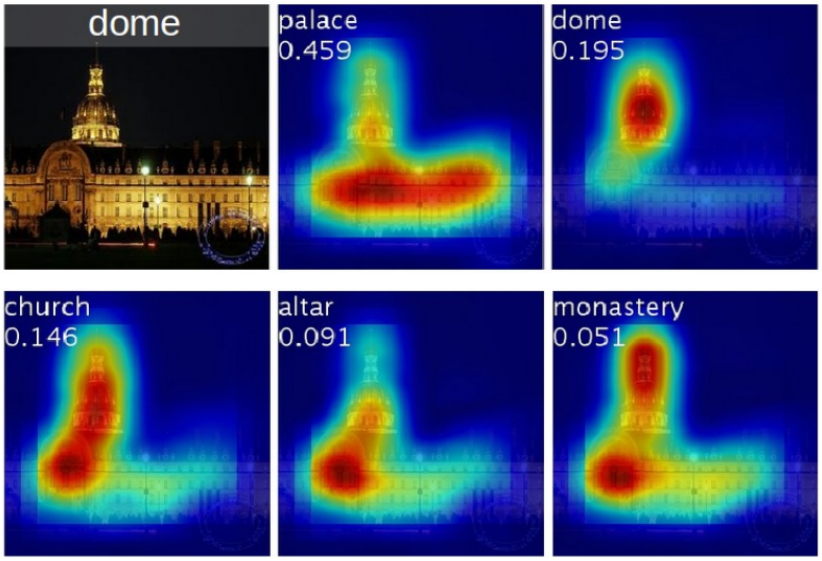


FIGURE 4.5: CAM mapping of the top 5 responding object. Image from [68]

We then consider another approach to exploit deep neural networks: the use of the first layers of a well-trained network as a feature extractor. This way, we avoid instability problems, and the module is easily plugged in our architecture.

Description

In convolutional neural networks, the input image is send across the network and is filtered by a large number of convolutional kernels, whose parameters are determined by a feed-forward back-propagation procedure, similar to multi-layer perceptrons. The convolutional layers of a CNN can then be seen as a set of filters whose values have been optimized by the dataset it was trained for. Similar to Gabor filtering in Section 4.3.1, we here produce feature maps by a succession of convolutions that aim to discriminate salient features.

Several popular CNN-based architectures have been proposed to solve computer vision problems, and are publicly available with weights trained on very large datasets (ImageNet [24], Coco [176], *etc*...). Among them, Alexnet [4], VGG [197], GoogLeNet [198] or more recently SegNet [166] are the most widely used networks. Most computer vision approaches use these networks with the proposed weights and remove the last layers of the architecture to fit their own needs. They next retrain the network with their own dataset (potentially much more modest than the initial ones), taking the weights trained for the other problem as a good initialization. This process is called *fine-tuning* (described in Section 3.4.2).

We base our feature extraction upon this idea of fine tuning by following the ideas of Zhou *et al.* [68]. In their article, a GoogLeNet architecture is used and fine tuned to perform object localization. The end of the network is replaced by a global average pooling layer, followed by fully connected layers providing strong localization capacities and working on weakly supervised training data. This type or architecture is then able to produce, in some sense, class specific saliency maps. In addition, the weights of this network are publicly available.

We then use this available trained model and do not consider the layers after the global average pooling one. The feature extraction is done at the level of the

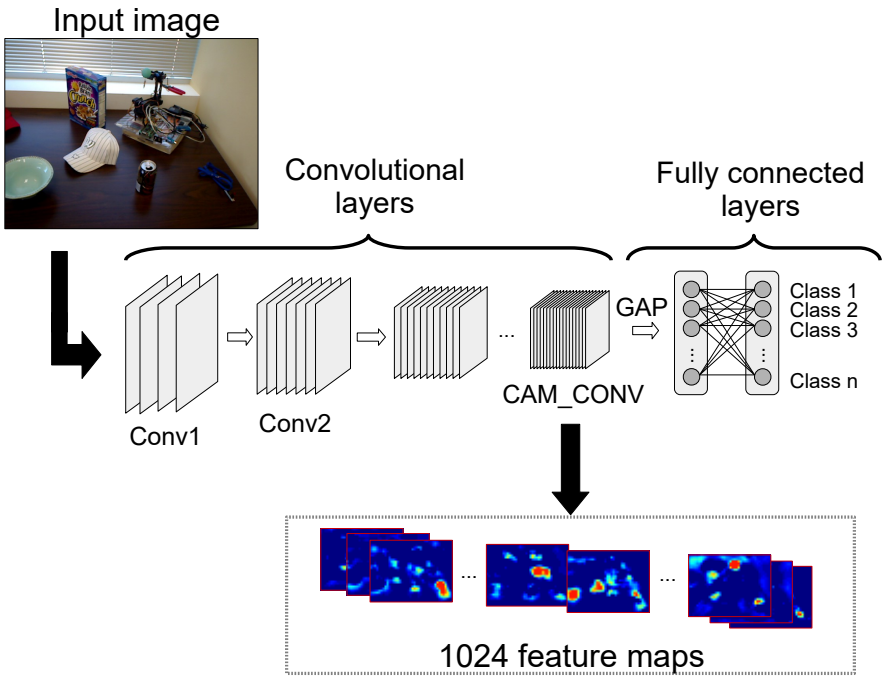


FIGURE 4.6: Example of deep feature maps extraction from an input RGB image

class activation mapping, or CAM layer (called "CAM-CONV" in the network). This corresponds to the last fully convolutional layer of the network. According to Zhou *et al.*, this layer is the one at which highly discriminative areas are enhanced (see Figure 4.5 for an example).

To get a set of feature maps, we then feed the network with the original RGB input image, without resizing it, and extract the 1024 maps of the CAM layer (See Figure 4.6).

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 4.5.2 a method to reconstruct saliency maps at the original scale.

Pros and cons

Deep features are very generic and robust features. They may have a better generalization capacity than the other proposed features. However, to be applied in real time on our images, a GPU is mandatory. Moreover, the successive convolutional layers produce a strong image downsampling, thus making small details hard to retrieve when computing the saliency map.

4.4 Salient objects discovery

To learn a model of saliency, we base our methods upon object detectors and classifiers that will constitute our learning signals. These learning signal have the properties to be reliable, but are only partial. The goal is then to generalize this signal to produce saliency maps in a more general case. Two different object detectors have been used in our two application scenarios.

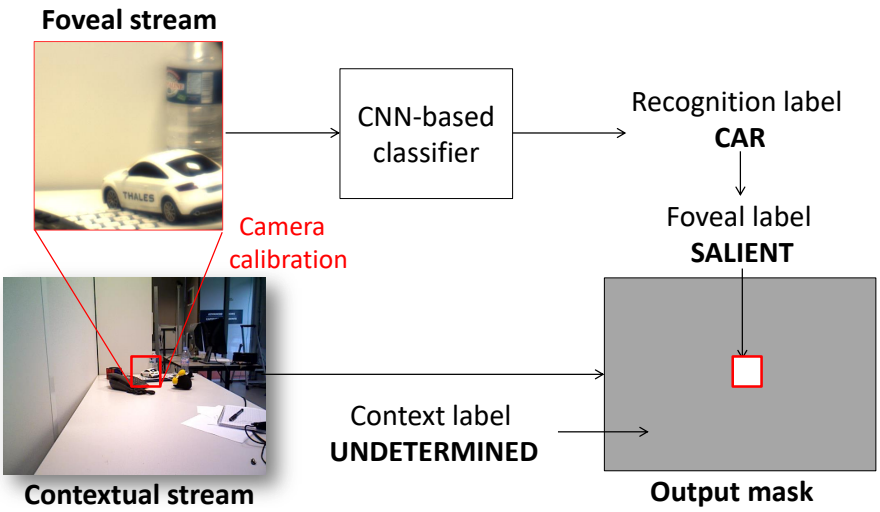


FIGURE 4.7: Segmentation mask from foveal object recognition

4.4.1 Foveated objects discovery

This approach has been designed for the *BioVision* platform and works on the principle of the foveated vision. The idea is to provide a saliency estimation in the fovea, and to transfer this information to the contextual view, as a segmentation mask integrating the foveal location. The method can then be applied on setups having both a foveal and contextual camera.

Figure 4.7 illustrates the general approach to obtain a segmentation mask in this context.

Object recognition in the fovea

The stream provided by the foveal camera is sent to an object classifier that has been described in Section 2.2.2. As a reminder, this classifier is based on a deep convolutional neural network and is able to identify 8 different categories of objects, plus a last class called "*other*", containing any other types of elements.

We take this classifier already trained, and simply use it to infer in real time the class of the object observed by the fovea. The classifier is considered robust enough to provide a reliable class estimation, that is to be used as our learning signal. In our case, we consider the eight objects as being salient (annotated with the "*salient*" label), and any other visual item as being "*not salient*".

Segmentation mask reconstruction

In this setup, the contextual view the main input stream, on which the saliency map is to be estimated. To make it compatible with the saliency learning algorithm, we then need to turn the object class estimation on the foveal image into a segmentation mask in the contextual view.

First, the label provided by the object recognizer of the fovea is tuned into a label compatible with our saliency learning framework. If the output label is among the nine objects, we transform it to a "*salient*" label. If the estimated class is "*other*" the new label is "*not salient*".

Second, the extrinsic calibration of the setup provides a correspondence between pixels of the fovea and pixels of the contextual view. Based on this information, we associate each pixel of the contextual view that are visible in the fovea

with the label ("*salient*" or "*not salient*") found earlier. Everywhere else in the contextual image, we assign the label "*undetermined*").

Note that in this configuration, learning is weakly supervised at two different levels. First, a saliency label is available only within the fovea, so that the learner should use several points of view to be able to generalize the estimation out of the fovea. Second, labels "*salient*" or "*not salient*" are assigned on the whole fovea, even if the salient element only represent a fraction of the foveal image. To learn saliency, the classifier must learn by itself to discriminate between what is really salient within the fovea and what is not.

4.4.2 3D segmentation for object detection

This type of object detector has been designed for RGB-D cameras such as Kinect or Asus Xtion, and mainly works for indoor environments. The method is a modified version of the depth-based object segmenter of Caron *et al.* [18] and is used to produce a partial but accurate estimation of the saliency. The depth map is the only component considered in the segmentation process.

General idea

The object segmenter detects objects lying on planar surfaces (typically tables or floor), with a diagonal size between 10 and 180 centimeters. As a first step, the depth map is turned into a point cloud. Depending on processing step, we either consider the 3D points of the cloud, or their corresponding pixels in the depth map. We then follow the procedure below, further detailed in the next paragraphs

- The major plane (in our case the floor or a tabletop) is detected and tracked during the whole sequence to make sure that the same surface is used during the whole experiment. For simplicity, we call this major plane the floor in these explanations.
- Given the floor position, potential walls are detected. Floor and wall are then filtered out, and the remaining points of the cloud are likely to be part of salient objects, but could as well be undetected 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.
- To avoid false detections from large objects cut by the border of the frame, only candidates having no contact with the border of the depth field of view are kept and considered as salient objects.
- Lastly, the segmentation procedure is converted into a 2D segmentation mask, further used to produce "*salient*" and "*not salient*" labels.

The main steps of the segmentation are summarized in Figure 4.8. We also synthesize in Table 4.1 the parameters of the method as well as their numerical values.

Plane tracking

The segmentation algorithm is using a plane detection as a pre-processing step. To detect the major plane of an image, the RANSAC algorithm [158] is generally

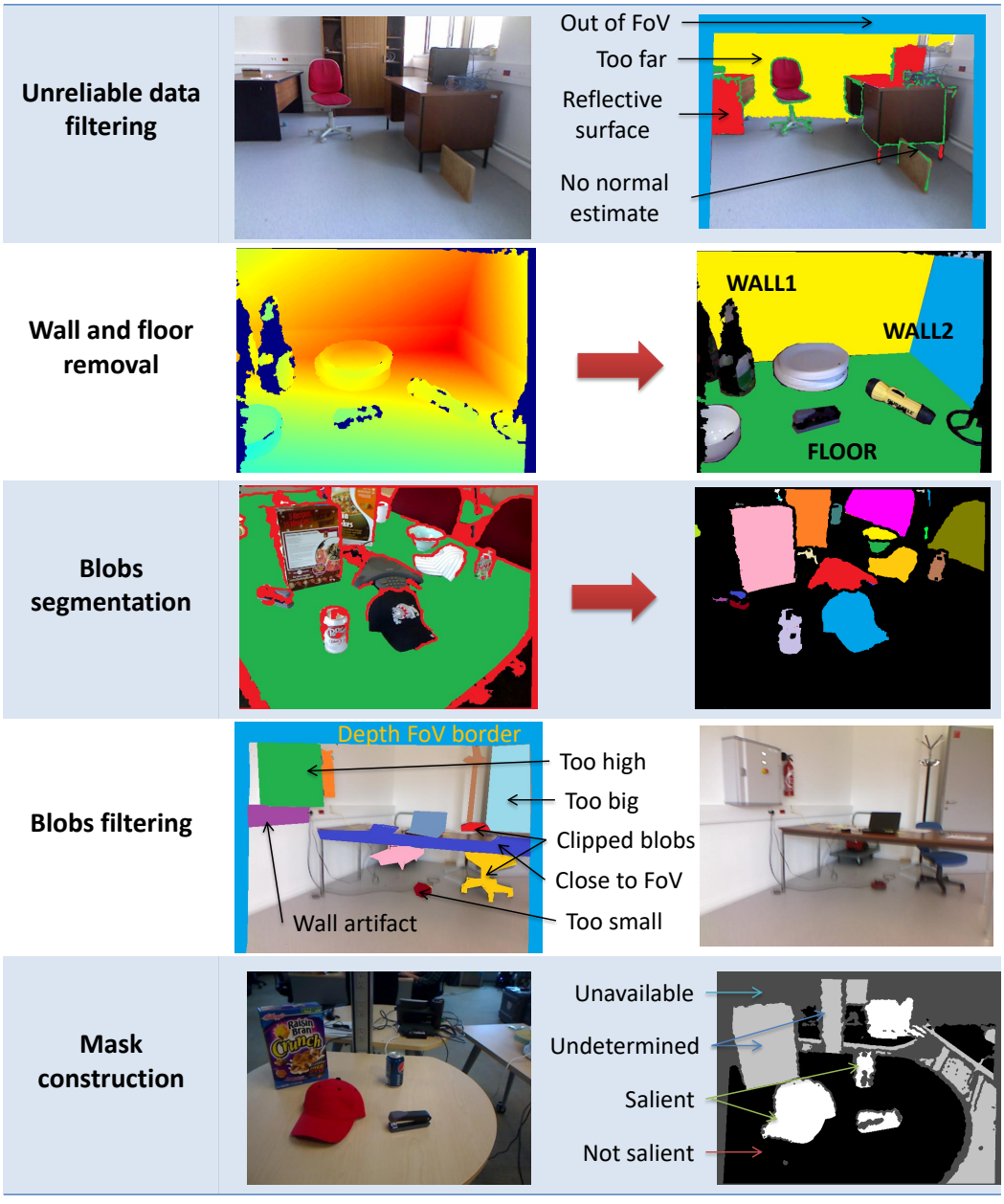


FIGURE 4.8: Main steps of the 3D segmentation

Parameter	Numerical value	processing step
Max dist. to kinect	4 m	Unreliable data filtering
ϵ	5 cm	Wall and floor removal
τ	0.95	Wall and floor removal
δ	0.05	Wall and floor removal
Max object diag. size	180 cm	Blobs filtering
Min object diag. size	10 cm	Blobs filtering
Blobs too high thresh.	150 cm	Blobs filtering
Plane artifact thresh.	5 cm	Blobs filtering

TABLE 4.1: Main parameters of the depth segmentation algorithm

applied on the whole point cloud of the frame. The detected major plane is then considered as the floor or tabletop. However, this is far from always being the case. Walls are very often detected this way and may conduct the algorithm to fail. In addition, computing a RANSAC from scratch on the whole image can be pretty slow, especially if the environment is complex. In Caron *et al.* [18], the RANSAC computation is avoided by determining the floor equation prior to the experiment, and supposing that orientation is the same during the whole experiment.

To overcome these limitation, we develop a method of plane tracking that returns after each frame an equation of this plane. Tracking is initialized by detecting the main plane in the whole image, and re-initialized the same way each time the tracking is lost. After initialization, the pixel position of the inliers (the points belonging to the plane) are turned into a binary mask (main floor/not main floor). Then, given this binary mask, the next frame is processed by applying a RANSAC to points of the cloud whose pixels of the mask are positive. Given the asumption that two successive frames are similar enough, this masking procedure provides an excellent initialization for the RANSAC. This way, the algorithm converges much faster and is almost sure to find the right plane, even when the major plane of the image has changed. An even better mask is obtained by filtering out pixels whose color value between the two successive frames are too far from each other. After each plane estimation, the points belonging to this plane our of the mask are retrieved, and a new mask is determined for the next frame. The main steps of this algorithm are illustrated in Figure 4.9.

Unreliable data filtering

As a first processing step, we remove points of the cloud that are likely to be hard to process, or for which data is not fully accessible.

Local normals are first estimated for each point of the cloud. However, pixels with too strong variations in their neighborhood (typically at the border of an object) cannot be assigned a normal and are therefore removed from the point cloud. They are represented in green in Figure 4.8. Second, points that were not assigned a numerical value, due to reflective surfaces, shadows (red color in the figure), or projection of the depth map on the rgb frame (denoted “Out of FoV”, in blue in the

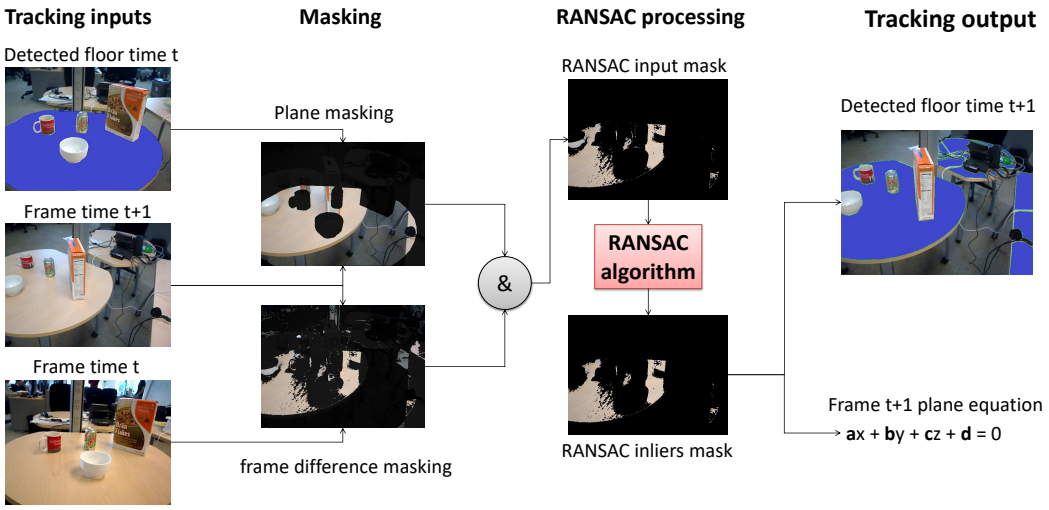


FIGURE 4.9: Main components of the plane tracker

figure), are also filtered out. Lastly, points whose estimated distance to the Kinect is above a certain threshold are removed due to their potentially noisy measurement.

Floor and wall removal

The plane equation provided by the plane tracker is a set of four values a, b, c and d so that each point (x, y, z) of the planes verifies the equation

$$\begin{cases} ax + by + cz + d = 0 \\ a^2 + b^2 + c^2 = 1 \end{cases} \tag{4.6}$$

Points belonging to the floor are first removed. We identify the points of the cloud that verify the conditions

$$\begin{cases} |ax + by + cz + d| < \epsilon \\ |n_x a + n_y b + n_z c| > \tau \end{cases} \tag{4.7}$$

with $\epsilon \approx 0, \epsilon > 0$ and $\tau \approx 1, \tau < 1$ two threshold values, and (n_x, n_y, n_z) the unit normal vector associated with (x, y, z). Those constraints forces both the point (x, y, z) to be close enough to the theoretical plane, and to have a normal (n_x, n_y, n_z) close enough to the plane normal (dot product close to 1).

Then, walls are identified and filtered out. We define walls as being large planes orthogonal to the floor plane, and having a contact with the border of the image. To detect them we run the RANSAC algorithm on the remaining point cloud. Whenever a plane is detected this way, the orthogonality condition is verified by the dot product between the plane normal and the floor normal:

$$aa' + bb' + cc' < \delta \tag{4.8}$$

with $\delta \approx 0, \delta > 0$ a threshold value. If the plane satisfies the equation, and if a portion of this plane is close enough to the border of the depth field of view, it is considered as a wall and removed from the point cloud. Otherwise, those points are removed from the next RANSAC computation, and another plane is detected

on the remaining point cloud. When the planes found have a too small number of pixels, the procedure is stopped.

Blobs segmentation

The remaining point cloud (recall that it contains points close enough to the kinect, being neither part of the floor, nor of the wall, and whose normal could be estimated) is turned into a set of blobs based on image processing. For this step, we consider the pixels corresponding to each 3D point of the cloud rather than the 3D points themselves. This way, we can apply simple image processing segmentation.

Recall that points at the border of an object are not considered because of their sharp depth variation in their neighborhood. This way, filtering them out of the point cloud makes a natural delimitation between objects (colored in red in Figure 4.8, section blobs segmentation). The blob segmentation step them simply consists in finding the connected components of the remaining pixels of the image. The set of 3D points associated with a connected component is now called a *blob* (or a cluster), and is used in the next processing steps.

These blobs are also used to generate the *SegBoxes* (see Section 4.5.3), considering their bounding boxes in the depth frame rather than their pixels.

Blobs filtering

A set of filters are then applied to the remaining blobs:

- Blobs whose centroids are too close to the floor or to a wall are filtered out as they are likely to be false RANSAC outliers.
- Blobs having a too large diagonal, or having a bottom too far from the floor are removed.
- Blobs having a contact with the border of the depth field of view are filtered out.

An optional final step is applied to clip together blobs belonging to the same object that would have been divided by the image-based segmentation (this is, for example, often the case for desk chairs like in Figure 4.8, section blobs filtering). For this step, we project all the remaining blobs to the floor, and we use a KD-Tree-based clustering to group pixels that are close enough or overlapping each other.

Object segmentation mask reconstruction

To convert this segmentation procedure into a 2-D 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. In these examples, pixels having no corresponding point cloud (because of reflectance, shadows or because visible points are too far from the Kinect) are labeled "*unavailable*" (dark gray in Figure 4.8, section mask construction). Points of the cloud that either belong to the major plane or to a wall are labeled "*not salient*" (colored in black in the figure). Points of the cloud belonging to a cluster detected as a salient object is labeled "*salient*" (colored in white). Lastly, all remaining points are categorized "*undetermined*" (light gray), as the algorithm was not able to determine their actual state of saliency. This includes candidates that are too close to a border and could be either walls or objects, or candidates bigger or smaller than the tolerance threshold, that could either be several

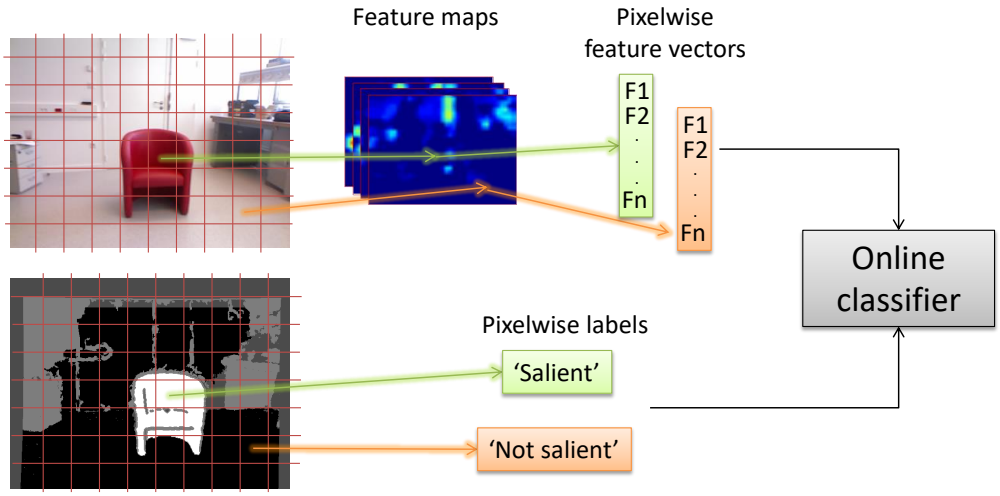


FIGURE 4.10: Pixel-wise data formatting

salient objects considered as one, or fragment of salient objects, isolated from the rest.

4.5 Saliency learning and saliency exploitation

In this section, we describe how feature extraction and saliency discovery are integrated into a learning framework to produce a saliency model. Moreover, we explain how saliency maps are retrieved based on this model, and how they can be exploited to produce bounding boxes around salient objects.

4.5.1 Incremental learning

Given, on the one hand, features continuously extracted from the main visual stream and, on the other hand, segmentation masks providing a partial estimation of the saliency, we integrate these cues into an online classifier to build a saliency model.

Data formatting

Learning is done at the pixel level, by turning the incoming feature maps and segmentation masks into a set of labels and features. Figure 4.10 illustrates this procedure.

First of all, the segmentation mask is resized at the same size as the feature maps. This way, each pixel of the feature maps and of the mask correspond to the same area of the visual field. Then, for a given pixel, the n features (n being the number of feature maps) are collected and turned into a feature vector, and the corresponding mask pixel value (which stands for a label) is retrieved. This way, a pair of features-label is associated with this pixel. We call it a *sample* in the following explanations.

The data send to the classifier is then a matrix of features and labels representing the samples collected at different pixel locations. However, creating a sample at each pixel would produce a very large amount of very redundant data to send to the classifier. For this reason, we downsample the number of extracted samples, by considering only the central pixels of the cells of a regular grid. In addition, pixels classified "*unavailable*" or "*undetermined*" cannot constitute relevant samples

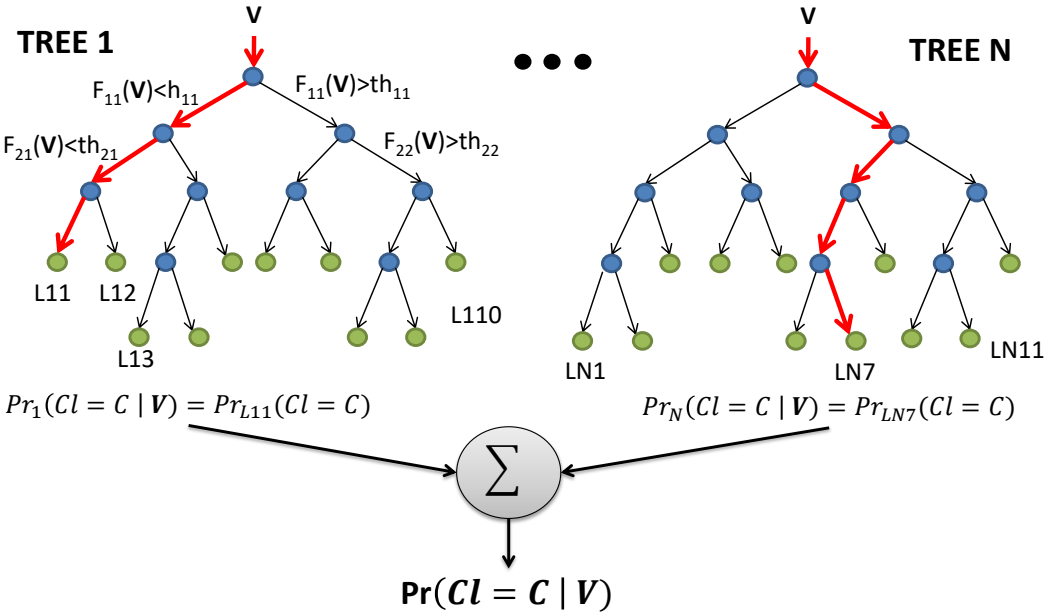


FIGURE 4.11: Class probability extraction from random forests

for saliency learning. Therefore, only samples labeled "*salient*" or "*not salient*" in the segmentation mask are selected to feed the saliency model.

Classifier

After each new observation, the classifier is updated based on the new incoming samples. We therefore need this update to be fast enough and stable in time. In spite of a few attempts with neural networks, we decided to use a random forest classifier that provided much better results while being decently fast to re-train.

Random forests have been introduced by Breiman *et al.* [199] and consist in a bagging approach based on an ensemble of decision trees. Unlike decision trees, random forests have the property of not overfitting their training set. In the case of classification with numerical attributes (called *features* in this text), a decision tree is a binary tree in which samples of the dataset are propagated to fall in a leaf providing an estimated label. Figure 4.11 is an illustration support for the following explanations.

Each node d of the tree is associated with a feature F_d and a numerical threshold th_d. To determine the label of a given sample $\mathbf{V}$, the numerical value $F_{11}(\mathbf{V})$ of the feature of the root node is extracted, and compared with the root threshold th_{11}. Depending whether $F_{11}(\mathbf{V}) > th_{11}$ or not, $\mathbf{V}$ ends up in the right of left node of the next level. In this node, the next feature $F_{2d}(\mathbf{V})$ is considered and compared with threshold th_{2d}, and so on until $\mathbf{V}$ reaches a leaf node. This leaf contains statistics about the labels of the training samples that have reached this node. Thus, the probability of a sample falling in a leaf L to be of class C is given by

$$Pr_L(Cl = C) = \frac{1}{|T_L|} \sum_{s \in T_L} Cl(s) = C \tag{4.9}$$

where T_L is the set of training samples that have reached L, and $CL(s)$ is the label associated with training sample s. A classical way to train a decision tree, is to grow from an iterative process where nodes are splitted based on their purity Ginny

Parameter	Numerical value	Comment
number of trees N	50	proportional to inference time
number of trees updated K	4	proportional to training time
number of features by tree	30	proportional to training time
max depth of the tree	10	proportional to inference time
max number of samples in memory	100 000	proportional to training time
max number of samples at each observation	500	
Fraction of the dataset used in each tree	70%	commonly used in random forests

TABLE 4.2: Main parameters for saliency learning

index [200]. The complexity of such algorithm is $\mathcal{O}(mn)$, n being the number of samples in the training set and m the number of features.

A random forest is constituted with a set of decision trees, each of them being trained with a fraction of the training set. This way, each tree is unique and has its own statistics about class probabilities. To retrieve the probability of a sample $\mathbf{V}$ to belong to a class C, the following formula is used

$$Pr(Cl = C|\mathbf{V}) = Pr(Cl = C) \times \frac{1}{N} \sum_{i=1}^{N} Pr_{L^i(\mathbf{V})}(Cl = C) \tag{4.10}$$

N being the number of trees in the forest, $Pr(Cl = C)$ being a prior on the class distribution, and $L^i(\mathbf{V})$ being the leaf node of tree i in which $\mathbf{V}$ is falling.

The major drawback of random forests in our system is that they are not designed for online training. No available version [185], [187] of online random forest was satisfying in terms of speed and performance, so that we adapted the offline version to make re-training fast enough.

To re-train random forests incrementally, we therefore use the following procedure: first, the whole dataset of extracted samples is kept in memory. To avoid memory issues or too slow updates, we forget the oldest samples when the training set exceeds a certain limit. Then, we update the classifier by only re-training a small fraction of the forest at a time: we randomly select K trees among the N in the forest, and we retrain those trees with a random fraction of the dataset available in memory. As a result, after each update, the classifier is able to estimate the saliency of an input based on the model trained with the previous observations.

Parameters

Our learning system depends on a certain number of parameters that must be carefully chosen. Most of them were empirically determined to allow a good trade-off between computation speed and classifier accuracy. We summarize in Table 4.2 the main parameters and their numerical values used in our implementation.

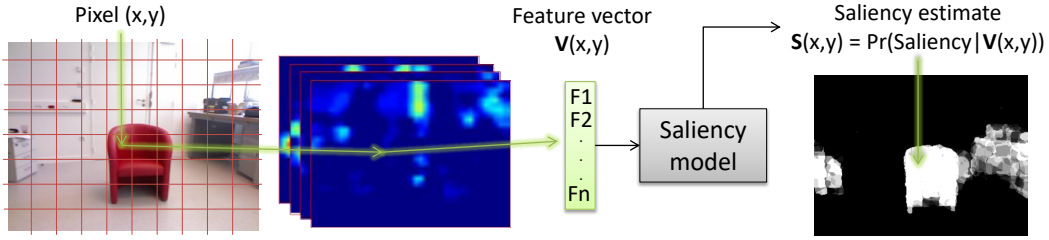


FIGURE 4.12: Saliency map reconstruction

Aside from these parameters, a statistical prior was added on the classes. In practice, the fraction of "*salient*" samples is around 5%, thus making the dataset highly unbalanced. We believe this is one the reason why neural networks failed at estimating saliency. To handle unbalanced dataset, the a prior probability of 0.95 was attributed to the "*salient*" class.

In Table 4.2, we define a parameter that restricts the max number of samples M at each observation. This parameter defines the cell length of the grids used to downsample the pixels from which we extract features and labels ($\frac{W \times H}{C^2} = M$, C being the cell length and W, H the image width and height). In practice however, this max number of samples is never reached, as a large portion of the image is labeled "*unavailable*" or "*undetermined*". The number of samples actually sent to the classifier after an observation is more realistically around 200.

4.5.2 Saliency inference and map reconstruction

Inference and saliency map reconstruction

The saliency maps are constructed by applying the saliency model to RGB images of the main visual stream. More precisely, the saliency map is obtained as follows: we first generate feature maps from the input RGB image, and associate each pixel of these maps with a feature vector, similar to the procedure of Section 4.5.1. For a pixel $P(x, y)$ with associated feature vector $V(x, y)$, we evaluate the probability of the pixel to be salient based on the classifier output. The random forest produces a probability for each class. As our problem is a binary one, we then obtain the probability of $P(x, y)$ to be salient by

$$S(x, y) = Pr(Salient|\mathbf{V(x,y)}) = \frac{1}{N}\sum_{i=1}^{N} Pr_i(Salient|\mathbf{V(x,y)}) \tag{4.11}$$

we then generate the saliency map by gathering the $S(x, y)$ results in an image of the same size as the feature maps. Unlike the partial estimation of the segmentation mask, we here obtain an estimation of the saliency for each pixel of the image. Figure 4.12 illustrates the construction of the saliency map.

Superpixel-based saliency recovery

The produced saliency map has the same resolution as the feature maps. For two of the three presented features, this has no influence on the output, but for feature maps produced with CNNs (see details in Section 4.3.3, recall that the deep feature extraction downsamples the image by a factor 16), the output saliency has a very low resolution.

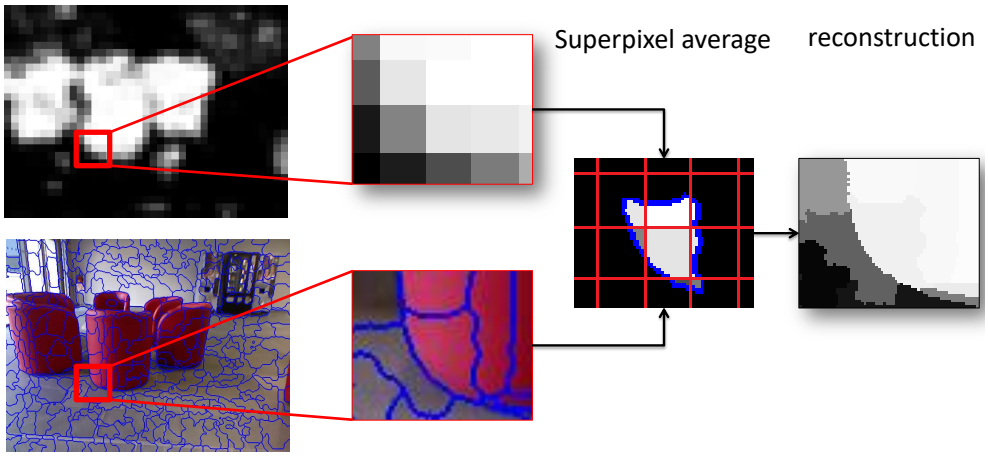


FIGURE 4.13: Illustration of the superpixel reconstruction technique

To rescale this low-resolution saliency map to the original input size, we generate SEEDS superpixels [140] from the original image (350 superpixels for 640×480 images in our experiments). Our goal is now to associate a saliency value to each of these superpixels. For that, we resize the saliency map at the original scale using a nearest neighbor interpolation approach, and we calculate the average saliency of this map within each superpixel. Mathematically, for a superpixel P such that $A_P = \{(x, y) \in P\}$ is the set of pixels contained in P, the superpixel saliency map S_{sp} is such that

$$S_{sp}(A_P) = \frac{1}{|A_P|} \sum_{x,y \in P} S(x, y) \tag{4.12}$$

Using this approach, the produced saliency map looks nicer, as saliency at the border of the objects is preserved. Figure 4.13 illustrate the procedure described in this Section.

4.5.3 Bounding box proposals

The saliency map provides an indication of the interestingness of a given pixel. In order to localize objects in an image, an additional step is necessary to group salient pixels into object candidates. To this end, we use two types of bounding box proposals, and we associate them a score that is strongly related to the obtained saliency map. This score is then used as a mean to determine boxes that are the most likely to contain a objects, or to reject boxes having a very low score.

EdgeBoxes

The first bounding boxes are obtained by the *EdgeBoxes* [169] algorithm. The edgeBoxes are among the existing techniques of agnostic bounding boxes, described in Section 3.3.2. They rely on a simple criterion of contours to produce these boxes, and are very fast to compute compared with most existing techniques. In addition, they are among the most efficient techniques of this field, and produce a confidence score.

To exploit this algorithm in our implementation, we compute for a given RGB input the 100 most likely EdgeBoxes, and store their associated edge confidence score, called h_b^{in} in the reference paper [169]. This score is already a good measure

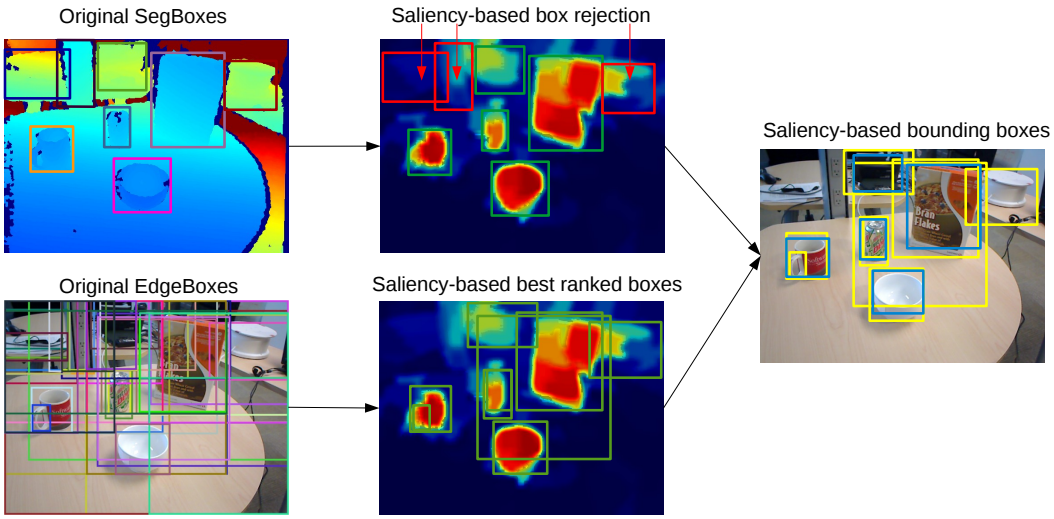


FIGURE 4.14: Bounding box proposals biased by our saliency map

to determine if a generated bounding box is likely to contain an object or not. Figure 4.14 illustrates the general idea of this technique.

SegBoxes

The second type of bounding boxes can only be obtained from RGB-D data, as they are obtained with the 3D object segmentation described in Section 4.4.2. To make explanations more clear, we call the bounding boxes obtained this way the SegBoxes.

During segmentation, pixels of the depth map are grouped in blobs in order to create object candidates. Among object candidates, some are labeled "*salient*", some should be salient objects but are labeled "*undetermined*" because they touch a border or are poorly segmented, and some are just artifacts. We define the Segboxes as the 2D bounding boxes around all blobs (labeled "*salient*" or "*undetermined*") of the segmentation mask. Additionally, blobs that were clipped together when projected to the floor plane are also added to the SegBoxes list.

Saliency score and re-ranking procedure

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:

$$SCscore(B) = \frac{1}{w_B \times h_B} \sum_{i,j \in B} S(i,j) \tag{4.13}$$

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. Lastly we filter out Segboxes and Edgeboxes whose final score is below a certain threshold. 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 alarms and false rejections.

4.6 Conclusion

In this chapter, we have presented the technical details of a method for learning a model of saliency incrementally. This approach is designed to be run on a robot exploring its environment. The saliency model is then improved as the robot moves and receives new observations.

The method is based on three modules. A feature extraction module processes an RGB stream and extracts relevant patterns to determine the state of saliency at a pixel level. A saliency discovery module provides, when possible, and estimation of the saliency that is reliable and converted to labels. Lastly, an online classifiers processes both labels and features to learn a model of saliency.

Once a model of saliency is available, the robot can now infer from the whole RGB image a state of saliency that is specialized for the explored environment. Lastly, the estimated saliency maps can also be used to produce bounding box proposals very likely to contain salient objects.

To demonstrate the modularity and the generalization of our approach, we have proposed different types of feature extraction, each having advantages and drawbacks. We also developed for two different setups, two strategies for discovering salient elements in the environment.

However, another question is raised to fully exploit the capacities of our technique: which exploration strategy should the robot adopt to increase and speed up the accuracy of the saliency model ? The problem of selecting relevant targets that the robot should explore constitutes the main problems of Part II of this manuscript.

Chapter 5

Experimental results

5.1 Introduction

The previous chapter was presenting the technical details of the proposed method to incrementally learn visual saliency. We propose in this chapter to evaluate the performance, behaviors and limitations of this approach. To this end, we rely on the four datasets described in Chapter 2, and conduct a set of experiment to evaluate different aspects of the method.

First, we study the incremental and self-supervised aspect of the method. We measure the execution time of each component and provide qualitative results on the evolution of the saliency model in time. In addition, we discuss the self-supervision process by quantitatively evaluating its reliability. We also demonstrate the ability of the learning process to generalize over the learning signal to create a reliable saliency model even when the supervision signal was partly missing, weak, or even wrong.

The second aspect of our evaluation is related to the quality of the saliency itself, once the model has been learned. We then evaluate the quality of each feature extraction technique, and the performance of our method versus state-of-the-art. Lastly, we show that our saliency approach can be used to produce bounding box proposals around salient object more efficiently than agnostic bounding box proposals methods would.

5.2 Setups and datasets

Our saliency learning technique is evaluated on the four datasets described in Chapter 2. We here provide a few recalls and explanations on the annotation procedure in lights of the method proposed in the previous chapter. Indeed, defining annotations that are relevant and compatible with our method is not completely straightforward.

5.2.1 Requirements for evaluation

Nature of the input images

Our method for learning saliency was developed for setups having at least two synchronized streams, called the main and addition visual streams in the previous chapter. Therefore, to train our system, we need datasets containing synchronized images from those two streams.

We described in our work two possible cases. The first one is the use of an RGB-D stream, where the RGB component is used as the main visual stream, and the depth as the additional one. The second one relies on a contextual camera playing the role of the main visual stream, and a foveal camera for the additional stream.

Our datasets are designed for those two configurations: The *ENSTA dataset*, *RGB-D scenes dataset* and *Ciptadi dataset* contain RGB-D images, while the *BioVision dataset* contains both foveal and contextual images.

Self-supervised training and objective evaluation

Our method is based on a self-supervised learning technique. This means that training is done without prior available annotation. We can then train our method from a dataset having only input images, and do not need annotations for the whole dataset.

Nevertheless, we also want an objective evaluation of our system, and cannot consider the self-annotations as a valid ground truth. Indeed, the self-annotations are only partial, and are not necessarily perfect (for example, the object classifier may misclassify some objects, the depth segmentation may fail sometimes, *etc*...). Therefore, we also need a portion of the available dataset to be associated with objective annotations to compare the produced saliency maps (and potentially our self-annotations) with a ground truth.

We therefore divided the *BioVision*, *ENSTA*, and *RGB-D scene datasets* into a training and a testing set. The training set constitutes the large majority of the dataset and is not associated with any annotation. Conversely, the testing set is composed with a small number of frames (100 to 150), all having an objective ground truth annotation.

Lastly, we wanted to test the generalization of our saliency models on environments they were not trained for. To this end, we relied on the *Ciptadi dataset*, composed of 80 annotated frames. In this case, training is done independently on another dataset, and the dataset is fully dedicated for evaluation.

Relevant annotations

Our last constraint was to define relevant annotations, compatible with our saliency models. Recall that our saliency technique produces top-down saliency maps focused on object in indoor environments. As mentioned in Chapter 3, these kind of datasets are not extremely frequent, and the teams evaluating top-down models of saliency usually build their own datasets to fit their own needs.

