N° d'ordre : 409

Année 2004

THÈSE

présentée en vue de
l'obtention du titre de

DOCTEUR
de
L'ÉCOLE NATIONALE SUPÉRIEURE
DE L'AÉRONAUTIQUE ET DE L'ESPACE

ÉCOLE DOCTORALE : Systèmes
SPÉCIALITÉ : Systèmes informatiques et image
par

Emmanuel ZENOU

Localisation topologique, amers visuels et treillis de Galois

Soutenue le 22 décembre 2004 devant le jury :

MM.	R.	CHATILA	Président
	M.	GHALLAB	Co-directeur de thèse
	R.	MOHR	Rapporteur
	M.	RICHETIN	Rapporteur
	J.	SALLANTIN	Rapporteur
	M.	SAMUELIDES	Co-directeur de thèse

Thèse préparée au sein de SUPAERO et du LAAS-CNRS.

THÈSE

préparée au
Laboratoire d'Informatique et d'Automatique de SUPAÉRO
et au
Laboratoire d'Analyse et d'Architecture des Systèmes du CNRS

en vue de l'obtention du titre de
Docteur de l'École Nationale Supérieure de l'Aéronautique et de l'Espace

par

- Emmanuel ZENOU -

LOCALISATION TOPOLOGIQUE, AMERS VISUELS ET TREILLIS DE GALOIS

Soutenance prévue : Décembre 2004

Président	**Raja CHATILA**
Rapporteurs	**Roger MOHR** **Marc RICHETIN** **Jean SALLANTIN**
Directeurs de thèse	**Malik GHALLAB** **Manuel SAMUELIDES**

Table des matières

Table des figures

Avant-propos

Ce travail de thèse s'est inscrit dans le cadre d'une coopération entre l'École Nationale Supérieur de l'Aéronautique et de l'Espace (SUPAERO), la Direction Générale de l'Armement (DGA) et le Laboratoire d'Analyse et d'Architecture des Systèmes (LAAS-CNRS). Depuis décembre 2001, date de mes débuts à SUPAERO, je me suis consacré à plein temps à ce travail de recherche, avec également de nombreuses activités d'enseignements qui m'ont procurées énormément d'enjouement et de satisfaction.

Il y a de nombreuses personnes que je tiens sincèrement et chaleureusement à remercier. En premier lieu, Manuel Samuelides, Professeur à SUPAERO et Directeur du Département de Mathématiques Appliquées. Pour plusieurs raisons, je n'en cite que trois. La première est de m'avoir donné les moyens de faire cette thèse dans le cadre de SUPAERO, de son investissement dans ce projet et de son aide précieuse lors de discussions fécondes et passionnantes. La seconde est la confiance qu'il a placée en moi pour les enseignements qu'il m'a confiés. La troisième, enfin, est son soutien dans des moments difficiles de ma vie personnelle.

En second lieu je tiens à remercier Malik Ghallab, Directeur de Recherche au CNRS et Directeur du Laboratoire d'Analyse et d'Architecture des Systèmes (LAAS-CNRS). Tout d'abord, pour m'avoir accueilli à plusieurs reprises au LAAS et pris ma motivation et mon engagement au sérieux avant de commencer ma thèse. Ensuite, pour m'avoir accueilli dans son laboratoire et dans le groupe Robotique et Intelligence Artificielle (RIA). Enfin, pour les discussions scientifiques fructueuses que nous avons eues ensemble tout au long de notre coopération.

Ensuite, je tiens à remercier Jean-Henri LLareus et Jacques Lamaison du Laboratoire d'Informatique et d'Automatique (LIA) de SUPAERO et Raja Chatila du groupe RIA du LAAS-CNRS pour leur accueil au sein du leur laboratoire, et pour m'avoir donné les moyens de travailler dans d'excellentes conditions. Les autres membres de chaque laboratoire sont aussi chaleureusement remerciés : Dominique Vizcaïno, Jérôme Dartigues, Éric Métral, Jean-François Dassieu pour le LIA, et tous les permanents et les doctorants du LAAS-CNRS pour avoir appris tant de choses à leur contact.

Enfin, pour terminer, je remercie vivement les rapporteurs pour avoir accepter de lire mon tapuscrit dans un délais aussi court, et les membres du jury pour avoir accepté de se déplacer à une date aussi tardive de l'année.

À ma mère,
et à mon oncle André.

Chapitre I

Introduction générale

“Adieu, dit le renard. Voici mon secret. Il est très simple : on ne voit bien qu’avec le cœur. L’essentiel est invisible pour les yeux.
- L’essentiel est invisible pour les yeux, répéta le petit prince, afin de se souvenir”.

Même si elle ne représente pas l’essentiel, la vue est certainement l’un des sens les plus riches et les plus développés chez l’être humain, comme chez beaucoup d’autres animaux. Certaines études comportementales montrent la capacité chez les primates par exemple à repérer dans une scène des aliments ou d’autres animaux, en quelques millisecondes à peine. Le sujet qui *interprète* alors la scène (en moins de $200ms$ chez le singe [Thorpe 96]) est un animal qui a appris à reconnaître ces éléments tout au long de son existence passée. Les motivations sont diverses (se nourrir pour la reconnaissance d’aliments, se protéger pour la reconnaissance d’autres animaux, *etc.*), mais les processus de reconnaissance mis en œuvre pour répondre à de telles sollicitations sont extrêmement complexes.

Bien que l’aspect physiologique de la vision humaine soit bien compris aujourd’hui, les processus neuronaux qui traitent l’information sont beaucoup moins maîtrisés, malgré les progrès récents d’analyse non intrusive qui permettent d’identifier les fonctionnalités de certaines zones “géographiques” du cerveau humain. Il est aujourd’hui prouvé par exemple que l’œil envoie au cerveau des informations plutôt simples : des couleurs, des contrastes de couleurs, des points particuliers, des segments de droites... Aussi le terme *voir* prend-il tout son sens à partir du moment où il est reconnu que le premier organe visuel, c’est le cerveau. Le meilleur exemple étant peut-être la couleur, qu’il *crée*.

On comprend alors toute la difficulté de prêter à une machine les capacités d'extraire de l'information utile à partir d'une image naturelle. L'analyse que nous en faisons est étroitement liée à notre connaissance préalable du monde, et témoigne de notre capacité à extraire l'essentiel, à synthétiser, à recouper l'information pour *comprendre* et réagir rapidement. Même si beaucoup de modèles bio-inspirés se développent ici ou là, ils sont encore loin d'expliquer les processus de vision humaine. Une machine n'a évidemment pas les mêmes capacités d'apprentissage et de généralisation que nous. L'approche biologique de la vision reste une source d'inspiration formidable pour les algorithmes de traitement d'image.

L'idée initiale de notre travail est de permettre à un robot de construire, en temps réel, une représentation de l'environnement sous forme d'un *alphabet*, c'est-à-dire d'un ensemble d'attributs combinés entre eux de manière caractériser et à identifier chaque pièce -ou site- d'un environnement structuré. L'approche que nous avons développé est donc plus mathématique. Bien que nous utilisons les mêmes primitives visuelles que le cerveau, nous combinons des attributs au travers d'un formalisme appelé "**treillis de Galois**" ou "**treillis de concept**" dans un objectif classique en robotique mobile autonome : la localisation, et plus précisément la **localisation topologique**.

La localisation est le processus de reconnaissance d'un lieu, nécessaire dans le cadre plus général de la navigation. Ici encore, l'inspiration biologique est très présente, et nous la retrouvons dans notre démarche au travers de la recherche d'**amers**, et plus précisément d'**amers visuels**. Aucune hypothèse n'est formulée sur ses amers visuels, en dehors des primitives extraites grâce au treillis. L'environnement est perçu comme un ensemble d'attributs potentiels, et non comme une superposition d'objets pré-définis ou de formes pré-définies.

Il s'agit donc de créer dans un robot un processus d'*apprentissage*, plus précisément de classification supervisée, afin que celui-ci soit capable, à partir de "simples" images, de construire ses propres amers visuels -de construire sa propre représentation du monde-, et de les reconnaître par la suite. Il s'agit également de développer des stratégies d'apprentissage, afin d'optimiser l'espace mémoire que le formalisme utilisé demande à la machine. Enfin, il s'agit par la suite de donner au robot les moyens de construire, de façon autonome, une carte topologique.

Les treillis de Galois, par l'entremise des concepts, permettent un apprentissage symbolique du monde. Ces concepts seront définis, comme nous allons le voir, par des combinaisons d'attributs visuels. Ce formalisme permet donc d'établir une expression *alphabétique* des amers, en *mots* construits avec des *lettres*, que sont les attributs d'une image. Le robot apprend le monde qu'il explore symboliquement en attachant ces *mots* à chaque site. Le principal intérêt des treillis de Galois réside donc dans l'aspect symbolique de l'apprentissage. D'autres avantages sont intrinsèques à ce formalisme : il est souple (facilité d'utilisation en terme de décision, en fonction d'un objectif, ou de mise à jour), modulaire (nous exploiterons cette propriété pour les treillis locaux), la structure de treillis à partir d'un jeu de données est déterministe (pas d'hypothèse initiale à formuler), et surtout leur construction est incrémentale (avec les

algorithmes utilisés), ce qui permet un apprentissage tout au long de l'expérimentation. En revanche, les limites sont montrées dans ce mémoire. La première d'entre elles est le grand nombre de concepts dans la construction du treillis, pour un flux d'images capturées importantes, ce qui nous a mené vers une approche locale de l'apprentissage. Ensuite, contrairement à un réseau de neurones par exemple, il n'y a pas de notion topologique intrinsèque à l'image (la position relative entre deux attributs n'est pas prise en compte). Un amer n'est donc pas toujours attaché à un objet du monde, mais peut se définir également comme une association entre des attributs non connexes. Enfin, les treillis sont construits à partir d'attributs binaires, et nécessitent donc la discrétisation des informations continues.