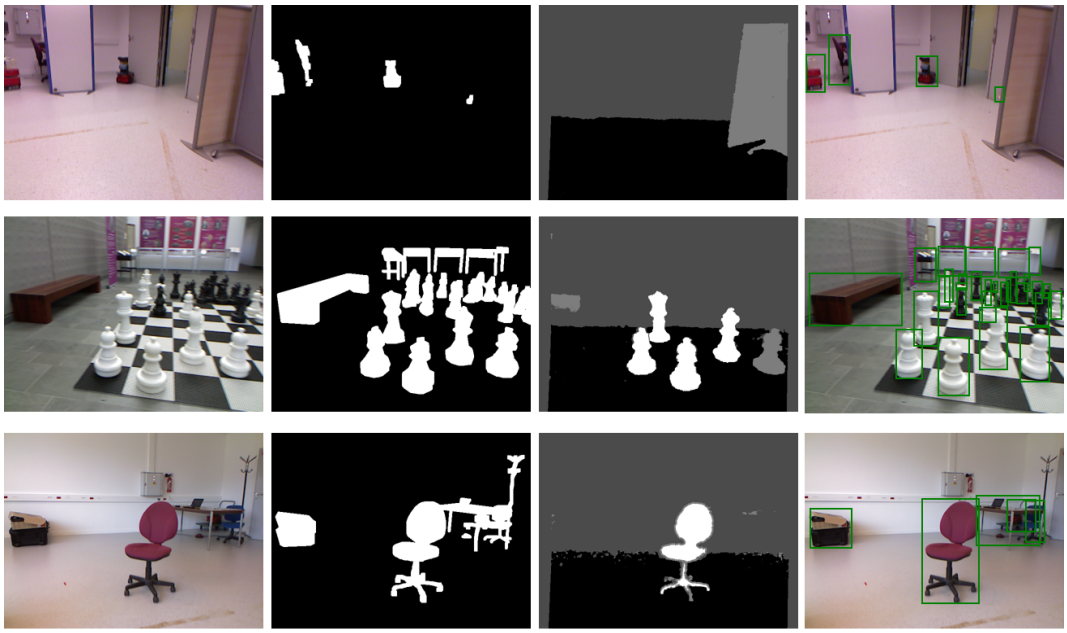
But there is more: the saliency maps produced with our technique are directly based on the supervision signal used to train the model. The annotations used for evaluation must then be compatible with our definition of saliency, represented by our learning signal. We then provide in the next section the annotation procedure used in each of our datasets.

5.2.2 Annotation procedure

BioVision dataset

In the *BioVision dataset*, the learning signal is produced from an object classifier, trained on eight different objects and a last class containing background elements. We define salient element as being exactly the eight objects recognized by the classifier. Thus, any distractor such as pens or notebooks are not supposed to be learned by our model as salient elements.

Therefore, we constructed our ground truth by manually drawing segmentation masks around those eight objects. Please refer to Section 2.3.1 to visualize examples of the produced segmentation masks.



(a) (b) (c) (d)

FIGURE 5.1: Samples of the *ENSTA dataset* and their associated ground truth. (a) RGB image, (b) manually annotated ground truth mask, (c) automatic segmentation mask, (d) manually annotated bounding boxes

ENSTA dataset

In the *ENSTA dataset*, a robot moves across a building and visualizes objects on the floor. The learning signal is obtained from a depth-based segmentation technique (described in Section 4.4.2) that isolates any element of a certain size (between 10 and 180 cm) on the floor. This includes for example desks, chairs or boxes. However, objects hanged on the walls (heaters, sprinkler, posters) are not considered as being salient.

Similar to the *BioVision dataset*, we manually annotated the *ENSTA dataset* by creating ground truth masks around those objects. We also needed ground truth bounding boxes around objects to evaluate our bounding box proposals method. Figure 5.1 provides sample frames of this dataset along with their ground truth masks and bounding boxes. We also display the result of the depth-based segmentation to get an idea of the learning signal on which our saliency technique is learning (recall that gray areas of this segmentation mask are either "*unavailable*" or "*unknown*" pixels, that are not integrated in the saliency process).

RGB-D scenes dataset

In the *RGB-D scenes dataset*, the learning signal is obtained from the same depth-based segmentation algorithm. However, we here look at objects lying on tabletops, so that detected objects are not the tables, but the objects on them (caps, cereal boxes, *etc*...). Therefore, chairs or objects out of the tabletop are not considered as salient elements.

The *RGB-D scene* is a publicly available dataset that already contains annotations. However, these annotations are not perfectly compatible with our definition of saliency. Indeed, only a few categories of objects are annotated, while other are

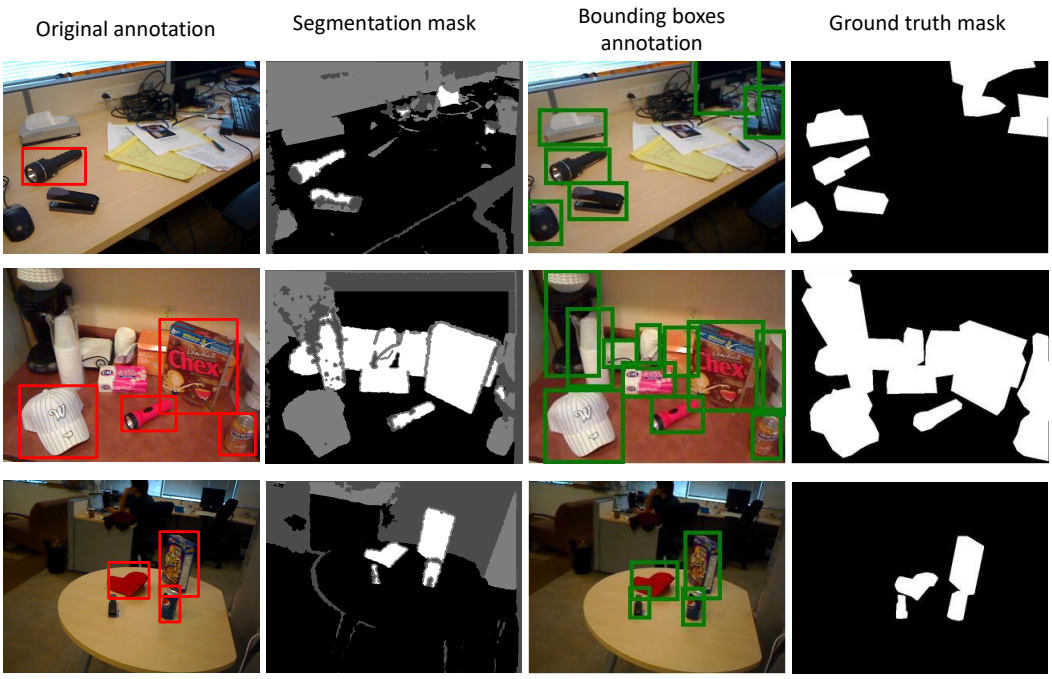


FIGURE 5.2: Ground truth masks and bounding boxes used in our experiments

considered as distractors. As our segmentation technique detects all objects lying on tabletops, a lot of them would have been classified salient by our algorithm and not salient by the dataset. We then had to manually re-annotate them. We followed the exact same procedure as for the *ENSTA dataset* to this end. Figure 5.2 illustrates a few examples of the original annotations, the result of the segmentation mask, and our own annotations.

To use the original annotations, an alternative solution would have consisted in using a specialized object detector for the objects originally annotated. The design would then have been very close to the *BioVision* setup. However, we did not exploited this idea in this work.

Ciptadi dataset

To make sure that our annotation system is not biased too much and that our saliency technique still provides relevant results in a general case, we used a last dataset on which annotations were already provided: the *Ciptadi dataset*. In this case, the ground truth of the dataset does not perfectly match our definition of saliency.

5.3 Incremental and self-supervised learning

To get a good insight of the way saliency is learned and how much the classifier can generalize the state of saliency given a partial learning signal, we provide qualitative and quantitative results.

5.3.1 Self-supervised learning

Evaluation of the segmentation process

A first set of experiments were conducted on the three datasets, *BioVision*, *ENSTA* and *RGB-D scenes*, to evaluate the performance of the self-supervision process, corresponding to the creation of a segmentation mask. To this end, we used the manually annotated ground truth mask of each dataset that we compared with the automatically generated segmentation masks. The segmentation technique used for *BioVision* was the one based on object classification (described in Section 4.4.1), while the depth-based technique of Section 4.4.2 was used for the two other datasets.

Experimental results are reported in Table 5.1. The first measure (**available data**) is the percentage of pixels labeled either "*salient*" or "*not salient*" over the total number of pixels. The *BioVision dataset* has a significantly small (0.5%) portion of the image available for labeling, while the other datasets are around 45%. Second, we measured the percentage of **salient pixels** among pixels for which data was available. This time, *ENSTA* and *RGB-D scenes* have a very small fraction (around 5%) of the pixels labeled salient. This confirms that the training dataset of the online classifier is highly unbalanced.

Lastly, we used the ground truth masks to measure precision and recall on a pixelwise basis. We here make a distinction between relative and absolute measures.

The **relative precision** and **relative recall** are obtained by considering only pixels for which segmentation provided a "*salient*" or "*not salient*" label. This measure is important as it reflects the quality of our learning signal: the precision is very high in both *ENSTA* and *RGB-D scenes datasets*, thus showing that the segmentation process is reliable. The relatively low recall is explained by two factors: on the one hand, segmentation is such that the bottom of the segmented objects are often considered as part of the floor. For small objects such as the one contained in *RGB-D scenes*, this may represent a non negligible portion of the object. On the other hand, big and flat objects (such as computer screens or shelves) are sometimes detected as walls, thus making objects of the *ENSTA dataset* potentially misdetected. Another interesting point is the low precision rate for the *BioVision dataset*, as the whole fovea is labeled with a single value. For example, if a salient object is contained in only 20% of the fovea, and if the provided label is "*salient*", the precision measure will be 20% whereas the label was correctly provided. A more relevant measure for *BioVision* is then to consider the accuracy of the object detector, which is of 94%. Nevertheless, this behavior does not prevent the classifier to have a good generalization capacity, as we discuss in the following sections.

The **overall precision** and **overall recall** are measured by comparing the ground truth with the whole segmentation mask. This measures reflects more the salient detection capacity of the segmentation process. As a large portion of the segmentation mask is not available, the overall recall are much lower (the precision is the same in both cases as the same number of false positives are obtained). Nevertheless, this poor overall performance does not influence the learning quality of our models.

Generalization capacities

We now demonstrate the capacity of the saliency model to generalize what was learned from the segmentation mask, and to 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 find

Measure (in %)	BioVision	RGB-D scenes	ENSTA
available data	0.52	46.2	45.5
salient pixels	17.4	7.1	4.1
relative precision	49.2	96.1	97.2
relative recall	83.3	78.2	75.8
overall precision	49.2	96.1	97.2
overall recall	2.53	62.4	58.8
object recognition accuracy	94.2%	n/a	n/a

TABLE 5.1: Features and performance measures of the segmentation process on the three datasets

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 points of view. This way, if the object detector fails at identifying an object for a given frame, this object may be detected under other points of view. The classifier then extrapolates those points of view and is able to retrieve undetected salient objects. The segmentation samples of this section were obtained with the deep features, by learning a model withe the entire available training dataset.

Generalization from depth segmentation: Figure 5.3, 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 at more 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 5, 1, 2) or badly clustered by the segmentation (sample 1). Conversely, the saliency algorithm provides an estimate of saliency even if the object is partially cut, occluded, or captured with a poor image quality.

Generalization from foveal observation: To demonstrate the generalization capability from foveal observations, we train and evaluate the model on the same sequence. To provide a better visual representation of the way saliency is mapped with objects of the room, we represent saliency map as a heatmap for which red are the most salient areas and blue are the least salient ones. In Figure 5.4, we display a sample frame (cropped around salient elements of the scene) and some foveal observations used to feed the saliency model. All of these observations (and therefore,

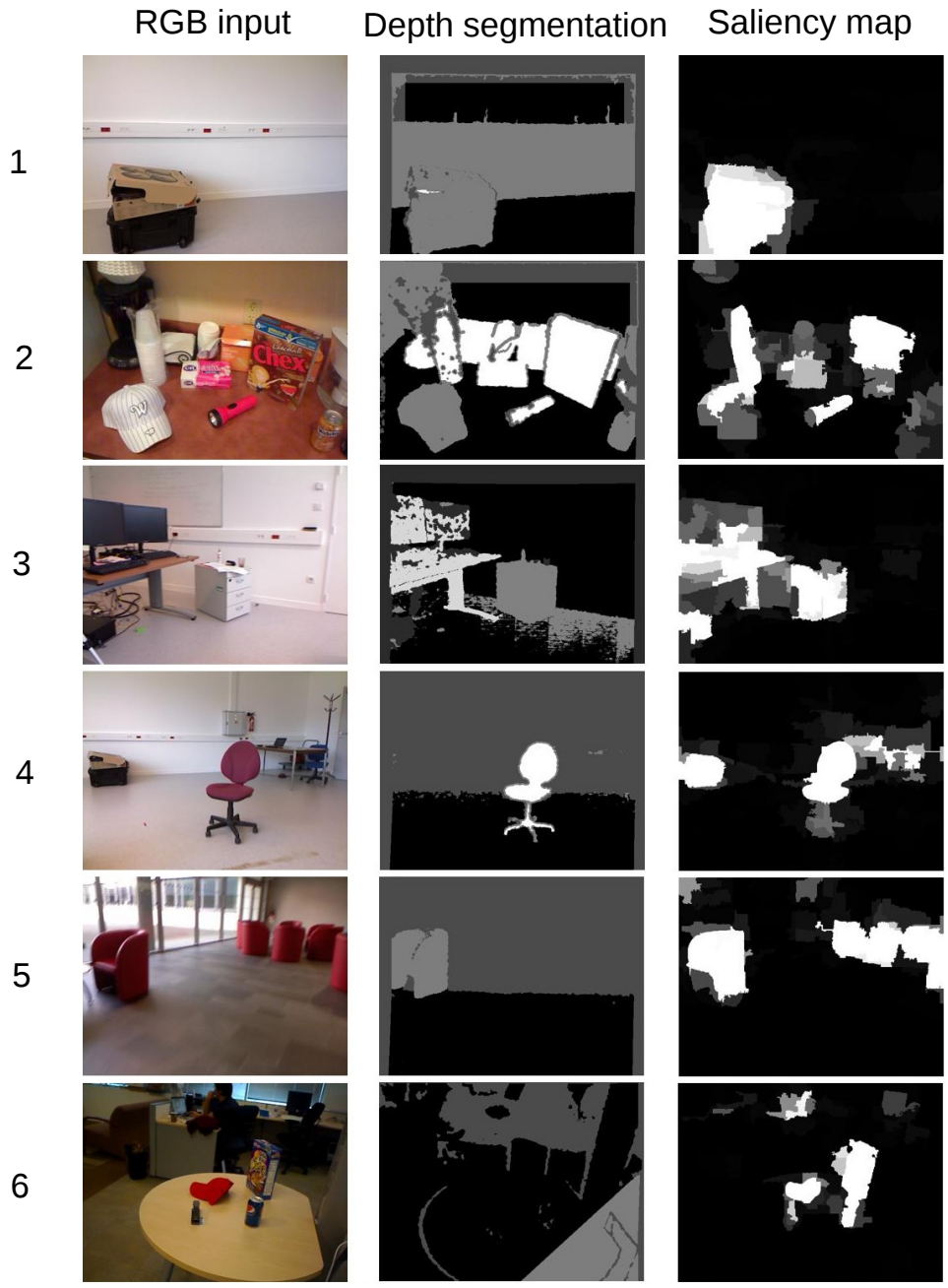


FIGURE 5.3: Generalization capabilities of the classifier. First column: the RGB input. Second column: segmentation mask. Third column: the estimated saliency map.

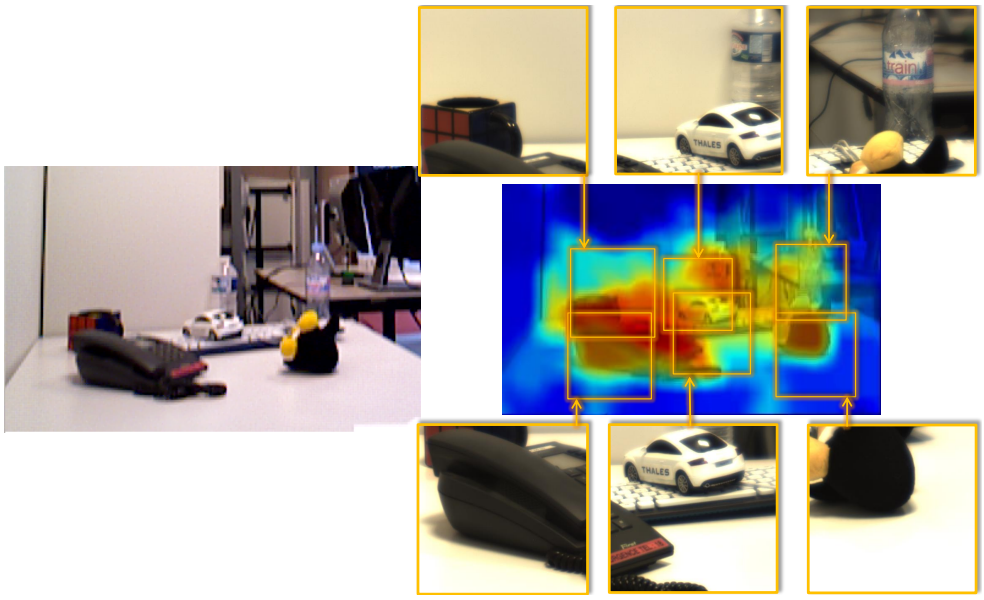


FIGURE 5.4: Generalization capabilities from a weakly supervised signal (the foveal image)

each pixels within the fovea) were labeled "*salient*" by the object recognizer, while it is clear that a large fraction of the pixels are not part of salient objects. Nevertheless, when looking at the resulting saliency map, the classifier was able to discriminate areas that were part of the objects, and areas from the background (walls behind or table). Of course, the discrimination process is not perfect: elements such as plastic bottle were not clearly learned as salient, and portions of the wall behind objects were partially classified salient, but the weak supervision process seems to be working reasonably well.

In figure 5.5, we point out two additional features of the saliency model. First, distractors such as the pen (sample 1) that would be naturally salient are correctly classified not salient. Second, the classifier is able to retrieve similarities in areas that have not been observed by the fovea. In sample 2, a computer keyboard and monitor are detected at the other side of the room.

5.3.2 Temporal aspect of the saliency learning

Our method has been designed to run online during the robot's exploration. It is therefore of paramount importance that the algorithm is executed in a reasonable amount of time, and to make sure that the saliency quality gets better as more examples are provided to the system. We cover these aspects in this section.

Execution time

The runtime performance was measured on each components of the method and is summarized in Table 5.2. To this end, we ran different sequences from the *BioVision dataset* on Ubuntu 14.4 with an Intel Core i3-3240, CPU at 3.4GHz quadcore processor. The code was implemented in C++ and adapted to be used under the ROS framework.

Feature extraction first depends on the type of feature considered, but also on the type of device used. Table 5.3 provides the numerical results associated with our experiments in that regard. We first tested each of the three features on a CPU.

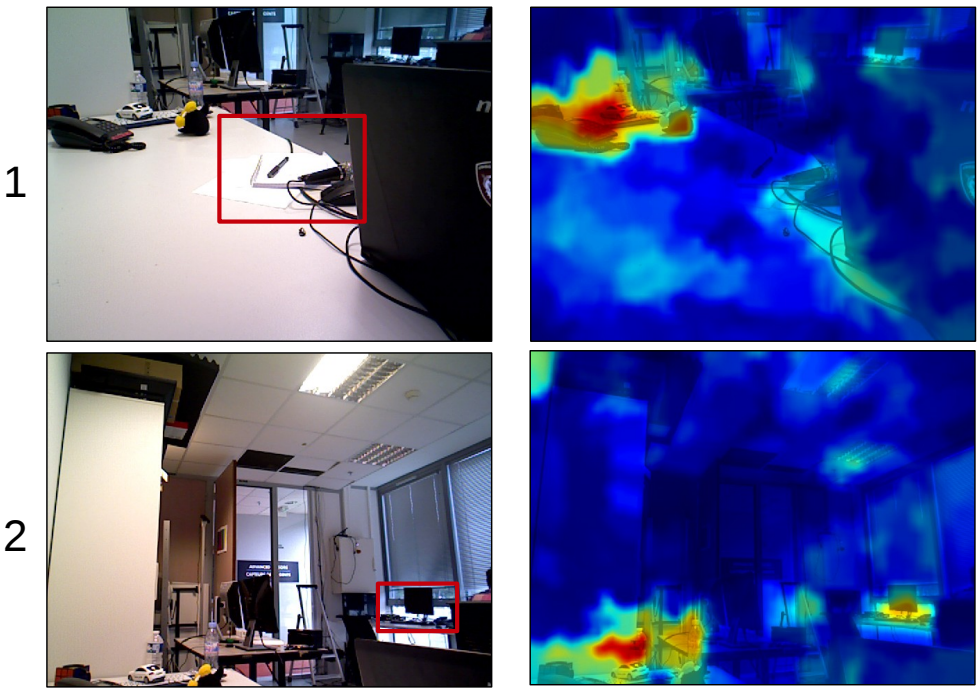


FIGURE 5.5: Sample saliency maps results for the *BioVision dataset*

Step	Min time	Max time	Comment
Feature extraction	26 ms	3 300 ms	See details in Table 5.3
Depth segmentation	70 ms	250 ms	Depends on the geometry of the point cloud
Foveal segmentation	12 ms	240 ms	Depends whether object recognition is calculated on GPU or CPU
Classifier update	23 ms	13 500 ms	See details in Figure 5.6
Saliency estimation	14 ms	82 ms	Depends on the features and if superpixel refinement is used

TABLE 5.2: Processing time of the main steps of the saliency learning algorithm

Features	Device	Average time
Make3D	CPU i3-3240	120 ms
Itti and Koch	CPU i3-3240	60 ms
Deep features	CPU i3-3240	3 300 ms
Deep features	GPU Quadro K600	240 ms
Deep features	GPU GTX Titan X	26 ms

TABLE 5.3: Processing time of the feature extraction on different devices.

For this device, the deep features are too long to process to be used in a real time scenario. However, the processing time is much more reasonable on a low-cost GPU (Quadro K600) and is particularly low on a powerful GPU (GTX Titan X).

The segmentation mask is obtained either from a geometrical approach based on the depth map, or from an object classifier based on CNN. The geometrical approach has an execution time that depends on the size of the major plane, the number of objects in the scene, or the difficulty to find walls. When the tracking is lost, additional time is also required to detect the major plane again. Segmentation based on the foveal observation mainly depends on the time required by the CNN to identify the class of the foveal image. The network is such that foveal images are resized to 240×240, thus making inference much faster than on 640×480 contextual images.

As explained in the technical Section 4.5.1, the classifier update mainly depends on the number of input features and samples. As random forest are not incremental classifiers, trees are re-trained from scratch with the whole dataset that is constantly increasing. Figure 5.6 represents the time measured for a classifier update depending on the number of samples in the training set. We evaluate the three types of features this way (recall that Make3D produces 39 feature vectors, Itti and Koch 42 and the deep features 1024). For 100 000 samples (above this number of samples, we randomly remove old examples), the classifier update is around one second for both Make3D and Itti and Koch features, while it is around 13 seconds for the deep features. We also clearly identify the linear relation between the execution time and number of samples. To enable a continuous sample acquisition and saliency evaluation during the classifier update, the classifier is trained on a separate thread (and a dedicated ROS node).

The time required to extract features is not considered in our measure of saliency estimation time. Saliency inference with a random forest is linear with the depth of the trees, it is then very fast. However, to construct the saliency map, this inference is done on each pixel of the image. For deep features, only 14 ms are required to generate the whole saliency map, as the resolution of the feature map is very low (40×30). For Itti and Make3D, we construct the saliency map by inferring only one pixel out of four. Processing time is then around 40 ms in this case. Last when using superpixels to retrieve the original resolution, the superpixels extraction requires an additional 60 ms.

Saliency quality evolution

We now look at the evolution of the saliency quality during the incremental learning. To get an idea of the way saliency is evolving as new observations are taken

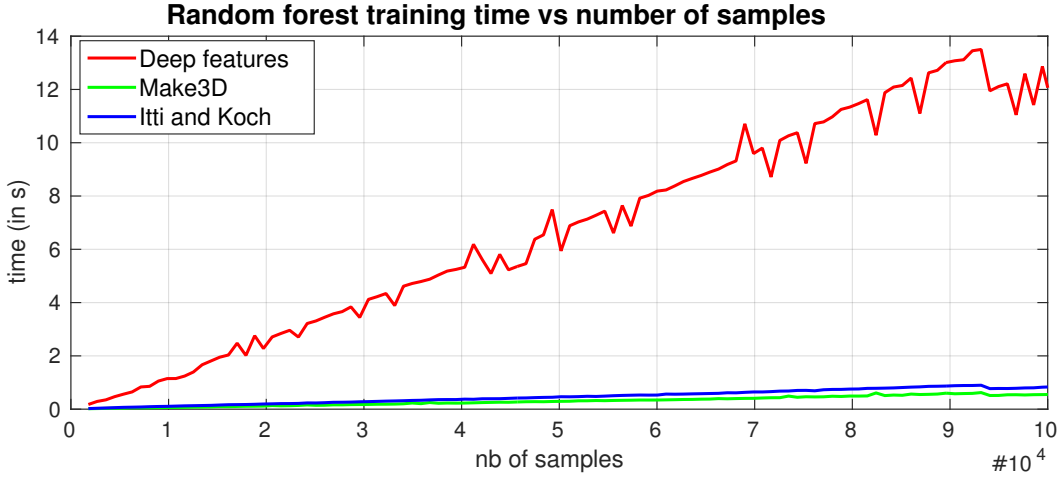


FIGURE 5.6: Execution time of the classifier update as number of samples increases

by the robot, we run the saliency algorithm with the deep feature extraction on a video of the *ENSTA dataset* only. During this sequence, the robot receives observations and updates the saliency model after each of them. We select in this sequence four frames, and we monitor the evolution the saliency produced by the model for those frames after each update. Results are reported in Figure 5.7.

After a single observation, the classifier is usually not able to produce a saliency map. After 5 observation, the produced saliency is very noisy but already able to detect portions of salient objects (the back of the chair in sample 304 for example). The model then makes fast progresses, and in less than 100 observations, the main salient elements of the scene are identified by the model, even before they are discovered by the robot. This property is due to the fact that salient object show common visual properties that can be distinguished from non salient elements. Between 100 and 400 observations, the saliency model is slowly refined. This time, the model makes more fine-grain improvements, by discovering each object separately and learning their visual properties in deeper details. A good example of this behavior is illustrated by frame 335: the mobile container at the back of the room is not naturally salient (almost white with a white wall behind). It is therefore not strongly identified by the saliency model after 100 observations. However, after 400 observations, the robot has reached a similar point of view, and the segmentation process has detected the container. The saliency model has then integrated the visual properties of the box, which is now recognized as salient. Conversely, the extinguisher of frame 260 is red and shows some salient properties. In observation 100, its state of saliency is not clearly defined by the model. After 400 observations, the robot has visited a similar point of view where the segmentation process identified the extinguisher as a part of the wall, and the model integrated the fact that this element is not salient. The model keeps evolving after 400 observations, but the visual evolution is not significant anymore.

Two videos representing the evolution of the saliency are available to better visualize this learning process. The first video [1] exactly represent the aforementioned experiment. Note that this video is not a live video. Training was done offline by

[1] https://www.youtube.com/edit?o=U&video_id=jNovc5jIIhw

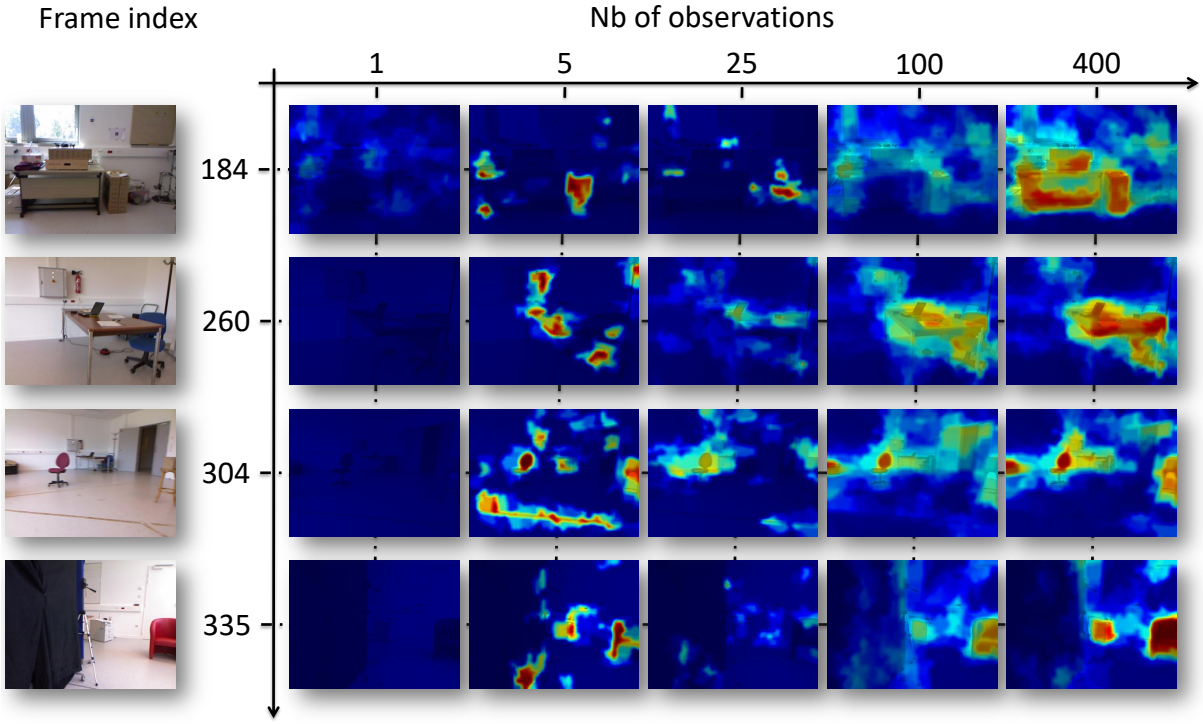


FIGURE 5.7: Evolution of the saliency model by generating saliency maps of fixed frames during a sequence.

updating the classifier after each frame. The second video [2] is a live one and represents a similar experiment with the *BioVision* setup and the Make3D features.

Chapter 8 is dedicated to the numerical evaluation of the saliency evolution, and considers several exploration strategies for that. We therefore limit this section to qualitative evaluation, and refer to this chapter for a more quantified one.

5.4 Comparison with state-of-the-art

To evaluate the saliency model versus existing state-of-the-art approaches, we analyze the final performance reached by the classifier when all samples of the training set are used. When not specified, all experiments conducted in this section were obtained with the deep feature extraction process (detailed in Section 4.3.3) and the superpixel refinement (detailed in Section 4.5.2). We also denote in this section our incremental saliency learning approach and produced saliency maps as ISL.

5.4.1 Saliency maps evaluation

Experiments and metrics

We measure the saliency performance based on the ROC curve evaluation. The ROC (receiver operating characteristic) curve is among the most common techniques to evaluate saliency [92]. The idea is to construct a curve representing the true positive rate versus the false positive rate of the method. Theoretically, a method having a higher area under the ROC curve should be more efficient.

[2] https://www.youtube.com/watch?v=bCdsGLT2UQw

To obtain such a ROC curve, we rely on the ground truth masks of each dataset, that we compare with a saliency map. A saliency map is composed with values between 0 and 1. The procedure then consists in applying thresholds to the saliency maps at regular intervals (10 thresholds in our case). Each threshold value provides a binary image that can be compared with the ground truth to produce a measure of *true positive rate* (TPR) and *false positive rate* (FPR)[3] by a pixelwise comparison.

The same procedure is applied for each threshold of an image. As a threshold produces a specific binary image, the batch of thresholds produces a set of TPR-FPR points that are used to draw the ROC curve of the image. To obtain the ROC curve of a dataset, the FPR values are discretized in bins. TPRs are collected in each bins and the average TPR value is calculated within each bin. The ROC curve is then plot such that the abscissa represents the FPR bins, and ordinate the corresponding average TPR.

Features performance

We first evaluate our three different features on the three datasets by producing the associated ROC curves. These results are reported in Figure 5.8. The three types of features are denoted as **Deep features**, **Make3D** and **Itti and Koch**. To simplify the comparison with results of the next section, we also report the performance of the method called **CAM** [68]. In each dataset, the deep features are always the best performing ones. The performance of the Make3D features are varying a lot depending on the datasets. This is most likely because these features have a moderate generalization capability, while deep features are known to be very good at it.

We also provide visual examples in Figure 5.9 to better understand the performances. On *BioVision* (sample 2), the deep features are the only ones able to correctly estimate saliency and to properly generalize. This is particularly visible in this dataset as the fovea provides a very localized and restricted learning signal. On the two other datasets, the deep features seems to be visually worse than the other. This is explained by the low resolution of the classifier unable to retrieve fine details (for example, the chair leg of sample 3). However, the ROC score is computed at the global image scale and does not really penalize fine details. Again, to demonstrate the good generalization capability of the deep features, we provide in sample 4 the very complex case of a chessboard, where salient elements (the pawns) are of the same color than the ground (black and white squares). In this case, both Itti and Make3D features fail at discriminating salient and non salient elements, while the deep features can.

State-of-the-art performance

To demonstrate the accuracy of our saliency model, we use the four datasets for which we compute ROC curves. For *BioVision*, *ENSTA* and *RGB-D scenes*, the results were obtained by training ISL on their associated training set, and evaluating on their evaluation set. For the *Ciptadi dataset*, we trained our model on *RGB-D scenes*, which is the dataset having the most similar points of view, and we compute the ROC curve based on *Ciptadi*'s images and ground truth masks.

For comparison, we select three publicly available saliency algorithms. First, **BMS** [66] is among the most accurate RGB saliency methods according to the MIT

[3] Recall that $TPR = \frac{tp}{tp+fn}$ and $FPR = \frac{fp}{fp+tn}$ with tp the number of true positive pixels, tn the true negative, and fn the false negatives.

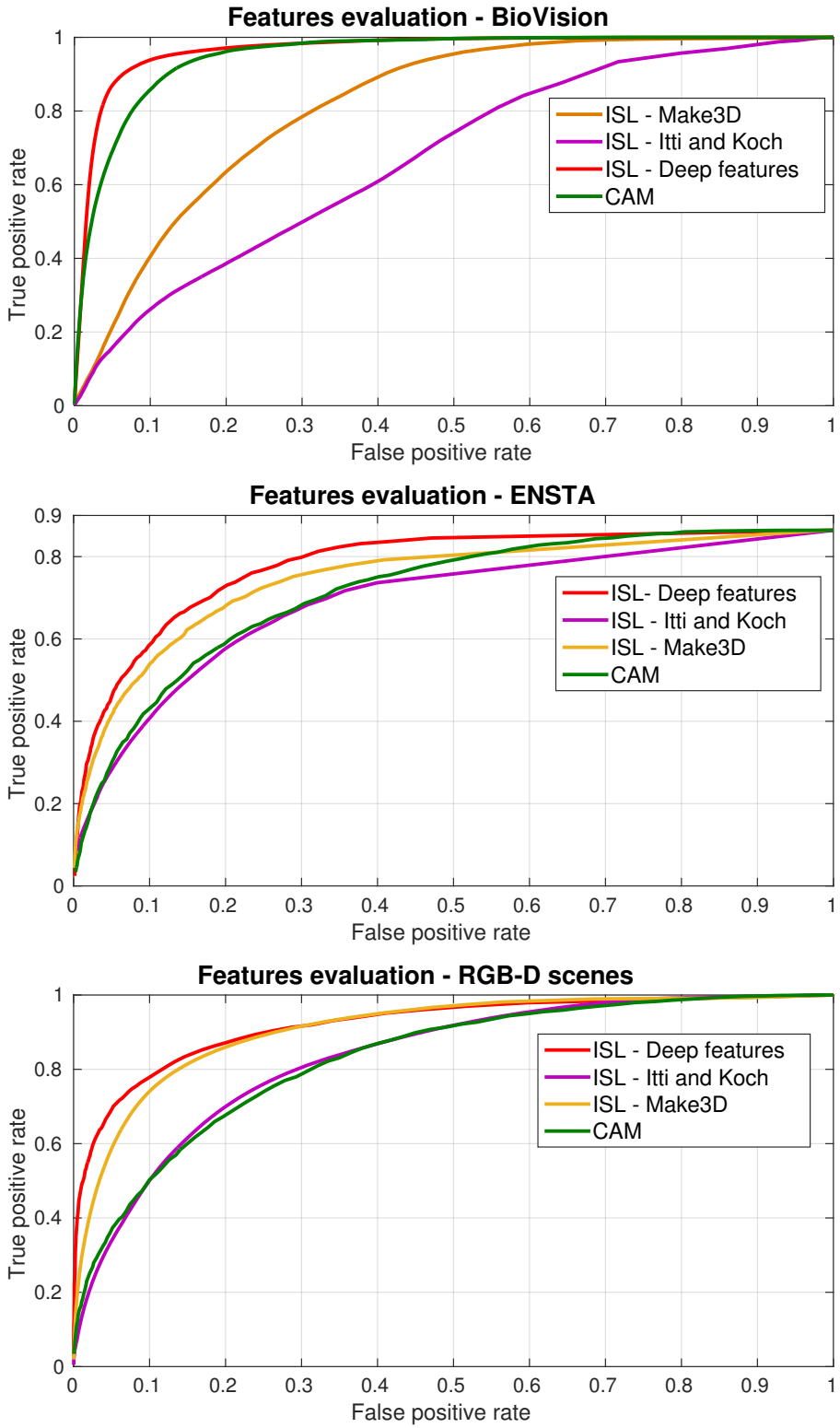


FIGURE 5.8: ROC curves comparing the different features performance on the three datasets

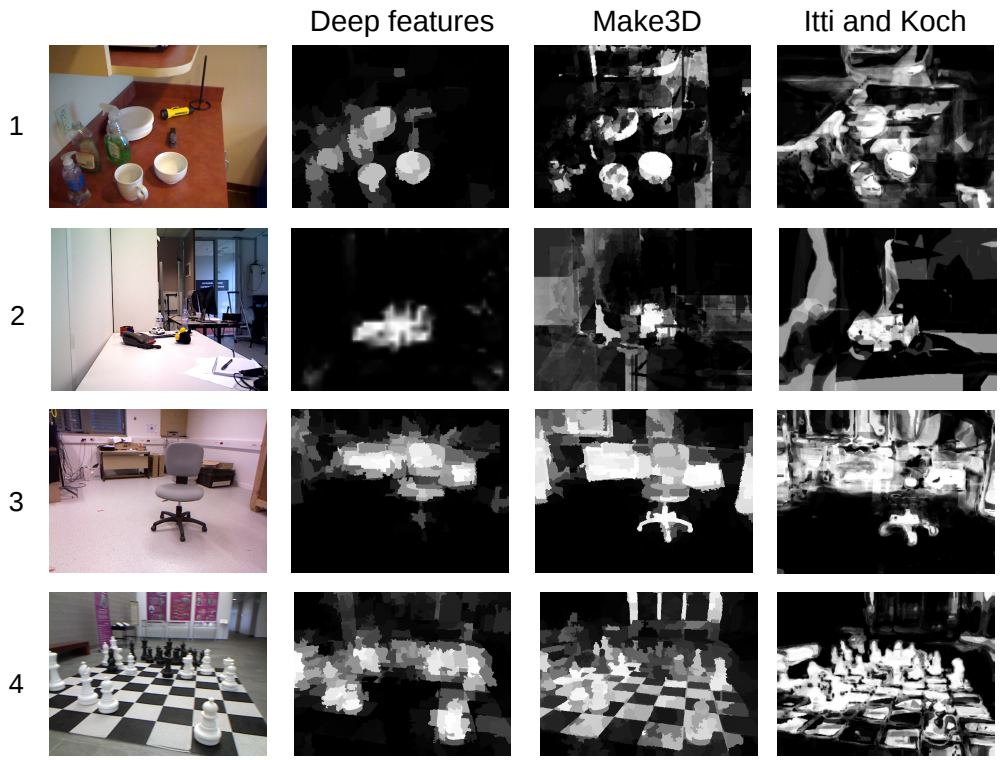


FIGURE 5.9: Sample results of saliency maps obtained with the three types of features

saliency benchmark [16]. We use BMS with the configuration that highlight salient objects rather than salient fixations. Second, we use the new version of the **VOCUS2** algorithm [48] along with the configuration file dedicated to the task of object detection in cluttered scenes (top-down saliency). Lastly, we compare our method with saliency maps produced with the **CAM** [68] 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 refinement technique presented in Section 4.5.2 to increase the resolution of these maps.

For the *ENSTA* and *RGB-D scenes datasets*, we also provide the performance of the segmentation. Recall that the segmentation mask provides only a partial estimation of the saliency. For that reason, we replace pixels that were neither labeled "*salient*" nor "*not salient*" by the procedure by a random value. Lastly, *Ciptadi*'s dataset also comes with associated saliency maps extracted by the method described in their article [31]. As their source code is not available, we only report their results on their dataset. This method has the particularity of being obtained from RGB-D images rather than RGB only.

The results of the ROC-based evaluation are reported in Figure 5.10 and suggest that ISL significantly outperforms the evaluated bottom-up and top-down techniques on all datasets. This result was expected for at least the three first datasets, because our model is trained for a dedicated environment, from a learning signal that is close to the ground truth. However, the good performance on *Ciptadi*'s

dataset shows that the method is still usable in environments in which it was not trained. 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, whereas pixels without saliency estimation in the segmentation mask (more than 50% of them) are replaced by noise.

In figure 5.11, a visual comparison between the methods 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. Sample 3 is taken from the *BioVision dataset* and represents an image with a highly textured background. In this case, ISL significantly outperforms other approaches as it is the only one for which these elements are part of the model and clearly defined as not salient items. 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 on 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, 2 and 3 without superpixel reconstruction). 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 6), or red power outlet (samples 5,7), that are visually salient but irrelevant for an indoor mobile robot. Lastly, it enhances elements that are not naturally salient (mobile container on sample 7) but consistent with our definition.