Au terme de notre travail, le robot est capable, dans une première phase d'exploration, d'étiqueter par un ou plusieurs amers symboliques les sites de l'environnement. Il est également apte à reconnaître ces amers dans une phase ultérieure de localisation. Cet apprentissage est donc supervisé. Dans une optique plus lointaine, un apprentissage semi-supervisé ou non supervisé est envisagé, afin de permettre la construction entièrement autonome d'une carte topologique de l'environnement structuré.

Ce mémoire reprend les grandes lignes de cette introduction. Le deuxième chapitre introduit la vision humaine et la vision par ordinateur. Il nous permet donc de présenter la façon dont l'image est utilisée, et de décrire les primitives visuelles et attributs qui serviront de base de notre apprentissage.

Le troisième chapitre traite de robotique et de navigation en robotique. Ce chapitre permet d'avoir un aperçu de la navigation en général et de la localisation en particulier d'un robot mobile.

Le quatrième chapitre donne une introduction à l'apprentissage en général et aborde les principaux outils de classification supervisée. Il expose différentes techniques de classification supervisée et permet de mettre en relief l'une des méthodes de classification utilisées dans notre application, les treillis de Galois ou treillis de concepts.

Le cinquième chapitre expose dans un premier temps le formalisme des treillis de Galois, puis commence notre contribution par l'introduction les notions importantes que nous avons développées dans le cadre de notre recherche : la classification supervisée à l'aide des treillis, la modification d'algorithmes de construction de treillis existants (Carpineto&Romano, Norris) permettant la classification de façon incrémentale, l'introduction des aspects probabilistes dans les treillis et enfin, dans une approche topologique, la decentralisation de ces treillis en treillis locaux.

Le sixième chapitre est le chapitre central de notre contribution. Nous y avons défini la notion d'amers visuels à l'aide des treillis de Galois, de couverture, d'amer plein et d'amer maximal. Suivent ensuite les différentes stratégies de localisation à l'aide d'amers visuels développées dans le cadre de cette thèse, notamment à l'aide d'amers globaux, d'amers probabilistes, et d'amers locaux, explicitement définis dans ce chapitre.

Ensuite, dans le chapitre sept, l'application directe des processus de localisation topologique a été faite sur des robots du LAAS-CNRS et de SUPAERO. On peut distinguer deux grandes parties. La première partie traite principalement de classification d'images, et des comparaisons sont effectuées avec d'autres classifieurs afin de valider théoriquement notre approche. La deuxième partie traite plus directement des applications robotiques, et les résultats relatifs aux différentes stratégies de localisation du robot sont exposés et discutés.

Enfin, dans le huitième et dernier chapitre, nous exposons nos conclusions sur la recherche effectuée dans le cadre de cette thèse, et les perspectives envisagées dans les laboratoires en termes d'apprentissage (semi- ou non supervisé) et de robotique mobile autonome.

Chapitre II

Vision et Image

Man Ray, "Tears".

"Toute la conduite de notre vie dépend de nos sens, entre lesquels celui de la vue étant le plus universel et le plus noble. Il n'y a point de doute que les inventions qui servent à augmenter sa puissance ne soient des plus utiles qui puissent être".
René Descartes, La dioptrique, 1633.

Pourquoi parler de vision humaine en robotique ? Il y a deux raisons principales. La première est que les algorithmes d'analyse d'images puisent leur origine dans la compréhension humaine des images. En effet, l'inspiration biologique est une mode "indémodable", tant que l'on ne percera pas les mystères du cerveau humain et ses stratégies d'analyse d'image.

La seconde raison est plus "philosophique" : Qu'est-ce que *voir* ? La question reste ouverte. Les dernières tendances montrent que la vision ne fonctionne pas seule, mais s'intègre dans un processus global qui réunit les sens, le contexte, les actions de l'homme, *etc.* C'est pourquoi les approches "visio-moteurs" font l'objet de recherches de plus nombreuses, même si globalement l'univers sensoriel de l'homme est essentiellement visuel.

Il n'en reste pas moins que ces mécanismes sont très complexes et ne sont absolument pas maîtrisés aujourd'hui, loin de là. Mais les nombreux "allers-retours" entre le monde du traitement d'image et le monde biologique montrent au cours du temps que cette deuxième approche est une source d'inspiration formidable pour l'analyse d'image et l'extraction d'information.

C'est pour ces raisons que nous avons voulu introduire ici, même si ce n'est pas notre spécialité, des notions relativement simples de la vision humaine.

Ce chapitre est découpé en trois parties. La première partie traite de vision humaine et décrit très succinctement la physiologie et des processus primaires liés à la vision. La deuxième partie reprend, ensuite, les bases de la vision par ordinateur et expose les outils les plus couramment utilisés pour l'analyse d'images. La troisième partie, enfin, expose les outils et algorithmes d'analyse d'images utilisés dans notre application.

II.1 Une Introduction à la Vision Humaine

Les processus humains de perception de la lumière et de la couleur sont des processus encore mal maîtrisés aujourd'hui, et ce pour une raison fort simple : le premier organe visuel est sans aucun doute le cerveau humain avec sa complexité et notre méconnaissance à son égard.

La vision est en général étudiée sous deux aspects complémentaires : l'aspect neurophysiologique et l'aspect psychophysique. La première approche est une approche microscopique, de la rétine aux aires visuelles, qui étudie la réponse des éléments de base à un stimulus visuel. La deuxième approche est une approche macroscopique et comportementale, qui étudie la réaction et le comportement du sujet dans sa globalité.

La plupart des ouvrages de vision offrent un chapitre relatif à la vision humaine, essentiellement sur les aspects physiologiques. Citons les ouvrages de Murat Kunt [Kunt 00] et de William Pratt [Pratt 01]. De nombreux autres ouvrages ont permis l'élaboration de cette partie. La très complète *Encyclopædia Universalis* donne une bonne introduction au domaine, ainsi que l'ouvrage de Bruce, Green et Georgeson [Bruce 03].

La référence en neuroscience est l'ouvrage collectif dirigé par Kandel, Schwartz et Jessell, *Principle of Neural Science* [Kansdel 00], dans lequel les différents articles, écrits par des spécialistes du domaine, sont actualisés à chaque nouvelle édition. La partie "perception" nous a appris l'essentiel des éléments apportés ici. S'ajoutent différents ouvrages [Gazzaniga 00, Gregory 00], un article écrit par un physicien du laboratoire d'optique des solides [Frigerio 01], ainsi qu'un mémoire de synthèse écrit par Dojat [Dojat 99].

II.1.1 L'œil humain

L'œil humain est l'organe de perception de la lumière. Sa physiologie est relativement bien connue à ce jour.

► Principaux éléments de l'œil humain

Les principaux éléments qui composent l'œil ($\simeq$ 8 grammes, 25 mm de diamètre, 6,5 cm^3, figure II.1) sont :

– la **cornée** : c'est une lentille fixe (11mm de diamètre, $\simeq$ 40 dioptries). Elle est privée de vaisseaux sanguins, et est nourrie par un liquide appelé **humeur aqueuse** ;

– le **cristallin** : contrôlé par le "zonule de Zinn", il permet l'accommodation de l'œil pour une vision nette ;

– l'**iris** (*"arc-en-ciel"*, en grec) : diaphragme qui contrôle le flux lumineux. Sa couleur est déterminée par la présence d'un pigment, la **mélanine**. L'iris est bleu si la mélanine est peu concentrée, et s'assombrit avec la concentration du pigment ;
– la **pupille** : ouverture contrôlée par l'iris et par laquelle passe par la lumière ;
– la **rétine** : zone de projection de l'image ;
– le **corps vitré** : corps par lequel transite l'image, et qui maintient la pression intra-oculaire ;
– le **nerf optique** : transmission des informations au cerveau.

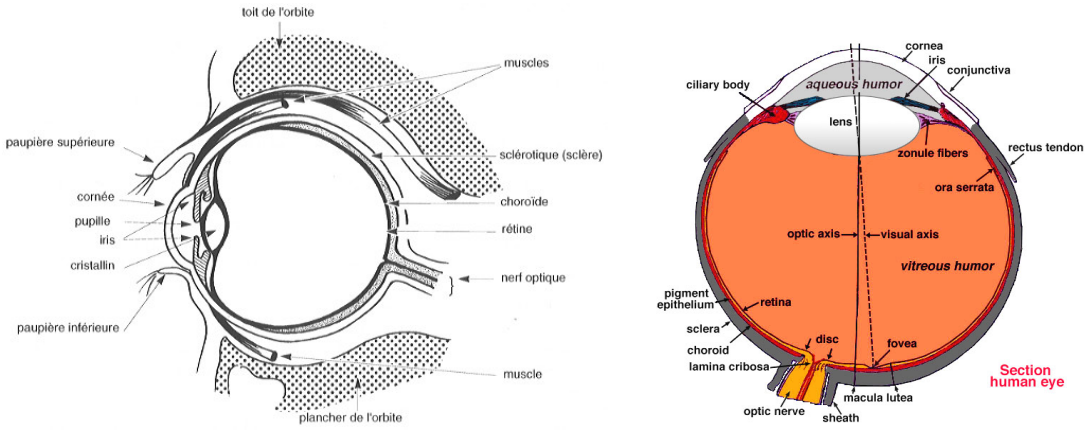


FIG. II.1 – L'œil humain.

L'homme est capable de coder les images alors que la plage de variation d'intensité peut varier considérablement : il existe un facteur supérieur à 10 milliards entre vision nocturne sans lune ni étoiles et vision diurne sur un glacier.

II.1.2 La rétine

La rétine est un élément essentiel de l'œil humain. Elle tapisse le fond et contient de nombreuses cellules. Les deux types de cellules photo-réceptrices présentes dans la rétine sont les *cônes* et les *bâtonnets*. Le fonctionnement de la rétine est décrit un peu plus loin.

► La macula et la fovéa

La **macula** (figure II.2) est la partie centrale de la rétine. Elle comprend une sorte de dépression dont le fond constitue la **fovéa** (ou **fovée**). La concentration de photo-récepteurs (cônes) est très importante dans cette zone, et moins importante dans le voisinage (implantation *log-polaire*, figure II.3). Il n'y a pas de vaisseaux dans la macula, qui est totalement dévolue à une acuité visuelle maximale.

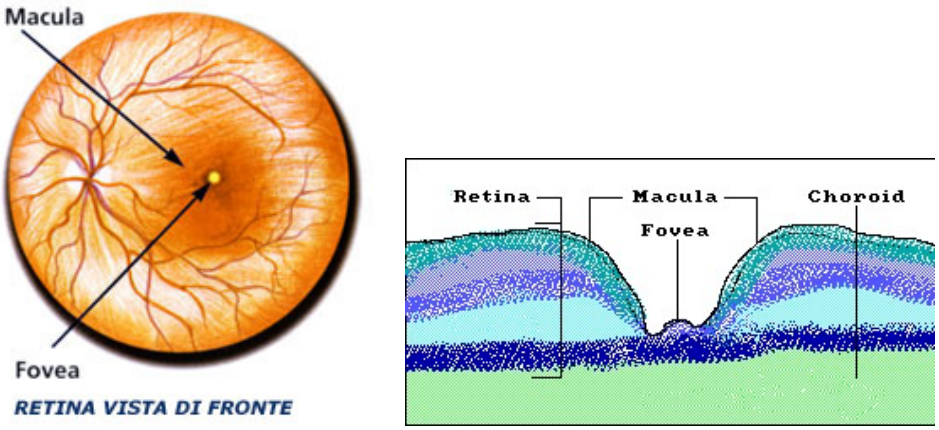


FIG. II.2 – Macula et fovéa.

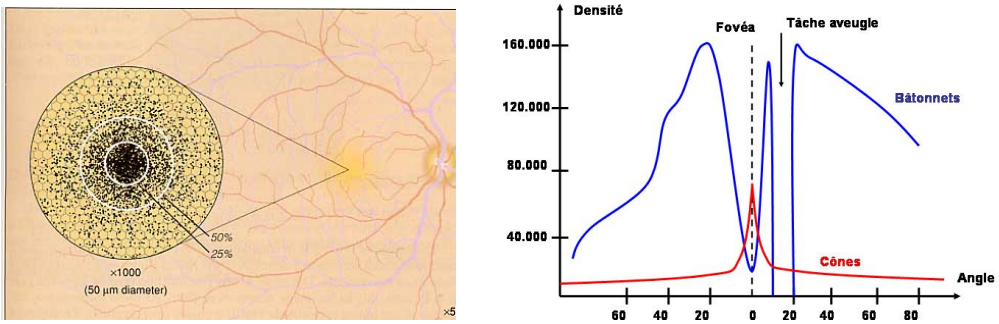


FIG. II.3 – Répartition log-polaire des cellules photo-sensibles

► La tâche aveugle

La **tâche aveugle** est la zone de la rétine où sort le nerf optique. Il n'y a donc pas de photo-récepteurs à cet endroit, ce qui signifie que l'on ne "voit" pas la lumière qui s'y projette. Dans la vie quotidienne, cette tâche aveugle n'est pas handicapante, car notre cerveau reconstitue l'image censée y être projetée.

► Les cônes

Les cônes sont des cellules de la rétine sensibles aux formes et aux couleurs. Ils nécessitent beaucoup de lumière pour fonctionner, et permettent une perception fine des détails spatiaux ainsi que la perception des couleurs.

On distingue trois types de cônes dans la rétine :
– **S** (short), de sensibilité centrée sur $430nm$ (bleu), 4 % ;
– **M** (Medium), de sensibilité centrée sur $535nm$ (vert), 32% ;
– **L** (Long), de sensibilité centrée sur $565nm$ (rouge), 64%.

On ne compte seulement que 4% de cônes "bleus", mais ils sont plus sensibles à la lumière...

Les cônes (5-7 millions) sont présents essentiellement (avec une très forte concentration) au centre du champs visuel (fovéa) où il n'y a pas de bâtonnets. Ils sont utilisés principalement pour la vision diurne.

► Les bâtonnets

Les bâtonnets (100-130 millions) sont des récepteurs de vision de faible luminosité ; ils sont si sensibles qu'exposés à la lumière du jour ils sont saturés et inutiles. Insensibles à la couleur (*"la nuit tous les chats sont gris"*), mais très sensibles au mouvement.

Les bâtonnets sont répartis en dehors de la fovée : vision périphérique. Ils ont une sensibilité maximale à 510 nm. Ils sont utilisés principalement pour la vision nocturne.

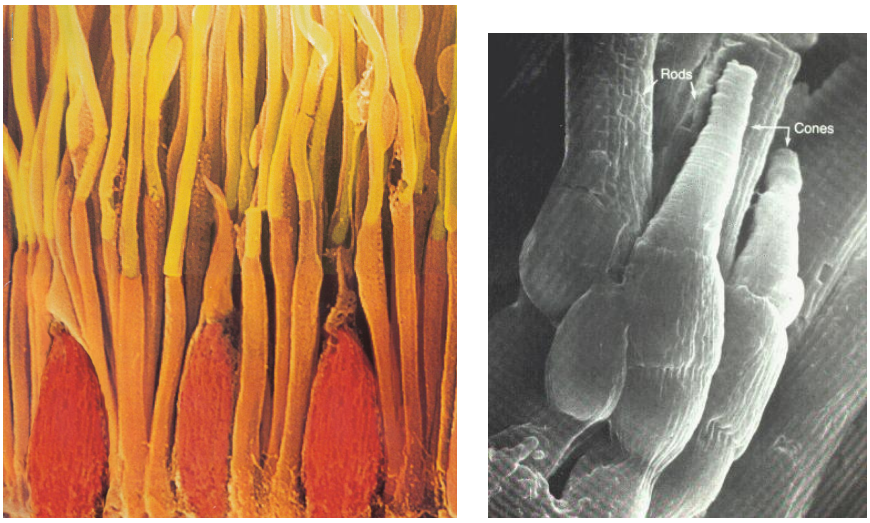


FIG. II.4 – Les cônes et les bâtonnets de la rétine humaine.

► Et ensuite...

Derrière, les cellules ganglionnaires (environ deux millions, constituant le nerf optique) répondent (fréquence de décharge) aux changements de luminance ou de couleur : le mouve-

ment des yeux rafraîchit les flux d'information, ils doivent donc être toujours en mouvement (à l'origine des saccades occulaires).

Ensuite, le traitement et la transmission de l'information visuelle sont assurés par deux millions de cellules ganglionnaires dont les axones constituent le nerf optique.

II.1.3 L'œil et la couleur

Comme nous l'avons vu précédemment, il existe trois types de cônes pour les trois couleurs primaires (c'est d'ailleurs une définition des couleurs primaires). La sensibilité spectrale des cônes et de l'œil est présenté figure II.5. Toutefois, la perception des couleurs est un processus physiologique complexe où **la longueur d'onde de la lumière est loin d'être l'unique déterminant de la sensation colorée**.

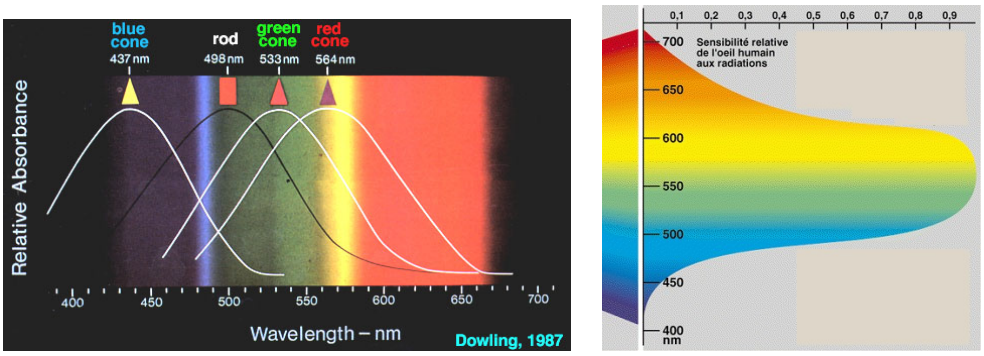


FIG. II.5 – Sensibilité spectrale des cônes et de l'œil.