5.4.2 Bounding box proposals

Experiments

We now demonstrate that ISL can be used to produce relevant bounding boxes around objects. To this end, we run the EdgeBoxes [169] 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 4.5.3:

$$SCscore(B) = \frac{1}{w_B \times h_B} \sum_{i,j \in B} S(i,j) \tag{5.1}$$

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. 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 saliency maps (denoted as **EB+ISL** and **EB+BMS** in Figure 5.12). Lastly, we generate the SegBoxes from the depth segmentation process, as described in Section 4.5.3. 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 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 5.12. In practice, a small number of SegBoxes are detected on each frame (between 0 and 7 on

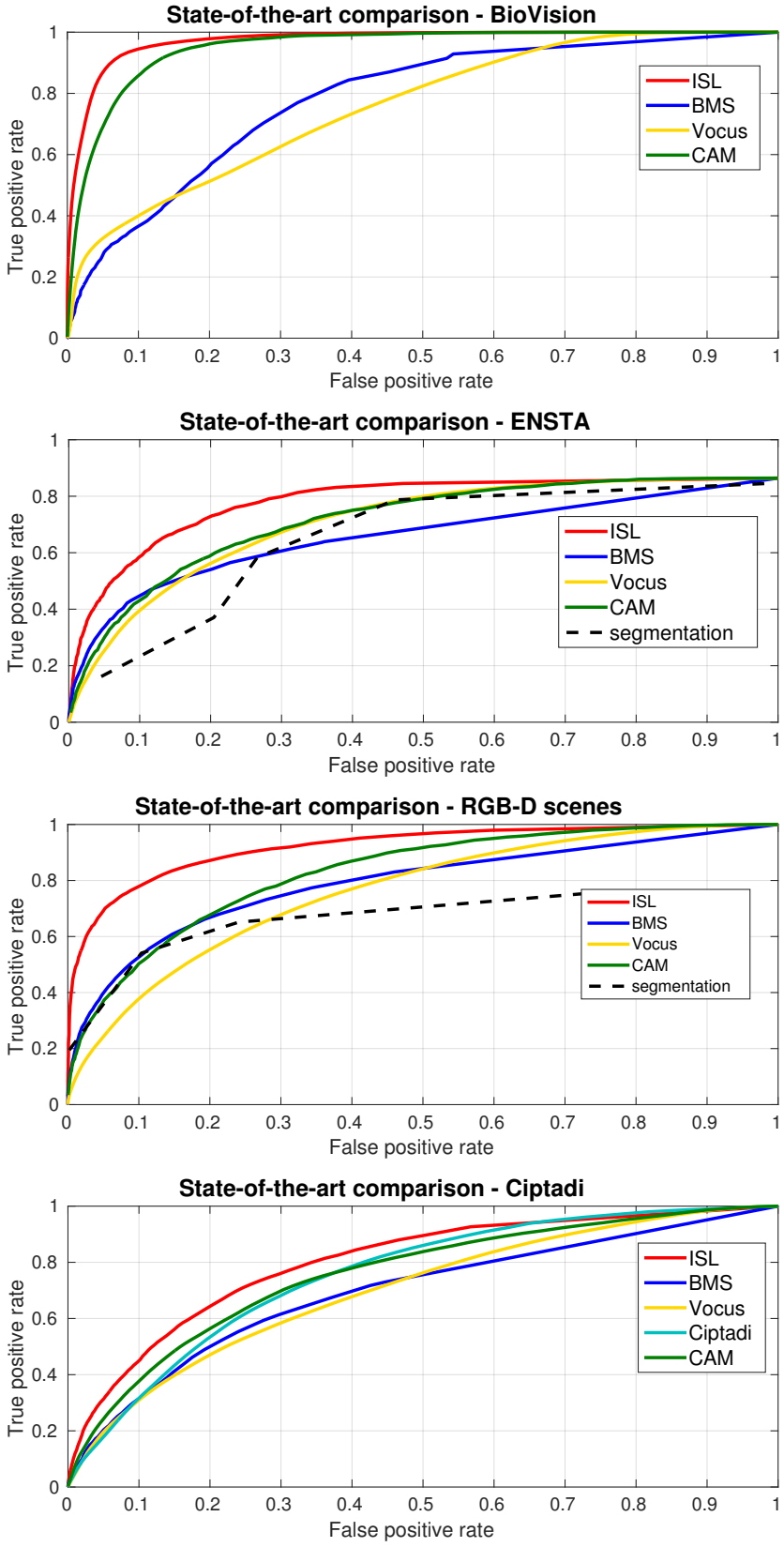


FIGURE 5.10: ROC curves comparing state-of-the-art approaches on four datasets

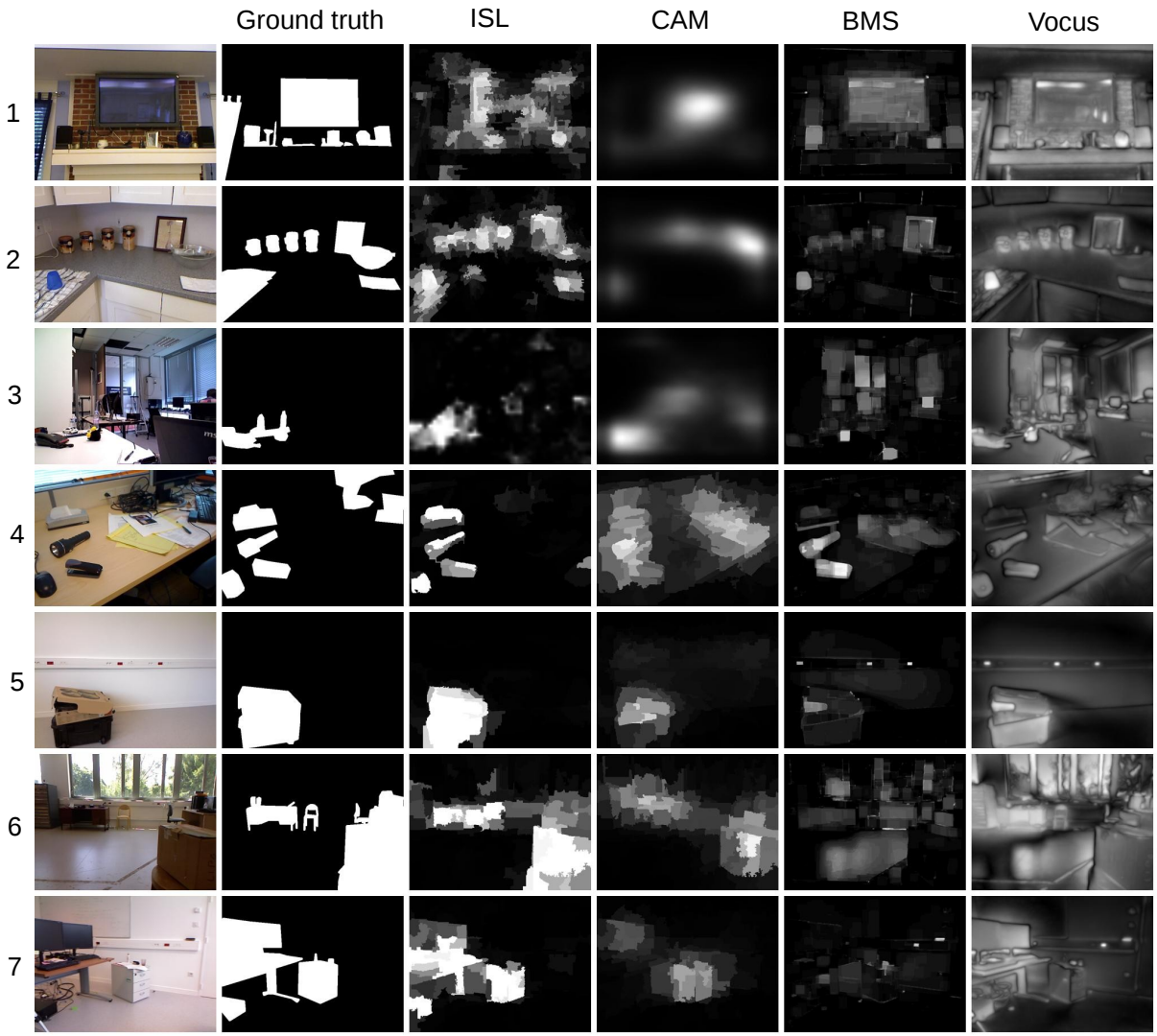


FIGURE 5.11: Sample results of saliency maps obtained compared with state of the art

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 5.12 and Figure 5.13 as **EB+ISL+SB**.

Metrics

The chosen evaluation metric is the detection rate versus the number of proposals, based on the *intersection over union* (IoU) measure to count the number of detections. This measure is used by Zitnick *et al.* [169] to evaluate their performance over state-of-the-art approaches. This measure first needs a ranking of the proposed bounding boxes. We then use the ranking obtained by the SCscore for both the EdgeBoxes and Segboxes. When combining EdgeBoxes and SegBoxes, we arbitrarily rank the SegBoxes higher than the EdgeBoxes, as they are much more likely to actually contain objects.

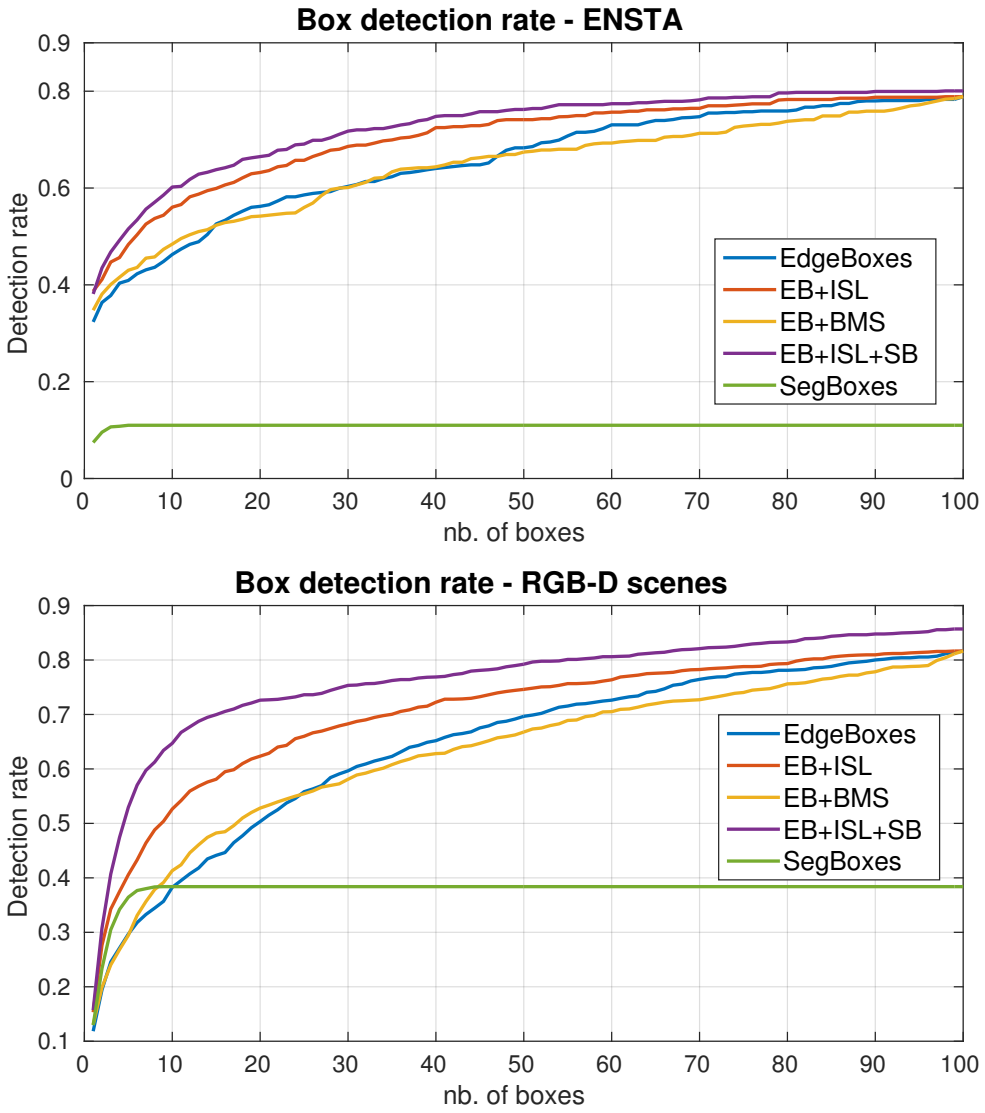


FIGURE 5.12: Detection rate vs number of box proposals comparison on the *ENSTA* and *RGB-D scenes datasets*

The detection rate for a set of calculated bounding boxes is obtained by comparing these proposed boxes with the ground truth (recall that *ENSTA* and *RGB-D scenes dataset* were annotated with bounding boxes around objects). The computation procedure is then as follows: for a given image, each bounding box GT_i of the ground truth is compared versus each proposed bounding boxes $Bbox_j$. For that, we produce for each pair of boxes an IoU score

$$IoU(GT_i, Bbox_j) = \frac{A_{GT_i} \cap A_{Bbox_j}}{A_{GT_i} \cup A_{Bbox_j}} \tag{5.2}$$

A_B being the area delimited by the bounding box. We consider GT_i and $Bbox_j$ to be matching if their IoU score is above 0.5. Therefore, the number of true positives TP_k in an image k is the number of ground truth bounding boxes matched with at least one of the proposed bounding boxes. The detection rate (or recall) for this

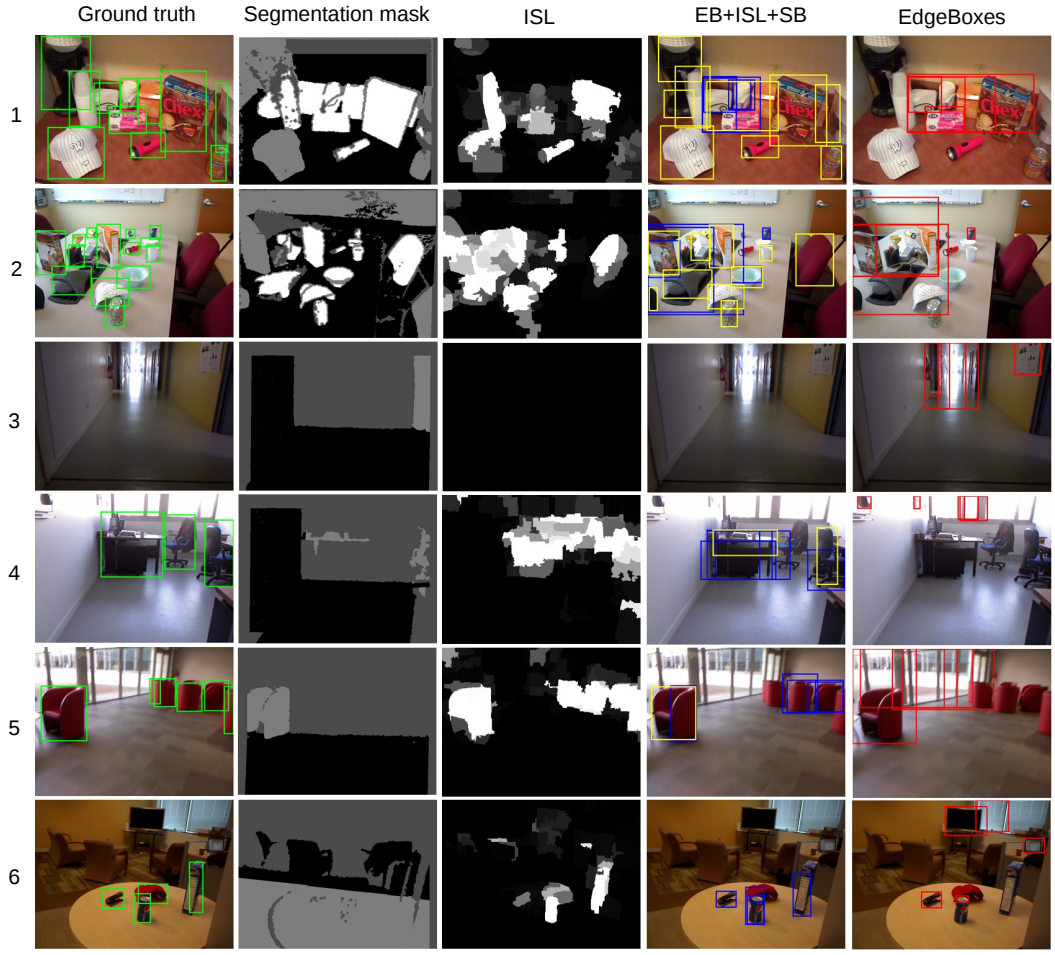


FIGURE 5.13: Sample results of bounding box proposals versus EdgeBoxes

image is then

$$DR = \frac{TP_k}{N_{GT}} \tag{5.3}$$

N_{GT} being the number of ground truth bounding boxes in this image. To draw the detection rate versus number of bounding boxes, we take for each $N \in [1..100]$ the N best ranked boxes for each image of the dataset, and we compute the average detection rate based on these boxes. The curve then keeps increasing with the number of proposed boxes.

Results

Numerical results are reported in Figure 5.12 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 our saliency 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 5.13 shows sample results of the top 5 EdgeBoxes (row **EdgeBoxes**), top 5 EbdgeBoxes re-ranked by the SCscore with ISL (displayed in row **EB+ISL+SB**, blue boxes), and Segboxes (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 5), or because segmentation failed (sample 6). 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 4, 5). Lastly, it is possible to cope with frames that do not contain any salient object (sample 3) by filtering boxes with an SCscore below a certain threshold (0.01 in our case).

5.5 Conclusion

In this chapter, we considered different aspects of our saliency learning technique and evaluated them on four different datasets. As a summary and concluding remarks, our experimental evaluation led us to the following findings:

First, the self-supervision signal on all datasets is such that the saliency model is able to generalize well. Provided that supervision is partial, strongly unbalanced and, in the case of *BioVision*, weakly supervised, this generalization capability was not straightforward. Aside from this result, we evaluated the supervision signal and found it reliable enough in terms of precision and recall to ensure a descent learning quality.

Second, we have shown that the execution time depends on many factors and still suffers from limitations. First, the deep features require a GPU to work at a reasonable framerate. Second, the random forest are such that the training time is increasing linearly with the dataset size. We partially solved the problem by training the model on a separate thread, but using a truly incremental classifier may be more appropriate.

Third, we illustrated the saliency evolution process, and explained that this model was evolving with several steps. After a few iterations required to get a relatively stable saliency model, naturally salient elements were first enhanced by the common properties they shared with already discovered object. The model is then progressively refined after each object is discovered separately.

Fourth, we demonstrated that the deep features were the features having the best generalization capability. This was especially visible on the *BioVision dataset* were the supervision signal was extremely weak. However, the poor resolution of the produced map cannot retrieve very fine details and could probably be improved in a future work.

Fifth, we outperform state-of-the-art techniques not only on our datasets, but also on an external one with a model not trained on it. Unlike other tested saliency techniques, our approach is really good at removing naturally salient elements that are not salient in our context, and conversely, enhancing salient elements in our context that are not naturally salient.

Lastly, our saliency technique can be used to produce relevant bounding boxes around objects by re-ranking EdgeBoxes according to our saliency model. In addition, using boxes from object segmentation (the SegBoxes) on top of the EdgeBoxes further improves the results.

Part II

Intrinsically motivated exploration

Chapter 6

Environment exploration on an autonomous robot

6.1 Introduction

This chapter provides an overview of the exploration problem in autonomous robotics when considered by three different fields of robotics that are mobile robotics, vision-based robotics (or active vision) and developmental robotics.

In mobile robotics, exploration is typically seen as a problem of path planning across the environment by maximizing different costs (time, energy, information, *etc*...). We focus more precisely on the problems of localization, mapping and planning for mobile robots.

We then concentrate our review around the exploration problem in a more general scope. We study in deeper details exploration techniques of mobile robotics and active vision systems.

In developmental robotics, a very specific type of exploration is considered, where the stress is put on the learning and development of skills and knowledge based on motivations. The problem is seen under a bio-inspired point of view, and many parallels are made with neuro-science and psychology. We then present an overview of the field of developmental robotics, and focus on the motivations as a drive for exploration. Lastly, we describe in details robotics architectures driven by a particular type of intrinsic motivation: the learning progress.

We show that for mobile, vision and developmental robotics, the philosophy and formalism may be different in the way to tackle exploration. However, they share an essential common idea: the acquisition of knowledge through actions impacting the environment.

6.2 Autonomous navigation on a mobile robot

A *mobile robot* is defined as a machine capable of locomotion. They have the capacity to move in their environment and are not limited to a simple physical location. Foveated systems that are not embedded in a platform able to move are then not considered as mobile robots. Mobile robots are then able to interact with their environment, and must therefore be careful of the geometry and behavior of the surrounding world.

To autonomously move in their environment, mobile robots must be equipped with an efficient navigation strategy to accomplish a specific mission. As depicted in Figure 6.1, the most general navigation scheme is to consider a sensorimotor loop in which the robot perceives data from the environment and exploits this data for self-localization. The robot is also equipped with a mission plan composed with

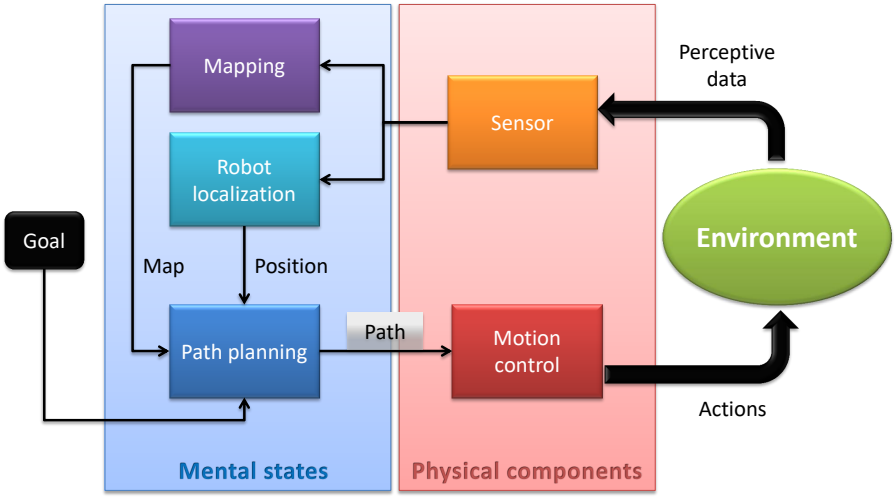


FIGURE 6.1: Basic architecture for autonomous navigation on a mobile robot. Localization, mapping and path planning are the three essential components of the control.

successive goals or tasks that should be achieved. The localization is then combined with the mission plan to generate motion control and displace the robot.

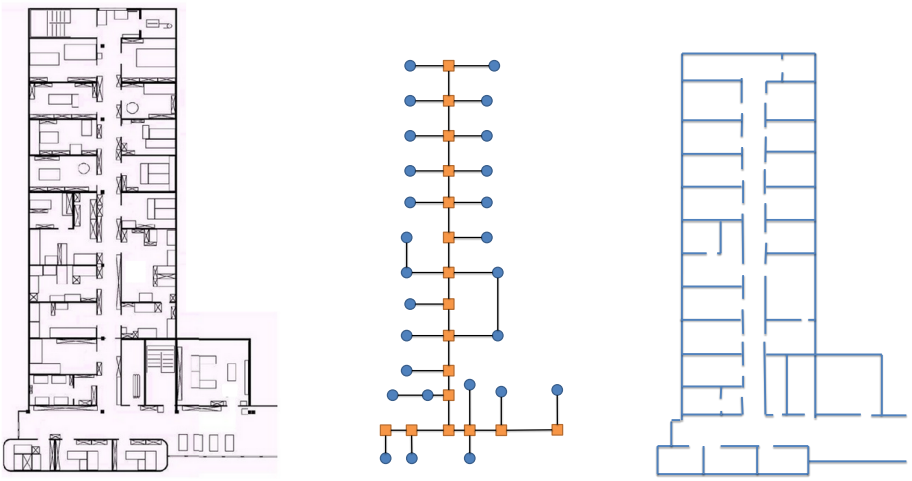
Most autonomous mobile robots use a mental representation of their environment as a map in which they can be localized. To accomplish a mission plan, a set of goals must be reached, and a trajectory must be found in that regard. We then describe and discuss in this section the three aspects of navigation that are *mapping*, *localization* and *planning*. The next two sections are widely inspired by Filliat's course on mobile robotics [201].

6.2.1 Mapping and localization

The approach used to determine a trajectory strongly depends on the way the environment is represented. Depending on the sensor used and available information, the map of the environment can either be *topological* or *metric* (See Figure 6.2 for examples).

Topological maps

In *topological maps*, the environment is represented as a graph (also called a *navigation graph*) where nodes are places that the robot can reach, and edges represent the available connections between nodes. In general the information stored in edges is related to the way to reach nodes of the graph, whereas nodes store identification information (for example, a set of images [202]) to localize the robot from sensory input. In topological maps, nodes may represent significant or meaningful places (such as doors, intersections or room centroids), that are either manually designed [22], [203] or automatically added when discovered by the robot [204]. Some approaches rather define nodes based on the robot exploration without any semantic meaning. Criteria for splitting nodes in this case can be the strong variation in the perceived inputs [205], or just euclidean distance boundaries [110]. Edges are defined either by adjacent relations (when the robot has the ability to directly attain a node from neighbor [22], [110], [123], potentially storing path commands for displacement), or by metric information containing relative positions of



(a) (b) (c)

FIGURE 6.2: (a) Real building map (b) Topological map (c) Metric map

the nodes [202]. Sometimes, maps even contain absolute positions of the nodes and are referred as *topometric maps* [110], [123], [202]. They are particularly well-suited for vision.

The advantages of topological maps is their simplicity to store and update information. Moreover, grouping information among distinct places make the representation of those places better-suited for a representation at a sensor's level. Lastly, discretizing the environment is a good representation for planning, where paths are directly found from the graph. A major disadvantage occurs when data need to be interpreted or integrated from places that have not been visited yet. Topological graphs usually require a very throughout exploration of the environment to avoid these type of cases. Moreover, in case of noisy data or dynamic environment, the localization may become a problem as identification is likely to fail.

Our exploration technique is based on a topometric map, where edges contain distances between adjacent nodes, and nodes contain information about the local model quality. As several approach rely on an already existing topometric map to guide their exploration [110], [123], other focus on the construction of this map online [202], [206]. We also propose in this work a simple technique to incrementally construct a topometric map adapted to our needs and considering the localization problem as solved and reliable.

Metric maps

In *metric maps* the environment is represented by a set of objects associated with positions in a metric space (typically a 2D map), where the position of the robot is estimated. Metric maps can be represented in terms of "objects" or *occupancy grids*. In the first case, "objects" are detected by sensors, localized from the robot's position and integrated to the map. Objects are either points [207] or oriented points [208] (also called anchors), or frontiers (also called features) composed of primitive shapes such as plans, lines or ellipses [209]. These objects are often associated with a measure of uncertainty to avoid navigation mistakes [207]–[209]. In the second case, occupancy grids [210] are based on fine grids segmenting the environment, on which

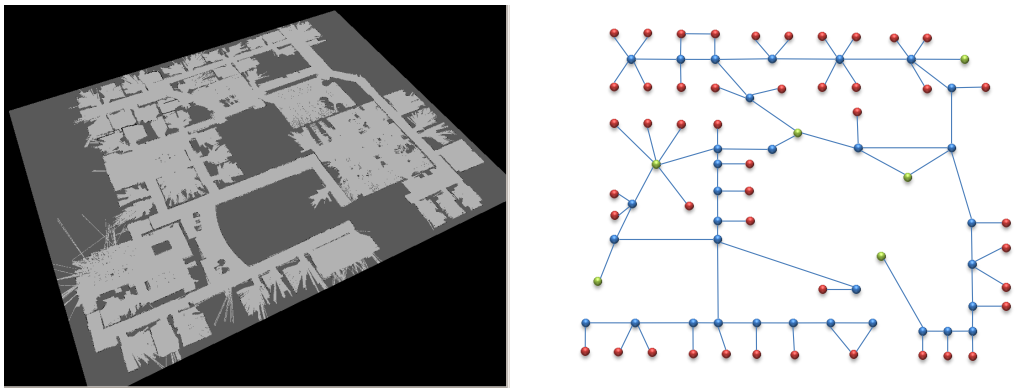


FIGURE 6.3: Occupancy grid obtained by a SLAM algorithm and possible associated navigation graph modeling rooms, halls and corridors.

each cell contains an occupancy probability. Although more memory consuming, occupancy grids do not require any object extractor.

Unlike topological maps, metric maps combine localization and sensory data at the same level. With metric maps, the whole environment is represented, and the robot can be localized more precisely. This kind of maps is also more easily usable by other types of robots, using different sensors, as the environment is not represented in terms of sensory data. It is also easier for a human user to interpret and follow the robot trajectory. However, metric map usually require a more efficient localization method. Planning strategies from these maps are also less straightforward, as the environment representation is continuous.

The navigation graph we rely on in our work is actually based on an occupancy grid produced by a laser range finder (similar to the example of Figure 6.3). Therefore, we rely on both metric and topological maps.

Simultaneous localization and mapping

Except for cases where the map is known beforehand and static during the whole experiment [123], [211], localization and mapping are strongly intertwined and should be jointly processed. The map of the environment is then incrementally constructed or updated, and localization within the map is inferred from the same observations. In the literature, this joint problem is referred as *Simultaneous Localization And Mapping* (or *SLAM*) [212], [213]. Localization may be obtained by *proprioceptive sensors* (wheel speed, inertial sensors) but is often subject to drift in the measurement. Therefore, *exteroceptive sensors* are required to perceive the environment of the robot and add a correction feedback to the displacement measures. SLAM based on laser range finders for example detect obstacles and perform a scan correlation with the previous measurements to estimate the displacement of the robot [25]. They typically generate occupancy grids. Another family of SLAM techniques are visual-based, and use the similarities of successive frames to estimate the displacement of the robot [214]. Of course, the combination of visual and geometrical cues, when possible, provides even more accurate results. The development of RGB-D SLAM techniques have then become popular in the last few years [215]. Visual and RGB-D SLAM are usually better-suited to construct "object maps".

Our exploration system relies on the hector mapping algorithm [25], based on a laser range finder and odometry combination. We consider this SLAM algorithm reliable enough for our needs.

6.2.2 Planning and path finding

During a mobile robot's mission, a sequence of tasks involving the displacement of the robot must be achieved. Tasks can then be divided into a set of target positions to reach that will move the robot to a certain location. In this section, we investigate the problem of planning a strategy to reach a goal, once mapping and localization are known. We also focus on planning approaches determined in two-dimensional maps. To be efficient, the chosen path should optimize a cost function (based on distance, time or energy), and be safe to the robot (collision-free).

Continuous and discrete environments

Finding an optimal trajectory with multiple obstacles in a continuous space is an NP-hard problem, but several approach tend to solve the problem locally, or by heuristic methods. Early approaches relied on potential fields [216], while more recent techniques such as improved version of $RRT^{\star}$ [217] rely on sampling a very large number of trajectories to approximate to the optimal solution. Another class of approaches use a "teach and repeat" strategy to make the robot learn its own path (not necessarily optimal) from experience and adapt them in new situations [218], [219].

To simplify the problem, many planning approaches are using techniques of path finding in a graph. If using a metric map, the space must then be turned into a topometric graph by discretization. A first approach to discretize maps is to cut them in segments, either by following the obstacle shapes [220], or according to a regular grid [221]. Nodes are then defined as centroids or corners of each segment, and adjacent segments are linked by edges if no obstacles are found in between. Another approach consists in pre-defining paths between points of the environment, either by visibility graphs [222], Voronoi diagrams to define path as far as possible from obstacles [123], [223], or generated by sampling approaches [224]. Points are then used as nodes and pre-defined path are links.

Our system uses a method of path finding in a graph, and therefore turns the occupancy grid obtained from a SLAM algorithm into a discrete navigation graph from regular cells.

Path finding in a graph

When trying to find an optimal path in a graph, the problem is typically solved by finding an optimal sequence of actions to follow. To do so, methods based on *direct path finding* or *policies* can be used (see Figure 6.4).

Direct path finding may be obtained by exact resolution methods using graph search, such as the Dijkstra [225] algorithm. If the number of nodes is too high for an extensive search, heuristics such as $A^{\star}$ [226] (or $D^{\star}$ in dynamic environments) are able to find an acceptable solution in a significantly reduced time. Alternative heuristic such as the particle swarm optimization [227] or the ant colony optimization [228] are also commonly used. In these types of approaches, the problem is supposed to be fully deterministic, and problems occur if the robot cannot reach the goal or deviates from its course. In this case, a re-planning process is required.

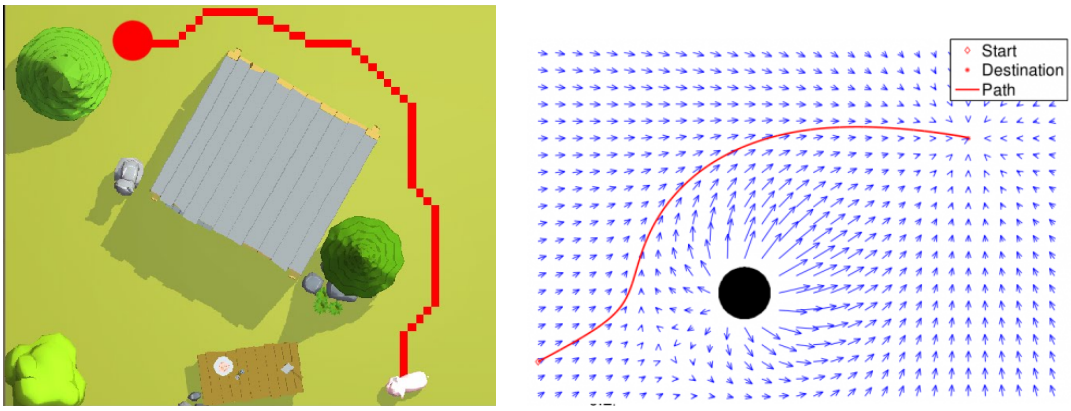


FIGURE 6.4: Direct path finding with the A* algorithm (source Sebastian Lague [229]) and potential field policy (from [230])

The second category of planning is based on policies. A *policy* associates for each node (whether the robot is in it or not) the best action to take from this node. To reach the goal with a policy, the robot will perform a sequence of node identification and action dictated by the policy at the current node. Although slower than a simple path finding, a policy does not have to be re-computed if the estimated position of the robot is wrong, and is better-suited if several specific goals are concurrent in the environment (even potentially no goal at all). Instead of considering a discrete set of actions for each state, a possible approach to determine a policy is to use a *potential field* that depends on the cost to reach the goal. The action is then calculated by following a gradient descent. To find such potential, Dijkstra algorithm can be used as well [225], or a value iteration procedure [231].

Our planning strategy is based on a path finding found from a policy, and does not depends on a single goal, but rather on attractive areas.

Path finding under uncertainty

In case of uncertainty (noisy observations, dynamic environment ...), the problem should not be modeled with a deterministic graph but rather as a *Markov Decision Process*. A probability is then associated with each state-action pair of the policy. If the environment is only partially known by the robot, the model is a *partially observable decision process*, or *POMDP*. The field of path finding in a POMDP context is very active, the main goal being to find the best action to take for the current state of belief [116], [232], [233] and to update this belief on the fly. POMDP is also introduced the context of reinforcement learning techniques that are further described in Section 6.3.2.

6.3 Exploration in mobile robotics and active vision

In robotics, the *exploration problem* is the one of maximizing knowledge over a working environment by means of a single or several robots. In many practical problems, the robot has a specific mission (such as delivering mail, monitoring a known area, driving a user to a specific address). Goals are specified by the mission, and path plans are generated toward these goals. When the mission is related to exploration, an exploration strategy is provided by the user, and the robot should autonomously

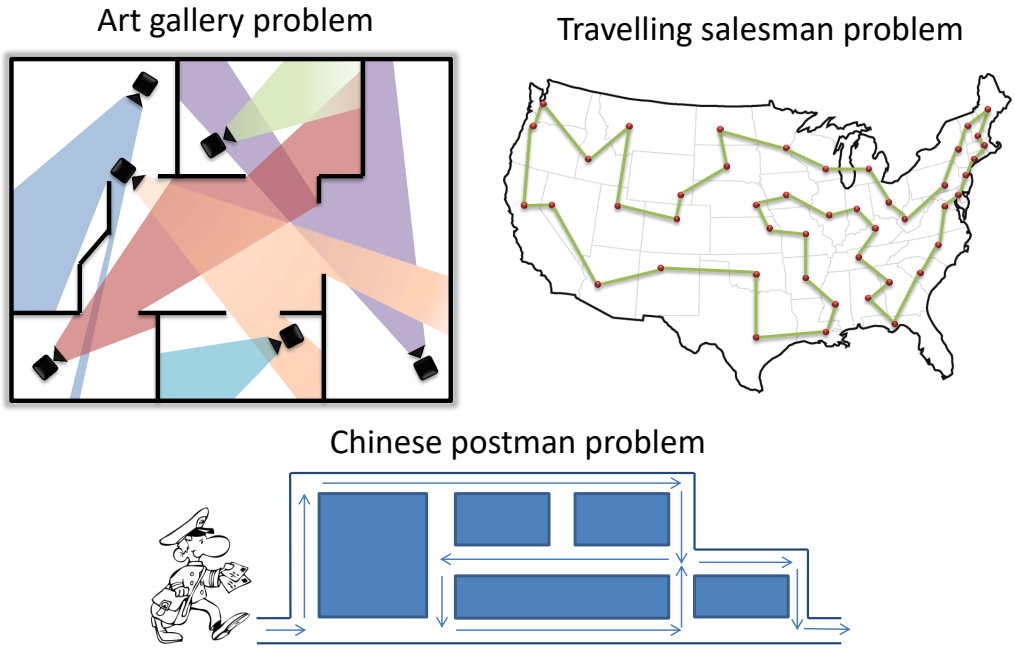


FIGURE 6.5: Example of exploration problem.

generate goals that are likely to increase its knowledge. Exploration is then seen as a mean to reach self-defined goals in the general purpose of maximizing knowledge.

It may happen that the robot performs a simultaneous exploration (for knowledge updates or initial discovery) and task. Several approaches raise the problem of trade-off between exploration and exploitation [234], [235], but these aspects are beyond the scope of this review.

We here present three possible types of exploration that could be used in most robotics systems. We however concentrate our explanations and examples on mobile robotics and active vision systems. A first purpose for exploration is the one of finding concrete elements of the environment, and memorizing knowledge about them. We call it *memory-oriented exploration*. Another purpose for exploring is the one probing the environment for learning. By learning, the robot acquires skills or predictive knowledge that are not directly representative of the environment, but useful for future tasks. We divide learning-based exploration between *reinforcement learning*, and *model-based exploration*.

6.3.1 Memory-oriented exploration

In a memory-oriented exploration, the goal is to have a good coverage of the environment towards elements of interest. The robot then keeps an instance of these elements in memory, varying from state variables, maps, lists of objects and events and their attributes.

In mobile robotics

In mobile robotics, building a map of the environment may be the most straightforward exploration application. In this case, exploration is related to the coverage whole environment rather than isolated targets of interest [123], [202], [236]. Autonomous SLAM approaches [237], for example, need exploration techniques to efficiently cover the environment and spend more time on more challenging areas.

Localizing and identifying objects in the environment is another typical purpose for exploration [102], [110], [181]. When no prior about the objects are given, the problem is called *object discovery*, and exploration should be such that areas likely to contain objects are examined. Lastly, in dynamic environments [235], [238], exploration should be such that places are constantly re-visited and memory updated accordingly.

When the robot is provided with a static map at the beginning of the exploration, the user can design pre-defined path plans prior to the experiment [101]. Otherwise, navigation path plans can be found by solving specific mathematical problems from graphs or continuous environments. Figure 6.5 presents a few examples of common problems to be solved for environment exploration. The most famous one is probably the *traveling salesman problem* [239], when an optimal path is found to join a set of areas of the map while minimizing the traveled distance. If the problem is to find an appropriate set of views to cover the whole environment (called *view planning problems*), the *art gallery problem* [240] is a good approach to consider. Otherwise, when the goal is to navigate to visit all edges of a graph, the *chinese postman problem* is the approach to considered [241]. As these problems are often NP-hard, solving them directly on a robot is not feasible. They are then either solved offline [123], require heuristics [236], or are replaced by more naive greedy algorithms [110]. Lastly, modern approaches usually combine these optimal path with sensory-based refinement to ensure that exploration is consistent with the true environment [123], [236].

When the map of the environment is not known beforehand, the robot can only rely on sensory data to drive exploration. A very common technique is the frontier-based exploration [236], [242], where sensory inputs are used to detect the boundaries (or frontiers) between known and unknown areas. Exploration is then done to visit these frontiers. Entropy reduction or information gain maximization [243] can also drive the robot to unexplored or highly informative regions from sensory inputs. The time spent in displacement can also be taken into account to speed up exploration [244], and planning to re-visit locations is sometimes considered to increase the knowledge quality [245]. In dynamic environments, temporal statistical approaches are used to determine how frequently exploration should be performed in a particular area [235].

In some cases, the problem of gathering information is considered jointly with the one of minimizing displacements. This is for example the case in Wang *et al.*'s work [246], where a combination of the traveling salesman and art gallery problems are sequentially solved to optimize both aspects of exploration. This idea of joint optimization is also raised by Tovar *et al.* [237] in the context of simultaneous localization and mapping. In their work, a utility function was proposed to measure the interestingness of potential location and balancing the potential information gain with the displacement cost. Jointly combining information and displacement cost is at the heart of the exploration technique proposed in the next chapter.

In active vision systems

Exploration can be guided by visual inputs as well, and is tightly linked with the mechanism of visual attention in Chapter 3 by targeting salient regions of the field of view. If the environment is static, the inhibition of return plays a major role in exploration, as it avoids the system to get stuck in the most salient location. When the robot is equipped with PTZ cameras (pan-tilt-zoom) [102], [109], [110] or with a combination of contextual (wide field of view) and foveal (sharp field of view)

cameras [60], [247], the actions performed for exploration are typically saccades, so that the zoomed camera is oriented to informative regions. If the robot can move across its environment, actions are displacements in order to get a closer or better point of view of areas of interest [110], [116], [181], [248].

6.3.2 Reinforcement learning-driven exploration

When the goal of the robot is to learn behaviors (for example, following an optimal trajectory towards a given area of the environment, performing saccades to identify an object, ...), a possible approach is to learn it by trying successive alternatives until the desired behavior is reached. *Reinforcement learning* exploits this principle of learning by successive trials, either leading to success or failure. This is made possible by receiving a feedback from the environment based on the quality of the attempt. This feedback, called a *reward*, will guide the robot to find which actions are good based on past experiments.

Applications

Reinforcement learning is a vast field of research and is used for many different purposes in robotics (see for example Kober and Kormushev reviews [249], [250]). In the context of visual attention, such techniques have been exploited to automatically guide the gaze toward salient areas [109], or to learn the optimal saccades for object recognition [49].

In mobile robotics, reinforcement learning is often used to find an optimal displacement policy in a given environment [251], [252] and for a specific task. In this context, reinforcement learning is equivalent to a planning tasks minimizing an objective function, except that this objective function is expressed in terms of rewards. In that regard, reinforcement learning is a possible technique to find optimal path plans by means of policies, as described in Section 6.2.2. In our approach, we use reinforcement learning in this context.

Lastly, some hybrid approaches combine saccades and robot locomotion in a reinforcement learning framework to jointly identify salient objects, and use them as visual anchors to navigate in the environment for reaching a target location. Borji *et al.*have proposed a set of publications in that regard [108], [211], inspired by the work of Mc Callum [253].

A large panel of reinforcement learning techniques have been proposed in the literature to solve problems in very different configurations (continuous or discrete states and actions, model-based and model-free techniques). A description of basic reinforcement learning problems has been proposed by Sutton and Barto [254].

Q-learning

To keep explanations easier, we describe in this section the Q-learning approach in a discrete state-action space only, that is further used in Chapter 7.

Reinforcement learning problems are modeled by an autonomous agent confronted to a given environment and having to select actions depending on its current state. To illustrate our explanations, let us consider the simple case of a robot able to move on a grid and whose goal is to reach a target G while avoiding a wall (See Figure 6.6). After making an action, the agent arrives in a new state and receives a *reward* (positive or negative) related to this state. This reward is the only signal that will guide the agent's learning, to progressively find the best action to

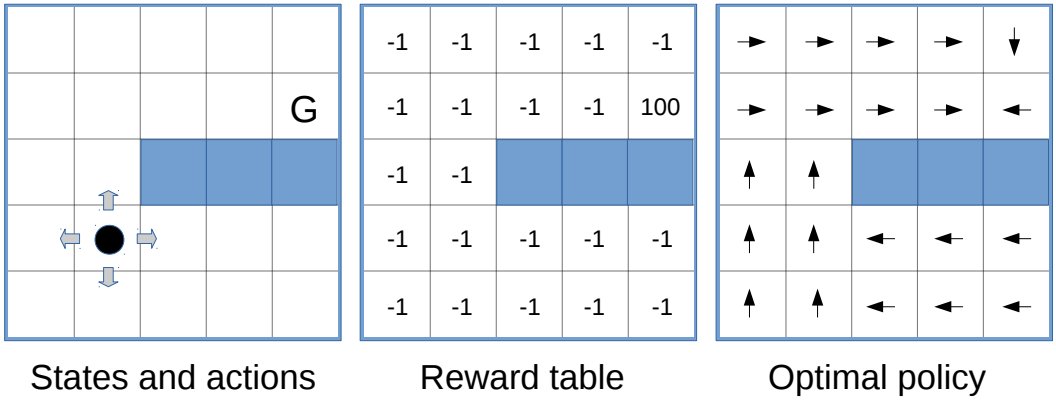


FIGURE 6.6: Environment, rewards and policy

take in each state of the environment. This correspondence between states and actions, that describes the optimal behavior to adopt is called the *optimal policy* (often called $\Pi^\star$).

Problem definition: Formally, the problem can be defined as a set of states S (the reachable squares of the grid), a set of actions A ("up", "down", "left" "right"), and a set of rewards $R = r(s, a)$. Typically, a high reward is associated with the goal state, and low reward are distributed everywhere else. Rewards are collected by the agent after taking an action while being in a given state. The agent starts in an initial state s_0, chooses an action a_1 and arrives in a new state s_1. If considering that reaching state s_{t+1} from s_t only depends on s_t and on action a_{t+1}, the problem can be modeled as a Markov Decision Process. An MDP is defined by a system (S, A, P, R), where $P = P(s'|s, a)$ are the transition probabilities between states given an action. Lastly, the policy $\Pi : S \rightarrow A$ is a function that associates an action to follow for each state of the environment. The goal is to iteratively modify the policy, and test its efficiency to progressively converge to an optimal one ($\Pi^\star$).

Action-value function: To converge to an optimal policy, a large number of experiments (or episodes) are conducted, in which the robot takes actions suggested by the current policy, receives rewards, and evaluates the benefits of these actions for the policy. This benefit is evaluated by considering an *action-value function* $Q^\pi(s, a)$ (which is a matrix in our case) that is progressively updated with experiments. This action value function represents the average cumulated reward during experiments.

$$Q^\pi(s, a) = E[R|s, a, \Pi] \tag{6.1}$$

where R is the cumulated reward received after taking action a in state s, and following policy Π thereafter. Mathematically, R is defined as

$$R = \sum_{t=1}^{N} \gamma^t r_{t+1} \tag{6.2}$$

where N is the number of successive actions in the sequence, r_{t+1} is the reward received at time step $t + 1$, and γ is the *discount factor* between 0 and 1. γ is used to make the agent reason about the long term consequences of its actions. It is a parameter that controls how much the value function should take future rewards into account. If $\gamma = 0$, then only immediate reward matters. If $\gamma = 1$, reward received in future steps all matter the same. The discount factor is an important parameter of our system, and its influence on the performance of our algorithm is discussed

in Chapter 8. The action-value function associated with the optimal policy $\Pi^\star$ must then be such that $Q^{\pi^\star}$, or $Q^\star$ is maximum for each state-action pair.

Exploration strategy: The Q-learning algorithm provides a way to guide the robot's exploration and to update the action-value function so that it naturally converges to $Q^\star$. The robot should, most of the time, follow the policy that is currently suggested by the current policy, and occasionally select a random action (this is called the *ϵ-greedy* strategy). Formally, the exploration strategy will select the next action a_t among A from state s_t to take by following the rule

$$a_t \begin{cases} \underset{a' \in A}{\text{Argmax}}(Q(s_t, a')) & \text{if } v > \epsilon \\ \underset{a' \in A}{\text{Rand}}(a') & \text{otherwise} \end{cases} \tag{6.3}$$

Update rule: Lastly, after each action a_t is performed and a reward r_{t+1} is received, the Q-matrix is updated according to the following rule:

$$Q(s_t, a_t) \Leftarrow (1 - \alpha)Q(s_t, a_t) + \alpha[r_{t+1} + \gamma(\max_{a'} Q(s_{t+1}, a'))] \tag{6.4}$$

where α is a *learning rate* (between 0 and 1) that controls how much the current action should be considered to update the model. Finally, to exploit the optimal policy to reach the goal, the agent in a state s should take the action a for which $Q^*(s, .)$ is maximum.