Ainsi, on observe ce résultat apparemment paradoxal : le mélange convenable d'une lumière bleue et d'une lumière jaune donne la sensation oculaire du vert, qui pourtant n'y est pas. De même l'addition d'un indigo et d'un jaune donne la sensation d'un blanc, dit *suggestif*, par rapport à la lumière blanche objective (toutes les raies du spectre présentes). Deux lumières sont dites *complémentaires* quand, agissant simultanément et en proportion convenable sur l'œil moyen, elles lui donnent la sensation d'une lumière blanche.

En première approximation, la sensation produite par une lumière quelconque vient du mélange, dans un rapport donné, d'une lumière blanche et d'une lumière monochromatique dite *dominante*. Cela permet de caractériser toute lumière par trois grandeurs :

- sa **luminance**, qui dépend du flux lumineux ;
- sa **teinte**, qui est fonction de la couleur pure dominante ;
- sa **saturation**, ou **pureté**, qui s'exprime par le rapport du flux lumineux de la couleur dominante sur le flux total. La pureté de la lumière blanche est de 0, celle d'une lumière monochromatique 1.

Ainsi, alors que l'œil lui envoie une information à trois composantes rouge-vert-bleu, le cerveau *interprète* selon une autre représentation des couleurs : luminosité-teinte-saturation. Ces deux représentations seront reprises par la suite pour le codage des images couleurs.

Pour terminer, rappelons que la couleur **n'existe pas**, mais est une sensation construite par le cerveau, au même titre que la douleur par exemple.

II.1.4 De l'œil au système nerveux central, le fonctionnement de la rétine

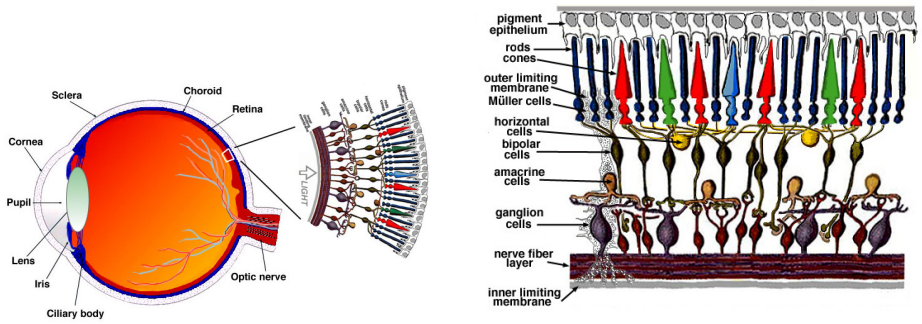


FIG. II.6 – La rétine

On estime à 200 millions le nombre de photo-récepteurs chez l'homme, alors qu'il ne possède, pour assurer la transmission de toutes les informations en sortie, que deux millions de ganglions. . .Il y a donc un traitement approfondi en termes de compression de l'information (environ 1 sur 100), qui est dû au fonctionnement de la rétine. Celle-ci est composée d'une dizaine de couches (figure II.6), dont :

– La couche des cônes et bâtonnets (ou couche des récepteurs). La couche des récepteurs est (paradoxalement) la couche la plus profondément située. Elles est composée, comme nous l'avons décrit, de cônes et de bâtonnets, qui répondent selon leur sensibilité à la lumière arrivant sur la cornée.
– La couche intermédiaire. La couche intermédiaire comporte trois types de cellules : les cellules **bipolaires**, les cellules **horizontales** et les cellules **amacrines**.
– La couche des Cellules Ganglionnaires.

Les deux dernières couches permettent de coder des contrastes en cotant "positivement" des informations issues d'une zone de projection (excitation), et négativement les zones autour de cette zone (inhibition) : on met alors en relief les contrastes de formes et de couleurs (notamment les contrastes vert-rouge et bleu-jaune).

La compression est telle qu'elle laisse à penser que les centres visuels de niveau supérieur doivent être des processeurs très efficaces pour récupérer les détails du monde visuel. On entre alors maintenant à proprement parler dans le cerveau humain.

II.1.5 Le traitement de l'information

► Le cerveau et ses neurones

Les informations visuelles issues de l'œil sont traitées par le cerveau humain. La communication et les traitements sont réalisés par des cellules nerveuses appelées **neurones**, au nombre approximatif de 10^{11} dans le cerveau humain. Il existe plusieurs types de neurones, mais ils ont tous une structure similaire (figure II.7).

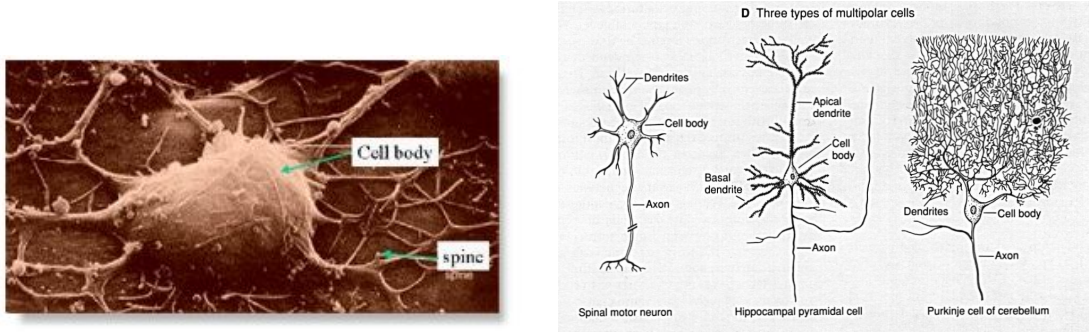


FIG. II.7 – Les neurones du cerveau

Un neurone est formé d'un corps central (ou corps cellulaire), duquel s'échappe une fibre principale appelée **axone** et un certain nombre de branches fibreuses, les **dendrites**. Un **synapse** est une jonction entre deux neurones, un axone ou plus rarement un dendrite. Les neurones émetteurs et récepteurs sont appelés respectivement **présynaptiques** et **postsynaptiques**. Les communications entre neurones se font sous forme électrique (variation de potentiels chimiques) et l'activité d'un neurone se mesure en terme de **potentiels d'action** à sa sortie. L'action d'un neurone sur un autre peut être de deux types : **excitatrice** et **inhibitrice**. Dans le premier cas, un neurone permet la génération des impulsions dans le neurone postsynaptique, donc il excite le neurone en question, alors que dans le second cas, le neurone empêche la génération des impulsions, donc inhibe le neurone.

► Après l'œil

Les axones des cellules ganglionnaires se réunissent en un faisceau, le nerf optique. C'est par ce nerf que les informations sont transmises au système nerveux central, via le **chiasma optique** puis les **corps genouillés latéraux** (CGL) qui accueillent 90% des fibres.

Il existe un relais des fibres nerveuses au niveau de deux amas de neurones d'une taille avoisinant celle des noisettes, les corps genouillés latéraux. Ce sont des relais qui permettent de trier plus finement encore les fibres nerveuses selon la zone visuelle qu'elles couvrent. Ensuite, la plupart de ces fibres arrivent à l'arrière du cerveau, au niveau du **cortex visuel**, situé dans le **lobe occipital**.

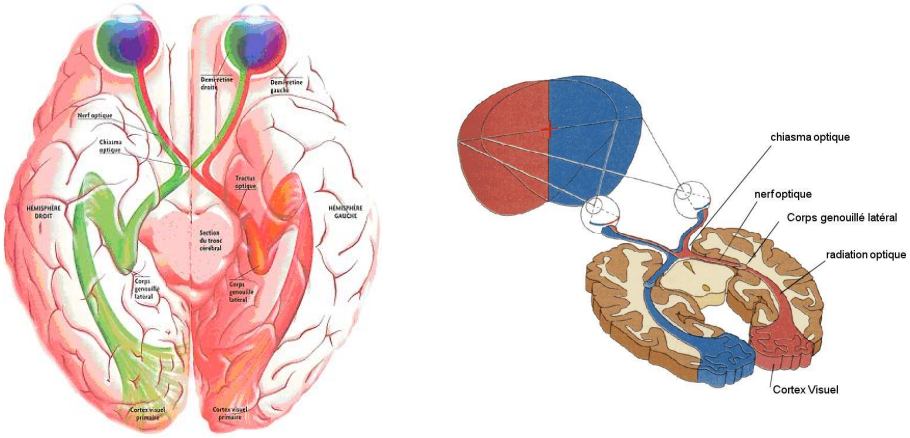


FIG. II.8 – Cheminement des informations sortantes de l'œil humain. On peut remarquer que tout le côté gauche d'une scène se projette dans l'hémisphère droit, et réciproquement tout le côté droit d'une scène se projette dans l'hémisphère gauche.

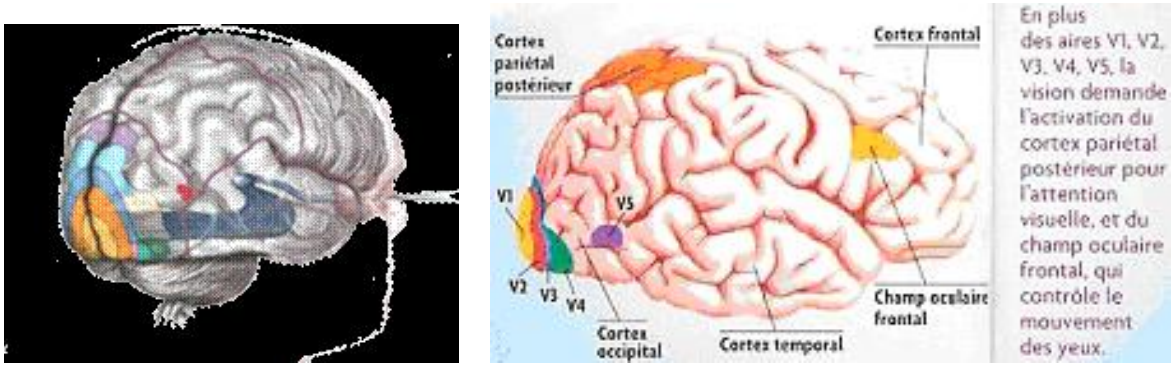


FIG. II.9 – Les aires visuelles dans le cerveau humain.

Ce cortex est composé d'une trentaine de régions différentes, appelées **aires corticales**, dont les rôles respectifs ne sont pas encore bien définis à ce jour.

► L'aire visuelle V1

L'**aire visuelle V1**, ou **aire primaire**, ou **aire 17**, est l'aire visuelle primaire. Les informations en provenance directe du CGL arrivent droit sur elle. V1 fait une première analyse des informations recueillies (forme, couleur, mouvement et disparité oculaire) et les distribue ensuite vers les autres aires.

L'aire visuelle V1 contient environ 500 millions de neurones, à comparer aux deux millions

de fibres du nerf optique arrivant de la cornée... 90% des connexions arrivant dans le cortex visuel V1 viennent du cerveau lui-même.

Il est intéressant de noter, dans ce cadre, les travaux de Bressloff *et. al.* [Bressloff 01] sur les hallucinations visuelles. Les auteurs ont modélisé mathématiquement les origines de ces hallucinations et de leur géométrie, à partir des connexions entre l'aire visuelle V1 et la rétine.

► L'aire visuelle V2

L'**aire visuelle V2**, ou **aire secondaire**, qui reçoit beaucoup d'informations de V1, les trie de façon encore plus fine. Cette aire traite à la fois les contours (cercle, carré, ovale, etc.), l'orientation (horizontale, verticale), les textures (lisse, rugueux, etc.) et les couleurs.

► Les autres aires visuelles

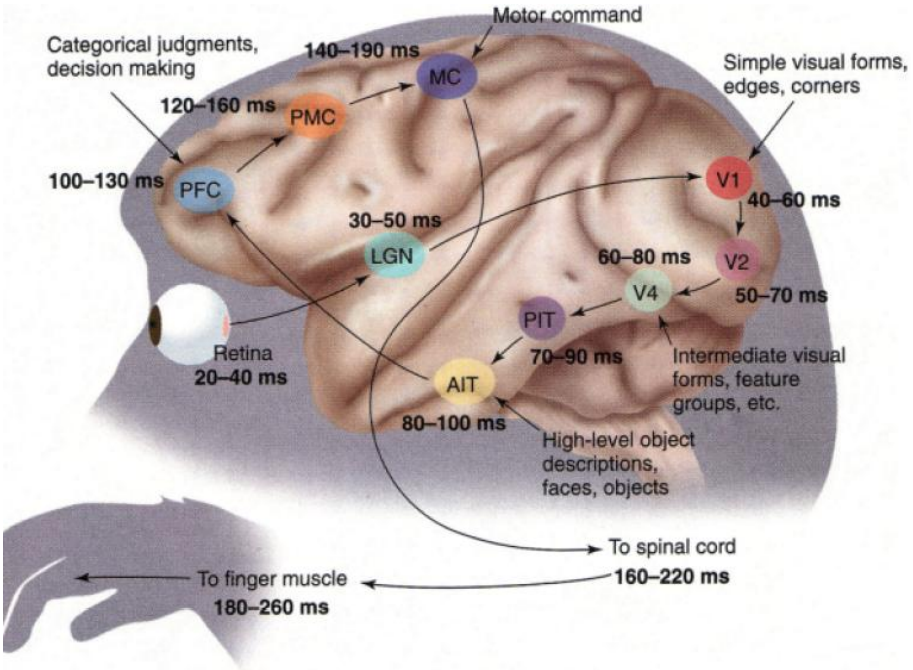
Les autres aires sont ensuite stimulées, chacune dans leur spécialité. V3 analyse les formes en mouvement et apprécie les distances. Elle est insensible à la couleur. V4 s'occupe du traitement des couleurs et des formes immobiles (orientations). V5, quant à elle, traite la perception des mouvements (direction et vitesse), etc.

Ensuite, l'information visuelle se sépare en deux voies bien distinctes :
– une partie emprunte une **voie ventrale** (le *"quoi"*) et se dirige vers le **lobe temporal**. Elle concerne l'identification des formes (contours et couleurs), donc en pratique elle permet de reconnaître les objets et les personnes ;
– l'autre partie emprunte une **voie dorsale** (le *"où"*) et se dirige vers le **lobe pariétal**. Elle concerne la localisation spatiale des objets et la coordination entre la vision et la motricité (déplacement, position et mouvement).

► Conclusion sur les aires visuelles

En conclusion, le système visuel apparaît comme un système multi-capteurs, intégrant progressivement des informations spatiales, de couleur, de profondeur et de fréquences spatiales extraites de notre environnement. La physiologie de l'aire visuelle primaire V1 nous montre qu'elle joue un rôle clef de préparation et de codage de tous ces types d'informations provenant de notre environnement sur une même carte spatiale. Cette organisation des informations, résultante de nombreux traitements bas niveau, constituerait une organisation adaptée pour des traitements de plus haut niveau tels qu'ils sont réalisés dans les aires visuelles supérieures, avec des vitesse d'analyse particulièrement élevées (figure II.10 [Thorpe 96]).

Il est important de souligner que même si certaines fonctions sont réparties entre ces aires, de très nombreuses interconnexions existent et il est encore aujourd'hui impossible d'avoir une lecture fonctionnelle très détaillée du système visuel humain. De plus, d'autres sens interviennent dans les mécanismes de perception, ce qui rend les recherches difficiles sur le sujet. De façon plus générale, de nombreux stimulus (internes, comme la sensation de faim dû à la baisse de dosage de sucre, ou externes, comme la réception de phéromones provenant d'un partenaire sexuel possible [Damasio 03]) entrent dans la boucle "perception-action", asssurant



From input to output. Monkeys can categorize complex visual stimuli very quickly, with reaction times that average 250 to 260 ms but that can be as short as 180 ms. Depicted is a plausible route between the retina and the muscles of the hand during a categorization task. Information from the retina is relayed by the lateral geniculate nucleus of the thalamus (LGN) before reaching V1, the primary visual cortex. From there, processing continues in areas V2 and V4 of the ventral visual pathway before reaching visual areas in the posterior and anterior inferior temporal cortex (PIT and AIT), which contain neurons that respond specifically to certain objects. The inferior temporal cortex projects to a variety of areas, including the prefrontal cortex (PFC), which contains the visually responsive neurons that categorize objects (*1*). To reach the muscles in the hand, signals probably need to pass via the premotor cortex (PMC) and primary motor cortex (MC) before reaching the motor neurons of the spinal cord. For each processing stage, two numbers (in milliseconds) are given: The first is an estimate of the latency of the earliest neuronal responses to a flashed stimulus, whereas the second provides a more typical average latency.

FIG. II.10 – Champs d'activité des techniques d'analyse fonctionnelle [Thorpe 96].

la survie de l'organisme homéostatique. Une architecture cognitive complexe permet au cerveau d'élaborer des images du corps, afin de connaître en quasi temps-réel l'état du système. Resntent les problèmes relatifs au temps de réponse entre l'action et la perception de l'action, soulevés par Damasio [Damasio 03].

II.1.6 Illusions d'optique

Quelques effets d'optiques sont intéressants à étudier. La plupart des analyses données dans cette partie sont issues de [Kansdel 00], du livre de Kanizsa [Kanizsa 97], et du site "Ophta-

surf"[1]

Tout d'abord, le *triangle de Kanisza* et autres figures similaires (figure II.11). Le triangle se perçoit très bien alors qu'il n'est pas dessiné. Le cerveau construit le contour entier en extrapolant les départs de ligne creusés dans les disques. Ce qui est d'autant plus remarquable, c'est qu'un triangle blanc se distingue très bien sur un fond blanc, et qu'un triangle noir se distingue très bien sur un fond noir.