Improving reinforcement-learning exploration

We now examine the possible exploration strategies to make reinforcement learning techniques converge rapidly. The worse case is the one of a pure random exploration. In this case, the value function converges to the optimal one, but very slowly. The ϵ-greedy technique takes advantage of the current value function to favor exploration in areas that have been the most rewarding. ϵ is then an arbitrary threshold to balance exploration (by random selection) and exploitation (by value function selection).

Several improvements have been proposed to further speed-up exploration while ensuring the convergence of the value function. A very basic, yet efficient approach is the "optimism towards uncertainty", also known as R-max algorithm [255]. This technique simply favors areas that have not been visited enough, thus ensuring a rapid coverage of the whole exploration space. Other techniques such as the *bayesian exploration bonuses* [256] follow a similar approach by rewarding state-action pairs inversely with their visitation counts.

The concept of intrinsic motivation has been proposed to speed-up reinforcement learning problems. Chentanez *et al.* [252] have proposed a technique where the learning agent is attracted by "salient events", while Lopes *et al.* [257] have use a strategy following learning progress. In these approach, an *intrinsic reward* is added to the exploration scheme and biases the value function to make the agent focus more on areas that really matter. This intrinsic reward decreases during exploration, thus avoiding any bias in the final policy. The field of intrinsic motivation is actively studied in a developmental robotics context and is further discussed in Section 6.4.2.

In our work, we only use reinforcement learning as a mean to solve a planning task. Therefore speeding up RL convergence is not our first priority. Nevertheless,

we also use a type of ϵ-greedy displacement rule, and our problem initialization strongly resembles the R-max approach.

6.3.3 Model-oriented exploration

In model-oriented exploration, the goal is to learn something from the environment by building predictive models. The approach in this case is to collect sensory information, turn it into exploitable data, and apply machine learning techniques to make inferences about the world. The exploration strategy is then oriented towards finding samples, able to build a good representation. The question of speeding up exploration to learn faster is another aspect considered when designing exploration strategies in that regard. This kind of exploration may be considered as a type of active learning problem, also known as the optimal experiment design in statistics [258].

Applications

Model-oriented exploration is used in number of applications. In mobile robotics, one can for example think of learning the visual aspects of the environment to predict loop closure [99], or to predict what unexplored areas may look like based on a limited number of evidences [259]. Learning object representations in an open-ended manner [117], [260], or trying to predict the localization of objects based on contextual input data are other possibilities [261]. Those models could be learned without any particular exploration strategies (by randomly sampling the environment for example), but very slowly and potentially poorly in complex environments or high dimensional spaces. Exploration strategies in these types of setups are of two kinds.

Exploration following extrinsic criteria

The first category centers exploration towards elements of interest, relevant for what needs to be learned. As an example, for the task of learning object representations, saliency maps are a good mean to isolate objects and guide exploration [133]. Very often approaches used for memory-based exploration also reveal efficient for model learning [259]. This is mainly because elements of interest are often the same in both cases, the only difference being the way they are analyzed by the system and integrated to the internal representation.

Exploration following intrinsic criteria

The second category of exploration chooses actions based on the internal state of the model. For example, if two distinct objects have a very close representation in the internal model of the robot, a good approach would be to sample additional data from these objects and increase the discrimination capacity of the model [117]. An early implementation of this idea was developed by Schmidhuber [262] in the scope of artificial curiosity, who proposed to guide the robot's exploration by looking at the prediction error of the neural network trying to learn a model. This kind of behavior cannot be obtained from the first type of exploration, unless of a hand-designed strategy dedicated to this goal. As this type of exploration is based on the internal representation of the knowledge, it is much closer to the way biological

exploration is conducted. This exploration technique is actually at the heart of developmental robotics, relying on intrinsic motivational systems. These aspects are more deeply described and discussed in the next section.

6.4 Exploration in developmental robotics

This section presents autonomous exploration from a developmental robotics point of view. We first provide an overview of the field, main goals, challenges and areas of research. We then focus on the problem of motivations to guide the exploration, and more specifically the use of learning progress as an intrinsic reward.

6.4.1 An overview of developmental robotics

Background

Developmental robotics, also called epigenetic robotics, aims to study the mechanisms that would be able to provide a robot with a lifelong and autonomous learning skills or knowledge. Pioneering thoughts on developmental robotics were formulated by Alan Turing himself [1]: "instead of trying to produce a program to simulate the adult mind, why not rather try to produce one which simulates a child's? If this were then subjected to an appropriate course of education one would obtain the adult brain"

Developmental robotics then suggests to find inspiration from human and animals, and from the way they gain knowledge of increasing complexity. Many works, either in biology or psychology, study the development and learning of humans [263]. Developmental robotics builds on those approaches, ranging from infant behavior to cortical structures involved in learning.

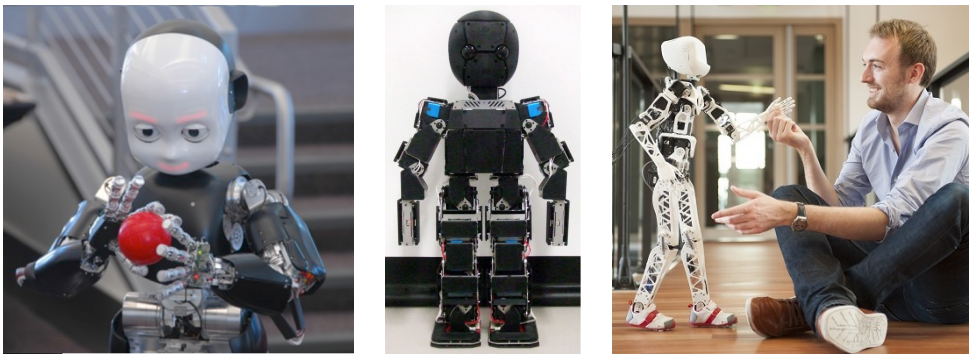
This is not a one way inspiration, as developmental robotics also aims to validate theories about development and learning. Applying a developmental psychology theory on a robot usually requires a deep reflection and understanding to make operational a weakly formalized theory. Implementation on a robot then allows a deeper study on original aspects.

Although Turing's ideas date back to the 50's, no concrete investigation was done in that regard for almost 40 years. Schmidhuber proposed in the 90's preliminary ideas on the way a robot should explore based on artificial curiosity [262]. However, the field of developmental robotics has really emerged in the late 90's with Weng's work on *autonomous mental development* [11].

Today, developmental robotics strongly builds upon the fields of artificial intelligence or machine learning, and widely overlaps cognitive robotics and computational neuroscience. It has emerged at the crossroad of those fields following the idea that development and learning is the result of interactions between the brain, the body, and the environment.

Goals and research in developmental robotics

Goals and challenges: Developmental robots should be designed so that a certain number of behaviors can emerge. A few robotics platforms have been proposed as developmental robotics frameworks in that regard, such as the iCub robot [12], Erato's baby robot [264], or the opensource Poppy robot [265] (See Figure 6.7). More specifically, engineers are interested in the following components:



(a) (b) (c)

FIGURE 6.7: Example of developmental robotics platforms. (a) iCub, (b) baby robot, (c) Poppy

- Building a task-independent learning mechanism that could be applied to any type of tasks, even the one that are unknown to the robot's designer.
- Build a system able to tackle *lifelong learning* in an open-ended manner. This implies that robots should be able to acquire new skills continuously during their whole operation time, in an unconstrained world.
- Acquire skills of increasing complexity, and use the previously learned knowledge to acquire them.

One of the main difficulty is to equip robots with a developmental plan that is not manually designed, and to let the robots decide by themselves the type of activity they should follow. This is a very challenging issue, as programmers should accept to be removed from the learning process and rather focus on how robot can select relevant tasks and learn from them. On the other hand, letting robots decide on their own also raises ethics and safety issues that should be considered by both programmers and users.

Lastly, a consequence of making a robot learn in a developmental way is the emergence of subjectivity [266]. What robots actually learns directly depends on what they have been experiencing. As a results, two robots with the same architecture may not have the same world representation in the end. This may be a problem in terms of performance evaluation, as there is no way to warranty that the system learns a model that is well-suited for the tasks it was trained for.

Research directions: As one of the main challenges of developmental learning is to learn a growing number of skills in unconstrained environments and within a limited lifetime, many research investigate how to constrain exploration to make learning possible and efficient. A first approach is the use of motivations able to drive the robot's exploration towards relevant areas of the space [13]. Motivations are defined by means of rewards, intrinsic or extrinsic and are further described in Section 6.4.2. An emblematic research in this field is the *Intelligent Adaptive Curiosity* (or *IAC*) proposed by Oudeyer *et al.* [13] and has been evaluated on a AIBO robot discovering how to interact with its environment by playing on a baby playground (the setup was called the *playground experiment*, see Figure 6.8). IAC was re-used and improved by many research work [22], [117], [190], [267], [268] and is the fundamental mechanism of the work described in Chapter 7.

FIGURE 6.8: The playground experiment (image from [13])

A second way of guiding the robot's exploration and learning is to study social guidance. In this context, a human user [117], or a peer [269] has the role of a teacher that guides the robot for learning desired skills.

A third area of research is related to the way skills and knowledge should be represented and should evolve in time [270]. The question of how to create high-level abstractions of skills to be used in more complex applications is another critical point [271].

Lastly, maturational constraints is also a way to efficiently guide exploration accordingly with the robot's skills. Instead of providing the robot with a fully operational body and brain, it is possible to progressively enable higher degrees of freedom, or better representation capacities to help the robot focus on simple tasks first [272].

6.4.2 Motivation and rewards to guide exploration

A challenge of developmental robotics is to make robots learn skills by exploring a complex environment, with many degrees of freedom for exploration. To be able to learn in a reasonable time, exploration should be guided by mechanisms able to engage and disengage attention towards areas of the space where learning is possible, and keep trying until a given skill is acquired. This process of engaging the implication for a given task is called *motivation*.

Intrinsic and extrinsic motivation

Motivation is guiding our everyday life's action, whether we need to learn something, or just accomplish a task. Ryan and Decy [273] have distinguished two types of motivations, namely *intrinsic motivation* and *extrinsic motivation*. Considering Ryan and Decy's work:

"Intrinsic motivation is defined as the doing of an activity for its inherent satisfaction rather than for some separable consequence. When intrinsically motivated, a person is moved to act for the fun or challenge entailed rather than because of external products."

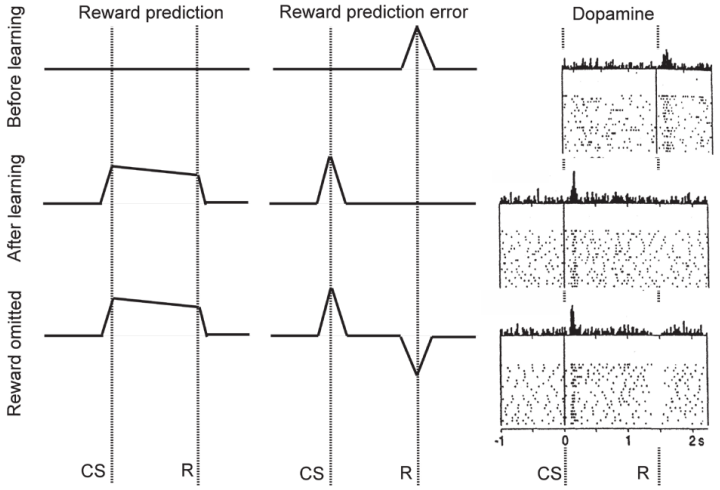


FIGURE 6.9: TD learning prediction error reward and dopamine emission. (image from [274])

Children are perfect example of intrinsically motivated living beings. They have the faculty to engage in an activity for no particular purpose. Indeed, young children are constantly trying to grasp, shoot, bite or squash any new object of their environment. They may be deeply involved in interacting with an object, and, for no external reason, suddenly drop it and find another exciting focus.

Intrinsic motivation is opposed to extrinsic motivation, defined as

"Extrinsic motivation is a construct that pertains whenever an activity is done in order to attain some separable outcome. Extrinsic motivation thus contrasts with intrinsic motivation, which refers to doing an activity simply for the enjoyment of the activity itself, rather than its instrumental value"

Extrinsic motivation is then associated with any activity where an external goal must be achieved. In robotics, systems relying on reinforcement learning with a simple reward function are typically extrinsically motivated.

Motivation, whether intrinsic or extrinsic, is strongly related to the notion of *reward*. Schlutz *et al.* [274] have revealed that a strong correlation exists between the reward function used in reinforcement learning and the dopamine emission in the midbrain (See Figure 6.9). Although this was demonstrated for extrinsic kind of reward, additional studies suggested that dopamine could also be involved in intrinsic motivation [275].

Lastly, there is a major difference in the role of intrinsic and extrinsic reward in the learning process. In reinforcement learning (extrinsic reward), the reward is the only learning signal provided by the environment, and this learning signal is entirely integrated into the acquired knowledge (for example, the value function providing a policy to follow). Intrinsic rewards are rather used as a guide for exploration and are not part of the model in the end.

Exploration strategies based on intrinsic reward

The degree of motivation towards an activity in the process of learning can be highlighted by the *theory of flow*. Figure 6.10 represents the mental state associated with a skill level and a challenge level. To reach the full state of concentration required for efficient learning conditions, the activity should be both challenging and just

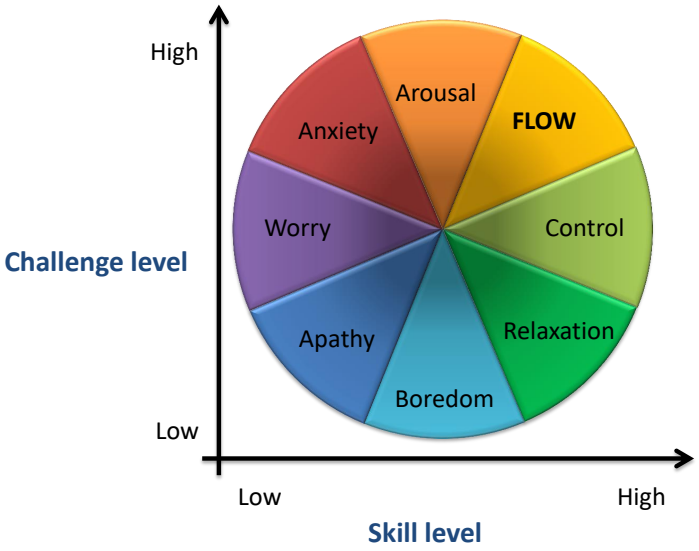


FIGURE 6.10: Stages of the theory of flow

above the skill capacities of the learner. Most intrinsically motivated exploration techniques then rely on defining how challenging an activity is likely to be.

Many types of intrinsic motivation: The list of intrinsic motivation strategies used in the literature is huge. Among them, the traditional *novelty* [255], [276], *surprise* [252], [277], or *uncertainty* [255], [278] are widely used. However, more exotic rewards may be attributed to *information gain* [273], *confidence* [279], *progress* [13], *curiosity* [262], *interest* [280], *achievement* [281], *saliency* [252] (not to be confused with visual saliency of Part I), *diversity* [282], or *familiarity* [283]. Many of them have been reviewed and compared in recent surveys [284], [285]. However, these denominations should be used very carefully as they are sometimes used by different teams under very different mathematical definitions. To our knowledge, there is no generally accepted definition (mathematically speaking) for each of these terms.

Many implementations of those strategies are in fact heuristics aiming at biasing the exploration towards areas that are potentially interesting. For example, *novelty* would reward states that have not been recently visited [255], *surprise* (or *contextual novelty*) targets attention towards salient events [277], and *uncertainty* is related to the high variance or entropy in states or actions [278]. This kind of approaches are efficient in the sense that they may speed up exploration by focusing on interesting areas only, but there is no warranty that learning may be possible. Regarding the theory of flow, they are focused on finding challenging areas, and do not really consider the skill level of the agent. They are most of the time similar to exploration techniques described in Section 6.3.1, except that they are examined under a neuroscience point of view (the term "reward" designating in fact the criterion to maximize in Section 6.3.1).

Prediction error and learning progress: To be consistent with the theory of flow, an evaluation of the skill level is required. For that, a possible measure is the so-called *prediction error* [13] (sometimes also called *novelty* [276] or even *surprise* [252]), evaluating the difference between what the robot is predicting, and what is actually measured by sensors. Though, a strong limitation of the prediction error is that the robot may be stuck in unlearnable situations, especially in the presence of noise. Focusing only on areas with a high level of prediction error may then have disastrous consequences on learning.

Another approach, early suggested by Schmidhuber [262] and implemented by

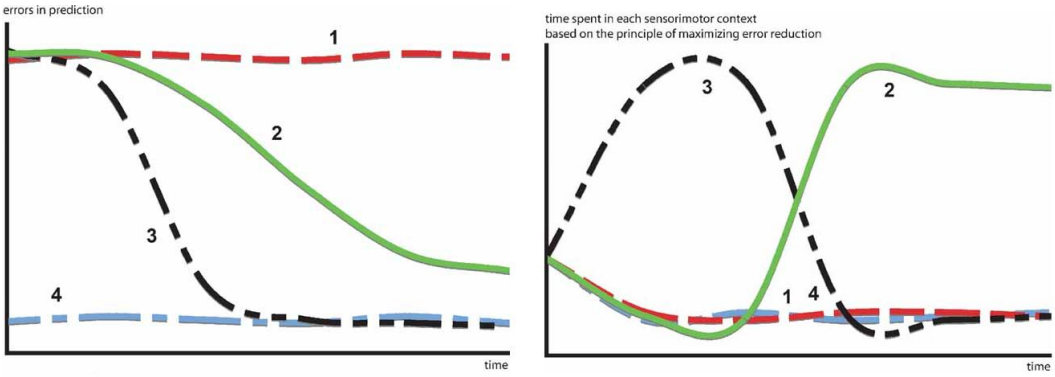


FIGURE 6.11: Prediction error of 4 different activities, and time dedicated to each of them by following the progress (image from [286])

Kaplan and Oudeyer [13], [283] in a developmental learning context suggests to rely on the *evolution* of the prediction error rather than current state of this error. This evolution can be seen as a sort of derivative and is called the *learning progress*. Progress is particularly interesting as it avoids both trivial (low error) and unlearnable (high error) areas of the space, while focusing on tasks that are neither too easy, nor too hard. Progress also has interesting properties on the order tasks should be accomplished, starting with easy ones (where error decreases quickly) and progressively increasing their complexities (having a slower error decrease), thus being consistent with a lifelong learning scenario. Figure 6.11 illustrates this behavior by considering the time spent in 4 activities based on their prediction errors. Activities 1 and 4 are trivial and unlearnable, so that the robot only spends time on them at the beginning of the experiment. Then, activity 3 is prioritized as learning progress is the highest. When the competence associated with activity 3 is acquired, learning progress becomes low, and focus is concentrated on activity 2, whose progresses where lower.

Learning progress is then a particularly interesting intrinsic motivation. In IAC, a measure of learning progress is proposed to drive the robot's exploration. We then rely on this measure in our approach.

Visual attention and intrinsic motivation

Visual attention is an excellent study case for intrinsic motivation. In a general case, eye saccades and fixations can be seen as a way to actively sample information. This sampling is generally oriented to reduce uncertainty in future states of observation, which can be seen as a form of intrinsically motivated exploration strategy [287], [288].

In neuroscience, several studies have highlighted the fact that eye movement patterns are not the same when trying to learn a skill, and once this skill is acquired [289]. This suggests that saccades are used as a mean of exploration. Moreover, studies on monkeys have shown that the brain is weighting visual information based on the collected reward they convey [287]. This type of weighting approach strongly resembles saliency map computation approaches.

In developmental robotics, visual attention is also considered as a common study case, most of the time used as a way to develop proprioceptive skills (predict the position of the hand in a field of view [267], head orientation toward a light [283]) or visual servoing skills (learning options from visual inputs [290],

smooth pursuit [291], simultaneous gaze control and reaching [292]). However, these methods mainly focus on developing motor skills rather than visual skills. For example, the discovery and learning of what should be salient in a environment has not been examined so far in a developmental robotics framework, although clear evidences show that such saliency differs from age groups [293], [294] and culture [295]. It is therefore, at least partially, learned.

6.4.3 Implementations of progress-based exploration

In this section, we highlight the main components of existing implementations of learning progress-based exploration strategies. For an alternative classification and formalism of the problem, one could refer to the strategic student problem formulation [296].

Architectures

Similar to any robotics system trying to learn from the environment, systems relying on progress for exploration need at least inputs sensory data to produce, when learning is finished, an outcome that is to be used in future tasks. Another common element of those systems is that they learn to make predictions about the external world. This prediction process is performed by a *learner*, an internal learning machine (classifier, regression system, POMDP, *etc*...) predicting output states Y given input signals X.

A general internal architecture of intrinsically motivated systems based on learning progress has been described by Oudeyer *et al.* [13] and is represented in Figure 6.12. The *learner*, is the central element of the architecture, learning to make predictions. The learner is iteratively updated by the *input signals*, and produces from each input signal an *output state* prediction ($\tilde{Y}$ in the figure). The *prediction error* E is obtained by measuring the distance with the learning signal (Y in the figure). Second, a *meta-learner* learns to predict the error in prediction of the learner. This time, the meta-learner receives successive prediction errors E as a learning signal and tries to predict how this error will evolve in a near future. Lastly, a *knowledge gain assessor* receives the estimated future prediction error E' and considers past E and future E' to compute the progress measure.

Lastly, the progress measure is transferred to a controller to derive a policy and guide the future robot's actions.

Sensory and proprioceptive signals: To make its own world representation, the robot perceives and analyzes both sensory and proprioceptive signals. All or part of them will constitute the input signals (X) of the learner. In vision-based systems, these predictions are made from raw [22], [127], [290], [297] or transformed [13], [283] visual signals providing a higher level information (for example, the position of a spotlight in the field of view). When learning sensorimotor skills [13], [190], proprioceptive signals (such as motor commands) are more likely to be used as input signals. Lastly, those signals are sometimes discretized before being processed by the learner, to simplify their representation [257].

Learning signal: Then, for the learner to learn something, a *learning signal* is required. This learning signal should be seen as a ground truth to be compared with the output state Y the learner is predicting, and should be removed from the prediction process once learning is finished. These learning signals may be obtained by other sensory data [22], temporal consistency of successive observations [127]

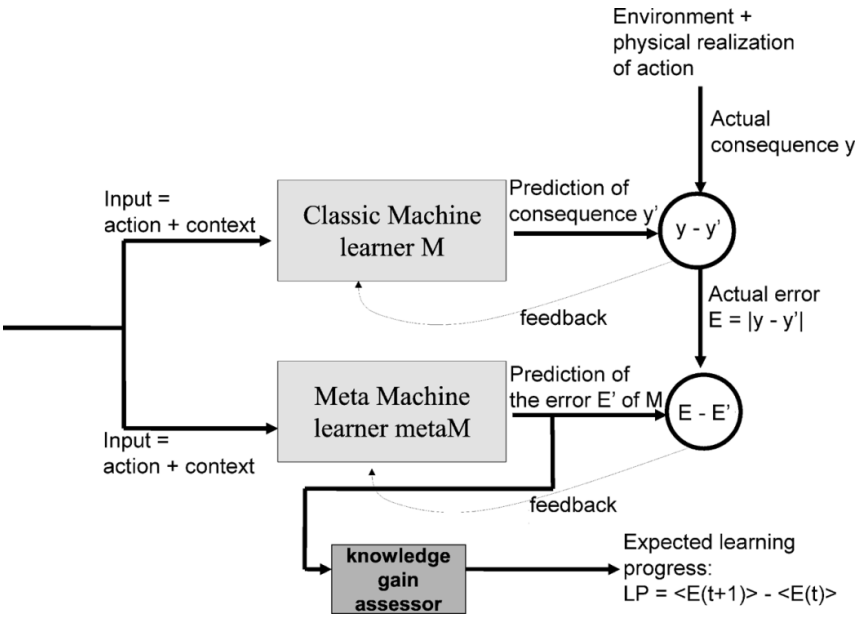


FIGURE 6.12: Global architecture of progress-based motivation systems, from [13]

(two successive views contain the same object), sensory measurements after a particular event occurred (for example, after performing an action [257], [290], [298] or after discovering a salient event [252]), or based on an external assistance from an oracle [127]. Note that this learning signal may be different than what the system should learn in the end. For example, if an intrinsically motivated system is using extrinsic reward to learn a policy, the learning signal associated with the learner would just be a hint to guide exploration, not what the system should learn in the end [257], [270].

System outcome: The *expected outcome* of the system should be the final product of what it was trained for. A sensorimotor mapping in the case of forward and backward models [190], [298], classifiers in the case of recognition systems [22], [127], abstraction libraries of sensorimotor primitives [290], or optimal policies for problems of goal reaching [257], [297]. Most of the time, the outcome of such systems is the learner, but it sometimes happens that the learner is just here to guide exploration for a better outcome [257], [290].

Learning progress implementation

A local evaluation: The idea behind progress is to guide exploration towards areas where the prediction error decreases fast. However, simply measuring the prediction error (and consequently, learning progress) at a global scale does not provide any information about areas that should be explored. It is therefore extremely important to provide a progress estimation at a local scale. A critical point is then to find a way to estimate the progress by grouping similar examples and providing a local measure for each group. Exploration will then favor examples in a group having a high measure of learning progress. These groups are called *regions* in Oudeyer's work [13]. Finding the space where those regions should be defined has a strong influence on the efficiency of the method. In a discrete world representation with a small number of states, this may be done by considering the progress in each unit (state-action for example) of the environment [257], [270]. In continuous

environment, regions may be defined in the motor or sensorimotor space [13]. Local estimation may also be part of the learner internal representation, considering for example the activation units of a self-organizing map [299]. For learning sensorimotor skills, Baranes *et al.* [190] have shown that defining regions in the tasks space, or goal space was much more efficient than considering regions in the motor space. This finding was consistent with the goal babbling approach proposed by Rolf [300]. Another way to consider regions is to see what the robot should learn as a set of problems (identifying several objects [117], using several tools [268]). A single region is then associated for each of these problems, and the capacity of the robot to solve each of them is used to estimate learning progress.

Prediction error measurement: The *prediction error* measure is easily defined as a distance (euclidean, for example) between the prediction and the learning signal when the learner is making a regression [13], [190], [299]. In classification mode, distances may represent probabilities of the correct class [117] or be based on a set of classification output to produce an average accuracy (in our system, we use a more exotic measures, the F1 score, that is also a classification score). If the goal is to build a set of representative features, a distance to the nearest representant is appropriate [270]. Lastly, in [257], a cross-validation measure was used to compare an estimated and a measured distributions.

Progress measurement: Lastly, once prediction error is evaluated, an estimation of the learning progress must be produced. The general idea is to build a predictor that estimates the next prediction error, and to consider progress as the difference between $E(t)$, the actual error and $E'(t+1)$, the estimated error at next step. Early approaches were using a meta-learner with complex prediction techniques (for example with neural networks [262]) to estimate $E'(t+1)$. However, the progress can be seen and defined in a more natural and simple way. Kalplan [283] suggested to consider, for guiding exploration, the *evolution* of the error measure rather than the error itself. Mathematically, the evolution corresponds to a variation, a sort of derivative of the error. If the error is quickly decreasing, progresses are high, and a definition of the progress may be the inverse derivative of the error $-\frac{dE(t)}{dt}$. Based on this idea, progress is then usually defined from a sort of linear regression over the last few local examples [22], [290]. Most of the time, progress is simply obtained by [190], [298], [299]

$$progress(t) = \sum_{k=t-\tau}^{t-\frac{\tau}{2}} E(k) - \sum_{k=t-\frac{\tau}{2}}^{t} E(k) \tag{6.5}$$

where τ is a time scale used to segregate recent and older events. An average over a few samples is preferred to a single example, as there is often a high variance in successive error measures.

Exploration policy

Last but not least, we raise the question of how to integrate progress estimation in the exploration strategy. To better understand the different proposed approaches, let us recall the basic exploration sequence of intrinsically motivated systems.

1. Initially, the robot is in a given state and a given region (that may or may not be the same object)

2. The robot takes an action and arrives in a new state (and potentially a new region)
3. From this state, the robot gets sensory (or sensorimotor) data and learning signals
4. This data is used to update the learner
5. The update leads to a new local progress estimation in the region
6. Based on progress estimation and environment constraints, the robot computes a new policy
7. The robot follows this policy to select the next action to take.

The policy then depends on the estimated progress, but above all, on the type of actions the robot can take.

In the case where any state can be reached from any other state by a single action, and if all of these actions have a roughly similar execution time, there is no cost associated with reaching a given state. In this context, the general idea is to focus directly on the most progressing region by using an ϵ-greedy policy [13], or probabilistic rules based on the proportion of learning progress [117], [267], [299]. These approaches were shown to provide a set of example that is quasi-optimal, under the condition that learning functions are submodular [296].

When reaching a particular state requires a sequence of actions, the problem must be seen as a Markov decision process. Reinforcement learning may then be used to determine the policy to follow. In this case, the learning progress represents another source of reward and is integrated in the policy calculation as it would be done for extrinsic reward. Q-learning [257], [262], or model-based least square policy iteration [270], [297] are typical algorithms used to find these policies. In this context, robots are able to explore their environment without any extrinsic reward.

In our system, we use two types of configurations to determine the policy. The first one follows an ϵ-greedy policy, and is not well-suited for all of our setups. We then build a second policy based on reinforcement learning.

6.5 Summary and conclusion

In this chapter, we have examined the problem of exploration in both mobile robotics and developmental robotics. Although having different constraints and goals, and using a different formalism, many overlaps can be found between those fields.

For mobile robots, exploration is usually constrained by the robot's displacements that need to be safe and efficient. The navigation problem is typically formalized on a map in which the robot is localized. Path finding techniques are then considered to determine a good trajectory fitting the environment constraints. Exploration usually tries to optimize trajectories, either for map coverage, goal reaching, or models learning.

Exploration in developmental robotics is focused on the learning and skill acquisition process in a lifelong perspective. The field of intrinsic motivations to drive the robot's decisions and constrain exploration is very active.

The next chapter is dedicated to the description of exploration strategies improving the quality of our models of saliency. Our approach is based on the IAC algorithm that we applied on a foveated system (*BioVision*) and on a mobile robot.

IAC has not been applied before on a mobile platform, and is not perfectly adapted to mobile robotics constraints. The junction of these two field then conducted us to the creation of a new algorithm, RL-IAC, that is one of the main contributions of our work.

Chapter 7

Proposed approach

7.1 Introduction

In the scope of an autonomous, life-long exploration and learning, the robot must be equipped with an exploration strategy able to drive the robot's displacements. This one should allow the robot to collect interesting examples, while avoiding problems such as *catastrophic forgetting*. In the meanwhile, exploration should be organized and efficient, so that the time spent traveling across the environment to reach goals is minimized.

We studied in Part I the capacity of a robot to learn a model of visual saliency on the fly during exploration. This approach has proved to be efficient when a sufficient number of samples are provided to feed the model. However, nothing is specified in this approach about the way samples should be collected and how representative they should be of the environment. We investigate here the strategy the robot should apply to visit relevant places to learn saliency efficiently. The previous part presented an open-loop framework for learning such a model, and this part is a closed-loop framework to refine and improve the learning capacity of the system.

We design our method based on the *Intelligent Adaptive Curiosity* (IAC), an algorithm able to guide the robot's actions depending on its current state of knowledge, to make it learn better and faster. We first explain how to apply such a system to the task of online saliency learning. Then, we show the limits of this algorithm when applied to a mobile robot exploring a building, and propose an extension of this approach. This extension, called *Reinforcement Learning-Intelligent Adaptive Curiosity* (RL-IAC) is now able to find good trade-offs between the time spent taking actions, and the time spent for learning and improving the internal knowledge of the robot.

7.2 Overview of the method

7.2.1 Position versus state-of-the-art

In this section, we make a link with Chapter 6 by positioning our work versus state-of-the-art. For more clarity, this positioning is sometimes discussed directly within the corresponding sections. In this case, we simply make references to these sections.

Regarding visual attention

A major difference between our approach and traditional visual attention systems is the way we consider salient regions. Our purpose here is to consider saliency as something to learn rather that something to direct attention towards. We then direct

our attention towards areas able to improve learning rather than towards salient elements themselves. As discussed all along this chapter, our focus of attention is focused at highly progressing areas.

Second, some visual attention techniques rely on a reinforcement learning setup to learn a sequence of saccades and displacements [49], [181]. We here learn some visual aspects of our environment in a reinforcement learning context, but the model learned in the end is completely independent from any action (saccade or displacement) of the robot.

Regarding mobile robotics

When considering experiments with a mobile robot, we rely on a SLAM algorithm [25] that we consider reliable enough to position the robot. This SLAM techniques builds metric maps and localizes the robot in terms of position and orientation on this map.

In Section 7.4.3, we rely on topometric maps, and provide a possible approach to build these maps incrementally. We also make in Section 7.4.2 a positioning of RL-IAC with respect to classical path finding approaches. More precisely, RL-IAC relies on path finding techniques that are not directed towards external goals, but towards intrinsic ones. Goals are then chosen to optimize learning rather than area coverage. Our path finding is also based on a Q-learning approach, able to find a good policy given some learning quality measurements.

Regarding intrinsically motivated approaches

Lastly, our method relies on intrinsic motivation and is an implementation of the IAC algorithm. The main differences with other IAC implementations are discussed in Sections 7.3.1, 7.3.2, 7.3.2 and 7.4.2. We also propose two main contributions in that regard.

The first is a measurement of the learning progress for the case where the learner is shared among different regions, that we call the *backward progress estimation*. More importantly, the second is an extension of IAC, containing a more general action policy that the one originally proposed. Based on this extension, IAC is now usable on mobile robots exploring large environments and spending a lot of time in displacements across the building. We call this extension *RL-IAC*.

7.2.2 Mechanisms and contributions

General framework

IAC can be seen as a way to guide the robot's actions when exploration is dedicated to the task of learning something. Unlike exploration strategies whose aim is to have an extensive coverage of the exploration space, IAC focuses on particular areas of the space so that learning is done efficiently. The essential component of this technique is a local measure of the learning progress, that catches the attention of the robot until knowledge has been acquired. The general architecture and different components of such approaches were described in Section 6.4.3. We here focus on describing the mechanism in our particular framework.

IAC is then constituted of four main components:

- a *learner* that learns a particular model;
- a way to divide the exploration space into *regions*;

- a *meta-learner* that monitors the learning evolution in each region and estimates progresses[1];
- a *policy* that determines the next action to take given the state of progress in each region.

In our implementation, the learner is the online classifier used in the saliency learning technique described in Chapter 4. The goal is to be able to predict, given an RGB image, the portions of the image containing salient objects. To this end, the saliency technique uses a *supervision signal* (also called *learning signal*) that provides, when possible, the state of saliency of a pixel of the image.

The exploration space strongly depends on the problem and varies a lot in the different implementations of IAC. The only constraint is that regions should be reachable by the different actions the robot can take (meaning that taking an action leads the robot to a given region), and wisely divide the space to locally estimate learning progress. In our case, regions are defined in the low dimensional space of positions the robot can reach. Unlike early approaches proposed by Oudeyer *et al.*[13] where regions are incrementally divided to refine learning progress estimation in relevant areas, regions are, in our case, separated along a regular grid. They are progressively discovered and updated during the experiment.

The meta-learner is the module that estimates the learning progress in each discovered region. To evaluate learning progress, the learning error (the difference between the supervision signal and the model prediction) is evaluated. In our approach, the supervision signal is the segmentation mask for pixels that have been segmented "*salient*" or "*not salient*", and the model prediction is simply the estimated saliency map. To obtain the learning progress, we consider the history of the error in each region as a function of time, and we define the opposite slope of this function as a progress measure.

Lastly, based on the learning progress estimation, the decision of the next action to take is determined. In our setup, an action is a displacement of the robot that leads to a new point of view, and, as a result, a new *observation*. The basic idea of IAC is to prioritize actions towards the most progressing regions. To this end, these techniques typically use a greedy approach in the regions with highest learning progress. We first follow a similar approach and highlight its potential inefficiency when applied to a mobile robot. We then propose an extension of the IAC architecture, that we call RL-IAC (for *Reinforcement Learning-Intelligent Adaptive Curiosity*) containing an additional module able to follow learning progress while being still efficient when a greedy policy fails.

Figure 7.1 summarizes the main components of the architecture and the way they interact with each other. We here provide a brief recall on the basic procedure for using it. At time t, the robot receives observations from the main and additional visual streams, as described in Section 7.3.1. They are, on the one hand, turned into a segmentation mask S, and, on the other hand, sent to the online classifier to estimate the saliency map $\widetilde{S}$. Then, the learner and meta-learner are both updated: S is first turned into labels and sent with the extracted features F to the online classifier (learner) for a model update. Second, as this observation was taken at a particular point of view, in a region R, the prediction error $\|S - \widetilde{S}\|$ is added to the error history of R, and the progress in region R is re-evaluated. Lastly, the next action of the robot is calculated from a policy depending on the learning progress.

[1] For more simplicity, we consider the meta-learner and the knowledge gain assessor as a unique module, both monitoring error and deriving progress.

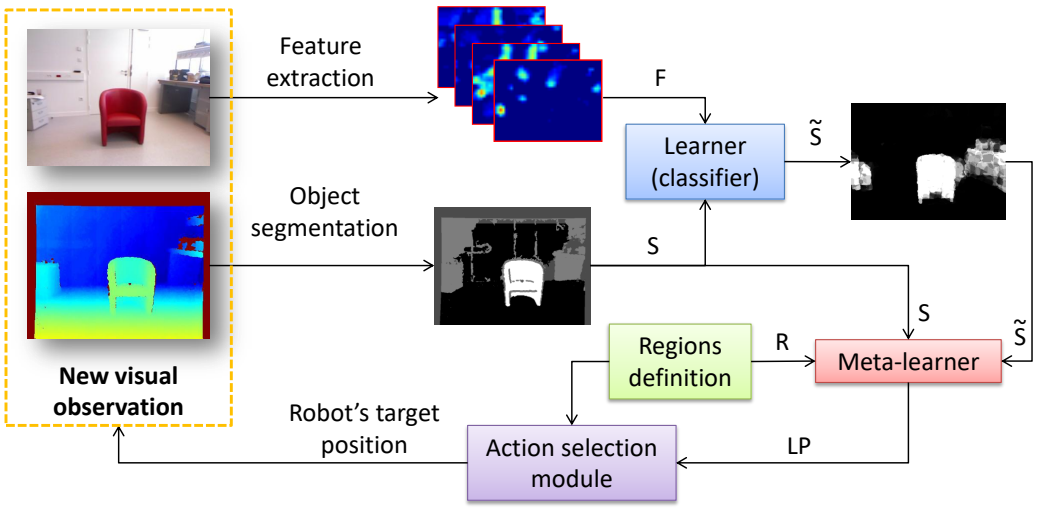


FIGURE 7.1: General architecture of the IAC algorithm for saliency learning.

The robot then takes the action, ends up in a new position and receives an new visual input that is to be processed, and so on.

Two application scenarios

The implementation of IAC in the scope of saliency learning has been tested on two different types of robots. Although based on the same mechanism, the exploration does not have the same constraints, and the process of action selection to guide exploration is different.

The first robot is *BioVision*, described in Section 2.2.1. In this case, the robot is an anthropomorphic head put on a table, watching by fixations and saccades at a static scene. The only actions the robot can take are motor commands modifying the pan and tilt orientations of the head.

The second robot is a mobile robot described in Section 2.2.1. This time, the robot is able to move across the rooms of the experimental environment. Actions taken in this case are based on reachable positions on a 2D map representing the environment, and the orientation of the robot within the environment.

In both scenarios, exploration is made possible by displacing the robot's field of view to a new position. Thus, the salient elements to be observed are not the same, and can be integrated to the model. Action then consists in sending commands to reach a particular point of view. In both scenarios, the learner is the online classifier, and the meta-learner calculates learning progress in a given region the same way. Section 7.3 is dedicated to the description of the components of IAC that apply to both setups.

The major difference between the two scenarios is related to the nature of the actions. For *BioVision*, reaching a position of the pan-tilt space is relatively fast, whatever the current position of the robot. Actions can be considered negligible with respect to the processing time required for segmentation, saliency computation and model update. For the mobile robot, the size of the environment does not depends on the architecture of the robot, but on the choice of the experimenter. In small environments (such as rooms) where different points of view are reachable by many rotations and little displacements, the same kind of approximation can apply.

However, if the space to consider is a building, the displacement required to reach a position at the other end of the environment may be time consuming and much longer than the processing step. The question of the time for performing actions is one of the main problems of Section 7.4. We there propose RL-IAC, an extension of the traditional IAC implementations in which the time required to displace the robot to a given position is considered. This way, a good trade-off is found between the time spent for updating the saliency model, and the time spent to get a new observation.

7.3 IAC and saliency learning

We provide in this section the implementation details for applying IAC to the case of saliency learning.

7.3.1 Learner

The online classifier described in Chapter 4 that learns the saliency model plays the role of the learner. As most implementation details were provided in this chapter, we only give a brief overview on the way it is used in the IAC framework.

Saliency learning

Our learner tries to construct a prediction function $I \rightarrow \widetilde{S}$, able to estimate the saliency $\widetilde{S}$ of the visual field given an input RGB image I. This process is done by moving in the environment and collecting observations after each displacement.

Each observation is characterized by inputs from the main and additional visual streams. Features F_I are extracted from the main visual stream, and a segmentation mask S is obtained from the additional visual stream to estimate the saliency label of each pixel of the image. Samples are obtained on a pixel-wise basis by associating pairs of features and labels.

These features and labels are used in two different modes, either to update the saliency model (rather called the learner's model in the context of IAC), or to infer saliency maps from the current model. Figure 7.2 presents the saliency learning mechanism and these two modes used in the IAC framework.

After an observation at time $t+1$, a set of samples are extracted from the main and additional streams. The features are first sent to the learner to infer the saliency map $\widetilde{S}$ given the model updated at time t. $\widetilde{S}$ is used to evaluate the learning quality, further discussed in Section 7.3.3. Note that the focus here is not the quality of the saliency model itself, but the way this quality evolves with new observations. Second, both the features and labels are sent to the learner in learning mode to update the model at time $t+1$. Therefore, a new model is available and ready to be used for inference once the update is finished.

Comparison with other IAC implementations

In our context, the learner is designed for a single activity, the one of saliency learning. Additional tasks could be easily integrated by adding classes to the online classifier, but they are not considered here.

Our approach then learns a mapping from a high dimensional space (the number of dimensions being the number of extracted feature maps) to a simple binary

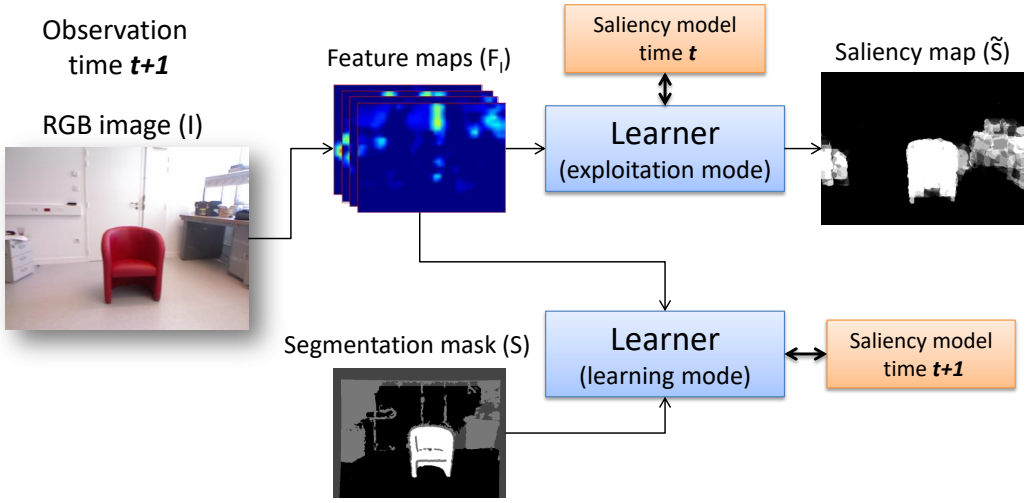


FIGURE 7.2: Integration of the saliency learning module in the IAC framework

classification. What is learned is a pretty low-level signal, learned at the pixel level, without any semantic meaning.

Unlike most techniques where each action provides a single sample, each observation provides hundreds of samples at the same time. The learning progress must then be updated by considering the integration of this large number of samples at the same time. In addition our self-supervision technique provides samples that are only partially supervised to evaluate the progress. This partial observation biases the learner as a large portion of the environment cannot be labeled and integrated to the model. This bias may have consequences on the learning progress estimation, as it is not representative of the whole environment.

7.3.2 Regions definition

Finding appropriate regions to estimate learning progress is an essential element. For each implementation of IAC, regions can be defined in several different spaces. Moreover, regions should be bounded by parameters that are defined prior to the experiment (size of the region, maximum number of samples, splitting criteria, and so on). We also detail in this section the reflection that conducted us to use this kind of regions in our implementation.

Proposed definition

To define appropriate regions, we considered subdivisions of a space containing all reachable visual points of view. Actions are defined in our scenarios by a displacement of the robot from a given point of view to a new reachable one. These points of view are more simply defined by positions in a proprioceptive or exteroceptive frame.

BioVision: For *BioVision*, the robotic head is placed on a table. It is therefore not moving within the environment as a mobile robot would. The only possible way to modify the point of view is to send a pan or tilt motor command so as to change the inclination of the cameras.

Positions of the reachable points of view are then defined as a pan and tilt position (p, t). Regions are defined by cutting the pan and tilt spaces in regular intervals.

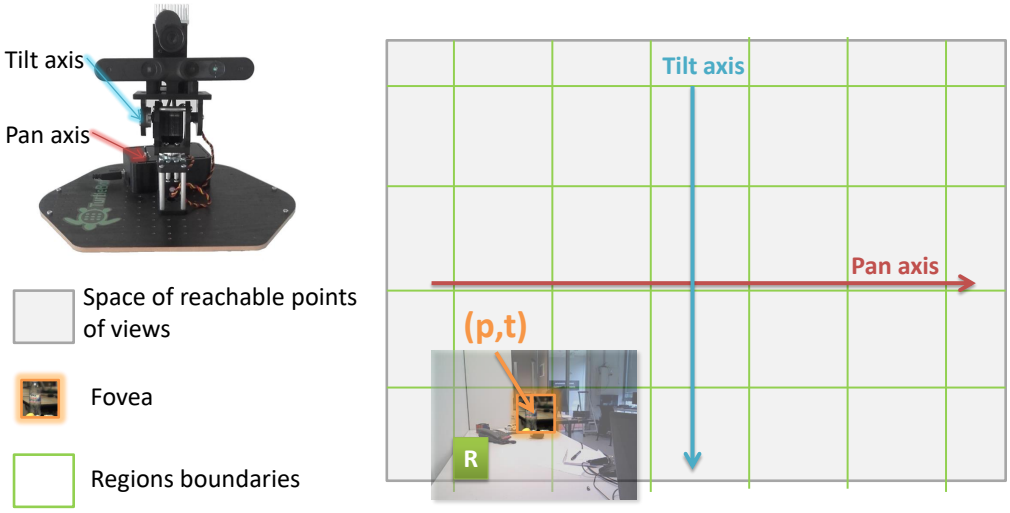


FIGURE 7.3: Regions definition in the pan-tilt space

The robot is then in region R if the pan tilt position falls within the boundaries of this region. Figure 7.3 illustrates the regions boundaries and makes the link between the pan-tilt axes and the foveal point of view.

Of course, pan and tilt positions are not so easily convertible to a foveal point of view. However, we make the assumption that objects are far enough to consider the optical center of the foveal camera to be at the interserction of the pan and tilt axes. Pan and tilt values are easy to obtain and actions are simple pan and tilt motor commands. Our second assumption is to consider each displacement command to fall in the right position which, given the precision of the motor, is realistic.

In practice, we divide the pan positions (ranging from -60° to 60°) into 4 intervals and the tilt (ranging from -30° to 30°) into 4 intervals as well. We then have the space divided into 16 regions.

Mobile robot: A mobile robot has the ability to modify the point of view of its cameras by moving within the environment. We here consider the 2D map space, that is a projection of the environment to the floor. In this space, the robot's position is represented by a triplet (x, y, θ) where (x, y) are the coordinates of the robot on the map, and θ is the orientation. This triplet is enough to define a unique point of view, as the camera mounted on the robot is static. Again, we define regions in this space by cutting each dimension in regular intervals. Figure 7.4 presents the trajectory of a robot in a room, and a possible way to define regions in the map space. Here, the translation dimensions are split in 2×2 intervals, and the orientation in 4 intervals. In total, the space is then divided in 16 regions.

An action in this configuration is given by a position (x', y', θ') to reach from the current position (x, y, θ) of the robot. In our work, we do not consider the conversion of this position into a trajectory avoiding obstacles, and the conversion of this trajectory to motor commands. We suppose these steps as being processed by a reliable plug-and-play module. Similarly, position and mapping obtained by a SLAM algorithm is considered as providing a perfect localization.

As the map of the environment is not evolving so much in time (at least for walls), regions may be defined manually before the experiment from a reliable and static map of the environment. However, in section 7.4.3, we present a way to obtain

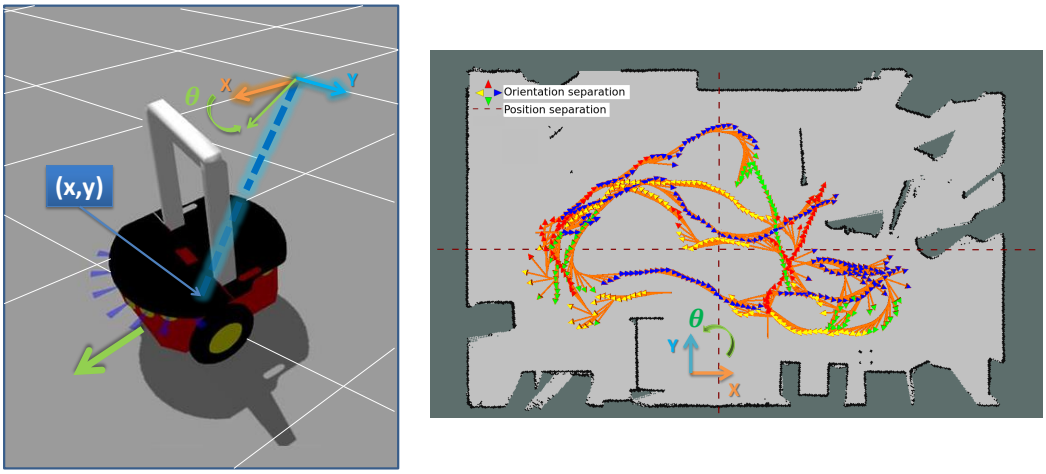


FIGURE 7.4: Regions definition for a mobile robot. On the left, representation of the position of the robot by (x, y, θ). On the right, subdivision of each space for an experiment in a room. Each arrow represents a position of the robot trajectory.

these regions incrementally directly during the robot's exploration. This is particularly interesting when the robot has to explore a large environment (for example, a building).

Remarks: This choice of regions was motivated by the low dimensionality of the represented space (two dimensions for *BioVision*, three for the mobile robot). Because dimensionality is small, cutting the space along a regular grid provides a very simple and safe way to generate regions, while providing consistent groups of samples within regions. In addition, regions boundaries are chosen to be small enough to group similar samples within a region, and large enough to limit the number of regions to explore.

In recent implementations of IAC [117], [268], the algorithm defines regions in a *space of problems* (or *activities*) that are accessible to the robot. We may consider our regions definition as a set of activities as well: In this case, a problem would be the one of learning saliency within a bounded set of reachable points of view. The robot then has to switch between activities by performing displacement actions.

Discussion 1: role of the actions for saliency learning

In many IAC implementations, where the goal is to anticipate the sensor state after performing an action, the learner's model integrates the different actions taken by the robot. For example, if the goal is to predict the position of the hand Y of the robot in the visual field given a motor command of the arm X, then the learner constructs a function $X \rightarrow Y$ based on any action X taken by the robot. In this case, the action space is used both by the learner and the meta-learner. On the one hand, the meta-learner uses learning progress to guide actions. These actions are performed to provide samples to the learner. On the other hand, actions are used directly as inputs for the learner to predict future sensor states.

In our implementation, the robot's actions are just used as a way to modify the point of view. These actions are essential to provide a good panel of visual inputs and construct a representative saliency model, but they are independent from the learner in the inference stage as the learner does not integrate any data related to actions or motor commands. Recall that we try to learn a mapping $I \rightarrow \widetilde{S}$

between images and saliency maps. Instead of using a sensorimotor loop to learn a sensorimotor mapping, we learn a sort of sensori-sensori mapping, where I would be a low-level sensor (pixels are RGB colors without any meaning) and $\widetilde{S}$ a high-level sensor (in the segmentation mask, each pixel intensity represents the interest of the entity - floor, walls or objects - they belong to).

A consequence is that, as visual signals (the input space of the learner) and actions (output space of the meta-learner) are independent, our version of the algorithm is somehow compatible with dynamic environments. If a salient object is moved somewhere else in the environment, it will still look roughly the same in the new position, and is therefore likely to be estimated salient by the learner.

Discussion 2: on defining regions in the learner spaces

The first versions of IAC [13] were using a *motor babbling* approach by defining regions in the motor space, that was the input space of the learner. Other implementations [190] preferred a *goal babbling* approach by defining regions in the goal space, the output space of the learner. It is then naturally interesting to consider the possibility of using the input or output space of our learner to define regions for our problem.

The input space of our learner is related to the RGB image of the main visual stream. A first approach would then split the image space into regions. Given the very high dimensionality of this space, finding relevant clusters is potentially complex and not so appropriate. More realistically, we could consider the input of the classifier that receives feature vectors. The dimensionality is here much lower (39 to 1024 depending on the method), but finding an appropriate distance to determine clusters in this space remains difficult. The few attempts we conducted in that regard did not show any promising result.

But more importantly, selecting regions in the feature space raises a problem for reaching samples. An implicit requirement for defining a region is that the robot should know the action (or sequence of actions) required to get a sample in this region. If not, the policy of action selection based on learning progress does not make any sense. For features describing pixels of an image, there is no direct way to associate a set of features with a set of positions in the exploration space, unless the space has been fully explored and analyzed beforehand. Thus, the image feature space is not well-suited for regions definition in the IAC framework.

As the classifier is binary, the output space only has two states ("*salient*" and "*not salient*"). A goal babbling approach would then select samples that are salient or not salient depending on the progress in each of these categories. However, the same problem of reachability occurs and cannot provide a good action policy.

7.3.3 Meta-learner

The meta-learner aims to monitor the local error made by the learner, and derive an estimate of the learning progress. The local estimation is made possible by grouping and making statistics on samples collected within the same region. Recall that the robot is in region R_i at time t if its current position falls within R_i's boundaries.

We provide in this section two methods for estimating learning progress. The first one is the most natural one, and the most frequently used in general in IAC implementations. We call it the *forward evaluation*. We then demonstrate that this kind of evaluation has some limitations when the learner is shared among regions, and we propose a *backward evaluation* able to overcome this issue.

Forward evaluation of the learning progress

The error made by the learner at time t is computed by comparing the segmentation mask $S(t)$ (obtained from the additional visual stream and standing for the ground truth in this case) with the corresponding saliency map $\widetilde{S}(t)$ (the estimated signal with the current available model). More precisely, from the RGB main visual stream $I(t)$, we extract features $F_I(t)$. In parallel and from the additional visual stream, we compute a segmentation mask $S(t)$. We then consider the *image label set* $L(t)$ by keeping only pixels labeled "*salient*" or "*not salient*" in $S(t)$. We estimate and binarize the saliency response $\widetilde{S}(t)$ for each of these pixels to obtain the *image estimation set* $E(t)$[2]. Lastly, we compute the estimated learning error $Err(t)$ of a particular frame based on Equation 7.1:

$$Err(t) = 1 - F_1(L(t), E(t)) \tag{7.1}$$

where $F_1(.,.)$ is the *F1 score*:

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} = \frac{2tp}{2tp + fp + fn} \tag{7.2}$$

with tp, fp and fn the true positives, false positives and false negatives[3]. We call this error evaluation is called "forward", as the samples used to estimate the error are not yet integrated to the model. The evaluation then considers the prediction of saliency for samples that have never been seen before.

To evaluate learning progress, the meta-learner stores a history of the prediction error for each region. Suppose now that at time t, the robot is in region R_i and makes an observation in this region. Suppose that in this region, $n-1$ observations were already recorded and added to the history from the beginning of the experiment. The observation at time t is then the nth of region R_i, and the learning error $Err_i(n)$ associated with this observation is then added to the history of R_i.

The estimation of the learning progress in R_i, is obtained by exploiting the error history sequence. For that, we apply a linear regression of the error history over the last τ samples:

$$\begin{pmatrix} Err_i(n-\tau) \\ \vdots \\ Err_i(n) \end{pmatrix} = \beta_i(n) \times \begin{pmatrix} n-\tau \\ \vdots \\ n \end{pmatrix} + \begin{pmatrix} \epsilon(n-\tau) \\ \vdots \\ \epsilon(n) \end{pmatrix} \tag{7.3}$$

with $\epsilon(n)$ the residual error and $\beta_i(n)$ the regression coefficient. $\beta_i(n)$ then represents the derivative of the learning error after n observations. The learning progress being defined as the derivative of the learning curve (opposite of the prediction error), we obtain the progress $LP_i(n)$ in region R_i by Equation 7.4:

$$LP_i(n) = \frac{2}{\pi} |atan(-\beta_i(n))| \tag{7.4}$$

We transform the slope β_i with an arc tangent to have the learning progress normalized between -1 and 1, and we consider the absolute value of the arc tangent to

[2] Recall that each pixel of $\widetilde{S}(t)$ is a probability of being salient. We binarize these probabilities by a threshold at 0.5

[3] We use the F_1 score as our error metrics, because "*not salient*" pixels are representing more than 90% of the samples, making accuracy inappropriate for error estimation.

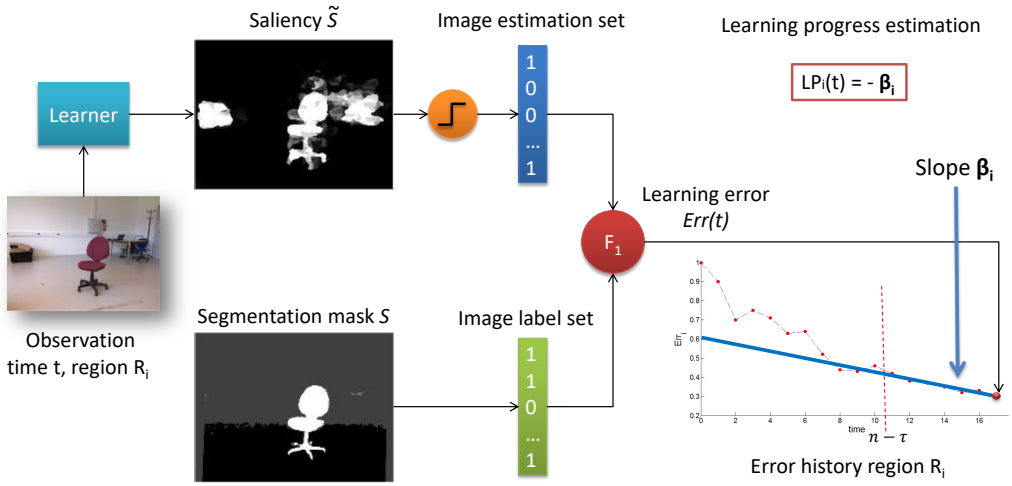


FIGURE 7.5: Forward evaluation of the learning progress

force the robot to explore regions where learning is decreasing as well. Figure 7.5 illustrates this procedure.

Shared learner and backward evaluation

In some implementations of IAC [13], [267], a dedicated learner is associated with each region, so that the model learned in a given region is not influenced by any sample collected in another region. The forward evaluation is well-suited for this case, as the learner is not influenced by observations collected in other regions.

Conversely, saliency shows common properties in different areas of the environment (for example, walls are often uniform and white, whatever the point of view at which you observe them). It is then more interesting to have a common learner shared by different regions. The drawback is that taking samples in a region may influence the performance of the learner in other regions, and therefore, their learning progress.

With a forward evaluation, the consequence of adding samples from outside a region R_i are measured only when new samples are observed in R_i, thus making the learning progress not evaluated as frequently as it should. Moreover, this measurement is clearly biased, as the difference between $Err_i(n-1)$ and $Err_i(n)$ does not measure how much the nth sample helped improving the model, but rather how much the whole observations (in any region) taken between sample $n-1$ and and n did. To avoid poor estimation of the learning evolution in the case where a region is not visited frequently enough, we define a way to monitor learning progress based on previously observed examples, that we call the *backward progress estimation*.

The proposed backward evaluation relies on a set of previously observed feature maps and segmentation masks in each region. Error estimation is then obtained from previously observed data in each region, making progress evaluation possible even when the robot is taking observations outside of the region. In deeper details, let us consider that at time t, the learner is associated with a model M_t constructed with previous observations taken from all regions of the environment. At that time, let us consider that the robot has already taken n observations in region

R_i from the beginning of the experiment (the robot may or may not be in that region at time t, it does not matter).

We then define a *regional label set* $\mathcal{L}_i(n) = \{L_i(1), L_i(2), ..., L_i(n)\}$ for region R_i, where $L_i(k)$ is the *image label set* obtained from the kth observation (See Section 7.3.3 for the detailed procedure related to the image label set, denoted $L(t)$ in the explanations). This regional label set is updated each time a new observation is taken in R_i.

The *regional feature set* $\mathcal{F}_i(n) = \{F_i(1), F_i(2), ...F_i(n)\}$ is obtained in a similar manner by considering the *image feature set* $F_i(k)$ of the kth observation. Similar to the *image estimation set* of the forward estimation, $F_i(k)$ only contains features for which the pixels were stored to create the image label set. The regional label and feature sets are then stored for each regions and updated each time a new observation is collected in this region.

To obtain the backward evaluation of the error $Err_i^{back}(t)$ in R_i at time t, we now need to generate a set of saliency estimates based on the current model M_t. To this end, we simply use each feature vector of $\mathcal{F}_i(n)$, binarize each produced estimate, and constitute this way the *regional estimation set* $\mathcal{E}_i^{M_t}(n) = \{E_i^{M_t}(1), E_i^{M_t}(2), ...E_i^{M_t}(n)\}$.

The backward evaluation is now estimated after each observation in each region (whatever the position of the robot), by considering the regional label and estimation sets:

$$Err_i^{back}(t) = 1 - F_1(\mathcal{L}_i(n), \mathcal{E}_i^{M_t}(n)) \tag{7.5}$$

The error history is also updated at the same time, and the learning progress re-evaluated as well.

The procedure to obtain the *backward progress estimation* is very similar to forward version. The main difference is that the history is represented as a function of time rather than as a function of observations in the region. Taking the same notations as Equation 7.3, the backward learning progress is obtained by

$$\begin{pmatrix} Err_i^{back}(t-\tau) \\ \vdots \\ Err_i^{back}(t) \end{pmatrix} = \beta_i(t) \times \begin{pmatrix} t-\tau \\ \vdots \\ t \end{pmatrix} + \begin{pmatrix} \epsilon(t-\tau) \\ \vdots \\ \epsilon(t) \end{pmatrix} \tag{7.6}$$

and

$$LP_i^{back}(t) = \frac{2}{\pi}|atan(-\beta_i(t))| \tag{7.7}$$

For practical reasons, we also limit the size of $\mathcal{L}_i$ and $\mathcal{F}_i$ by keeping at most 10 frames in memory from R_i, and by randomly replacing them when new inputs are available. The backward evaluation procedure is summarized in Figure 7.6.

7.3.4 Displacement policy

Now that learning progress is evaluated in each region, the measure is sent to a module able to decide the next region to visit, and more precisely, the next action to take in that regard.

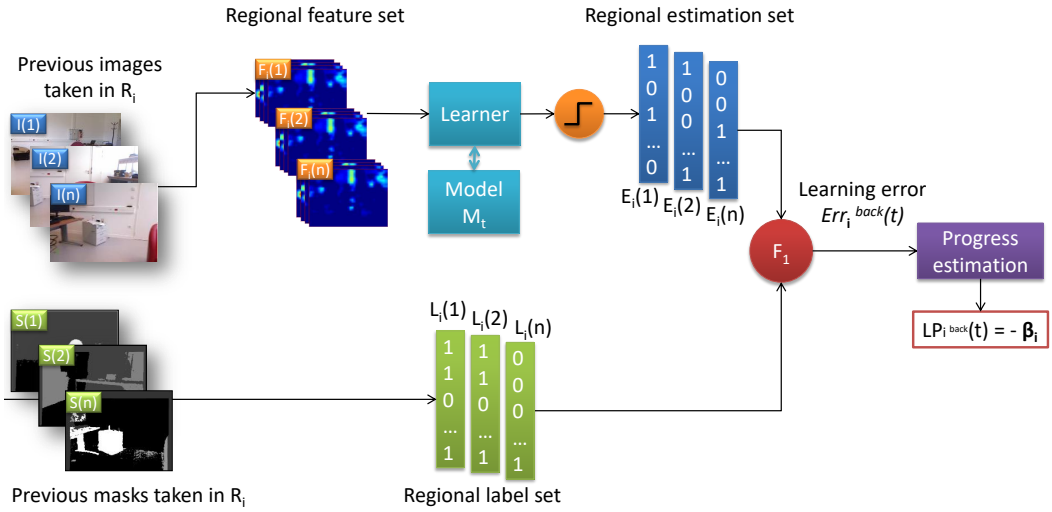


FIGURE 7.6: Backward evaluation of the learning progress in the region R_i

A greedy approach

In most IAC implementations, learning progress is used directly as an intrinsic reward the robot should follow. More precisely, many implementations select the region having the highest learning progress and randomly choose an action among all possible actions leading to that region. As the next action is selected based on immediate learning progress, without any long term planning consideration, we call this behavior "greedy".

We also follow an ϵ-greedy procedure to select the next action. However, we do not directly select the most progressing region, but select the region with a probability proportional to the learning progress. This idea was already proposed by Baranes *et al.* [267], and suggests that the probability of the next region to visit r being region R_i is given by equation 7.8

$$Pr(r = R_i) \begin{cases} \dfrac{LP_i(t)}{\sum_{j=1}^{N} LP_j(t)} & \text{if } v > \epsilon \\ \dfrac{1}{N} & \text{otherwise} \end{cases} \tag{7.8}$$

where $LP_i(t)$ is the learning progress (forward or backward) described in Section 7.3.3, v is a uniform random variable, and $\epsilon = 0.25$ is used to select a random region 25% of the time.

Once the next region is selected, a position within this region is randomly determined as the target to reach. The corresponding action to reach this position is then used to displace the robot. Figure 7.7 presents an example of the next position selection given a current position.

Limitations

In the approach we described in this section, we never mentioned the time required to take an action, and this time is never integrated in the model to decide the next action. The IAC actually considers each action to be costless. This hypothesis holds

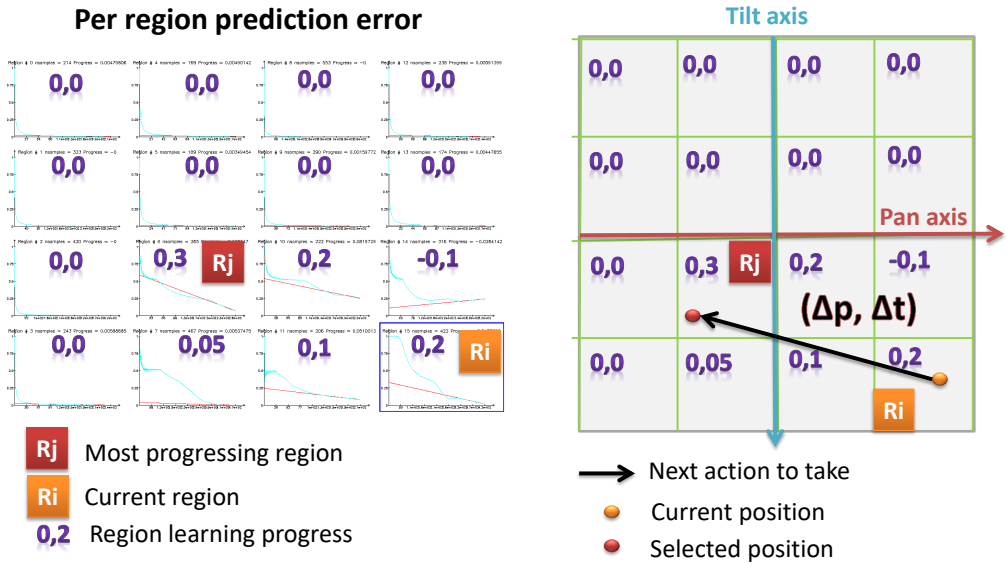


FIGURE 7.7: Next position selection example with a greedy policy

when the time required for learning is longer than the time required to perform an action, or when all actions take the same time.

For a mobile robot moving in a large environment (e.g., a building), the displacements between two regions can be extremely time consuming, thus making the proposed greedy policy inefficient. We therefore propose to extend the IAC policy with a module able to estimate the best trade-off policy between progress and displacement. This is the topic of the next section.

A last point worth mentioning is related to learning progress evaluation when a shared learner is used. Our definition of learning error actually measures how good the model is performing in a given region, and learning progress measures how much it should progress in this region. However, the true quantity the learning progress should measure is how much an observation in a given region should help improving the overall quality of the model. For a learner dedicated for any region, the forward evaluation provides this quantity (supposing that every region has the same weight), but this is not the case for a shared learner. We did not investigate here how to measure such a quantity in the case of a shared learner, but we believe this problem is worth exploring in a future work.

7.4 Learning on a mobile robot: the RL-IAC approach

We propose in this section an extension of IAC able to cope with problems where actions have a different cost. We here consider the case of a mobile robot exploring a large environment (in our case, a building), in which reaching position B from position A can be extremely time consuming.

In this section, we first describe our problem as an extension of the strategic student problem, and we describe a new algorithm called *Reinforcement Learning-Intelligent Adaptive Curiosity* (or RL-IAC) that is based on IAC, but proposes two additional modules to integrate the time required for taking actions in the decision process.

7.4.1 An extension of the strategic student problem

Initial problem

The *strategic student problem* (SSP) was introduced by Lopes and Oudeyer [296] as a theoretical and general framework for lifelong learning problems. The SSP considers the case where a student has to study K topics within a time N before an exam, and to optimize his final grade by allocating a limited amount of time in each topic. The problem is modeled by considering the learning function $q_i, i \in [1..K]$ such that spending time n_i studying topic i makes the student end up with grade $q_i(n_i)$. Each grade may be weighted by a factor C_i to compute the global final grade. The optimization problem is then formulated as follows:

$$\begin{cases} \underset{n_1,\dots,n_K}{\operatorname{argmax}} \sum_{i=1}^{K} C_i q_i(n_i) \\ \sum_{i=1}^{K} n_i = N \end{cases} \tag{7.9}$$

The strategic student problem can be solved by simply looking at each time step the progresses made in each topic (considering the derivatives of the q_is), and allocating the next slot of time on the topic where learning progress is the highest. Experimental results demonstrated that this approach is optimal. This idea is consistent with the IAC approach of greedily following learning progress for exploration. As a bonus, Lopes and Oudeyer demonstrated that this approach was quasi-optimal if the learning curves are monotonic sub-modular[4].

Analogy with a learning mobile robot

We now consider the problem of a mobile robot in a building, moving across K rooms and corridors to learn a model of saliency in each of them. Each room has its own learning function, and the strategic student problem can easily be applied to this case. However, a major difference is that the displacement of the robot between rooms has a non negligible time during which the robot does not learn. In the SSP, the transition between topics actually not considered. However, the problem could be made more complex by considering that the student learns a given topic by consulting books on site in different specialized libraries in town. In this configuration, switching between topics is pretty time consuming, and the student must optimize the time spent in displacements on top of the time allocated for learning. The problem would then extent the SSP as follows:

$$\begin{cases} \underset{n_1,\dots,n_K}{\operatorname{argmax}} \sum_{i=1}^{K} C_i q_i(n_i) \\ \sum_{i=1}^{K} n_i + \sum_{i,j} T(i,j)\psi(i,j) = N \end{cases} \tag{7.10}$$

where $T(i,j)$ is the time required to move from library for topic i to library for topic j, and $\psi(i,j)$ is the number of time this displacement has been done. Thus, available time N is divided between the time to learn different topics, and the time to switch between topics. The problem is now about jointly optimizing learning and displacement. If $\psi(.,.)$ is constant, the problem to solve is a simple SSP. However,

[4] A function $f : X \to \mathbb{R}$ is a submodular function if the following property is verified: $\forall Y, Z \subset X$, $f(Y \cup Z) + f(Y \cap Z) \leq f(Y) + f(Z)$

switching between topics modifies $\psi(.,.)$ and the cumulated displacement time, which is equivalent to solve a Traveling Salesman Problem (TSP).

This problem cannot be solved in practice for several reasons. First, the learning functions are not known beforehand, so that only short-term estimates can be used to define the time allocation in the near future. A planning from time 0 to time N is not feasible. Second, the TSP is known to be NP-hard and is solvable in reasonable time only though heuristics. The problem is even more complex here as constisting in a joint optimization. Previous works already tried to jointly solve a TSP with the art gallery problem [246], but they solve both problems sequentially and with heuristics.

In RL-IAC, we then use an empirical evaluation of the learning progress that we update as soon as possible, and we rely on path planning techniques to estimate, given the current configuration of the environment and learning progress, the best path to follow. This path is re-estimated jointly with progress estimation update.

7.4.2 RL-IAC: General idea

RL-IAC works on the same principle than IAC in the sense that the exploration space is divided in regions, that a learner tries to make predictions about the environment, and that a meta-learner monitors the error made in each region and derives a learning progress. However, two additional components are considered in RL-IAC, and are meant to provide an action policy able to handle the time spent in performing actions.

IAC vs RL-IAC

Figure 7.8 presents the general architecture of RL-IAC. As compared with Figure 7.1, representing the general mechanisms of IAC, the only differences are about the region module and the action policy modules. Provided as an input for the meta-learner in the case of IAC, the regions here also help in the action policy process. In addition, they are updated by the robot displacement by an incremental procedure.

First, in the region module, we now rely on a graph describing the connectivity between regions and encoding the average time required for displacements between regions. Given this representation, all displacements are not possible from a given region. To reach an arbitrary position from another position, a sequence of displacements may be required. In addition, the distance between regions is integrated in the action policy module given this graph representation. In mobile robotics, this kind of representation is called a *navigation graph*.

Second, the action policy module is a more complex one (called the reinforcement-learning module), that integrates the navigation graph and the learning progress to calculate a displacement policy that optimizes the amount of collected learning progress. To this end, the environment is modeled in a reinforcement learning context, where learning progress is a reward to be collected, and the navigation graph models the states and actions of the environment. The RL module is re-trained at each step to provide the robot with a displacement policy.

Aside from these two aspects, the learner, the region space definitions and the meta-learner work exactly the same way as in IAC. Please refer to the corresponding paragraphs of Section 7.3 for more details.

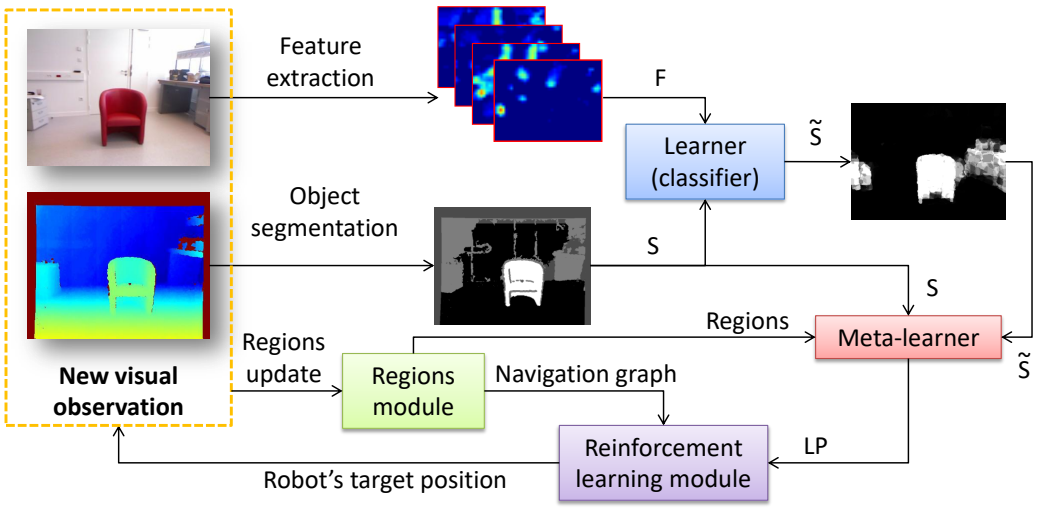


FIGURE 7.8: General architecture of RL-IAC

Link with path planning approaches on a mobile robot

The problem of action policy is turned in our approach as a problem of path planning where the robot has to move in the environment to collect some rewarding learning progress. The displacement of the robot is seen as a sequence of actions performed to collect this reward, that can be solved by techniques of path finding.

As mentioned in Section 6.2.2, path finding can be solved by many different approaches. Dijsktra [225] or LSPI [301][5] algorithms may have been used instead of Q-learning and would probably have lead to similar results given the inputs of the problem. Q-learning however has the advantage of being model-free and then potentially applicable to more complex scenarios.

Unlike most path finding problems, our robot does not have to reach a specific goal. In addition, the learning progress is, by nature, progressively vanishing during the experiment. We then consider our problem in a reinforcement learning context, without any external goal, and with a non stationary intrinsic reward. This kind of configuration is not very common in the literature.

Link with artificial curiosity

A few research works have tackled the problem of mixing learning progress or intrinsic motivation with reinforcement learning [252], [257], [262], [290], [297] to provide the robot with an intrinsically motivated exploration policy. In these approaches, there is no warranty on the optimality of the final policy. Indeed, the proofs of convergence towards an optimal policy only holds for a constant reward. Luciw reported this aspect in a previous work [297], and Lopes *et al.* provided a mathematical proof of the exploration efficiency of their approach [257], without claiming optimality. So far, the benefit of using learning progress with reinforcement learning has only be empirically demonstrated.

The main difference between these approaches and RL-IAC is the way reinforcement learning is considered for action selection. In the aforementioned work, reinforcement learning is part of the experiment and is what makes the robot select a new action. As actions are taken and reward is collected, the policy is refined, and

[5] Least Square Policy Iteration

this new policy guides the future robot's actions. In our case, a new reinforcement learning problem is initialized and solved after each model update. The optimal policy is obtained by internal simulations, and the next action to take is decided based on the optimal policy. In other words, we follow a philosophy of optimal control[302], where Q-learning is used as a tool to find an optimal policy given available data at time t, and use this policy to determine a single action.

7.4.3 Regions and navigation graph

The traditional region module is extended by a navigation graph representing the spatial relationship between regions. We discuss in this section the construction of such a navigation graph, and propose a method to build it incrementally as the robot explores the environment.

Regions and navigation graph formatting

Regions in RL-IAC still represent a set of reachable points of view in an appropriate space. We mainly reason here in the 2D map space of a mobile robot, as the method was designed in that purpose. However, the navigation graph definition still holds in other kind of spaces.

In Section 7.3.2, we defined regions in the (x, y, θ) space of the robot's position, divided into regular intervals. Here, we use a slightly different definition of regions, although simple considerations could make the former definition compatible with the description of this section.

Here, regions are defined as an area of the 2D map represented by a set of (x, y) positions that the robot can reach. The θ component representing the orientation of the robot is then not considered. This means that the region space is not formally the one of all reachable positions of the robot. However, when selecting an action to reach a given region, we add a θ component that fully describes the future state of the robot. In additions, regions are not necessarily obtained by regular subdivisions of the space.

To define a navigation graph from these regions, we consider the centroid of each area describing a region. These centroids stand for the nodes of the navigation graph. Regions are connected with each other if they share a common frontier. This is represented in the navigation graph by connecting the nodes by an edge. Each edge is associated with a weight that represents the euclidean distance between the two centroids. The proposed navigation graph then represents the environment as a topometric map. Figure 7.9 presents an example of regions defined by areas on a 2D map and the associated navigation graph. In this representation, nodes of the navigation graph are represented at the physical centroid position of the region, and weights of the edges are represented by their length.

The navigation graph is then a representation of the displacement cost, as time for displacement is roughly proportional to the distance. It also represents the set of reachable positions from a current position by considering the edges associated with the current region. This representation has the advantage of being compact and simple to obtain. However, this is only a rough approximation of the displacement cost, as only distances between centroids are considered. Moreover, the sequence of actions proposed to join two points of the map is not necessarily the shortest path.

Lastly, the way to obtain regions and navigation graph is flexible. If the environment map is known beforehand, a manual definition of the regions may be

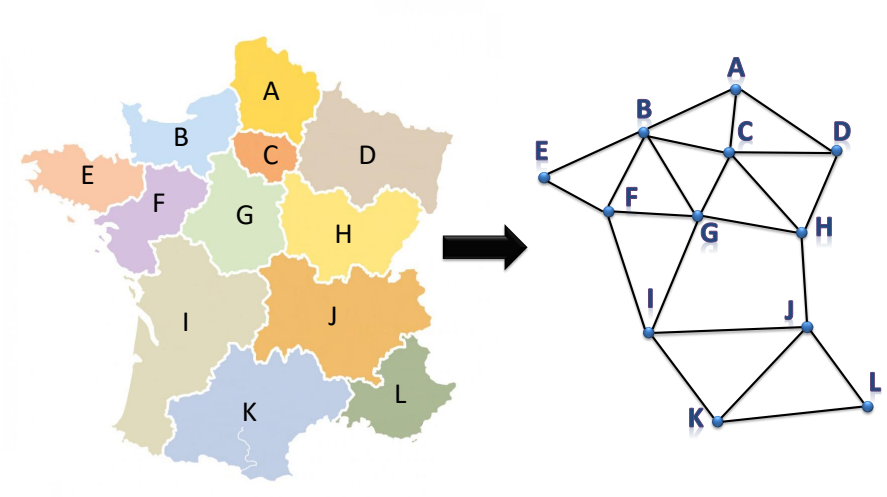


FIGURE 7.9: Example of arbitrarily defined regions (on the left) and associated navigation graph (on the right)

provided in order to define areas that are likely to be consistent (for example, a region could be defined as a room of a building). However, nothing in RL-IAC forces the regions to be static, so that we developed a technique to determine regions and navigation graph on-the-fly during the robot's exploration.

Incremental navigation graph building for a mobile robot

The method described in this section is able to jointly define regions and build a navigation graph as the robot moves in the environment. Both are then updated when new areas of the environment are discovered, or when known areas have changed (opening of a closed door for example). Note that this method has been designed for a mobile robot equipped with an RGB-D camera.

More precisely, 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 7.10 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 in adjacent proto-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. This way, each position of the occupancy grid is then 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:

FIGURE 7.10: Example of regions and navigation graph after exploration.

- 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.

Figure 7.11 provides an illustration of each aforementioned cases. In this configuration, 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.

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

FIGURE 7.11: Update procedure example

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.

7.4.4 Reinforcement learning module

To determine an action policy able to take into account the displacement cost, we use the navigation graph, as it encodes the relative distances between regions and the possible displacements to reach them. We use this navigation graph to model states and actions of a Q-learning problem [303]. The reward is represented by 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.

We suppose in this work that the positions of the robot are perfectly known and that a target position is perfectly reached. As Q-learning is designed for setups where the transitions between states are probabilistic, the integration of noise in the position measurement could be considered in the algorithm as a straightforward extension.

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

States, actions and reward definition

Q-learning problems are typically defined by states, actions and rewards. The theoretical background about the Q-learning algorithm is recalled in Section 6.3.2. In our setup, the states are the node of the navigation graph. They then represent the regions, and are defined by the region centroids.

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 action among "*up*", "*down*", "*left*" or "*right*", depending on the graph edges, and moves to the corresponding neighbor region.

A reward r is associated with each region R_j. We take for each region the last calculated learning progress. Thus, at time t, we define reward as:

$$r(R_j) = LP_j(t) \tag{7.11}$$

LP_j being the forward or backward learning progress defined by Equations 7.6 and 7.7.

State-Action policy computation

To determine a state-action policy, 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 travelled distance exceeds $N = 1km$ [6].

During the batch of episodes, the action-value function Q (also called the Q matrix for simplicity) is updated according to the following rule

$$Q(s_k, a_k) = r(s_k) + \gamma \underset{a' \in A}{\text{Max}} Q(s_{k+1}, a') \tag{7.12}$$

with γ the discount factor, 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.

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 define the next action a_t taken by the robot by Equation 7.13:

$$a_t = \underset{a' \in A}{\text{Argmax}}(Q(R_i, a')) \tag{7.13}$$

Here again, an ϵ-greedy policy is used and 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 by executing action a_t. To perform the action in reality, a position (x_j, y_j) in R_j is randomly selected and constitutes the next target to reach. Additionally, an orientation θ is also randomly determined to fully describe the next position of the robot. The robot then moves to this position, updates the navigation graph, grabs a new RGB-D input, updates the learner and meta-learner, and launches a new batch of simulation to determine 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 get any better). However, the assumption is accurate enough to provide a reasonable 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 there. This additional constraint has the same effect as the R-max [255] exploration policy in reinforcement learning.

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

7.4.5 Remarks

To conclude this section, we point out two features of RL-IAC that could be further investigated in a future work.

IAC as a trivial case of RL-IAC

The action policy module proposed in RL-IAC is noting more than an extension of the greedy action policy proposed by IAC. The greedy behavior of IAC could be obtained in RL-IAC by providing the action policy module with a navigation graph fully connected between regions, and having a displacement cost of 0 between each region. In this kind of graph, the best policy is such that the action to take from any region makes the robot end up in the most progressing region, which is exactly the behavior proposed by IAC.

Of course, using Q-learning on a fully connected graph with a large number of regions is not very efficient. The direct selection of the most progressing regions without running the whole Q-learning simulations would lead to the same result much faster, but this is just a matter of implementation.

Optimality and possible improvements

In RL-IAC, we calculate a policy at each step, apply this policy and recalculate a new one, so that the robot follows in the end a sequence of displacement that is not associated with a unique policy.

The policy is actually not the right tool for defining an optimal displacement sequence of the robot, as learning progress in the different regions are evolving.

In a future work, the computation of the next states to reach should integrate the diminution of the learning progress as new observations are taken in a given region, and a different representation of the states should be considered for calculating the optimal sequence of actions to take.

7.5 Conclusion

In this chapter, we proposed an exploration strategy dedicated for the task of learning saliency. More precisely, we have adapted and implemented the IAC algorithm for our particular problem and setup. Equipped with this exploration strategy, the robot is intrinsically motivated, and can efficiently explore and learn in a lifelong learning perspective.

This approach was developed for two types of setups: first, a fixed but orientable head equipped with a set of cameras. Second, and a mobile robot able to move across a building. We then detailed and discussed the implementation choices we took, as well as their innovation towards previous implementations.

A first contribution was the definition of the backward evaluation of the learning progress, which provides a better error evaluation in problems having a learner shared between regions. The second contribution (probably the major contribution of our work) is the creation of RL-IAC as an extension of the IAC algorithm, able to take into account long and costly displacements across the environment. RL-IAC is still, however, an open problem with many possible improvements.

Chapter 8

Experimental results

8.1 Introduction

The previous chapter was presenting the technical details of the proposed method to efficiently and autonomously explore the environment. We propose in this chapter a quantitative and qualitative evaluation of this approach. We then exhibit the behavior, advantages and drawbacks of both IAC and RL-IAC for the task of saliency learning.

First, we propose to evaluate IAC applied in the context of saliency learning based on several datasets and robotics platforms. We particularly exhibit the need for a methodical exploration of the environment to rapidly obtain a reliable model. We compare the efficiency of our approach with other exploration strategies to emphasize this finding. Aside from these results, We also examine the behavior of IAC in terms of time allocation in each region. We also study how and to which extend the method can adapt to a significant change in the environment.

Second, we study in more details the RL-IAC approach in the case where the displacement time is non negligible. We present in this section three environments in which the robot can jointly navigate and learn saliency. Based on experimental results on these environments, we first exhibit the need for using RL-IAC instead of IAC in this context. We then examine the influence of some parameters of the system on the exploration efficiency. Lastly, we demonstrate that RL-IAC outperforms other exploration techniques, and that learning progress as an intrinsic motivation factor is efficient in spite of estimation biases.

8.2 IAC

In this section, we provide a set of experiments aiming to evaluate the use of the Intelligent Adaptive Curiosity in the scope of saliency learning. To this end, we produce results from three different setups, and we evaluate both the performance of the algorithm and its behavior in this problem.

8.2.1 Experimental setups and evaluation procedure

Construction of the experiments

Our method is evaluated by conducting exploration in semi-simulated environments. This is done by using the recorded sequences presented in Chapter 2. We call them semi-simulated as saliency is learned from real images taken from these sequences, but actions taken by the robot are simulated.

More precisely, to make the robot reach a position in our setup, we simply select one of the frames recorded in the sequence. This process has in practice no physical

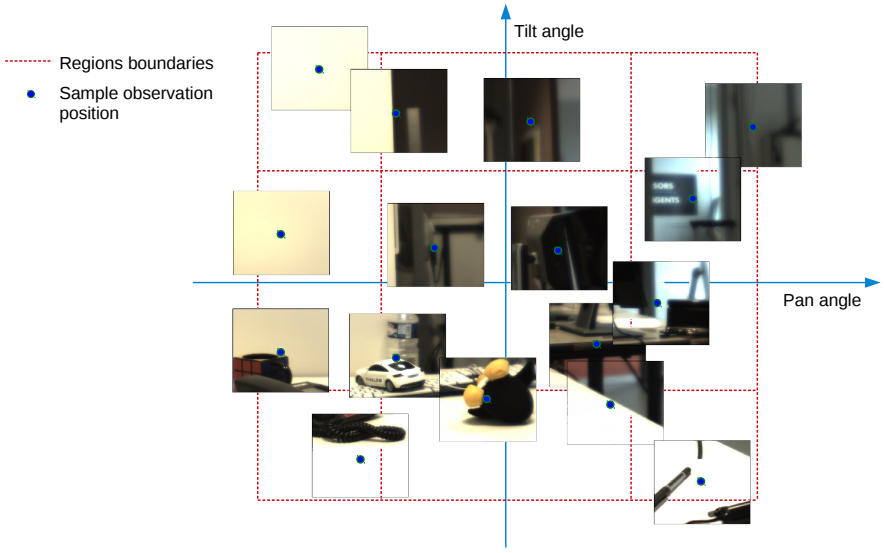


FIGURE 8.1: *BioVision*'s 16 regions and associated sample foveal images.

cost and do not take into account the time required to take this action in real life. Once selected, we just consider that the robot has reached this position and can start processing the associated observation.