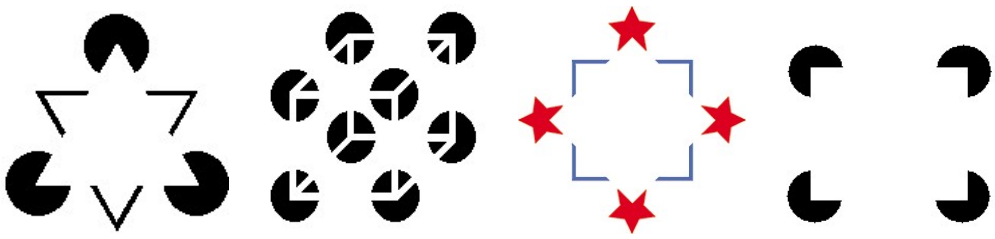


FIG. II.11 – Intégration de contours.

Ensuite, la célèbre illustration du psychologiste *Edgar Rubin* (figure II.12, à gauche) dans laquelle on peut percevoir soit un vase, soit deux visages en vis-à-vis. Apparemment, nous avons les deux visions simultanément. En réalité, nous passons alternativement très rapidement d'une interprétation à l'autre. Le tableau du peintre Dalì (figure II.12, à droite) montre également ce phénomène de double interprétation. En dehors de tout aspect artistique...

FIG. II.12 – Double interprétation. À gauche, l'illustration d'Edgar Rubin. À droite, le *"Marché d'esclaves avec apparition du buste invisible de Voltaire"*.

[1] http ://ophtasurf.free.fr

Curieusement, la longueur perçue peut être différente de la longueur réelle. Les images de la figure II.13 illustrent ce phénomène avec, à gauche, l'illusion de *Müller-Lyer*. À droite, on constate également que la taille perçue dépend des autres objets dans le champ visuel et des effets de fuite. Ainsi, le "grand" et méchant poursuivant est de même taille et aussi apeuré que le "petit" poursuivi.

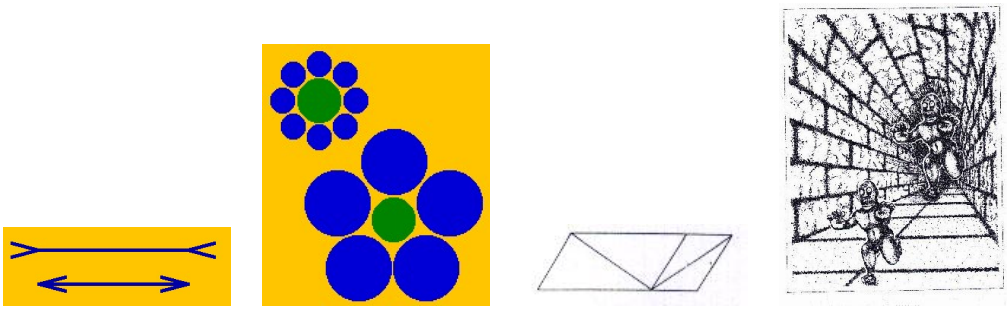


FIG. II.13 – Comparaisons de grandeurs.

Autre curiosité, les *effets d'angles* sont parmi les plus spectaculaires. Ils induisent une perception globale déformée des figures géométriques. Les exemples de la figure II.14, avec à gauche l'illusion de *Hering* et à droite l'illusion de *Zöllner*, sont impressionnants… alors que les lignes sont droites ! L'explication vient d'une double tendance, celle de sur-estimer les angles aigus et de sous-estimer les angles obtus d'une part, et de sur-estimer les cotés d'un angle obtus et de sous-estimer les côtés d'un angle aigu d'autre part.

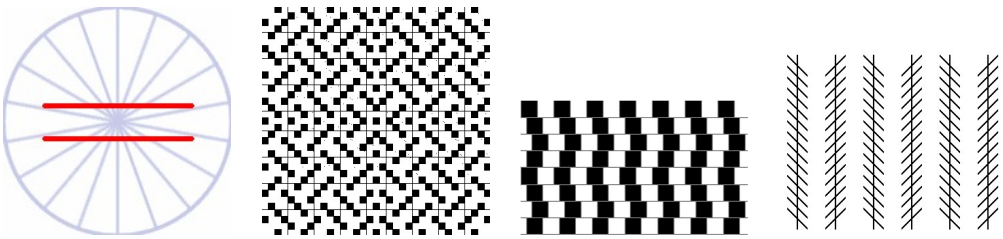


FIG. II.14 – Effets d'angle.

De la même façon, le cerveau adapte l'information concernant la luminosité d'une zone en fonction des zones voisines. Ainsi, sur la *grille d'Hermann* (figure II.15), on voit apparaître des points noirs dans certaines intersections blanches.

Enfin, pour terminer, il existe également des images fixes qui donnent une impression très nette de mouvement (figure II.16). En fait, le cerveau a du mal à exprimer les images en fonction de ce qu'il connaît, et n'arrive pas à déterminer les contours. D'où cette impression de

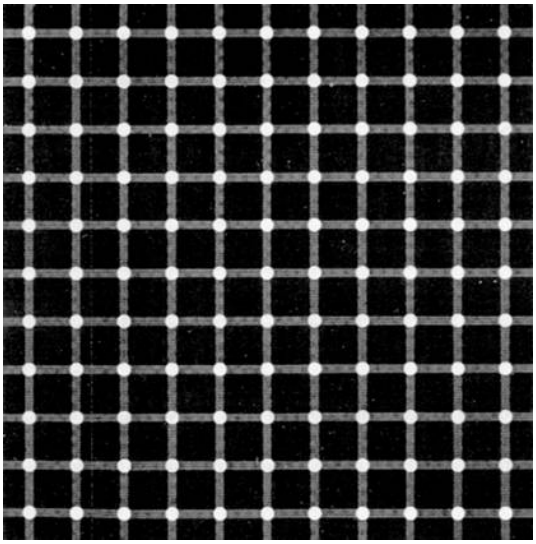


FIG. II.15 – Grille d'Hermann.

mouvement.

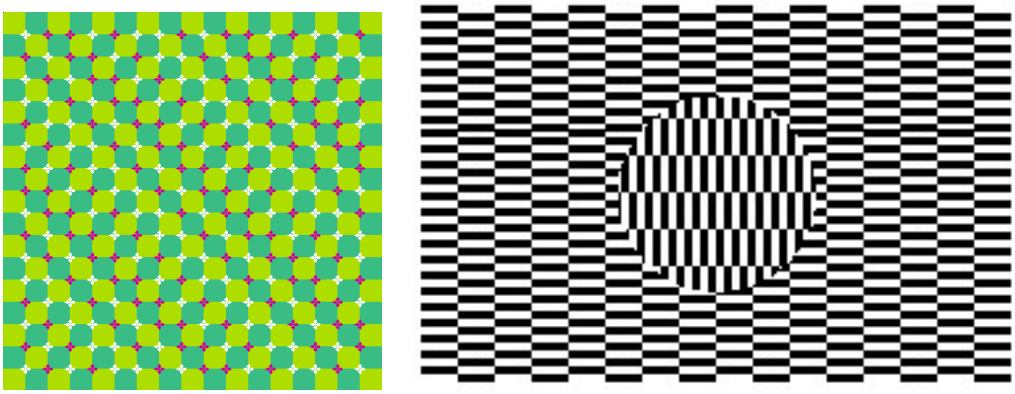


FIG. II.16 – Illusions de mouvement. À gauche, *"Trick eyes"* d'A. Kitaoka.

II.2 Vision par Ordinateur

Les deux sens les plus importants utilisés par l'homme pour communiquer avec ses semblables sont le son et l'image, domaines pour lesquels la recherche est de plus en plus active grâce notamment aux capacités de calcul -en croissance exponentielle- des machines de traitement de l'information.

Même si l'image est utilisée depuis très longtemps (les premières télévisions sont apparues dans les années 40) et analysée par des *traitements analogiques*, le *traitement numérique des images* est, quant à lui, beaucoup plus récent.

Les domaines d'application sont aujourd'hui très variés : en dehors de l'aspect "communication" évident, le traitement des images est très présent en médecine, robotique, intelligence artificielle, contrôle qualité, défense, sécurité, agriculture, météorologie, planétologie, astronomie, télédétection, *etc.*

Nous nous intéressons dans ce chapitre aux signaux discrets multidimensionnels : l'intensité $\mathcal{I}(u, v)$ d'une image a pour paramètres les coordonnées du pixel concerné ; une séquence d'images, quant à elle, est un signal tridimensionnel $\mathcal{I}(u, v, t)$ où le temps est un facteur de lien entre les images.

Dans cette section sont exposées que des notions directement en rapport avec notre activité. En plus des ouvrages précédemment cités ([Kunt 00, Pratt 01]), l'ouvrage collectif coordonné par Cocquerez et Philipp [Cocquerez 95] ansi que le livre de Jain, Kastury et Schunck [Jain 95] sont très complets sur la vision par ordinateur et les algorithmes associés.

II.2.1 Codage des images

Il existe de nombreux moyens de coder une image. Chaque codage est plus ou moins "efficace" par rapport à la quantité d'information stockée, à la vitesse de traitement, ou bien à la qualité de l'image retrouvée. Dans la pratique, le codage dépend essentiellement de l'utilisation qui est faite de l'image. L'unité élémentaire spatiale d'une image est le pixel.

Le codage spatial (matriciel) et d'intensité (sur 8 bits) est très classique. Arrêtons-nous maintenant sur les codages des couleurs que nous avons utilisés.