BioVision

The first evaluation setup uses the *BioVision* platform. Considering the dataset described in Section 2.3.1, we only work with one of the recorded sequences. This sequence contains both RGB contextual and foveal images to estimate saliency and update the model. In addition, we also use for the experiments the pan-tilt positions associated with each observation.

Regions are defined before the experiment by dividing the reachable pan-tilt positions into 4 pan intervals and 4 tilt intervals (for a total of 16 regions). The size of each region is defined so that regions are not equally interesting. Some of them only contain white wall, without any salient object, while other are mainly composed with them. Figure 8.1 illustrate a sample of foveal images reachable in each region.

For the action selection process, we attach to each region the list of observations falling within the region's boundaries. When the robot selects a random action within this region, one of these frames is randomly selected.

ENSTA

To exploit the *ENSTA dataset* described in Section 2.3.2 in this context, we consider a sequence recorded in a single room of the building. The room considered is the laboratory, represented in Figure 8.2.

During the sequences recorded in the ENSTA building, the robot was able to estimate its position in the environment from a SLAM algorithm. We then consider for each recorded observation the associated position and orientation of the robot.

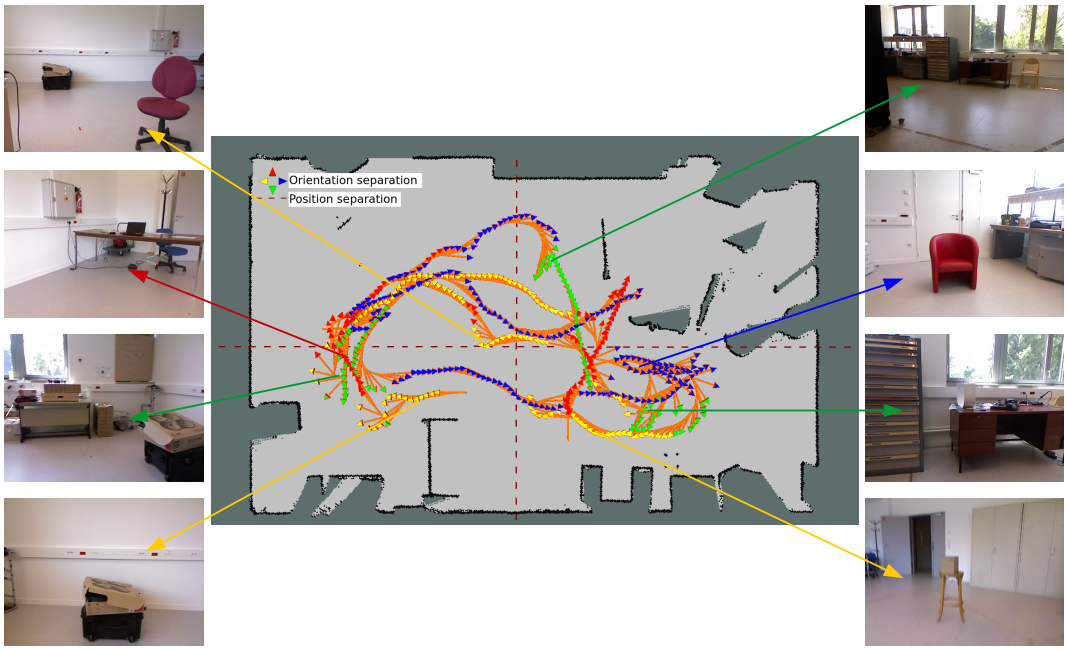


FIGURE 8.2: The 16 regions of the *ENSTA* laboratory sequence, and sample observations for a few regions

Here, 16 regions are also defined before the experiment. This time the space is divided along the x and y axes of the map (two subdivisions for each), and the θ angle (four subdivisions) representing the orientation of the robot.

Here again, each observation recorded during the sequence acquisition is associated with a position lying within a region interval. Each region then contains a list of observations that the robot can reach by random selection. Here again, the time for displacement and rotation is not considered.

RGB-D scenes

The last evaluation dataset is constructed from the *RGB-D scenes dataset*, and more precisely the *table_small_2* sequence. This time, we do not consider any information about the position of the camera, and simply consider the sequence as a temporal succession of observations.

In this case, the space where to define regions is simply the temporal axis. We then define five different regions by spliting the sequence into 5 sub-sequences of equal length. Some of these regions were more challenging than other because of the local points of view. In this configuration, the robot reaches an observation by simply selecting a frame index in a given region. Figure 8.3 shows sample images for each region of the sequence.

Evaluation procedure

In general, the performance of IAC is hard to evaluate [13] as intrinsic motivation systems are designed to develop skills autonomously and independently of any task. The traditional approach that consists in evaluating the performance level of a task does not make any sense in this case. As alternative evaluations, Oudeyer *et al.* [13] have proposed a set of measures based on the robot's point of view, external point of view and the observation of successive stages.

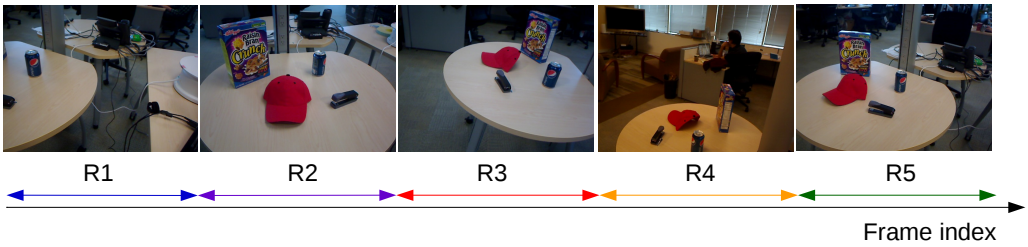


FIGURE 8.3: The 5 regions cut along the temporal axis of the *table_small_2* sequence

In our case, our system uses a developmental approach for the single task of learning a saliency model. An objective evaluation based on the saliency map quality is then possible.

The measure of the saliency quality is evaluated using the evolution of the *overall error rate* of the system: based on the reference frames of the datasets for which a ground truth is available, we produce an error measure of the estimated saliency maps. To this end, we produce after each learner update the saliency maps for each reference frame, and we compare them with the available ground truth. We use for that the F1 score and a very similar formula than the one proposed for the backward evaluation in Section 7.3.3:

$$Err_{ovl}(t) = 1 - F_1(\mathcal{GT}, \mathcal{E}^{M_t}) \tag{8.1}$$

where $\mathcal{GT}$ is the *ground truth label set*, produced in a similar fashion than the *regional label set* of Section 7.3.3, with the set of available ground truth masks. $\mathcal{E}^{M_t}$ is the *estimation set* produced with the available model M_t at time t in a similar way than the *regional estimation set* of the same section.

Given this measure, we can monitor the evolution of the error rate all along the experiment and objectively compare the different exploration approaches. Note that the *overall error rate* has a different signification than the *regional error rate* used to compute learning progress. Indeed, the *regional error rate* is an intrinsic metrics, whose evaluation is based on segmentation rather than on an external ground truth. Second, we use the F1 score, rather than the ROC curve for evaluation to be consistent with our evaluation of the learning progress.

8.2.2 Behavior of IAC in the task of saliency learning

Learning curves in different regions

For IAC to be efficient and useful, the learning curves should be different in each region. If not, the learning progress would be the same everywhere, and a uniform exploration of the environment should lead to similar results much more easily.

We therefore estimate the error rate evolution on the *table_small_2* sequence for each region taken separately. To this end, we run five experiments (one for each region) in which the robot is allowed to explore only in a single region. The corresponding curve then represents the evolution of the error rate in that particular region. Results are reported in Figure 8.4.

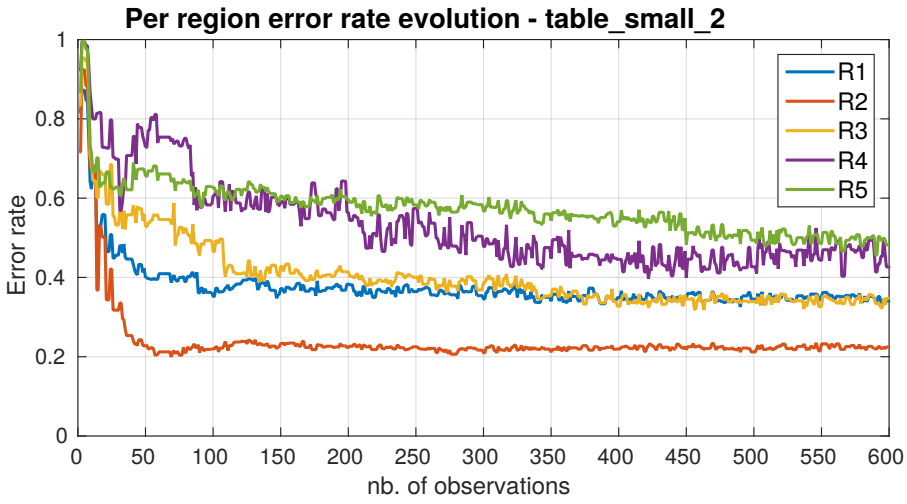


FIGURE 8.4: Error rate in each of the five regions of the *table_small_2* sequence

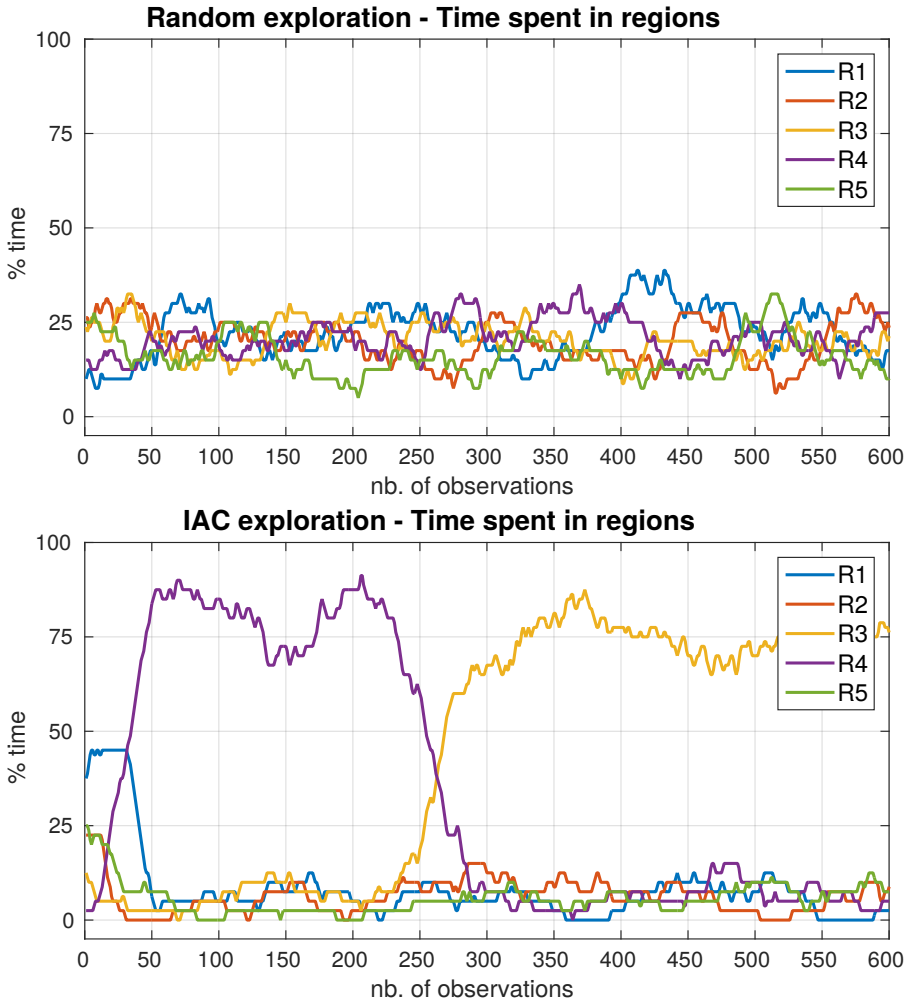


FIGURE 8.5: Time spent in each region for random exploration and IAC-based exploration

We notice that some regions converge very quickly (**R2** for example), while other are much more challenging (**R4** and **R5**). These results were somehow expected, given the average point of view in each of these regions. In **R2**, the objects and the table represent a large fraction of the input images. The table is easy to find by the segmentation process, and objects on it are easily segmented. Conversely, regions 4 and 5 show the table and objects on it at a smaller scale. The background is much more present, and significant illumination changes (due to the auto white balance) make some frames harder to classify.

As a result, the variability in the error evolution among the different regions suggests that an intrinsically motivated exploration of the environment should lead to better and faster learning.

Time allocation in regions

We now demonstrate that the IAC approach provides such a methodical exploration of the environment, and that the exploration procedure taken by the robot follows some stages of development. To this end, we use again the *table_small_2* sequence and compare the exploration using IAC and the random exploration.

Figure 8.5 represents the proportion of time spend in each region during the exploration. To build these graphs, we monitored during the whole sequence the regions associated with each observation of the robot. Then, we measured the local number of observations in each region by applying a sliding window of 50 observation all along the sequence. Each curve then represents the percentage of time spent in a given region.

As opposed to random selection, where the time spend in each region is almost constant, IAC shows a clear methodological exploration, consistent with the error rate presented in Figure 8.4. For example, region **R2** progresses very fast in the beginning, then becomes flat after 50 observations. For that reason, the time spent in **R2** with IAC exploration is slightly higher at first, then significantly drops to a very low rate. Region **R1** and **R5** have a similar behavior. Then, as **R3** and **R4** have a much slower learning rate (and, as a result, a lower progress), much more energy is spent in these regions to improve the model. In addition, a switch is clearly visible between regions **R3** and **R4** precisely when **R4** starts making fewer progress than **R3**.

As a result, in IAC, the robots organized its exploration to visit regions of increasing complexity. To better visualize this process, a video is also available[1], showing the evolution of the error in each region and the switch between regions.

Adaptation to a change in the environment

Most of our experiments use a static environment. We now want to examine the evolution of the exploration in the case where the environment is changing. Suppose now that *BioVision* is learning the saliency of the environment, when someone comes and suddenly moves the positions of the objects on the table. The learning curve in the regions containing objects should be drastically modified, and the exploration strategy should exhibit different patterns.

To produce such behavior, we consider three sequences of the *BioVision dataset*: we make *BioVision* start learning with a certain sequence, and we switch to a new one during the experiment. We use for that a first sequence with a moderate number of objects, and switch with two different sequences: first, a sequence in which

[1] https://www.youtube.com/edit?o=U&video_id=w-nGugA5SS4

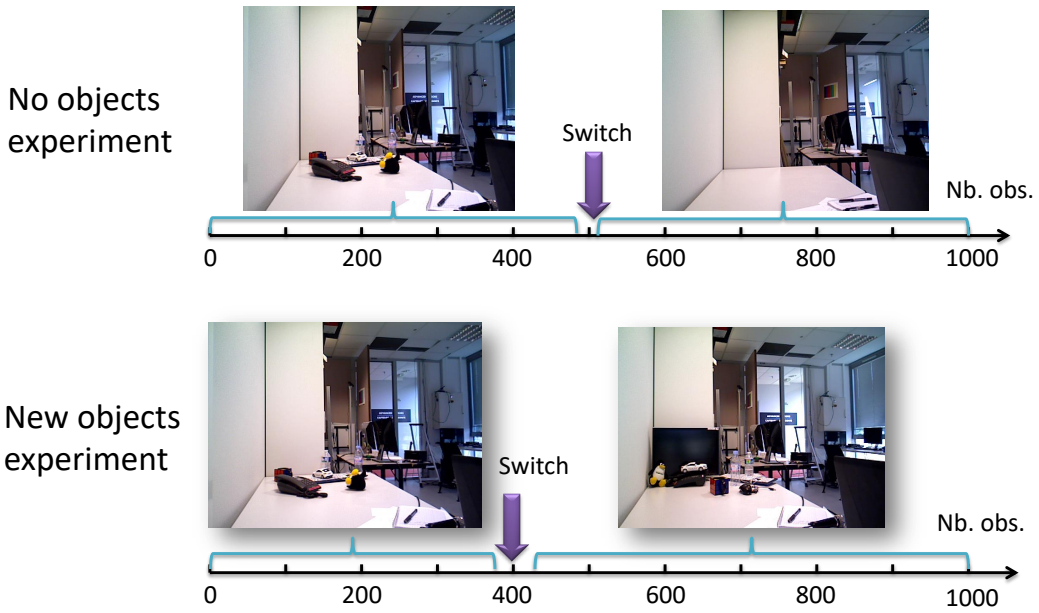


FIGURE 8.6: Switch between datasets to simulate a change in the environment of the robot

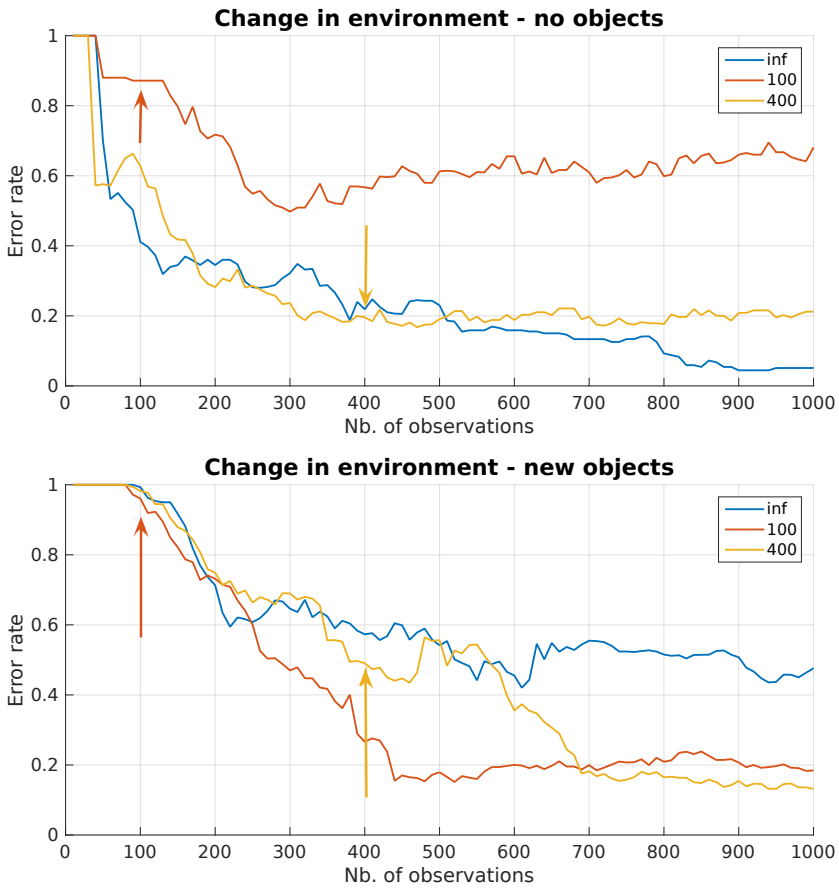


FIGURE 8.7: Error rate evolution when modifying the environment of the robot

all objects were removed from the table (called "**no objects**" in the explanations). Second, a sequence where additional objects were put on the table("**new objects**" in the explanations). Figure 8.6 illustrates the switch principle to simulate the change in the environment. To further analyze the behavior of IAC in this configuration, we run several experiments by changing the time at which the switch operates: after 100 or 400 observations, or never.

We first present in Figure 8.7 the evolution of the learning curve when operating a switch in the environment after 100 and 400 observations (curves **100** and **400** in the figure). We also display the resulting error rate without any change (curve **inf**). In the experiments, the error rate is evaluated based on the ground truth of the sequences constituting each experiment. As a result, the **no objects** and **new objects** experiments are not evaluated on the same set of ground truth and do not have the same error rate curves. As expected, the change in the environment (indicated by arrows in the plots) strongly modifies the learning curve quality: when the environment switches to a table without any object, the scene is simplified, and the robot cannot get additional samples to learn the visual aspect of salient objects. As a result, the later the switch occurs, the better the learning. This is verified by the first plot of the figure, where the switch after 100 observations produces a very bad model. Conversely, when the new configuration of the environment is a more complex one, the robot is able to learn the aspect of additional objects, which is likely to improve the learning quality. This time, the switch is supposed to make the model better. This is verified by the second plot of the figure, where the switch at 100 observations enables a fast decrease of the error rate. However, when the switch never occurs (represented by the **inf** curve), the error rate cannot reach a descent performance. Lastly, the curve corresponding to a switch at 400 observations seems to have the best final performance (as compared to the switch at 100). This results would need to be confirmed with additional experiments, but this may be due to the greater number of samples from the first configuration that produces a better balanced model.

Second, we represent the amount of time spent in each region by following a similar procedure than in Section 8.2.2. To better analyze the results, we group the 16 regions in four *areas*: first, regions containing salient objects at the beginning of the experiment. Second, the one containing a background easy to learn (white walls and table). Third, the one containing a complex background. Lastly, the one for which the more complex environment contains salient objects and the initial one does not (See Figure 8.6 to visualize the different environments, and Figure 8.8 to visualize the groups of regions called areas in these explanations). We now display in Figure 8.8 the evolution of the proportion of time in each of these groups, for all of the experiments.

Regarding the curves obtained for both the **no objects** and **new objects** environments, the following behaviors can be observed.

- Between 0 and 200 observations, the exploration time drops in each area, except the one containing salient objects. This area is the most progressing one.
- After 900 observations, the area containing salient object decreases as well for the **'inf'** curve. This is because the model does not make much more progress in this area.

When looking in more details at the first row of the figure, obtained when switching to the **no objects** environment:

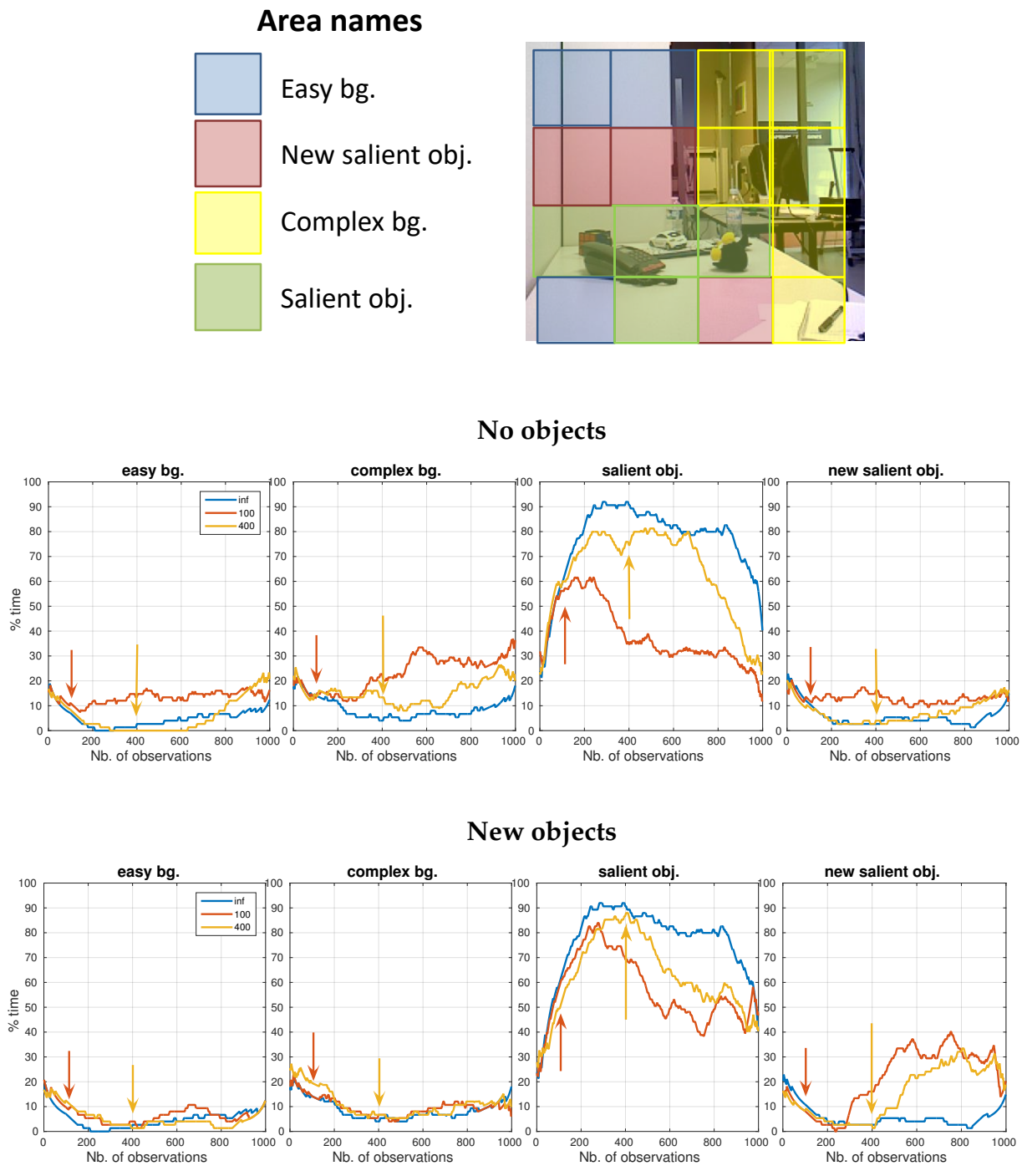


FIGURE 8.8: Time spent in each group of regions for each experiment. Top: **no objects**. Bottom: **new objects**

- For the curve **'100'**, after 200 observations, exploration decreases in the salient objects area and increases everywhere else. This is because this area does not contain any salient object anymore.
- After 200 observations, the complex background area now seems to be the most observed one.
- The same behavior is observed for curve **'400'**, after 600 observations.

Lastly, when examining the second row, obtained when switching to the **new objects** environment, the following comments can be made:

- For the curve **'100'**, after 200 observations, exploration decreases in the salient objects area and increases in the new salient objects area. This is because this area now contains much more salient objects and is now worth exploring more.
- The same behavior is observed for curve **'400'**, after 450 observations.

As a last general comment, we observe a certain delay (between 50 and 200 observations) between the actual switch of the environment, and the exploration behavior of the robot. This may be explained by the time required by the system to correctly assess the change in the learning progress. Therefore, the system is reactive to changes in the environment, but with a certain inertia.

IAC versus other exploration strategies

To demonstrate the benefits of exploring the environment using IAC, we compare the performance of the system with three other types of exploration strategies. For each dataset, results are reported in Figure 8.9 for the following explorations:

- **Chronological**: Frames of the sequence are integrated to the learner in chronological order;
- **IAC**: The next region to visit is determined with IAC and the next position to reach in it is selected randomly;
- **Random**: The next region to visit is determined randomly and the next position to reach in it is selected randomly as well.

Each of these exploration techniques were tested 10 times on each dataset, and the mean and standard deviation were considered and displayed in the result curves. As a reference, we also display the error rate obtained by training the classifier offline with all the dataset (**Offline** in the figure), and the error rate of the bottom-up saliency method BMS [66] already used in Chapter 5 (**BMS** in the figure).

As expected, the offline model is a lower bound of the error rate, and the system tends to reach this limit after a certain number of observations, whatever the exploration strategy. Our method rapidly outperforms BMS when enough observations are obtained. The chronological exploration is slower to converge than random exploration at the beginning. Lastly, IAC seems to be the exploration strategy for which learning is the fastest. Note that the difference between random and IAC-based exploration is less significant on the robot sequence, but the error variance of IAC is much lower than the one for random exploration.

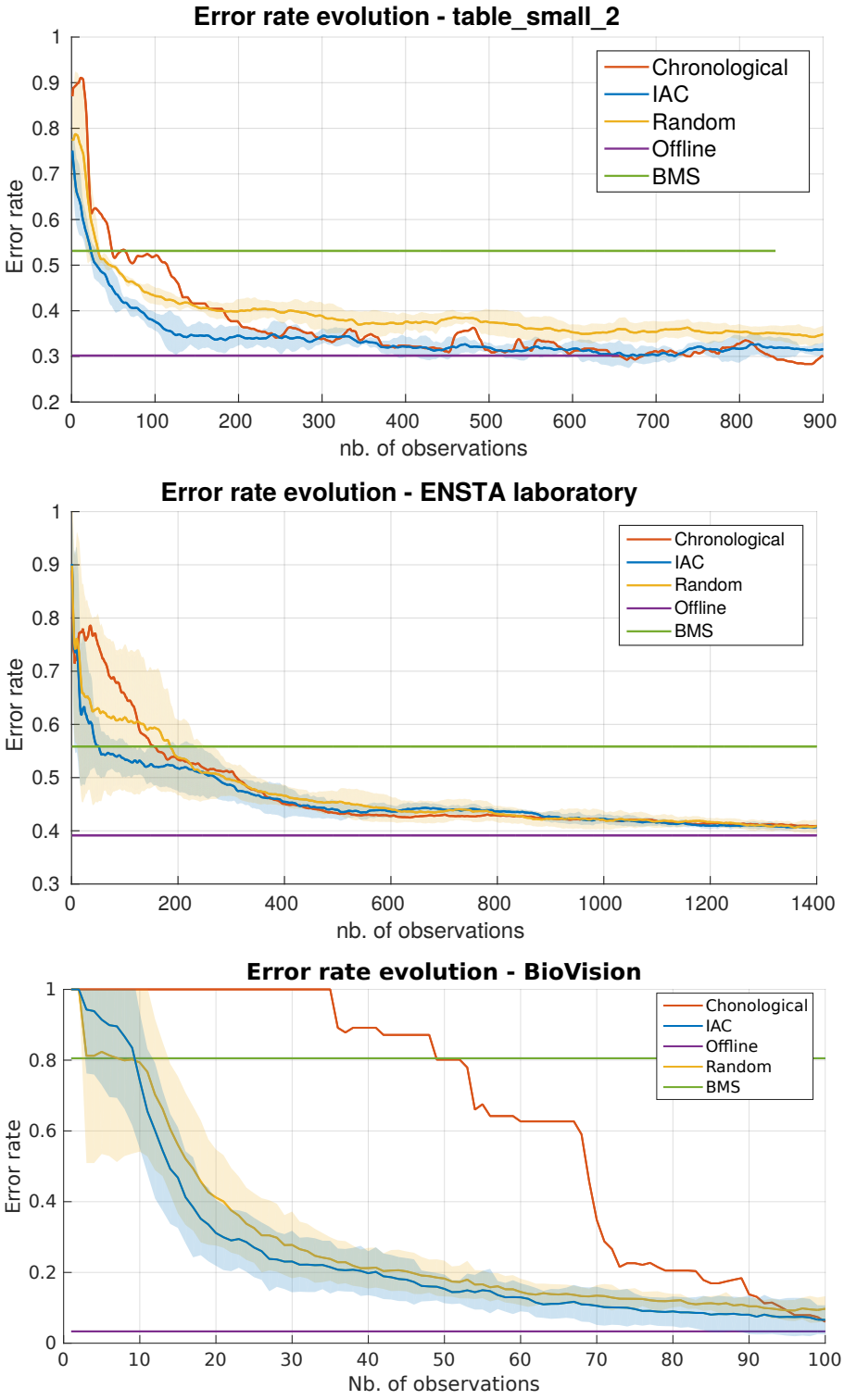


FIGURE 8.9: Evolution of the error rate for several exploration strategies

8.3 RL-IAC

In chapter 7, we described the limitations of IAC, and the need for a new algorithm in the case of a mobile robot moving across a large environment. This section aims to provide quantitative evidences on this need, and evaluates the performance of RL-IAC as compared to IAC and other exploration strategies.

8.3.1 Experimental setup

To evaluate our results, we also rely on different experimental setups to simulate the displacement of the robot across the environment.

ENSTA

The first dataset was recorded in the ENSTA building and presented in Chapter 2. This time, we take the longest sequence recorded in the whole building (See Section 2.3.2 to get an overview of the environment and the trajectory taken by the robot). This sequence contains both the RGB-D images and the positions and orientations of the robot associated with these images.

From this sequence, we build a navigation graph based on the technique described in Section 7.4.3. Since this method constructs a navigation graph incrementally as the robot is getting new RGB-D inputs from its environment, it could work in parallel with the saliency learning model and the RL-IAC exploration strategy. However, to simplify the experimentation, we use the sequence once to construct a navigation graph, and we start the RL-IAC exploration with this navigation graph as a static input. The navigation graph used in all our experiments is depicted in Figure 8.10. Proto-regions were arbitrarily defined to be of 5 meters length.

Similar to the exploration procedure for IAC, we consider the positions of each observation recorded during the sequence, and we associate each of them to a region. When selecting a position in a given region, we then 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[2].

Note that some of the regions here did not contain any frame. In these regions, when the robot attempted to take an observation, no update was done in the model, so that the prediction error was always high, and the region not attractive in terms of learning progress.

RGB-D scenes

The two other datasets consist in artificial buildings constructed from the *RGB-D scene dataset* [30]. 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 8.11. In these maps, the navigation graph is superimposed on the map of the building, where regions are represented by a circle, and allowed transitions between regions

[2] https://www.youtube.com/edit?o=U&video_id=BZgcF-iIBDY

FIGURE 8.10: Navigation graph of the *ENSTA building*

are edges of the graph. The first artificial building, denoted as the *short corridor*, is composed with five of the video sequences, and contains a short corridor of three regions to switch between rooms. Some artificial doors between rooms also enable a pathway between them without using the corridor. The second one, denoted as the *long corridor*, is composed with the eight video sequences and contains three long corridors. The regions constituting the corridor represent almost half the total number of regions in the building.

To construct the navigation graph in each room, we first tried to apply a SLAM algorithm to the video sequences and use the (x, y) positions of the camera to define regions. However, the displacements of the camera were much smaller than the variation of the camera orientation, thus making the (x, y) positions not well-suited to define regions boundaries. To simplify the region definitions, we then cut each of the sequence into five of six sub-sequences of equal length (similar to what was done in Section 8.2.1), and we created an arbitrary trajectory to travel across the sequence. The navigation graph between regions is also arbitrarily designed. The main constraints were to have equally-spaced regions in the corridor, and a building with a plausible size. Lastly, we 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*".

In both datasets, the observations that the robot can take in the corridors are such that nothing salient can be detected. The only observation to be taken in the corridor is a black RGB frame and a segmentation mask of only non salient pixels. This way, it is very uninformative for the robot to travel across the corridor. The large proportion of corridor in the second datasets should then emphasize the behavior of IAC.

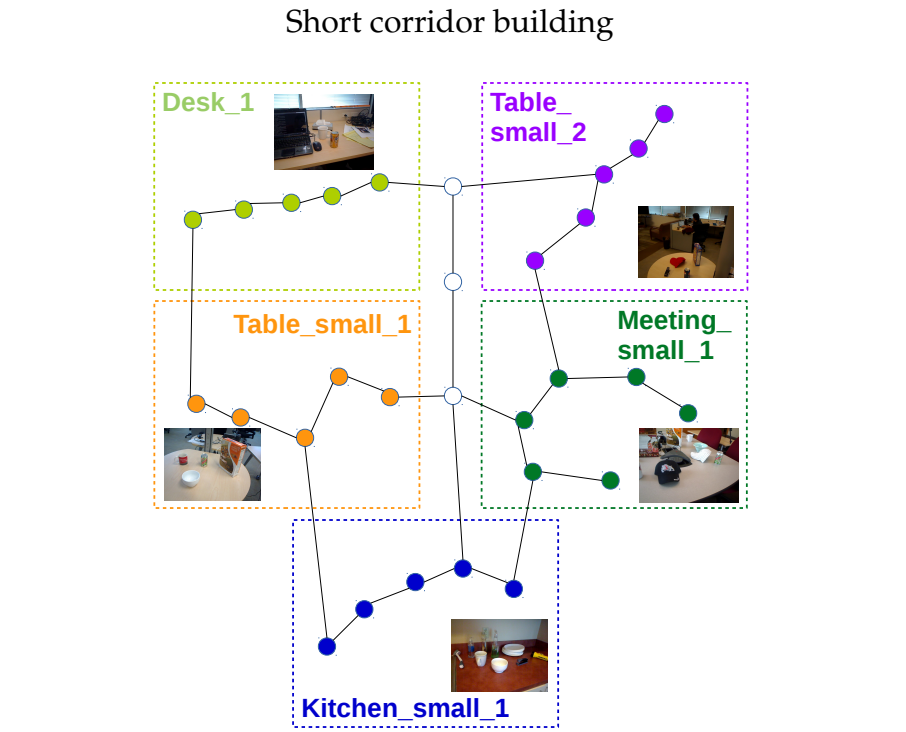


FIGURE 8.11: *short corridor* and *long corridor* artificial buildings created from the *RGB-D scene dataset*

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

TABLE 8.1: Processing time for the main steps of RL-IAC

Displacement simulation procedure

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

1. The robot takes an observation, 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 observation;
6. before taking the next observation, 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 simulated time starting at the beginning of the experiment. This timestamps was then used to plot our results.

To obtain the simulated time, we measure for each iteration the time spent by the system to compute steps 1 to 3 (not simulated), and we add 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 8.1 provides additional measurements to get a better overview of the execution time for each step.

The meta-learner update is much longer for backward than forward evaluation, because saliency evaluation is done on a single frame in one case, and on a set of samples in the other case. This step contains the time required for feature extraction (deep features on GPU), segmentation, and learning progress estimation (which is much faster than the first two steps). The Q-learning training is relatively constant in time, and the learner update processing time was detailed in Chapter 5.

For the robot displacement, we consider 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 points of two adjacent regions. Of course, this measure is an approximation, and more considerations should be taken into account to produce more accurate simulations (path planning for obstacle avoidance, non constant speed, rotation time ...)

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.

The evaluation procedure is very similar to the one used in Section 8.2.1. The *overall error rate* is once again our main evaluation metrics. However, this time, the overall error rate is displayed versus time rather than versus the number of observations. Again, we run 10 times each experiment and consider the average and standard deviation to plot the result curves.

8.3.2 RL-IAC parameters influence

To evaluate and analyze the behavior of RL-IAC, we first examine the influence of some parameters of the system. Because many parameters could be investigated and fine tuned, we only select the most interesting ones and provide a few insight on the way they influence the learning quality.

Robot's displacement speed

To validate the need to consider displacement time in the exploration strategy, we first examine the error rate evolution when the robot is moving at 0.5m/s, and the corresponding evolution in the scenario where the robot can teleport across the building and instantaneously reach a given position.

To simply observe such a behavior, we run experiments on *short corridor* building with four exploration strategies, namely

- **RL-IAC** : the RL-IAC approach described in Chapter 7
- **Random pos.** : after each observation, the robot selects a random position in the building, determines the best sequence of actions (from the navigation graph) to reach it, and moves to that position.
- **Random act.** : after each observation, the robot selects a random action to reach a neighbor region in the navigation graph. It then selects a random position in that region and moves to that position.
- **IAC** : after each observation, the robot selects the most progressing region and a random position in that region. It then determines the best sequence of actions to reach it, and moves to that position.

Note that in the case of **IAC** and **Random pos.**, the robot does not learn in all visited regions, but only the one targeted, once the displacement across the building is finished.

We then display the results of these experiments in two different ways. First, we plot the error rate evolution versus simulated time, based on the considerations described in the previous section. This plot correspond to the behavior of the robot with a $0.5m/s$ speed. Second, we plot the error rate versus the number of observations, without considering the displacement of the robot. This representation is the one used in the IAC evaluation part and roughly corresponds to the evolution if the robot could teleport across the building.

Based on results in Figure 8.12, **RL-IAC** is the most efficient approach when considering time, and IAC is the best when considering observations. As RL-IAC is designed to find a compromise between learning and displacements, the progress between two consecutive observations is not as good as the one obtained with **IAC**.

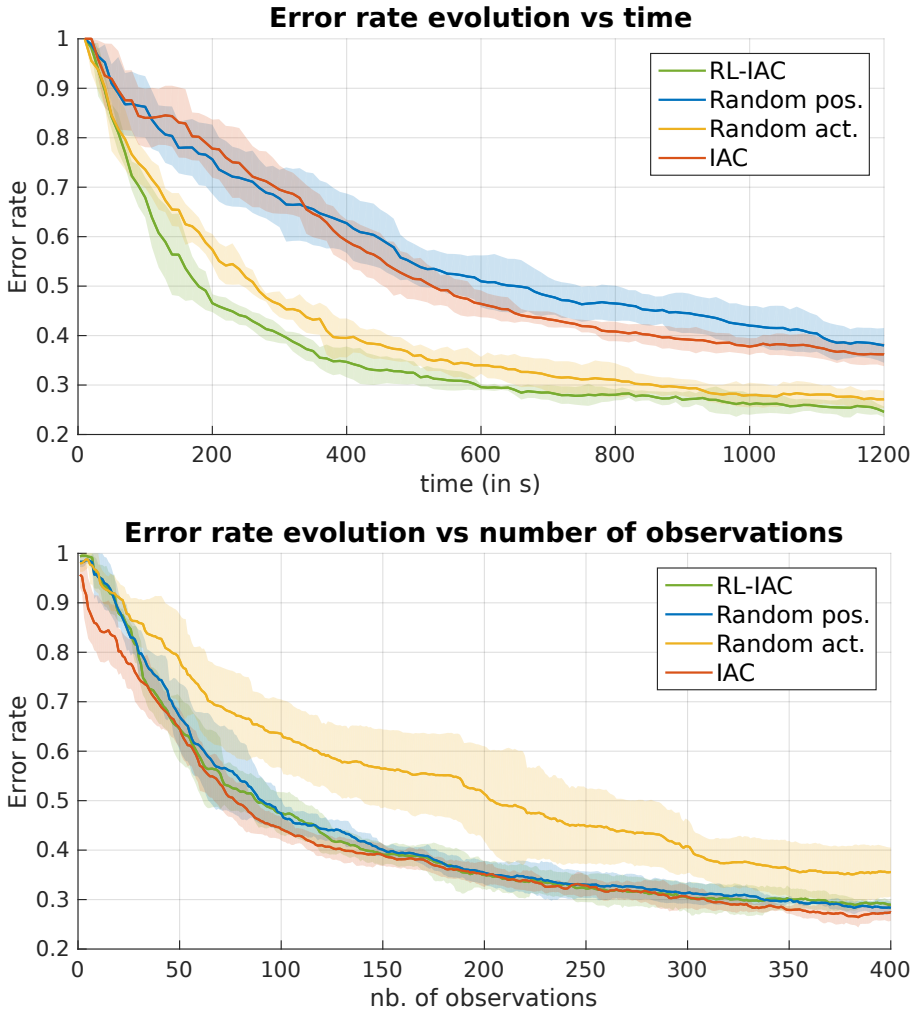


FIGURE 8.12: Error rate evolution versus time or number of frames

Therefore, when displacement time is not represented, **RL-IAC** is not doing much better than random exploration. Lastly, the difference in the performance of **Random pos.** and **Random act.** mainly comes from the fact that **Random pos.** makes long displacements across the building without learning in between, thus having a shape similar to **IAC**. However, **Random pos.** enables a better sampling of the overall environment, as **Random act.** only chooses a displacement among the immediate neighbor regions, thus explaining the better performance when considering the number of observations rather than displacement time.

Discount factor

The second parameter of our discussion is the *discount factor* used in the reinforcement learning process. Recall that the goal of reinforcement learning algorithms is to maximize the amount of cumulated reward R such that

$$R = \sum_{t=1}^{N} \gamma^t r_{t+1} \tag{8.2}$$

r being the immediate reward received at a certain time, and γ being the discount factor. γ is a value between 0 and 1, and represents how much the long term reward

matters in the calculation of the cumulated one. Thus, a small discount factor favors an immediate reward, while a large discount factors favor all rewards equally.

To measure the influence of the discount factor, we run RL-IAC on the three datasets by varying the value of γ (successively 0.1, 0.3, 0.5, 0.7 and 0.9). We report our experimental results in Figure 8.13.

The clearest result is on the *long corridor* dataset: the larger the discount factor, the better. This is probably due to the large amount of uninformative regions the robot has to come across. With an immediate reward the robot will probably never tend to visit rooms that are far away in the building. This behaviour is also visible in the *ENSTA dataset*, where a discount factor of 0.9 seems to provide the best exploration, while a discount factor of 0.1 provides the worst. However, the results are not as clear as they are in the *long corridor dataset*. Lastly, for the *short corridor* dataset, the discount factor does not have a strong influence in the exploration performance. Based on these results, we favor a high discount factor of 0.9 for the remaining experiments.

Forward versus backward evaluation

Third, we compare the performance of RL-IAC using a forward and a backward progress evaluation. We suggested in Chapter 7 that the backward evaluation was better suited to accurately measure the learning progress in the case of a learner shared among regions. To verify this hypothesis, we run again RL-IAC on the three datasets.

We compare the performance obtained with the forward and backward evaluation in Figure 8.14. At least for the *ENSTA* and the *long corridor datasets*, the backward evaluation provides a much better exploration strategy. The results are not as convincing for the *short corridor dataset*. This is probably because of the configuration of this building and the number of regions to visit. Indeed, the restricted number of regions makes the update of the forward estimation much more frequent, and the monitoring of the learning quality in this case more realistic.

8.3.3 Performance evaluation

As a last set of evaluations, we would like to compare the performance of RL-IAC versus other types of exploration strategies and other types of intrinsic motivation. To run the experiments in this section, we took a discount factor of 0.9 and considered the backward evaluation only.

Versus other exploration strategies

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 mobile robotics navigation, the goal is generally to have a good coverage of the environment to explore so as to get an accurate mapping. This coverage approach may be solved by heuristics of NP hard problems (traveling salesman, art gallery problem ...). For example, Osswald *et al.* [123] have used a pre-defined map and navigation graph, and solved a traveling salesman problem before exploration to obtain an efficient exploration route. This route was refined during exploration based on the sensory inputs of the robot. Our goal is not to make a mapping of

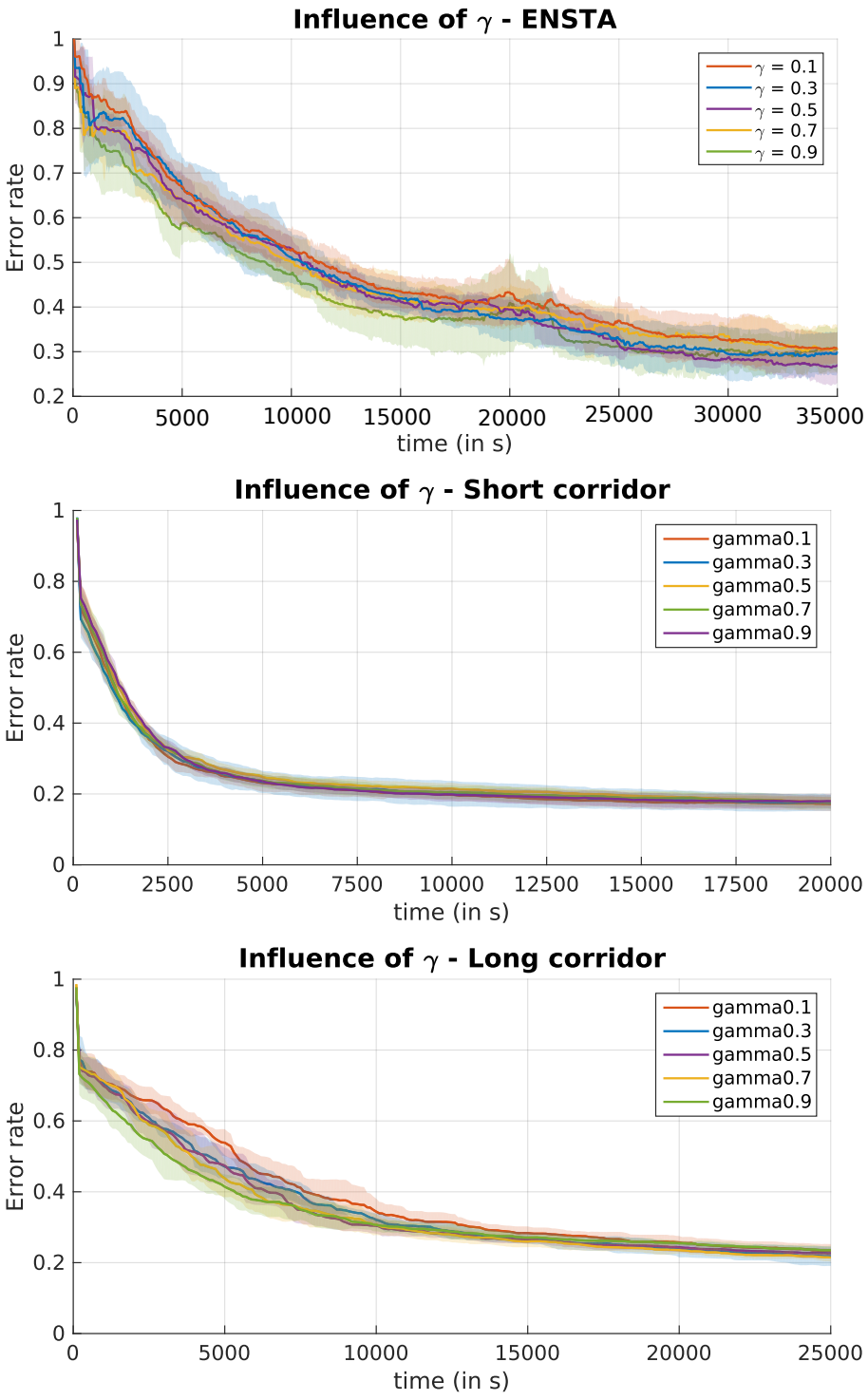


FIGURE 8.13: Influence of the discount factor γ in the action selection process

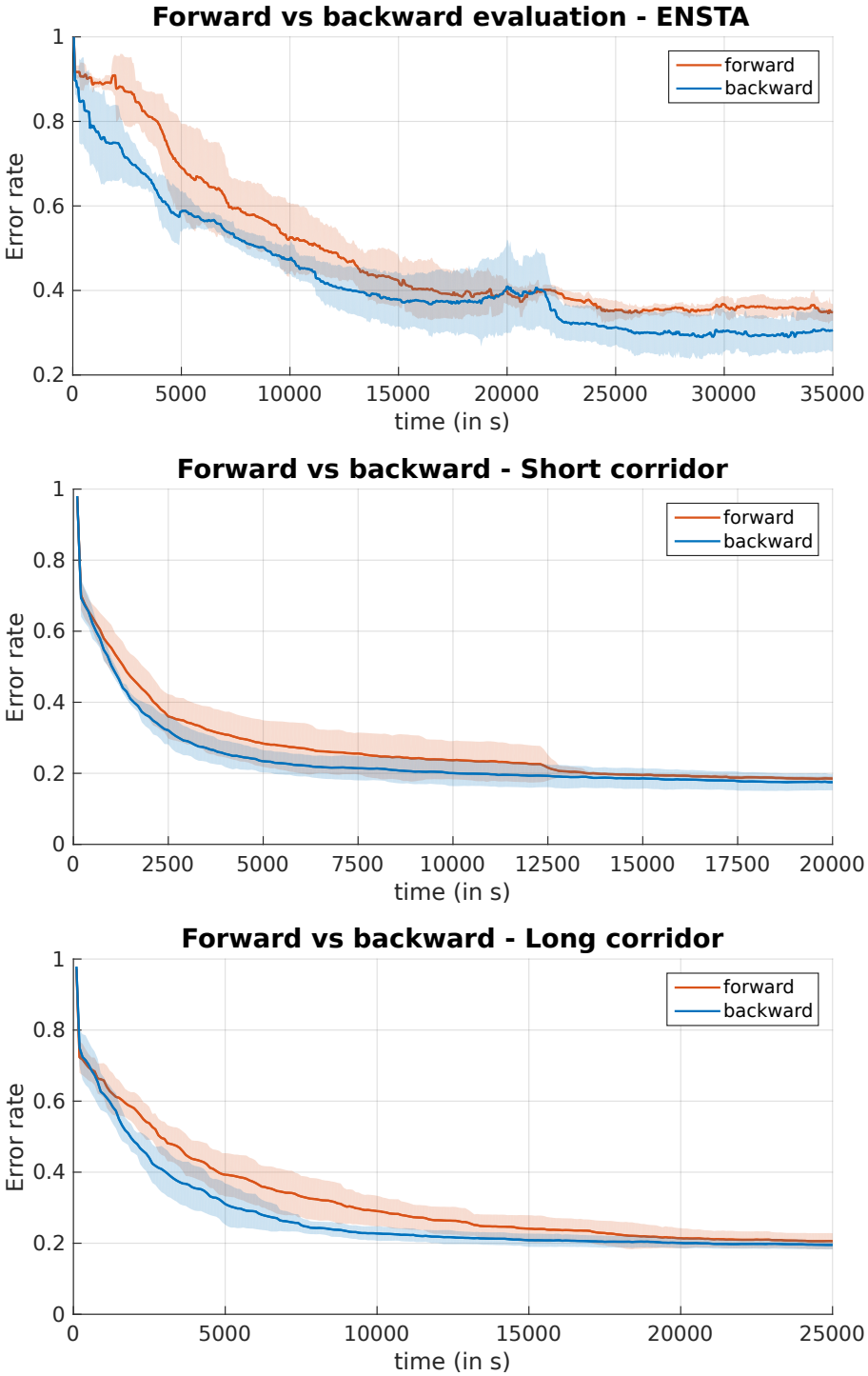


FIGURE 8.14: Forward versus backward learning progress evaluation

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.*, the global path was found with the Concorde software [304] that solves the traveling salesman problem with various heuristics. We used the same software with our regions configurations, and slightly adapted the resulting route to be consistent with the allowed displacements defined by our navigation graphs.

A few approaches rely on learning progress to guide exploration in a reinforcement learning context. In particular, Schmidhuber [262] or Lopes *et al.* [257] 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 running virtual displacement simulations to train our Q-matrix, we simply update it based on Bellman's equation after collecting reward $r(s_k)$ in the current region (s_k)

$$Q(s_k, a_k) = r(s_k) + \gamma \underset{a' \in A}{\text{Max}} Q(s_{k+1}, a') \tag{8.3}$$

We also use an ϵ-greedy approach (50% random) to decide the next action to take.

Figure 8.15 shows the evolution of the *overall error rate* in time on both environments, for the 4 exploration strategies:

- **RL-IAC**: As described in Chapter 7. Selects the next region 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. 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.

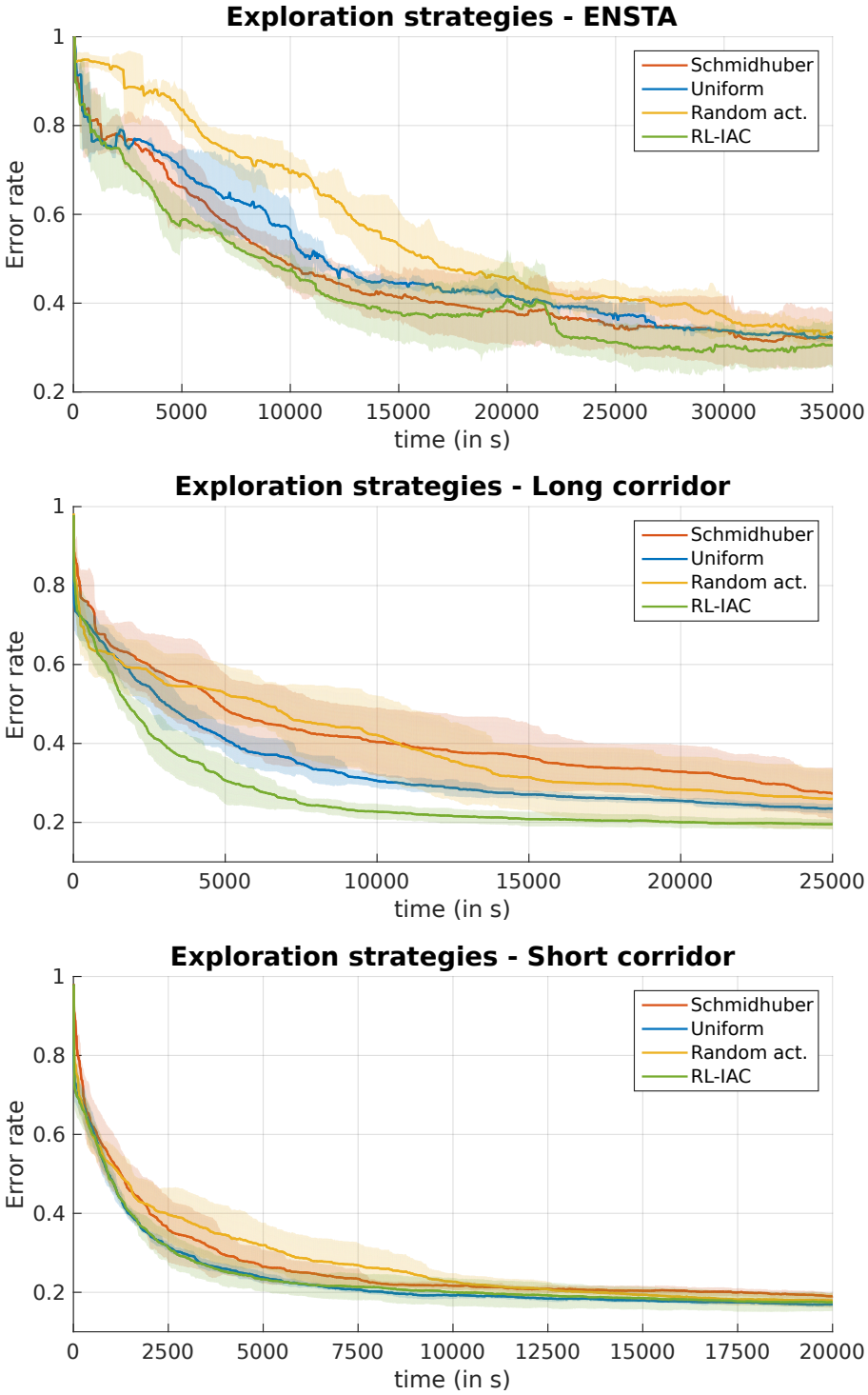


FIGURE 8.15: Error rate evolution for several exploration strategies

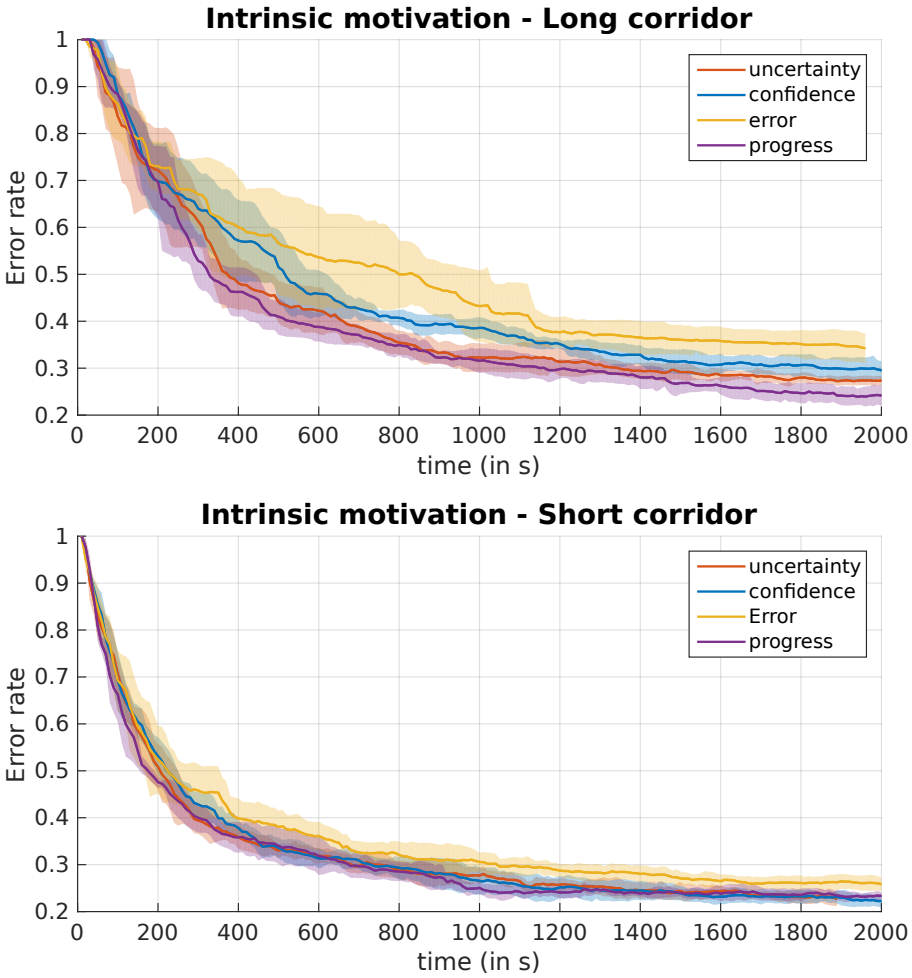


FIGURE 8.16: Comparison of the learning progress versus other kinds of intrinsic motivation techniques

Versus other kinds of intrinsic motivation

So far, we only considered learning progress as our intrinsic motivation factor. The theoretical reasons for such a choice have been widely discussed in the literature [13], [296], but learning progress estimation suffers from several biases in practice. First, a sufficient number of samples is required to make the evaluation of the error statistically significant. Second, at time t, the progress is estimated based on the evolution of the error over the last N samples. The progress estimated this way then represents the progress at time $t - \frac{N}{2}$, so that progress estimation is delayed in time. Therefore, the larger N is, the more accurate, but also delayed in time the progress estimation is. On the other hand, using learning progress makes the robot focus on areas where learning is actually possible and avoids loosing time in unlearnable or trivial ones. This makes learning progress even more efficient when learnable areas represent a small portion of the environment. As a result, a bigger gap should be observed between learning progress and other strategies in environments that are essentially unlearnable.

We are then interested in comparing exploration based on learning progress with other sources of intrinsic motivation. We then propose four factors, namely **Error**, **Confidence** and **Uncertainty**) to replace the learning progress in the RL-IAC

procedure. More precisely, we replaced the backward learning progress $LP_i^{back}(t)$

$$LP_i^{back}(t) = \frac{2}{\pi}|atan(-\beta_i(t))| \tag{8.4}$$

as defined in Section 7.3.3 and used as the reward in the Q-learning training by the following terms:

- **Error**: $Err_i^{back}(t)$ (see Equation 7.1). The agent is rewarded by exploring regions with high prediction error.
- **Confidence**: $Conf_i(t) = \frac{1}{t_i}$, t_i the number of frames observed in region R_i. The agent is rewarded by exploring regions in which confidence is low (having the fewest number of observations).
- **Uncertainty**: $Unc_i(t) = \sum_{j=1}^{t_i} -|E_j^{L_t} - 0.5|$, where $E_j^{L_t}$ are the samples of the *regional estimation set* $\mathcal{E}_{t_i}^{L_t}$, described in Section 7.3.3. These samples are values between 0 and 1 and represent the probability of a sample to be salient. As a result, the closer to 0.5, the most "uncertain" the learner is about the sample. This measure was already used and detailed in Craye *et al.* [20]. In this configuration, the robot is then rewarded by exploring regions producing fuzzy saliency maps.

We compare the performance of RL-IAC with the aforementioned intrinsic motivations on the *long corridor* and *short corridor datasets*. The comparison is displayed in Figure 8.16. In both configurations, progress seems to be the best source of intrinsic motivation, but the difference is much more significant in the *long corridor* configuration. In this configuration, the **error** fails as the absence of positive samples keeps the error rate very high in the whole corridor[3]. **Confidence** spends as much time in the corridor as in the rooms, thus making learning much slower. **Uncertainty** is the only one able to avoid the corridor, as, in the absence of positive samples in the model, saliency maps are set to '0' by default everywhere, thus making output saliency maps not fuzzy at all.

8.4 Conclusion

In this chapter, we separately evaluated IAC and RL-IAC for the task of saliency learning. To demonstrate the generalization capability of our approach, we evaluated them on several datasets. As a summary and concluding remarks, our experimental evaluation led us to the following findings:

First, IAC shows a clear behavior of methodical exploration of the environment. When driven by IAC, the robot first focuses at regions where learning progress is high. Then, as learning progress decreases in these regions, the robot tends to explore more complex ones and spends more time in them to acquire knowledge.

Second, when a significant change occurs in the environment (for example, if all salient objects are removed), IAC makes the robot still able to adapt to the most progressing regions. Nevertheless, this adaptation is not immediate because of a delayed evaluation of the learning progress.

Third, when displacement time is not negligible, IAC can reveal very inefficient. Indeed, focusing on the most progressing region forces the robot to perform long

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

displacements that could be used to learn in other regions nearby. As a result, the use of RL-IAC makes sense in large environments such as buildings. Nevertheless, when measuring the learning evolution in terms of number of observations rather than in time, IAC still performs better than RL-IAC.

Fourth, the parameters of the system have a strong influence on the performance. We demonstrated that the backward evaluation was, in general, better than the forward one, and that a high discount factor was preferable in a large environment. Nevertheless, IAC and RL-IAC's parameters are complex to optimize, and there is no rule of thumb to determine them before the experiment.

Lastly, both IAC and RL-IAC have been found to be efficient exploration strategies as compared with more extensive and uniform kinds of exploration. In addition, learning progress has been found to be an appropriate source of intrinsic motivation, as compared with other sources unable to detect unlearnable areas of the environment. Nevertheless, those findings are much more significant on large environments where the proportion of uninformative regions is high.

Chapter 9

Discussions and perspectives

In this works we have presented several contributions, both in the field of visual attention and intrinsically motivated exploration. We summarize in this concluding chapter the proposed approach and highlight the main contributions of this work. We then discuss the strength and limitations of our system under a theoretical and experimental points of view. Lastly, we open the discussion by suggesting future work directions and possible applications of our method.

9.1 Contributions of this work

9.1.1 Incremental learning of visual saliency

Summary

The first part of the manuscript was dedicated to the description and evaluation of a method for learning a model of visual saliency directly on a robot. In our experiments, the saliency is specialized in the detection of objects in indoor environments, but could be generalized to other applications.

To do so, we have used a modular block architecture of three main components. First, a feature extractor able to encode the characteristics of an image at the pixel level. Second, a self-annotation process able to provide a partial but accurate state of the saliency on the image. Lastly, a classifier combining features and partial saliency annotation to learn a model and generalize over the partial signal.

Our system then works in two modes. On the one hand, a learning mode, integrating on the fly the visual inputs perceived by the robot into a model of visual saliency. On the other hand, an inference mode, using the learned model to generate saliency maps.

Contributions

Unlike most saliency methods proposed by the state-of-the-art, our approach is incremental. This means that the current available model of saliency can be learned directly on the robot, within its operating environment. The advantages of such approach are threefold: first, it does not require a long dataset preparation and learning period. Unlike saliency approaches based on machine learning techniques, ours does not require any prior model before the experiment, and gets better as exploration goes. Second, the produced saliency maps are adapted to the environment and the task the robot should accomplish (this is a top-down kind of saliency). Lastly, the saliency model remains flexible to any change in the environment by simply updating the model continuously.

Unlike object detectors based on geometrical segmentation, our saliency model is able to provide a probability of a pixel to belong to an object on the whole image.

Indeed, geometrical segmentation techniques are often unable to provide a result on the entire image frame, because of the quality of the sensor, or the reliability of the segmentation algorithm.

9.1.2 Intrinsically motivated exploration

Summary

In the second part of the manuscript, we investigated the methods to drive the robot exploration in order to improve the quality of the saliency model. We directed our research towards developmental robotics and the area of intrinsic motivations. We got particularly interested in the Intelligent Adaptive Curiosity algorithm, and designed an adapted version for the task of saliency learning.

IAC is a mechanism able to drive a robot's actions, based on the current state of knowledge, towards areas of the space likely to help improving this knowledge. This method conveys a methodical and efficient exploration strategy, by choosing actions that are adapted to the current skills of the robot: not to complex, not too trivial.

IAC relies on a measure of the local learning progress, estimated in regions of the exploration space. The actions taken by the robot are then selected so as to favor regions having the highest learning progress. This means that the robot focuses the attention on the most progressing region and tries to gain more knowledge in there. As knowledge increases, the learning progress vanishes, until a more progressing region catches the robot's attention.

Contribution

Our contribution here is the adaptation of the IAC algorithm for the task of saliency learning in general, and saliency learning on a mobile robot. This adaptation involved a number of design choices such as regions definition, learning error measurement, progress estimation, or action policy. This required a deep understanding of the algorithm to make IAC consistent and efficient for our application. In particular, we proposed a new way of estimating learning error in cases where the model to learn by IAC was shared among the regions. We called this error estimation the *backward error estimation*.

Moreover, we found that IAC was not integrating the time for processing action in the decision scheme. For a mobile robot, this factor was too important to be omitted. We therefore created an extension of IAC, called RL-IAC (for *Reinforcement Learning-Intelligent Adaptive Curiosity*) to overcome this limitation and ensure an efficient exploration strategy for mobile robots. RL-IAC then relies on reinforcement learning to find a good trade off between learning progress attraction and displacement time.

Aside from the improvements of the algorithm itself, this work was also the occasion to propose an original exploration strategy to the fields of foveated visual attention, and the one of mobile robotics. In visual attention setups, exploration is typically guided by saliency maps, and the focus is targeted at the most salient regions. We took in this work the problem the other way around by learning saliency with intrinsic motivation. In addition, the use of intrinsic motivation is not very common for mobile robotics. We then proposed an exploration approach based on knowledge optimization rather than the amount of information collected in the environment.

9.1.3 Experimental setups

As a last contribution, our work was tested and evaluated on several experimental setups. In particular, we constructed two different datasets from two robotics platforms.

The first platform is a bio-inspired head placed on a table and able to observe its environment by performing saccades and fixations. Equipped with both foveal and contextual cameras, this robot is an excellent platform for demonstrating bio-inspired visual attention mechanisms.

The second one is a mobile robot equipped with an RGB-D sensor and a laser range finder. This robot is able to move across indoor environments such as rooms or entire buildings while building a map and localizing itself in this environment.

Our contribution then consisted in creating sequence datasets on those two platforms so as to be able to efficiently evaluate our work. We also processed all the recorded data, formatted them and annotated them to perform a numerical evaluation.

To evaluate the performance of IAC and RL-IAC, we also adapted our datasets to create a simulated environment in which the robot was able to move and collect images, while avoiding the constraints of a robot navigating across a real building. Moreover, these simulated environments allowed us to repeat the experiments several times, to precisely analyze the behavior of our algorithms, and to compare our approach with other exploration strategies.

9.2 Discussions and current limitations

This work is still an ongoing research that could be largely improved and extended. We discuss in this section the strength and limitations of our approach.

9.2.1 Incremental learning of visual saliency

Strengths of the method

The incremental learning of visual saliency was evaluated on four different datasets and has shown a very similar behavior in each of them. We demonstrated that the method was able to generalize saliency over a weak supervision signal, and that the produced saliency maps, when evaluated on a similar environment than the one the model was trained in, were more accurate than state-of-the-art techniques. Lastly, we evaluated the saliency quality on a dataset with a model of saliency trained on another environment. The results were still acceptable and outperforming state-of-the-art. As a results, the similar behavior in each dataset suggests that the method is robust and should work in many types of indoor environments.

In addition, the modular architecture of our system makes it easy to replace components with more efficient ones. We already demonstrated this ability by testing several types of feature extractors and two types of segmentation masks. Using a new classifier, or even using the system for another related problem (joint object detection and recognition for example) should not require a strong re-implementation effort.

Current limitations

A major limitation of our approach is the strong dependence on the learning signal (the product of the object segmentation process). For saliency to be correctly

learned, the segmentation needs to be reliable and adapted to the environment. If the robots explores an environment where so many objects are piled-up that the floor cannot be detected, the whole process will fail. As a result, the design of a reliable segmentation process is a potentially complex procedure that must be carefully treated prior to any experiment. This is the cost for having a self-supervised learning rather than a supervised one.

Another potential problem is that the algorithm is not completely incremental. As random forests are not designed to be incremental, the training set and the training time necessarily increases as exploration progresses. This aspect is limited by the process of forgetting oldest samples, but this approach is not viable in a lifelong learning scenario.

Lastly, the generalization capacity of the model is hard to predict. When testing the saliency algorithm in a new environment, there is absolutely no warranty about the performance.

9.2.2 Intrinsically motivated exploration

Strengths of the method

The implementation of IAC for the task of saliency learning has demonstrated a set of interesting behavioral properties and good performance on several datasets.

We demonstrated that exploration with this principle was methodical, but also logical. When monitoring the learning progress in each region, the exploration priority followed by the robot was intuitive and understandable. Equipped with this kind of exploration drive, our experiments have shown that the robot was behaving in a developmental way, thus reinforcing the findings of Oudeyer *et al.*in previous experiments on other problems [14], [190], [268].

The integration of reinforcement learning in the process of action selection was also a successful solution to the problem of long and varying displacement times to get new observations. We demonstrated with several experiments that exploration with RL-IAC was still efficient in this type of scenarios.

We also demonstrated the ability of IAC to be flexible to changes in the environment, by adapting the exploration strategy and refocus after some time on the areas of high learning progress. This property is extremely important in the scope of a realistic lifelong learning experiment.

Current limitations

In spite of promising and interesting results, IAC and RL-IAC still suffer from a few limitations. First, this kind of exploration strategies only show a strong advantage over other techniques when the environment contains a large fraction or trivial or unlearnable areas. Otherwise, a regular exploration pattern is likely to show the same performance.

In addition, IAC depends on many factors, and uses learning progress measurement at the heart the process. By definition, learning progress is obtained by approximating the derivative of an approximation of the learning error. This succession of approximation is likely to make the measure of progress unstable and unreliable. Our own experience feedback is that IAC is globally hard to set up, and required a lot of trials to get a good insight on the influence of the parameters.

Moreover, the action policy proposed by RL-IAC is not optimal and only provides a reasonable solution given the current configuration. We believe that further

theoretical and experimental investigations should be conducted to create a more satisfying model.

Lastly, we highlighted the adaptation behavior in the case of a change in the environment of the robot, but the environment still needs to be globally static. In a very dynamic environment, the measure of progress will most likely not be reliable.

9.3 Perspectives

To conclude this manuscript, we suggest a few short and long term perspectives that would be worth investigating in a future work.

9.3.1 Short-term perspectives

Coupling saliency with object recognition

Saliency maps are not an end in themselves. We provided in this work a way to learn them, but they are meaningless if not used in another context. A first possible plan would be to integrate them into an object recognition framework. We already suggested a way to use them to improve the proposal of agnostic bounding boxes around objects, but this could be largely improved to provide a framework able to both localize and identify objects within their environments.

Make RL-IAC fully incremental

In this work, we presented separately a method for building incremental maps and navigation graphs of the environment, and a method using a navigation graph to explore the environment. Those two are used one after the other, but everything has been done to build the navigation graph and autonomously explore at the same time. A few implementation efforts should be sufficient to make RL-IAC fully incremental.

Using more biologically plausible models

For practical reasons, the regions defined in RL-IAC are very arbitrarily splited and could be largely improved with biologically plausible components. For example, considering gist aspects to make regions more visually and biologically consistent is a possible approach. In addition, the use of an ϵ-greedy approach to guide exploration could be replaced by more biologically plausible stochastic transition models. Thus, the displacement policy could rely on the way human deploy their visual gaze in a scene which is not steered by a greedy process. In the BioVision scenario, fixation and saccades could then be driven by curiosity, but also the distribution of saccade amplitudes, orientations, the fixation durations and so on measured on human populations. To go one step further, and for the future research, it would make sense to consider some aspects used by saccadic models of the literature.

9.3.2 Long-term perspectives

Integrating the method in an end-to-end deep architecture

In this work, we have shown the promising performance of deep features and highlighted the limitation of random forests as a non incremental classifier. The next

algorithmic step would be to integrate the saliency learning technique into an end-to-end deep learning architecture. The biggest challenge would then probably be the one of making the network both incremental and stable.

Because of the flexibility of neural networks, this architecture would have very interesting properties. First, the network could not only learn saliency, but also bounding boxes detection, and even eventually recognition. Ideally, the system should perform similar tasks than an end-to-end system like YOLO [105], but incrementally learn this task for a dedicated environment.

Additionally, coupling neural network and intrinsic motivation (and especially learning progress) would be a very exciting area of investigation. Research in this area are at their early stages [305], but there are for sure many interesting ways to explore in that direction.

Testing RL-IAC in a real environment

Our experiments have been so far limited to the recording of video sequences on a robot, and to the simulation of the robot displacements on virtual environments. Now that our method seems to be efficient in this context, it would be interesting to verify its performance in a real life scenario.

Given the estimations provided in the last experimental chapter, we suggested that the experiment should last for about eight hours for 3000 observation. This is a considerably long experiment for a mobile robot that will very likely lack batteries before the end, and needs a very reliable hardware and software to run for so long.

To make this possible, the robot would also need an efficient obstacle detection and avoidance mechanism, able to determine a realistic and reliable path between two locations, while avoiding potential obstacle on the way. The navigation graph should also be constantly updated based on the evolution of the environment (doors closed then opened, presence of humans, *etc*...).

The problem of discovering the exploration space at the first time is also a question to consider. Should we define boundaries of the experiment to make sure the robot does not explore dangerous or unexpected places (stairs for example).

To simplify, but also to enhance the experiment, the presence of a human assistant or teacher, helping the robot would be an interesting alternative.

A potential application of IAC for active image annotations

A last interesting future work could be related to the integration of IAC in an interactive manual annotation software.

Using active learning to simplify the task of image annotation is not a new idea [306]. Nevertheless, IAC would be perfectly suited to this task, and the work done in the scope of this thesis would be a good starting point to develop this idea.

Suppose that a video sequence needs to be manually annotated by providing bounding boxes around objects of interest. IAC would then be applied in the following manner: the sequence could be divided into subsequence of arbitrary length to define the regions. IAC would propose to the user selected frames and ask for a manual annotation. This annotation would serve as a learning signal, and would replace the segmentation mask of our current approach. Thus the user would not have to annotate the entire dataset, but only a set of frames selected by the algorithm to improve its own model. IAC should naturally avoid frames that are trivial

(containing no annotation for example), and progressively focus on the most complex cases. Lastly, once enough confident in the model, the remaining frames could be annotated automatically.

Index

Bibliography

[1] A. M. Turing, "Computing machinery and intelligence", *Mind*, vol. 59, no. 236, pp. 433–460, 1950.

[2] B. Goertzel, M. Iklé, and J. Wigmore, "The architecture of human-like general intelligence", in *Theoretical Foundations of Artificial General Intelligence*, Springer, 2012, pp. 123–144.

[3] H. Sak, A. Senior, K. Rao, and F. Beaufays, "Fast and accurate recurrent neural network acoustic models for speech recognition", *arXiv arXiv:1507.06947*, 2015.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks", in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[5] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané, "Concrete problems in ai safety", *arXiv preprint arXiv:1606.06565*, 2016.

[6] L. M. Gonçalves, A. A. Oliveira, and R. A. Grupen, "A framework for attention and object categorization using a stereo head robot", in *Computer Graphics and Image Processing, 1999. Proceedings. XII Brazilian Symposium on*, IEEE, 1999, pp. 143–152.

[7] M Bjorkman and D. Kragic, "Combination of foveal and peripheral vision for object recognition and pose estimation", in *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, IEEE, vol. 5, 2004, pp. 5135–5140.

[8] J. Hawkins and D. George, "Hierarchical temporal memory: Concepts, theory and terminology", *Whitepaper, Numenta Inc*, 2006.

[9] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition", *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[10] C. D. Gilbert and W. Li, "Top-down influences on visual processing", *Nature Reviews Neuroscience*, vol. 14, no. 5, pp. 350–363, 2013.

[11] J. Weng, J. McClelland, A. Pentland, O. Sporns, I. Stockman, M. Sur, and E. Thelen, "Autonomous mental development by robots and animals", *Science*, vol. 291, no. 5504, pp. 599–600, 2001.

[12] V. Tikhanoff, A. Cangelosi, P. Fitzpatrick, G. Metta, L. Natale, and F. Nori, "An open-source simulator for cognitive robotics research: The prototype of the icub humanoid robot simulator", in *Proceedings of the 8th workshop on performance metrics for intelligent systems*, ACM, 2008, pp. 57–61.

[13] P.-Y. Oudeyer, F. Kaplan, and V. V. Hafner, "Intrinsic motivation systems for autonomous mental development", *Evolutionary Computation, IEEE Transactions on*, vol. 11, no. 2, pp. 265–286, 2007.

[14] P.-Y. Oudeyer, A. Baranes, and F. Kaplan, "Intrinsically motivated learning of real-world sensorimotor skills with developmental constraints", in *Intrinsically motivated learning in natural and artificial systems*, Springer, 2013, pp. 303–365.

[15] L. Itti, "Visual salience", *Scholarpedia*, vol. 2, no. 9, p. 3327, 2007.

[16] Z. Bylinskii, T. Judd, F. Durand, A. Oliva, and A. Torralba, *Mit saliency benchmark*, http://saliency.mit.edu/.

[17] J. Harel, C. Koch, and P. Perona, "Graph-based visual saliency", in *Advances in neural information processing systems*, 2006, pp. 545–552.

[18] L.-C. Caron, D. Filliat, and A. Gepperth, "Neural network fusion of color, depth and location for object instance recognition on a mobile robot", in *Computer Vision-ECCV 2014 Workshops*, Springer, 2014, pp. 791–805.

[19] C. Craye, D. Filliat, and J.-F. Goudou, "Apprentissage incrémental de la saillance visuelle pour des applications robotique", in *Journées francophones des jeunes chercheurs en vision par ordinateur*, 2015.

[20] ——, "Exploration strategies for incremental learning of object-based visual saliency", in *2015 Joint IEEE International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob)*, IEEE, 2015, pp. 13–18.

[21] C. Craye, D. Filliat, and J. Goudou, "Environment exploration for object-based visual saliency learning", in *Robotics and Automaton (ICRA), 2016 IEEE International Conference on*, 2016, pp. 3140–3148.

[22] ——, "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.

[23] C. Craye, D. Filliat, and J.-F. 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.

[24] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, *et al.*, "Imagenet large scale visual recognition challenge", *arXiv preprint arXiv:1409.0575*, 2014.

[25] 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, 2011.

[26] G. Bradski and A. Kaehler, *Learning opencv: Computer vision with the opencv library*. " O'Reilly Media, Inc.", 2008.

[27] R. B. Rusu and S. Cousins, "3d is here: Point cloud library (pcl)", in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, IEEE, 2011, pp. 1–4.

[28] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding", in *Proceedings of the 22nd ACM international conference on Multimedia*, ACM, 2014, pp. 675–678.

[29] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: An open-source robot operating system", in *ICRA workshop on open source software*, Kobe, vol. 3, 2009, p. 5.

[30] K. Lai, L. Bo, X. Ren, and D. Fox, "A large-scale hierarchical multi-view rgb-d object dataset", in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, IEEE, 2011, pp. 1817–1824.

[31] A. Ciptadi, T. Hermans, and J. 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)*, 2013, pp. 9–13.

[32] L. Itti and C. Koch, "Computational modelling of visual attention", *Nature reviews neuroscience*, vol. 2, no. 3, pp. 194–203, 2001.

[33] J. M. Henderson, "Human gaze control during real-world scene perception", *Trends in cognitive sciences*, vol. 7, no. 11, pp. 498–504, 2003.

[34] S. Frintrop, E. Rome, and H. I. Christensen, "Computational visual attention systems and their cognitive foundations: A survey", *ACM Transactions on Applied Perception (TAP)*, vol. 7, no. 1, p. 6, 2010.

[35] A. Borji and L. Itti, "State-of-the-art in visual attention modeling", *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 35, no. 1, pp. 185–207, 2013.

[36] A. L. Yarbus, *Eye movements during perception of complex objects*. Springer, 1967.

[37] Depawn, *Sights and sighs*, http://www.depauw.edu/news-media/ latest-news/details/28688/.

[38] L. Itti, C. Koch, and E. Niebur, "A model of saliency-based visual attention for rapid scene analysis", *IEEE Transactions on pattern analysis and machine intelligence*, vol. 20, no. 11, pp. 1254–1259, 1998.

[39] O. Le Meur, P. Le Callet, D. Barba, and D. Thoreau, "A coherent computational approach to model bottom-up visual attention", *IEEE transactions on pattern analysis and machine intelligence*, vol. 28, no. 5, pp. 802–817, 2006.

[40] X. Hou and L. Zhang, "Saliency detection: A spectral residual approach", in *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, IEEE, 2007, pp. 1–8.

[41] X. Hou, J. Harel, and C. Koch, "Image signature: Highlighting sparse salient regions", *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 34, no. 1, pp. 194–201, 2012.

[42] N. Bruce and J. Tsotsos, "Attention based on information maximization", *Journal of Vision*, vol. 7, no. 9, pp. 950–950, 2007.

[43] S. S. Kruthiventi, K. Ayush, and R. V. Babu, "Deepfix: A fully convolutional neural network for predicting human eye fixations", *arXiv arXiv:1510.02927*, 2015.

[44] A. Borji, "Boosting bottom-up and top-down visual features for saliency estimation", in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, IEEE, 2012, pp. 438–445.

[45] B. Rasolzadeh, M. Björkman, K. Huebner, and D. Kragic, "An active vision system for detecting, fixating and manipulating objects in the real world", *The International Journal of Robotics Research*, vol. 29, no. 2-3, pp. 133–154, 2010.

[46] S. Frintrop, *Vocus: A visual attention system for object detection and goal-directed search*. Springer, 2006, vol. 3899.

[47] F. Orabona, G. Metta, and G. Sandini, "A proto-object based visual attention model", in *Attention in cognitive systems. Theories and systems from an interdisciplinary viewpoint*, Springer, 2007, pp. 198–215.

[48] S. Frintrop, T. Werner, and G. M. García, "Traditional saliency reloaded: A good old model in new shape", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 82–90.

[49] L. Paletta, G. Fritz, and C. Seifert, "Q-learning of sequential attention for visual object recognition from informative local descriptors", in *Proceedings of the 22nd international conference on Machine learning*, ACM, 2005, pp. 649–656.

[50] O. Le Meur, P. Le Callet, and D. Barba, "Predicting visual fixations on video based on low-level visual features", *Vision research*, vol. 47, no. 19, pp. 2483–2498, 2007.

[51] H. Peng, B. Li, W. Xiong, W. Hu, and R. Ji, "Rgbd salient object detection: A benchmark and algorithms", in *Computer Vision–ECCV 2014*, Springer, 2014, pp. 92–109.

[52] G. M. García, E. Potapova, T. Werner, M. Zillich, M. Vincze, and S. Frintrop, "Saliency-based object discovery on rgb-d data with a late-fusion approach", in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2015, pp. 1866–1873.

[53] F. Rea, G. Metta, and C. Bartolozzi, "Event-driven visual attention for the humanoid robot icub", *Neuromorphic Engineering Systems and Applications*, p. 23, 2015.

[54] J. Najemnik and W. S. Geisler, "Optimal eye movement strategies in visual search", *Nature*, vol. 434, no. 7031, pp. 387–391, 2005.

[55] W. Wang, C. Chen, Y. Wang, T. Jiang, F. Fang, and Y. Yao, "Simulating human saccadic scanpaths on natural images", in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, IEEE, 2011, pp. 441–448.

[56] H. R. Tavakoli, E. Rahtu, and J. Heikkilä, "Stochastic bottom–up fixation prediction and saccade generation", *Image and Vision Computing*, vol. 31, no. 9, pp. 686–693, 2013.

[57] O. Le Meur and Z. Liu, "Saccadic model of eye movements for free-viewing condition", *Vision research*, vol. 116, pp. 152–164, 2015.

[58] O. Le Meur and A. Coutrot, "Introducing context-dependent and spatially-variant viewing biases in saccadic models", *Vision research*, vol. 121, pp. 72–84, 2016.

[59] Q. Zhao and C. Koch, "Learning a saliency map using fixated locations in natural scenes", *Journal of vision*, vol. 11, no. 3, p. 9, 2011.

[60] S. Vijayakumar, J. Conradt, T. Shibata, and S. Schaal, "Overt visual attention for a humanoid robot", in *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, IEEE, vol. 4, 2001, pp. 2332–2337.

[61] Y. Xie, L.-f. Liu, C.-h. Li, and Y.-y. Qu, "Unifying visual saliency with hog feature learning for traffic sign detection", in *Intelligent Vehicles Symposium, 2009 IEEE*, IEEE, 2009, pp. 24–29.

[62] N. Imamoglu, W. Lin, and Y. Fang, "A saliency detection model using low-level features based on wavelet transform", *IEEE transactions on multimedia*, vol. 15, no. 1, pp. 96–105, 2013.

[63] K. Gao, S. Lin, Y. Zhang, S. Tang, and H. Ren, "Attention model based sift keypoints filtration for image retrieval", in *Computer and Information Science, 2008. ICIS 08. Seventh IEEE/ACIS International Conference on*, IEEE, 2008, pp. 191–196.

[64] A. Torralba, "Modeling global scene factors in attention", *JOSA A*, vol. 20, no. 7, pp. 1407–1418, 2003.

[65] M.-M. Cheng, G.-X. Zhang, N. J. Mitra, X. Huang, and S.-M. Hu, "Global contrast based salient region detection", in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, IEEE, 2011, pp. 409–416.

[66] J. Zhang and S. Sclaroff, "Saliency detection: A boolean map approach", in *Computer Vision (ICCV), 2013 IEEE International Conference on*, IEEE, 2013, pp. 153–160.

[67] E. Erdem and A. Erdem, "Visual saliency estimation by nonlinearly integrating features using region covariances", *Journal of vision*, vol. 13, no. 4, p. 11, 2013.

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

[69] G. Li and Y. Yu, "Visual saliency based on multiscale deep features", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 5455–5463.

[70] D. A. Klein and S. Frintrop, *Salient pattern detection using w 2 on multivariate normal distributions*. Springer, 2012.

[71] J. Li, Y. Tian, and T. Huang, "Visual saliency with statistical priors", *International Journal of Computer Vision*, vol. 107, no. 3, pp. 239–253, 2014.

[72] T. Judd, K. Ehinger, F. Durand, and A. Torralba, "Learning to predict where humans look", in *2009 IEEE 12th International Conference on Computer Vision*, IEEE, 2009, pp. 2106–2113.

[73] Y. Kavak, E. Erdem, and A. Erdem, "Visual saliency estimation by integrating features using multiple kernel learning", *CoRR*, vol. abs/1307.5693, 2013.

[74] F. 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*, vol. 100, no. 1, pp. 64–106, 2005.

[75] S.-B. Choi, S.-W. Ban, and M. Lee, "Biologically motivated visual attention system using bottom-up saliency map and top-down inhibition", *Neural Information Processing-Letters and Review*, vol. 2, no. 1, 2004.