► Codage des couleurs

Nous avons vu que plusieurs types de photorécepteurs tapissent la rétine de l'homme, et en particulier pour les couleurs trois types de cônes coexistent : S (*Short*) pour le bleu, M (*Medium*) pour le vert, et L (*Long*) pour le rouge. Chaque cône fournit ainsi une information correspondante à l'intensité de la lumière restreinte à un spectre donné, en fonction de sa sensibilité.

Trois informations arrivent donc au niveau du cerveau afin que celui-ci *crée* de la couleur.

► Le système R-V-B

Le système R-V-B (Rouge-Vert-Bleu) ou R-G-B en anglais est le système le plus simple et le plus "naturel" (figure II.17). Il se base sur le principe de la perception humaine (cônes) et utilise donc une information de rouge, une information de vert et une information de bleu.

La plupart des couleurs perceptibles par l'œil humain peuvent être codées en fonction de ces trois composantes. Un "rouge pur", par exemple, sera codé $(255, 0, 0)$; un "jaune pur" sera codé $(255, 255, 0)$ et un "bleu-moyen" $(153, 204, 255)$.

La figure montre un échantillon de couleurs, en forme de cube (appelé *cube des couleurs*), codées à partir de ces trois couleurs primaires.

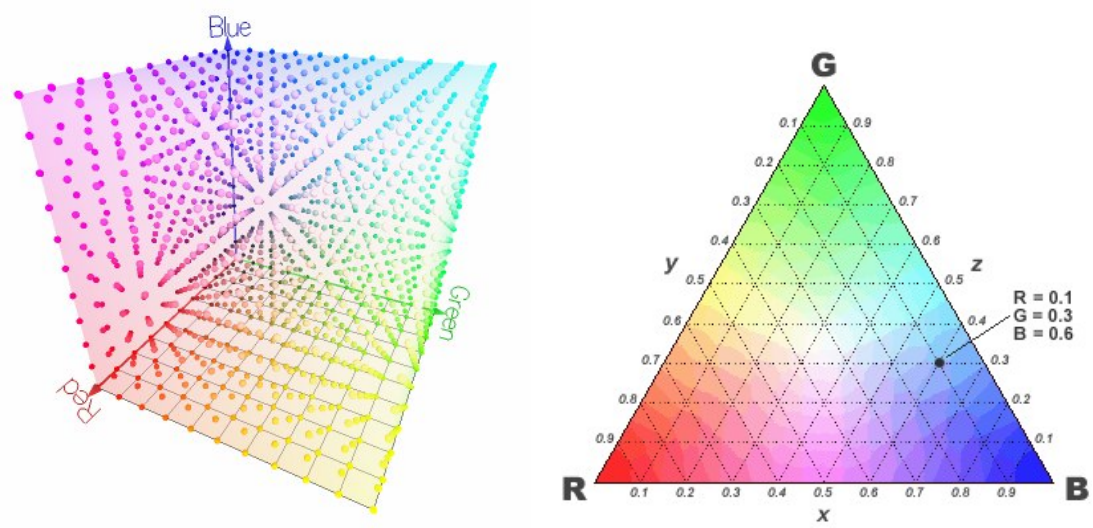


FIG. II.17 – Le système RVB et le *triangle de Maxwell*.

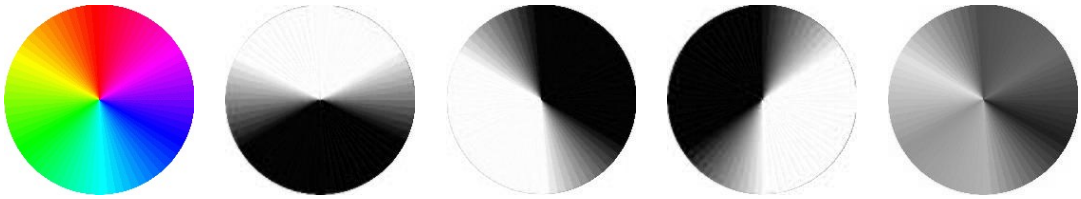
Le *triangle de Maxwell* est inscrit dans ce cube des couleurs, il est défini par les trois points de couleurs primaires. L'*axe chromatique* est l'axe traversant le triangle de Maxwell allant du noir au blanc.

La figure II.18 montre un disque tapissé du spectre visible en entier. Sur les disques qui suivent, sont reportées les composantes rouge, verte et bleue du spectre initial. Enfin, sur le dernier disque est reportée la luminance Y.

La luminance Y (ou efficacité lumineuse) de la lumière se définit de la façon suivante :

$$Y = 0,299 \times R + 0,587 \times V + 0,114 \times B \tag{II.1}$$

les coefficients étant calculés de façon à garder un spectre équi-énergétique pour chaque composante.



Spectre entier Image “rouge” Image “vert” Image “bleue” Image N&B

FIG. II.18 – Image des composantes “rouge”, “vert” et “bleu” qui composent l’image originale ; à droite, image N&B.

► Synthèses additive et soustractive

Il existe trois couleurs primaires, rouge, vert, bleu, trois couleurs secondaires, jaune, magenta, cyan, et par ailleurs 6 couleurs tertiaires, orange, vert citron, émeraude, bleu pervenche, violet, et framboise.

La synthèse additive (figure II.19) correspond à l’addition de couleurs, c’est-à-dire l’addition spectrale de signaux. La couleur ainsi obtenue peut être multispectrale, mais surtout elle est perçue par l’œil humain comme une couleur d’une certaine teinte, d’une certaine saturation et d’une certaine luminance. Ainsi la composition de la lumière *vue* peut être complètement différente du spectre *attendu*. Par exemple, un jaune *vu* peut être

- un signal monochromatique de $580nm$ de longueur d’onde,
- un signal bichromatique rouge ($650nm$) + vert ($530nm$),
- ...

Sur un ordinateur, la synthèse additive correspond à la somme des composantes RVB des couleurs de chaque pixel. Et la somme des couleurs correspond également à la somme des spectres correspondants.

La synthèse soustractive (figure II.20) permet de modéliser la résultante d’une lumière initiale par un filtre. En effet, un “filtre bleu” par exemple est un filtre qui ne laisse passer que du bleu. Si la lumière initiale est blanche, la résultante sera donc bleue. Si la lumière initiale est une lumière spectrale rouge, la résultante sera probablement nulle.

Ce phénomène est typiquement illustratif de la peinture : la couleur d’une peinture est en réalité le filtre de la lumière blanche (soleil) par cette peinture. Ainsi plus il y a de “couleurs”, plus il y a de filtres, et moins il reste de luminosité résultante. Les mélange de plusieurs couleurs donne du noir, et non du blanc. C’est également pour cette raison que les couleurs “primaires” en peinture sont le jaune, le magenta (appelé “rouge”), et le cyan (appelé “bleu”).

En vision par ordinateur, les couleurs étant affichées sur un écran ou projetées sur un écran, on travaillera toujours en couleurs additives.

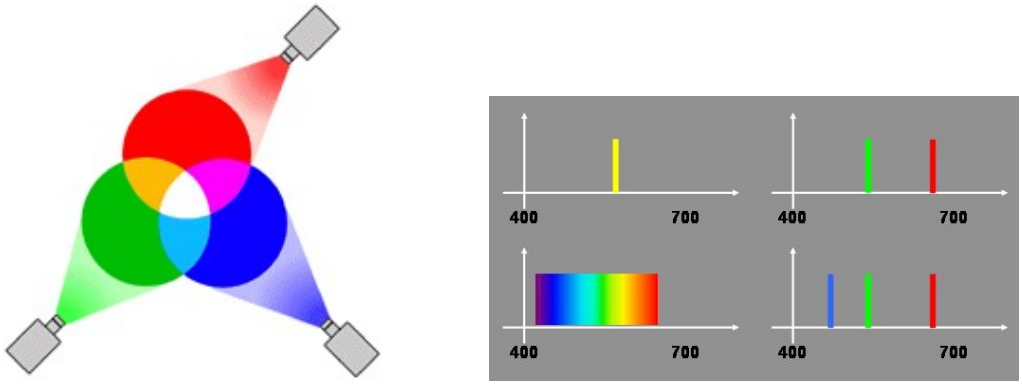


FIG. II.19 – Synthèse additive et identité de couleurs. À gauche, l'addition des trois couleurs primaires ; à droite, le jaune *perçu* peut être soit un signal monochromatique jaune, soit un signal bichromatique rouge et vert ; de même, le blanc *perçu* peut être soit une couleur blanche réelle (toutes les longueurs d'onde ont même énergie), soit un signal trichromatique rouge+vert+bleu.

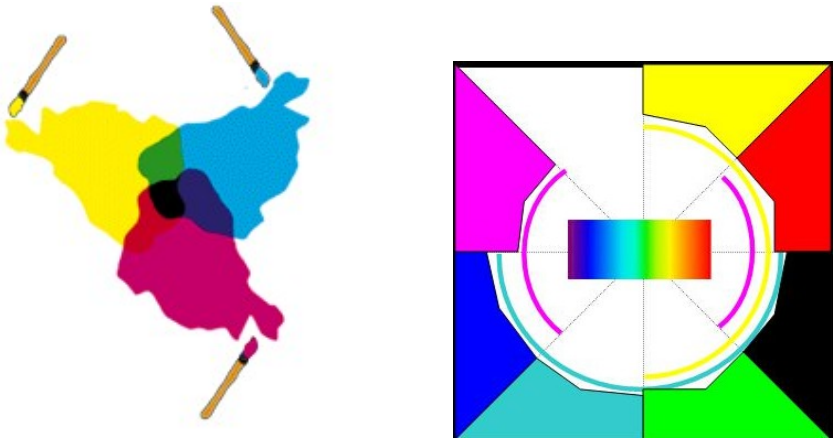


FIG. II.20 – Synthèse soustractive et filtrage de couleurs.