[76] J. K. Tsotsos, S. M. Culhane, W. Y. Kei Wai, Y. Lai, N. Davis, and F. Nuflo, "Modeling visual attention via selective tuning", *Artificial intelligence*, vol. 78, no. 1, pp. 507–545, 1995.

[77] M. Cerf, J. Harel, W. Einhäuser, and C. Koch, "Predicting human gaze using low-level saliency combined with face detection", in *Advances in neural information processing systems*, 2008, pp. 241–248.

[78] H. Li and K. N. Ngan, "Saliency model-based face segmentation and tracking in head-and-shoulder video sequences", *Journal of Visual Communication and Image Representation*, vol. 19, no. 5, pp. 320–333, 2008.

[79] S. Goferman, L. Zelnik-Manor, and A. Tal, "Context-aware saliency detection", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 10, pp. 1915–1926, 2012.

[80] R. Zhao, W. Ouyang, H. Li, and X. Wang, "Saliency detection by multi-context deep learning", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1265–1274.

[81] G. W. Humphreys and H. J. Muller, "Search via recursive rejection (serr): A connectionist model of visual search", *Cognitive Psychology*, vol. 25, no. 1, pp. 43–110, 1993.

[82] M. Kümmerer, L. Theis, and M. Bethge, "Deep gaze i: Boosting saliency prediction with feature maps trained on imagenet", *arXiv preprint arXiv:1411.1045*, 2014.

[83] M. Oquab, L. Bottou, I. Laptev, and J. 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*, 2015, pp. 685–694.

[84] Y. Gitman, M. Erofeev, D. Vatolin, and B. Andrey, "Semiautomatic visual-attention modeling and its application to video compression", in *2014 IEEE International Conference on Image Processing (ICIP)*, IEEE, 2014, pp. 1105–1109.

[85] A. Borji and L. Itti, "Cat2000: A large scale fixation dataset for boosting saliency research", *arXiv preprint arXiv:1505.03581*, 2015.

[86] T. Liu, Z. Yuan, J. Sun, J. Wang, N. Zheng, X. Tang, and H.-Y. Shum, "Learning to detect a salient object", *IEEE Transactions on Pattern analysis and machine intelligence*, vol. 33, no. 2, pp. 353–367, 2011.

[87] T. Judd, F. Durand, and A. Torralba, "A benchmark of computational models of saliency to predict human fixations", 2012.

[88] K. Koehler, F. Guo, S. Zhang, and M. P. Eckstein, "What do saliency models predict?", *Journal of vision*, vol. 14, no. 3, pp. 14–14, 2014.

[89] M.-M. Cheng, N. J. Mitra, X. Huang, P. H. Torr, and S.-M. Hu, "Global contrast based salient region detection", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 3, pp. 569–582, 2015.

[90] Y. Li, X. Hou, C. Koch, J. M. Rehg, and A. L. Yuille, "The secrets of salient object segmentation", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 280–287.

[91] P.-H. Tseng, R. Carmi, I. G. Cameron, D. P. Munoz, and L. Itti, "Quantifying center bias of observers in free viewing of dynamic natural scenes", *Journal of vision*, vol. 9, no. 7, pp. 4–4, 2009.

[92] N. Riche, M. Duvinage, M. Mancas, B. Gosselin, and T. Dutoit, "Saliency and human fixations: State-of-the-art and study of comparison metrics", in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 1153–1160.

[93] A. Borji, D. N. Sihite, and L. Itti, "Quantitative analysis of human-model agreement in visual saliency modeling: A comparative study", *IEEE Transactions on Image Processing*, vol. 22, no. 1, pp. 55–69, 2013.

[94] J. Aloimonos, I. Weiss, and A. Bandyopadhyay, "Active vision", *International journal of computer vision*, vol. 1, no. 4, pp. 333–356, 1988.

[95] C.-J. Westelius, "Focus of attention and gaze control for robot vision", PhD thesis, Linköping University Electronic Press, 1995.

[96] R. M. Klein, "Inhibition of return", *Trends in cognitive sciences*, vol. 4, no. 4, pp. 138–147, 2000.

[97] M. Begum and F. Karray, "Visual attention for robotic cognition: A survey", *IEEE Transactions on Autonomous Mental Development*, vol. 3, no. 1, pp. 92–105, 2011.

[98] N. Ouerhani, T. Jost, A. Bur, and H Hugli, "Cue normalization schemes in saliency-based visual attention models", in *International cognitive vision workshop*, 2006.

[99] K. L. Ho and P. Newman, "Loop closure detection in slam by combining visual and spatial appearance", *Robotics and Autonomous Systems*, vol. 54, no. 9, pp. 740–749, 2006.

[100] A. Kim and R. M. Eustice, "Real-time visual slam for autonomous underwater hull inspection using visual saliency", *IEEE Transactions on Robotics*, vol. 29, no. 3, pp. 719–733, 2013.

[101] D. Meger, P.-E. Forssén, K. Lai, S. Helmer, S. McCann, T. Southey, M. Baumann, J. J. Little, and D. G. Lowe, "Curious george: An attentive semantic robot", *Robotics and Autonomous Systems*, vol. 56, no. 6, pp. 503–511, 2008.

[102] J. M. Cañas, M. M. de la Casa, and T. González, "An overt visual attention mechanism based on saliency dynamics", *International Journal of Intelligent Computing in Medical Sciences & Image Processing*, vol. 2, no. 2, pp. 93–100, 2008.

[103] S. Gould, J. Arfvidsson, A. Kaehler, B. Sapp, M. Messner, G. R. Bradski, P. Baumstarck, S. Chung, and A. Y. Ng, "Peripheral-foveal vision for real-time object recognition and tracking in video.", in *IJCAI*, vol. 7, 2007, pp. 2115–2121.

[104] T. Serre, L. Wolf, S. Bileschi, M. Riesenhuber, and T. Poggio, "Robust object recognition with cortex-like mechanisms", *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 29, no. 3, pp. 411–426, 2007.

[105] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection", *arXiv preprint arXiv:1506.02640*, 2015.

[106] S. Eslami, N. Heess, T. Weber, Y. Tassa, K. Kavukcuoglu, and G. E. Hinton, "Attend, infer, repeat: Fast scene understanding with generative models", *arXiv preprint arXiv:1603.08575*, 2016.

[107] M. Björkman and J.-O. Eklundh, "Vision in the real world: Finding, attending and recognizing objects", *International Journal of Imaging Systems and Technology*, vol. 16, no. 5, pp. 189–208, 2006.

[108] A. Borji, M. N. Ahmadabadi, and B. N. Araabi, "Learning sequential visual attention control through dynamic state space discretization", in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, IEEE, 2009, pp. 2258–2263.

[109] S. Minut and S. Mahadevan, "A reinforcement learning model of selective visual attention", in *Proceedings of the fifth international conference on Autonomous agents*, ACM, 2001, pp. 457–464.

[110] D. Kragic, "Object search and localization for an indoor mobile robot", *CIT. Journal of Computing and Information Technology*, vol. 17, no. 1, pp. 67–80, 2009.

[111] A. Mian, "Realtime face detection and tracking using a single pan, tilt, zoom camera", in *2008 23rd International Conference Image and Vision Computing New Zealand*, IEEE, 2008, pp. 1–6.

[112] S. Frintrop and M. Kessel, "Most salient region tracking", in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, IEEE, 2009, pp. 1869–1874.

[113] D. Kragic and M. Bjorkman, "Strategies for object manipulation using foveal and peripheral vision", in *Computer Vision Systems, 2006 ICVS'06. IEEE International Conference on*, IEEE, 2006, pp. 50–50.

[114] G. Metta and P. Fitzpatrick, "Early integration of vision and manipulation", in *Neural Networks, 2003. Proceedings of the International Joint Conference on*, IEEE, vol. 4, 2003, 2703–vol.

[115] A. J. Davison and D. W. Murray, "Mobile robot localisation using active vision", in *European Conference on Computer Vision*, Springer, 1998, pp. 809–825.

[116] M. Lauri and R. 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.

[117] S. M. Nguyen, S. Ivaldi, N. Lyubova, A. Droniou, D. Gerardeaux-Viret, D. Filliat, V. Padois, O. Sigaud, and P.-Y. 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*, IEEE, 2013, pp. 1–8.

[118] G. Heidemann, R. Rae, H. Bekel, I. Bax, and H. Ritter, "Integrating context-free and context-dependent attentional mechanisms for gestural object reference", in *International Conference on Computer Vision Systems*, Springer, 2003, pp. 22–33.

[119] A. Holzbach and G. Cheng, "A fast and scalable system for visual attention, object based attention and object recognition for humanoid robots", in *2014 IEEE-RAS International Conference on Humanoid Robots*, IEEE, 2014, pp. 316–321.

[120] D. Kragic, M. Björkman, H. I. Christensen, and J.-O. Eklundh, "Vision for robotic object manipulation in domestic settings", *Robotics and autonomous Systems*, vol. 52, no. 1, pp. 85–100, 2005.

[121] A. Thomas, V. Ferrar, B. Leibe, T. Tuytelaars, B. Schiel, and L. Van Gool, "Towards multi-view object class detection", in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, IEEE, vol. 2, 2006, pp. 1589–1596.

[122] K. Welke, J. Issac, D. Schiebener, T. Asfour, and R. Dillmann, "Autonomous acquisition of visual multi-view object representations for object recognition on a humanoid robot", in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, IEEE, 2010, pp. 2012–2019.

[123] S. Oßwald, M. Bennewitz, W. Burgard, and C. Stachniss, "Speeding-up robot exploration by exploiting background information", *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 716–723, 2016.

[124] G. Kootstra, J. Ypma, and B. de Boer, "Exploring objects for recognition in the real word", in *Robotics and Biomimetics, 2007. ROBIO 2007. IEEE International Conference on*, 2007, pp. 429–434.

[125] B. Browatzki, V. Tikhanoff, G. Metta, H. Bulthoff, and C. Wallraven, "Active object recognition on a humanoid robot", in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, IEEE, 2012, pp. 2021–2028.

[126] L. Natale, F. Orabona, F. Berton, G. Metta, and G. Sandini, "From sensorimotor development to object perception", in *Humanoid Robots, 2005 5th IEEE-RAS International Conference on*, IEEE, 2005, pp. 226–231.

[127] N. Lyubova and D. Filliat, "Developmental approach for interactive object discovery", in *Neural Networks (IJCNN), The 2012 International Joint Conference on*, IEEE, 2012, pp. 1–7.

[128] S. Ivaldi, N. Lyubova, D. Gérardeaux-Viret, A. Droniou, S. M. Anzalone, M. Chetouani, D. Filliat, and O. Sigaud, "Perception and human interaction for developmental learning of objects and affordances", in *Proc. of the 12th IEEE-RAS International Conference on Humanoid Robots - HUMANOIDS*, 2012, pp. 1–8.

[129] T. Kollar and N. Roy, "Utilizing object-object and object-scene context when planning to find things", in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, IEEE, 2009, pp. 2168–2173.

[130] H. Ali, F. Shafait, E. Giannakidou, A. Vakali, N. Figueroa, T. Varvadoukas, and N. Mavridis, "Contextual object category recognition for rgb-d scene labeling", *Robotics and Autonomous Systems*, vol. 62, no. 2, pp. 241–256, 2014.

[131] D. Walther and C. Koch, "Modeling attention to salient proto-objects", *Neural networks*, vol. 19, no. 9, pp. 1395–1407, 2006.

[132] M. Wischnewski, A. Belardinelli, W. X. Schneider, and J. J. Steil, "Where to look next? combining static and dynamic proto-objects in a tva-based model of visual attention", *Cognitive computation*, vol. 2, no. 4, pp. 326–343, 2010.

[133] U. Rutishauser, D. Walther, C. Koch, and P. Perona, "Is bottom-up attention useful for object recognition?", in *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, IEEE, vol. 2, 2004, pp. II–37.

[134] J.-Y. Zhu, J. Wu, Y. Wei, E. Chang, and Z. Tu, "Unsupervised object class discovery via saliency-guided multiple class learning", in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, IEEE, 2012, pp. 3218–3225.

[135] S. Frintrop, E. Rome, A. Nüchter, and H. Surmann, "A bimodal laser-based attention system", *Computer Vision and Image Understanding*, vol. 100, no. 1, pp. 124–151, 2005.

[136] T. Xu, N. Chenkov, K. Kuhnlenz, and M. Buss, "Autonomous switching of top-down and bottom-up attention selection for vision guided mobile robots", in *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, IEEE, 2009, pp. 4009–4014.

[137] Y. Guo, Y. Liu, A. Oerlemans, S. Lao, S. Wu, and M. S. Lew, "Deep learning for visual understanding: A review", *Neurocomputing*, vol. 187, pp. 27–48, 2016.

[138] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Susstrunk, "Slic superpixels compared to state-of-the-art superpixel methods", *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 34, no. 11, pp. 2274–2282, 2012.

[139] C. Hazirbas, "Feature selection and learning for semantic segmentation", Master's thesis, Technical University Munich, Germany, 2014.

[140] M. Van den Bergh, X. Boix, G. Roig, B. de Capitani, and L. Van Gool, "Seeds: Superpixels extracted via energy-driven sampling", in *Computer Vision–ECCV 2012*, Springer, 2012, pp. 13–26.

[141] M.-Y. Liu, O. Tuzel, S. Ramalingam, and R. Chellappa, "Entropy rate superpixel segmentation", in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, IEEE, 2011, pp. 2097–2104.

[142] Y. Zhang, R. Hartley, J. Mashford, and S. Burn, "Superpixels via pseudo-boolean optimization", in *2011 International Conference on Computer Vision*, IEEE, 2011, pp. 1387–1394.

[143] J. Papon, A. Abramov, M. Schoeler, and F. Worgotter, "Voxel cloud connectivity segmentation-supervoxels for point clouds", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 2027–2034.

[144] C. Xu, S. Whitt, and J. J. Corso, "Flattening supervoxel hierarchies by the uniform entropy slice", in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 2240–2247.

[145] D. Weikersdorfer, D. Gossow, and M. Beetz, "Depth-adaptive superpixels", in *Pattern Recognition (ICPR), 2012 21st International Conference on*, IEEE, 2012, pp. 2087–2090.

[146] A. Elgammal, D. Harwood, and L. Davis, "Non-parametric model for background subtraction", in *European conference on computer vision*, Springer, 2000, pp. 751–767.

[147] J. Badenas, J. M. Sanchiz, and F. Pla, "Motion-based segmentation and region tracking in image sequences", *Pattern Recognition*, vol. 34, no. 3, pp. 661–670, 2001.

[148] M. B. B. Mahsa Ghafarianzadeh and G. Sibley, "Efficient, dense, object-based segmentation from rgbd video", in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2016.

[149] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient graph-based image segmentation", *International Journal of Computer Vision*, vol. 59, no. 2, pp. 167–181, 2004.

[150] C. Rother, V. Kolmogorov, and A. Blake, "Grabcut: Interactive foreground extraction using iterated graph cuts", in *ACM transactions on graphics (TOG)*, ACM, vol. 23, 2004, pp. 309–314.

[151] M. Dubois, P. K. Rozo, A. Gepperth, O. F. A. González, and D. Filliat, "A comparison of geometric and energy-based point cloud semantic segmentation methods", in *Mobile Robots (ECMR), 2013 European Conference on*, IEEE, 2013, pp. 88–93.

[152] A. K. Mishra, Y. Aloimonos, L.-F. Cheong, and A. A. Kassim, "Active visual segmentation", *Pattern Analysis and Machine Intelligence, IEEE Transactions On*, vol. 34, no. 4, pp. 639–653, 2012.

[153] A. Mishra, Y. Aloimonos, and C. L. Fah, "Active segmentation with fixation", in *Computer Vision, 2009 IEEE 12th International Conference on*, IEEE, 2009, pp. 468–475.

[154] S. Christoph Stein, M. Schoeler, J. Papon, and F. Worgotter, "Object partitioning using local convexity", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 304–311.

[155] A. Karpathy, S. Miller, and L. Fei-Fei, "Object discovery in 3d scenes via shape analysis", in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, IEEE, 2013, pp. 2088–2095.

[156] E. Potapova, K. M. Varadarajan, A. Richtsfeld, M. Zillich, and M. 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)*, IEEE, 2014, pp. 4946–4952.

[157] C. F. Aleksandrs Ecins and Y. Aloimonos, "Cluttered scene segmentation using the symmetry constraint", in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2016.

[158] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography", *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[159] M. Everingham, S. A. Eslami, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge: A retrospective", *International Journal of Computer Vision*, vol. 111, no. 1, pp. 98–136, 2015.

[160] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from rgbd images", in *European Conference on Computer Vision*, Springer, 2012, pp. 746–760.

[161] L. Ladickỳ, P. Sturgess, K. Alahari, C. Russell, and P. H. Torr, "What, where and how many? combining object detectors and crfs", in *European conference on computer vision*, Springer, 2010, pp. 424–437.

[162] C. Zhang, L. Wang, and R. Yang, "Semantic segmentation of urban scenes using dense depth maps", in *European Conference on Computer Vision*, Springer, 2010, pp. 708–721.

[163] J. Tighe and S. Lazebnik, "Superparsing: Scalable nonparametric image parsing with superpixels", in *European conference on computer vision*, Springer, 2010, pp. 352–365.

[164] S. Gupta, P. Arbelaez, and J. Malik, "Perceptual organization and recognition of indoor scenes from rgb-d images", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 564–571.

[165] D. Grangier, L. Bottou, and R. Collobert, "Deep convolutional networks for scene parsing", in *ICML 2009 Deep Learning Workshop*, Citeseer, vol. 3, 2009.

[166] V. Badrinarayanan, A. Handa, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for robust semantic pixel-wise labelling", *arXiv preprint arXiv:1505.07293*, 2015.

[167] G. Lin, C. Shen, I. Reid, *et al.*, "Efficient piecewise training of deep structured models for semantic segmentation", *arXiv preprint arXiv:1504.01013*, 2015.

[168] G. Papandreou, L.-C. Chen, K. Murphy, and A. L. Yuille, "Weakly-and semi-supervised learning of a dcnn for semantic image segmentation", *arXiv preprint arXiv:1502.02734*, 2015.

[169] C. L. Zitnick and P. Dollár, "Edge boxes: Locating object proposals from edges", in *Computer Vision–ECCV 2014*, Springer, 2014, pp. 391–405.

[170] P. Viola and M. Jones, "Robust real-time object detection", *International Journal of Computer Vision*, vol. 4, 2001.

[171] P. Felzenszwalb, D. McAllester, and D. Ramanan, "A discriminatively trained, multiscale, deformable part model", in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, IEEE, 2008, pp. 1–8.

[172] J. Hosang, R. Benenson, P. Dollár, and B. Schiele, "What makes for effective detection proposals?", *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 4, pp. 814–830, 2016.

[173] K. E. Van de Sande, J. R. Uijlings, T. Gevers, and A. W. Smeulders, "Segmentation as selective search for object recognition", in *2011 International Conference on Computer Vision*, IEEE, 2011, pp. 1879–1886.

[174] B. Alexe, T. Deselaers, and V. Ferrari, "Measuring the objectness of image windows", *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 34, no. 11, pp. 2189–2202, 2012.

[175] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks", in *Advances in Neural Information Processing Systems*, 2015, pp. 91–99.

[176] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context", in *European Conference on Computer Vision*, Springer, 2014, pp. 740–755.

[177] S. Gidaris and N. Komodakis, "Locnet: Improving localization accuracy for object detection", in *Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on*, 2016.

[178] ——, "Object detection via a multi-region and semantic segmentation-aware cnn model", in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1134–1142.

[179] J. Hoffman, S. Guadarrama, E. S. Tzeng, R. Hu, J. Donahue, R. Girshick, T. Darrell, and K. Saenko, "Lsda: Large scale detection through adaptation", in *Advances in Neural Information Processing Systems*, 2014, pp. 3536–3544.

[180] S. Mitri, S. Frintrop, K. Pervolz, H. Surmann, and A. Nuchter, "Robust object detection at regions of interest with an application in ball recognition", in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, IEEE, 2005, pp. 125–130.

[181] A. Borji, M. N. Ahmadabadi, B. N. Araabi, and M. Hamidi, "Online learning of task-driven object-based visual attention control", *Image and Vision Computing*, vol. 28, no. 7, pp. 1130–1145, 2010.

[182] A. K. McCallum, "Reinforcement learning with selective perception and hidden state", PhD thesis, University of Rochester, 1996.

[183] D. T. Larose, "K-nearest neighbor algorithm", *Discovering Knowledge in Data: An Introduction to Data Mining*, pp. 90–106, 2005.

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

[185] A. Saffari, C. Leistner, J. Santner, M. Godec, and H. Bischof, "On-line random forests", in *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*, IEEE, 2009, pp. 1393–1400.

[186] M. Denil, D. Matheson, and N. De Freitas, "Consistency of online random forests.", *ICML (3)*, vol. 28, pp. 1256–1264, 2013.

[187] B. Lakshminarayanan, D. M. Roy, and Y. W. Teh, "Mondrian forests: Efficient online random forests", in *Advances in Neural Information Processing Systems*, 2014, pp. 3140–3148.

[188] G. Pasquale, C. Ciliberto, L. Rosasco, and L. Natale, "Object identification from few examples by improving the invariance of a deep convolutional neural network", in *International conference on intelligent robots and systems (IROS), 2016 IEEE Conference on*, IEEE, 2016.

[189] A. Saxena, S. H. Chung, and A. Y. Ng, "Learning depth from single monocular images", in *Advances in Neural Information Processing Systems*, 2005, pp. 1161–1168.

[190] A. Baranes and P.-Y. Oudeyer, "Active learning of inverse models with intrinsically motivated goal exploration in robots", *Robotics and Autonomous Systems*, vol. 61, no. 1, pp. 49–73, 2013.

[191] S. J. Pan and Q. Yang, "A survey on transfer learning", *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[192] A. Margolis, "A literature review of domain adaptation with unlabeled data", *Tec. Report*, pp. 1–42, 2011.

[193] A. Sharif Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "Cnn features off-the-shelf: An astounding baseline for recognition", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2014, pp. 806–813.

[194] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell, "Decaf: A deep convolutional activation feature for generic visual recognition.", in *Icml*, vol. 32, 2014, pp. 647–655.

[195] D. Gabor, "Theory of communication. part 1: The analysis of information", *Journal of the Institution of Electrical Engineers-Part III: Radio and Communication Engineering*, vol. 93, no. 26, pp. 429–441, 1946.

[196] A. Saxena, M. Sun, and A. Y. Ng, "Make3d: Learning 3d scene structure from a single still image", *IEEE transactions on pattern analysis and machine intelligence*, vol. 31, no. 5, pp. 824–840, 2009.

[197] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition", *arXiv preprint arXiv:1409.1556*, 2014.

[198] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1–9.

[199] L. Breiman, "Random forests", *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[200] C Gini, "Concentration and dependency ratios", *Rivista di politica economica*, vol. 87, pp. 769–792, 1997.

[201] D. Filliat, "Robotique mobile", *École Nationale Supérieure de Techniques Avancées*, 2011.

[202] S. Bazeille and D. Filliat, "Incremental topo-metric slam using vision and robot odometry", in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, IEEE, 2011, pp. 4067–4073.

[203] G. Dedeoglu, M. J. Mataric, and G. S. Sukhatme, "Incremental online topological map building with a mobile robot", in *Photonics East'99*, International Society for Optics and Photonics, 1999, pp. 129–139.

[204] D. Kortenkamp and T. Weymouth, "Topological mapping for mobile robots using a combination of sonar and vision sensing", in *AAAI*, vol. 94, 1994, pp. 979–984.

[205] P. Gaussier, C. Joulain, J.-P. Banquet, S. Leprêtre, and A. Revel, "The visual homing problem: An example of robotics/biology cross fertilization", *Robotics and autonomous systems*, vol. 30, no. 1, pp. 155–180, 2000.

[206] T. Krajník, J. P. Fentanes, O. M. Mozos, T. Duckett, J. Ekekrantz, and M. Hanheide, "Long-term topological localisation for service robots in dynamic environments using spectral maps", in *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, IEEE, 2014, pp. 4537–4542.

[207] H. J. S. Feder, J. J. Leonard, and C. M. Smith, "Adaptive mobile robot navigation and mapping", *The International Journal of Robotics Research*, vol. 18, no. 7, pp. 650–668, 1999.

[208] R. Smith, M. Self, and P. Cheeseman, "Estimating uncertain spatial relationships in robotics", in *Autonomous robot vehicles*, Springer, 1990, pp. 167–193.

[209] N. Ayache and O. D. Faugeras, "Maintaining representations of the environment of a mobile robot", *IEEE transactions on Robotics and Automation*, vol. 5, no. 6, pp. 804–819, 1989.

[210] H. Moravec and A. Elfes, "High resolution maps from wide angle sonar", in *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*, IEEE, vol. 2, 1985, pp. 116–121.

[211] A. Borji, M. N. Ahmadabadi, and B. N. Araabi, "Simultaneous learning of spatial visual attention and physical actions", in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, IEEE, 2010, pp. 1270–1276.

[212] J. J. Leonard and H. F. Durrant-Whyte, "Simultaneous map building and localization for an autonomous mobile robot", in *Intelligent Robots and Systems' 91.'Intelligence for Mechanical Systems, Proceedings IROS'91. IEEE/RSJ International Workshop on*, Ieee, 1991, pp. 1442–1447.

[213] R. Chatila and J.-P. Laumond, "Position referencing and consistent world modeling for mobile robots", in *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*, IEEE, vol. 2, 1985, pp. 138–145.

[214] J. Engel, T. Schöps, and D. Cremers, "Lsd-slam: Large-scale direct monocular slam", in *European Conference on Computer Vision*, Springer, 2014, pp. 834–849.

[215] F. Endres, J. Hess, J. Sturm, D. Cremers, and W. Burgard, "3-d mapping with an rgb-d camera", *IEEE Transactions on Robotics*, vol. 30, no. 1, pp. 177–187, 2014.

[216] R. Murphy, *Introduction to ai robotics*. MIT press, 2000.

[217] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Informed rrt*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic", *arXiv preprint arXiv:1404.2334*, 2014.

[218] N. Jetchev and M. Toussaint, "Trajectory prediction: Learning to map situations to robot trajectories", in *Proceedings of the 26th annual international conference on machine learning*, ACM, 2009, pp. 449–456.

[219] X. Jiang and M. Kallmann, "Learning humanoid reaching tasks in dynamic environments", in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2007, pp. 1148–1153.

[220] G. K. Kraetzschmar, G. P. Gassull, K. Uhl, G. Pags, and G. K. Uhl, "Probabilistic quadtrees for variable-resolution mapping of large environments", in *Proceedings of the 5th IFAC/EURON symposium on intelligent autonomous vehicles*, July, 2004.

[221] Y. L.L. L. Hengbo Tang, "Pose graph optimization with hierarchical conditionally independent graph partitioning", in *2016 IEEE International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2016.

[222] B. Kuipers and Y.-T. Byun, "A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations", *Robotics and autonomous systems*, vol. 8, no. 1, pp. 47–63, 1991.

[223] L. F.-L. J. Neira and J. A. Castellanos, "Path planning in graph slam using expected uncertainty", in *2016 IEEE International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2016.

[224] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning", 1998.

[225] E. W. Dijkstra, "A note on two problems in connexion with graphs", *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[226] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths", *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[227] Y.-Q. Qin, D.-B. Sun, N. Li, and Y.-G. Cen, "Path planning for mobile robot using the particle swarm optimization with mutation operator", in *Machine Learning and Cybernetics, 2004. Proceedings of 2004 International Conference on*, IEEE, vol. 4, 2004, pp. 2473–2478.

[228] T. Guan-Zheng, H. Huan, and A. Sloman, "Ant colony system algorithm for real-time globally optimal path planning of mobile robots", *Acta Automatica Sinica*, vol. 33, no. 3, pp. 279–285, 2007.

[229] S. Lague, *Github seblague path finding project*, https://github.com/SebLague/ Pathfinding.

[230] M. Schwager, *Multi-robot systems lab projects, boston university*, http://sites.bu.edu /msl/research/coordinated-aggressive-control-of-aerial-vehicles/.

[231] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*, 1. MIT press Cambridge, 1998, vol. 1.

[232] S. Ross, J. Pineau, S. Paquet, and B. Chaib-Draa, "Online planning algorithms for pomdps", *Journal of Artificial Intelligence Research*, vol. 32, pp. 663–704, 2008.

[233] A. Somani, N. Ye, D. Hsu, and W. S. Lee, "Despot: Online pomdp planning with regularization", in *Advances in neural information processing systems*, 2013, pp. 1772–1780.

[234] J.-Y. Audibert, R. Munos, and C. Szepesvári, "Exploration–exploitation tradeoff using variance estimates in multi-armed bandits", *Theoretical Computer Science*, vol. 410, no. 19, pp. 1876–1902, 2009.

[235] J. M. Santos, T. Krajník, and T. Duckett, "Spatio-temporal exploration strategies for long-term autonomy of mobile robots", *Robotics and Autonomous Systems*, 2016, ISSN: 0921-8890.

[236] I. Jebari, S. Bazeille, and D. Filliat, "Combined vision and frontier-based exploration strategies for semantic mapping", in *Informatics in Control, Automation and Robotics*, Springer, 2012, pp. 237–244.

[237] B. Tovar, L. Muñoz-Gómez, R. Murrieta-Cid, M. Alencastre-Miranda, R. Monroy, and S. Hutchinson, "Planning exploration strategies for simultaneous localization and mapping", *Robotics and Autonomous Systems*, vol. 54, no. 4, pp. 314–331, 2006.

[238] M. Levihn, J. Scholz, and M. Stilman, "Planning with movable obstacles in continuous environments with uncertain dynamics", in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, IEEE, 2013, pp. 3832–3838.

[239] M. Dorigo and L. M. Gambardella, "Ant colony system: A cooperative learning approach to the traveling salesman problem", *IEEE Transactions on evolutionary computation*, vol. 1, no. 1, pp. 53–66, 1997.

[240] H. González-Baños, "A randomized art-gallery algorithm for sensor placement", in *Proceedings of the seventeenth annual symposium on Computational geometry*, ACM, 2001, pp. 232–240.

[241] H. A. Eiselt, M. Gendreau, and G. Laporte, "Arc routing problems, part i: The chinese postman problem", *Operations Research*, vol. 43, no. 2, pp. 231–242, 1995.

[242] B. Yamauchi, "A frontier-based approach for autonomous exploration", in *Computational Intelligence in Robotics and Automation, 1997. CIRA'97., Proceedings., 1997 IEEE International Symposium on*, IEEE, 1997, pp. 146–151.

[243] C. Stachniss, G. Grisetti, and W. Burgard, "Information gain-based exploration using rao-blackwellized particle filters.", in *Robotics: Science and Systems*, vol. 2, 2005, pp. 65–72.

[244] C. Stachniss and W. Burgard, "Exploring unknown environments with mobile robots using coverage maps", in *IJCAI*, 2003, pp. 1127–1134.

[245] J. A. P. Fentanes, R. F. Alonso, E. Zalama, and J. G. García-Bermejo, "A new method for efficient three-dimensional reconstruction of outdoor environments using mobile robots", *Journal of Field Robotics*, vol. 28, no. 6, pp. 832–853, 2011.

[246] P. Wang, R. Krishnamurti, and K. Gupta, "View planning problem with combined view and traveling cost", in *Proceedings 2007 IEEE International Conference on Robotics and Automation*, IEEE, 2007, pp. 711–716.

[247] M. Björkman and D. Kragic, "Active 3d scene segmentation and detection of unknown objects", in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, IEEE, 2010, pp. 3114–3120.

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

[249] J. Kober and J. Peters, "Reinforcement learning in robotics: A survey", in *Reinforcement Learning*, Springer, 2012, pp. 579–610.

[250] P. Kormushev, S. Calinon, and D. G. Caldwell, "Reinforcement learning in robotics: Applications and real-world challenges", *Robotics*, vol. 2, no. 3, pp. 122–148, 2013.

[251] S. Jodogne and J. H. Piater, "Closed-loop learning of visual control policies.", *J. Artif. Intell. Res.(JAIR)*, vol. 28, pp. 349–391, 2007.

[252] N. Chentanez, A. G. Barto, and S. P. Singh, "Intrinsically motivated reinforcement learning", in *Advances in neural information processing systems*, 2004, pp. 1281–1288.

[253] A. K. McCallum, "Learning visual routines with reinforcement learning", in *AAAI Fall Symposium 1996*, 1996.

[254] A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 1998.

[255] R. I. Brafman and M. Tennenholtz, "R-max-a general polynomial time algorithm for near-optimal reinforcement learning", *The Journal of Machine Learning Research*, vol. 3, pp. 213–231, 2003.

[256] J. Z. Kolter and A. Y. Ng, "Near-bayesian exploration in polynomial time", in *Proceedings of the 26th Annual International Conference on Machine Learning*, ACM, 2009, pp. 513–520.

[257] M. Lopes, T. Lang, M. Toussaint, and P.-Y. Oudeyer, "Exploration in model-based reinforcement learning by empirically estimating learning progress", in *Advances in Neural Information Processing Systems*, 2012, pp. 206–214.

[258] A. F. Emery and A. V. Nenarokomov, "Optimal experiment design", *Measurement Science and Technology*, vol. 9, no. 6, p. 864, 1998.

[259] D. P. Ström, F. Nenci, and C. Stachniss, "Predictive exploration considering previously mapped environments", in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2015, pp. 2761–2766.

[260] Y. Chen, J.-B. Bordes, and D. Filliat, "An experimental comparison between nmf and lda for active cross-situational object-word learning", in *ICDL EPIROB 2016*, 2016.

[261] M. Pandey and S. Lazebnik, "Scene recognition and weakly supervised object localization with deformable part-based models", in *2011 International Conference on Computer Vision*, IEEE, 2011, pp. 1307–1314.

[262] J. Schmidhuber, "Curious model-building control systems", in *Neural Networks, 1991. 1991 IEEE International Joint Conference on*, IEEE, 1991, pp. 1458–1463.

[263] S. Chaiklin, "The zone of proximal development in vygotsky's analysis of learning and instruction", *Vygotsky's educational theory in cultural context*, vol. 1, pp. 39–64, 2003.

[264] T. Minato, F. DallaLibera, S. Yokokawa, Y. Nakamura, H. Ishiguro, and E. Menegatti, "A baby robot platform for cognitive developmental robotics", in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, vol. 9, 2009.

[265] M. Lapeyre, "Poppy: Plate-forme robotique open source, imprimée en 3d et totalement modulaire pour l'experimentation scientifique, artistique et pédagogique", PhD thesis, Bordeaux, 2015.

[266] R. A. Brooks, "Achieving artificial intelligence through building robots.", DTIC Document, Tech. Rep., 1986.

[267] A. Baranès and P.-Y. Oudeyer, "R-iac: Robust intrinsically motivated exploration and active learning", *Autonomous Mental Development, IEEE Transactions on*, vol. 1, no. 3, pp. 155–169, 2009.

[268] S. Forestier and P.-Y. Oudeyer, "Overlapping waves in tool use development: A curiosity-driven computational model", in *The Sixth Joint IEEE International Conference Developmental Learning and Epigenetic Robotics*, 2016.

[269] C. Moulin-Frier and P.-Y. Oudeyer, "Exploration strategies in developmental robotics: A unified probabilistic framework", in *Development and Learning and Epigenetic Robotics (ICDL), 2013 IEEE Third Joint International Conference on*, IEEE, 2013, pp. 1–6.

[270] V. Kompella, M. Luciw, M. Stollenga, L. Pape, and J. 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*, IEEE, 2012, pp. 1–8.

[271] A. Matricon, D. Filliat, and P.-Y. Oudeyer, "An iterative algorithm for forward-parameterized skill discovery", in *ICDL EPIROB 2016*, 2016.

[272] A. Baranes and P.-Y. Oudeyer, "The interaction of maturational constraints and intrinsic motivations in active motor development", in *2011 IEEE International Conference on Development and Learning (ICDL)*, IEEE, vol. 2, 2011, pp. 1–8.

[273] R. M. Ryan and E. L. Deci, "Intrinsic and extrinsic motivations: Classic definitions and new directions", *Contemporary educational psychology*, vol. 25, no. 1, pp. 54–67, 2000.

[274] W. Schultz, P. Dayan, and P. R. Montague, "A neural substrate of prediction and reward", *Science*, vol. 275, no. 5306, pp. 1593–1599, 1997.

[275] P. Dayan and B. W. Balleine, "Reward, motivation, and reinforcement learning", *Neuron*, vol. 36, no. 2, pp. 285–298, 2002.

[276] X. Huang and J. Weng, "Novelty and reinforcement learning in the value system of developmental robots", 2002.

[277] S. E. Boehnke, D. J. Berg, R. A. Marino, P. F. Baldi, L. Itti, and D. P. Munoz, "Visual adaptation and novelty responses in the superior colliculus", *European Journal of Neuroscience*, vol. 34, no. 5, pp. 766–779, 2011.

[278] C. A. Rothkopf and D. H. Ballard, "Credit assignment in multiple goal embodied visuomotor behavior", *Embodied and grounded cognition*, p. 217, 2010.

[279] H. Ngo, M. Luciw, A. Förster, and J. Schmidhuber, "Confidence-based progress-driven self-generated goals for skill acquisition in developmental robots", 2013.

[280] R. Saunders and J. S. Gero, "Curious agents and situated design evaluations", *AI EDAM: Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, vol. 18, no. 02, pp. 153–161, 2004.

[281] K. E. Merrick, "A computational model of achievement motivation for artificial agents", in *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 3*, International Foundation for Autonomous Agents and Multiagent Systems, 2011, pp. 1067–1068.

[282] F. Benureau and P.-Y. Oudeyer, "Diversity-driven selection of exploration strategies in multi-armed bandits", in *2015 Joint IEEE International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob)*, IEEE, 2015, pp. 135–142.

[283] F. Kaplan and P.-Y. Oudeyer, "Motivational principles for visual know-how development", 2003.

[284] P.-Y. Oudeyer, F. Kaplan, *et al.*, "How can we define intrinsic motivation?", in *proceedings of the 8th international conference on epigenetic robotics: modeling cognitive development in robotic systems*, 2008.

[285] K. Merrick, "Value systems for developmental cognitive robotics: A survey", *Cognitive Systems Research*, vol. 41, pp. 38–55, 2017.

[286] F. Kaplan and P.-Y. Oudeyer, "In search of the neural circuits of intrinsic motivation", *Frontiers in neuroscience*, vol. 1, no. 1, p. 225, 2007.

[287] J. Gottlieb, P.-Y. Oudeyer, M. Lopes, and A. Baranes, "Information-seeking, curiosity, and attention: Computational and neural mechanisms", *Trends in cognitive sciences*, vol. 17, no. 11, pp. 585–593, 2013.

[288] A. Baranes, P.-Y. Oudeyer, and J. Gottlieb, "Eye movements reveal epistemic curiosity in human observers", *Vision research*, vol. 117, pp. 81–90, 2015.

[289] U. Sailer, J. R. Flanagan, and R. S. Johansson, "Eye–hand coordination during learning of a novel visuomotor task", *The Journal of neuroscience*, vol. 25, no. 39, pp. 8833–8842, 2005.

[290] V. R. Kompella, M. F. Stollenga, M. D. Luciw, and J. Schmidhuber, "Explore to see, learn to perceive, get the actions for free: Skillability", in *2014 International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2014, pp. 2705–2712.

[291] C. Zhang, Y. Zhao, J. Triesch, and B. E. Shi, "Intrinsically motivated learning of visual motion perception and smooth pursuit", in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2014, pp. 1902–1908.

[292] M Hulse, S. McBride, J. Law, and M. Lee, "Integration of active vision and reaching from a developmental robotics perspective", *Autonomous Mental Development, IEEE Transactions on*, vol. 2, no. 4, pp. 355–367, 2010.

[293] H. A. Trukenbrod and R. Engbert, "Icat: A computational model for the adaptive control of fixation durations", *Psychonomic bulletin & review*, vol. 21, no. 4, pp. 907–934, 2014.

[294] O. L. Meur, A. Coutrot, Z. Liu, A. L. Roch, A. Helo, and P. Rama, "Computational model for predicting visual fixations from childhood to adulthood", *arXiv preprint arXiv:1702.04657*, 2017.

[295] A. Boduroglu, P. Shah, and R. E. Nisbett, "Cultural differences in allocation of attention in visual information processing", *Journal of Cross-Cultural Psychology*, vol. 40, no. 3, pp. 349–360, 2009.

[296] M. Lopes and P.-Y. Oudeyer, "The strategic student approach for life-long exploration and learning", in *Development and Learning and Epigenetic Robotics (ICDL), 2012 IEEE International Conference on*, IEEE, 2012, pp. 1–8.

[297] M. Luciw, V. Graziano, M. Ring, and J. Schmidhuber, "Artificial curiosity with planning for autonomous perceptual and cognitive development", in *Development and Learning (ICDL), 2011 IEEE International Conference on*, IEEE, vol. 2, 2011, pp. 1–8.

[298] P.-Y. Oudeyer, F. Kaplan, V. V. Hafner, and A. Whyte, "The playground experiment: Task-independent development of a curious robot", in *Proceedings of the AAAI Spring Symposium on Developmental Robotics*, Stanford, California, 2005, pp. 42–47.

[299] M. Lefort and A. Gepperth, "Active learning of local predictable representations with artificial curiosity", in *2015 Joint IEEE International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob)*, IEEE, 2015, pp. 228–233.

[300] M. Rolf, J. J. Steil, and M. Gienger, "Online goal babbling for rapid bootstrapping of inverse models in high dimensions", in *Development and Learning (ICDL), 2011 IEEE International Conference on*, IEEE, vol. 2, 2011, pp. 1–8.

[301] M. G. Lagoudakis and R. Parr, "Least-squares policy iteration", *Journal of machine learning research*, vol. 4, no. Dec, pp. 1107–1149, 2003.

[302] A. E. Bryson, "Optimal control-1950 to 1985", *IEEE Control Systems*, vol. 16, no. 3, pp. 26–33, 1996.

[303] C. J. Watkins and P. Dayan, "Q-learning", *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[304] D. Applegate, R. Bixby, V. Chvatal, and W. Cook, *Concorde tsp solver*, 2006.

[305] T. D. Kulkarni, K. Narasimhan, A. Saeedi, and J. Tenenbaum, "Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation", in *Advances in Neural Information Processing Systems*, 2016, pp. 3675–3683.

[306] M. Wang and X.-S. Hua, "Active learning in multimedia annotation and retrieval: A survey", *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 2, no. 2, p. 10, 2011.

université PARIS-SACLAY
ÉCOLE DOCTORALE
INTERFACES
Approches Interdisciplinaires : Fondements, Applications et Innovation

Titre : Apprentissage incrémental de la saillance visuelle par des mécanismes de motivation intrinsèque

Mots clefs : Saillance visuelle, robotique mobile, robotique cognitive, bio-inspiration, motivation intrinsèque, localisation d'objets

Résumé : La conception de systèmes de perception autonomes, tels que des robots capables d'accomplir un ensemble de tâches de manière sûre et sans assistance humaine, est l'un des grands défis de notre siècle. Pour ce faire, la robotique développementale propose de concevoir des robots qui, comme des enfants, auraient la faculté d'apprendre directement par interaction avec leur environnement. Nous avons dans cette thèse exploré de telles possibilités en se limitant à l'apprentissage de la localisation des objets d'intérêt (ou objets saillants) dans l'environnement du robot.
Pour ce faire, nous présentons dans ces travaux un mécanisme capable d'apprendre la saillance visuelle directement sur un robot, puis d'utiliser le modèle appris de la sorte pour localiser des objets saillants dans son environnement. Cette méthode a l'avantage de permettre la création de modèles spécialisés pour l'environnement du robot et les tâches qu'il doit accomplir, tout en restant flexible à d'éventuelles nouveautés ou modifications de l'environnement.
De plus, afin de permettre un apprentissage efficace et de qualité, nous avons développé des stratégies d'explorations basées sur les motivations intrinsèques, très utilisées en robotique développementale. Nous avons notamment adapté l'algorithme *IAC* à l'apprentissage de la saillance visuelle, et en avons conçu une extension, *RL-IAC*, pour permettre une exploration efficace sur un robot mobile.
Afin de vérifier et d'analyser les performances de nos algorithmes, nous avons réalisé des évaluations sur plusieurs plateformes robotiques dont une plateforme fovéale et un robot mobile, ainsi que sur des bases de données publiques

Title : Intrinsic motivation mechanisms for incremental learning of visual saliency

Keywords : Visual saliency, mobile robotics, cognitive robotics, bio-inspiration, intrinsic motivation, object localization

Abstract : Conceiving autonomous perceptual systems, such as robots able to accomplish a set of tasks in a safe way, without any human assistance, is one of the biggest challenge of the century. To this end, the developmental robotics suggests to conceive robots able to learn by interacting directly with their environment, just like children would. This thesis is exploring such possibility while restricting the problem to the one of localizing objects of interest (or salient objects) within the robot's environment.
For that, we present in this work a mechanism able to learn visual saliency directly on a robot, then to use the learned model so as to localize salient objects within their environment. The advantage of this method is the creation of models dedicated to the robot's environment and tasks it should be asked to accomplish, while remaining flexible to any change or novelty in the environment.
Furthermore, we have developed exploration strategies based on intrinsic motivations, widely used in developmental robotics, to enable efficient learning of good quality. In particular, we adapted the *IAC* algorithm to visual saliency leanring, and proposed an extension, *RL-IAC* to allow an efficient exploration on mobile robots.
In order to verify and analyze the performance of our algorithms, we have carried out various experiments on several robotics platforms, including a foveated system and a mobile robot, as well as publicly available datasets.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France