► Le système I-T-S

Le système I-T-S (Intensité-Teinte-Saturation) ou H-S-V (Hue-Saturation-Value) en anglais est le système basé sur trois paramètres : l'intensité, la teinte et la saturation (figure II.21).

L'**intensité** telle que définie ici représente la valeur de la composante maximale. Elle ne correspond en rien à la luminance précédemment décrite. Ce nom est en fait une traduction de l'anglais "value". Elle est quand même plus ou moins relative à une intensité lumineuse, même si cette définition n'est pas stricte. La **teinte** correspond à la longueur d'onde dominante λ, du

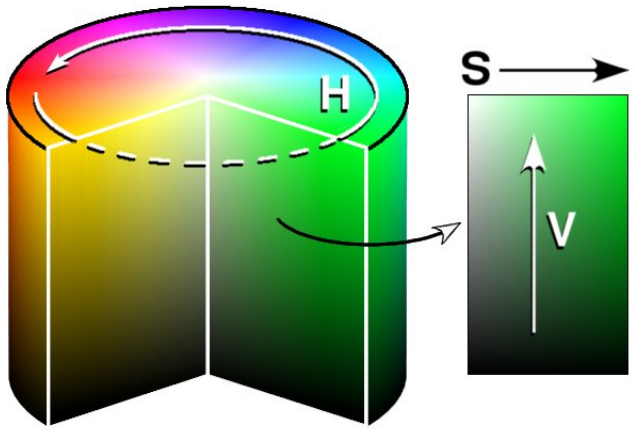


FIG. II.21 – La représentation de la lumière par intensité, teinte et saturation.

rouge ($T = 0$) au vert ($T \simeq 0,4$), puis bleu ($T \simeq 0,7$), puis de nouveau rouge ($T = 1$). Elle donne la couleur d'*impression* de la lumière. La **saturation**, ou **pureté**, correspond à la qualité de la lumière : plus la saturation est proche de 1, plus la couleur est *pure*. Plus elle est proche de 0, et plus elle est grise.

Ce code est proche de l'*interprétation* humaine, dans le sens où l'on interprète une image essentiellement en terme de teinte, saturation et luminance. On peut donc dire qu'à partir des informations des cônes, notre cerveau convertit ces données dans cette base.

Il existe plusieurs codages relatifs à cette décomposition de la couleur. Nous n'en exposerons qu'un. Les conversions entre les codages R-V-B (chaque composante est comprise entre 0 et 255) et I-T-S (la luminance et la teinte sont comprises entre 0 et 255, la saturation entre 0 et 1) s'expriment de la façon suivante :

$$\begin{aligned} I &= \max(R,V,B) \\ S &= \begin{cases} \frac{I-\min(R,V,B)}{I} & \text{si } I \neq 0 \\ 0 & \text{sinon} \end{cases} \\ T &= \begin{cases} \frac{60\times(V-B)}{S} & \text{si } I = R \\ 180 + \frac{60\times(B-R)}{S} & \text{si } I = V \\ 240 + \frac{60\times(R-V)}{S} & \text{si } I = B \end{cases} \\ \text{avec } T &= T + 360 \text{ si } T < 0 \end{aligned} \qquad \text{(II.2)}$$

La fonction réciproque est plus complexe, et résulte d'une algorithmique. Si $S = 0$, l'image

est achromatique (grise) ; on a alors chaque composante identique $R = V = B = I$. Si S est non nulle, on pose : $T_i = \lfloor T/60 \rfloor$ (partie entière), $f = (T/60) - T_i$ (f est donc le reste), et $p = L \times (1 - S)$, $q = L \times (1 - (S \times f))$, $r = L \times (1 - (S \times (1 - f)))$,

$$\begin{aligned}
&\text{si } T_i = 0, \text{ alors } R = I, V = r, etB = p \\
&\text{si } T_i = 1, \text{ alors } R = q, V = I, etB = p \\
&\text{si } T_i = 2, \text{ alors } R = p, V = I, etB = r \\
&\text{si } T_i = 3, \text{ alors } R = p, V = q, etB = I \\
&\text{si } T_i = 4, \text{ alors } R = r, V = p, etB = I \\
&\text{si } T_i = 5, \text{ alors } R = I, V = p, etB = q
\end{aligned} \tag{II.3}$$

► Le système Y-Cr-Cb

Lorsque les télévisions couleurs sont apparues à la fin des années 60 (la première émission en couleur diffusée en France date du 1 octobre 1967 à 14h15 devant 1500 privilégiés ; pour donner un point de repère, la première diffusion d'*Interville* date de 1962 et celle des *Shadocks* du 29 avril 1968), il a fallu trouver un codage qui soit compatible avec les anciennes télévisions N&B. Il a donc fallu "transporter" indépendamment les informations de luminance (la luminance est l'intensité pondérée avec la fonction de sensibilité de l'œil) avec les informations de couleur (*chrominance*).

Le système Y-Cb-Cr s'appuie donc sur la luminance Y et sur deux canaux de chrominance Cb-Cr. Les composantes de chrominance se définissent comme suit :

$$\begin{cases} Cr = \alpha_1(R - Y) + \beta_1(B - Y) \\ Cb = \alpha_2(R - Y) + \beta_2(B - Y) \end{cases}$$

Les variables $\alpha_1, \alpha_2, \beta_1$ et β_2 sont spécifiques aux différents standards (NTSC, PAL, ou SECAM.)

II.2.2 Filtrage et convolution

Dans cette section sont définis des outils de base utilisés régulièrement dans le traitement et l'analyse d'images. Sont présentés le *produit de convolution*, suivi de la notion de *filtrage linéaire* d'une image par un *masque de convolution*, base de la plupart des traitements d'image. Nous introduisons donc ici ces outils et le formalisme mathématique correspondant.

► Filtrage linéaire

On appelle filtre linéaire un système à la fois linéaire et invariant dans le temps. Ceci implique qu'une combinaison linéaire d'entrées entraîne la même combinaison linéaire des sorties, et de plus tout décalage temporel de l'entrée entraîne le même décalage en sortie. Dans le domaine temporel, l'entrée et la sortie sont liées par une opération de convolution introduisant

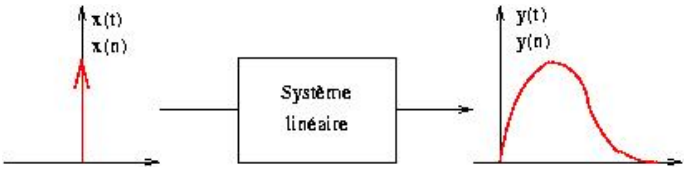


FIG. II.22 – Filtrage linéaire.

la réponse impulsionnelle $h(t)$ ou $h(n)$, qui définit le filtre linéaire (figure II.22).

La sortie $y(t)$ d'un signal $x(t)$ passant par un filtre linéaire de réponse impulsionnelle $h(t)$ est déterminée par la relation :

$$y(t) = \int_{-\infty}^{+\infty} x(\tau)h(t-\tau)d\tau \tag{II.4}$$

Nous retrouvons ici la formule classique de la convolution. On peut donc écrire, dans le domaine fréquentiel,

$$Y(f) = H(f).X(f) \tag{II.5}$$

Dans le cas discret, le raisonnement est identique et

$$y(n) = \sum_{k=-\infty}^{+\infty} h(k)x(n-k) \tag{II.6}$$

et les transformées en Z nous permettent d'écrire

$$Y(z) = H(z).X(z) \tag{II.7}$$

► Produit de convolution bidimensionnel

Afin de définir le filtrage linéaire d'un signal bidimensionnel $\mathcal{I}(u,v)$ par un filtre bidimensionnel de réponse impulsionnelle $H(u,v)$, on définit le produit de convolution bidimensionnel -à l'aide de l'opérateur "*" doublé- de la façon suivante :

$$\begin{aligned} Y(u,v) &= \mathcal{I} * *H \\ Y(u,v) &= \sum_{u'=-\infty}^{+\infty}\sum_{v'=-\infty}^{+\infty} \mathcal{I}(u',v')H(u-u',v-v') \\ Y(u,v) &= \sum_{u'=-\infty}^{+\infty}\sum_{v'=-\infty}^{+\infty} H(u',v')\mathcal{I}(u-u',v-v') \end{aligned}$$

► Masque de convolution

Dans le cadre du traitement d'image, on ne parle pas pour $H(u,v)$ de filtre mais de **masque de convolution** ou **noyau de convolution**, et on utilise le simple opérateur "*" pour définir la double convolution.

Ces masques de convolution sont des opérateurs sur l'image de type moyenneur (passe-bas), dérivateur (passe-haut), laplacien, *etc.* Nous les verrons en détail par la suite. Ils sont en général de petite taille (3×3 à 11×11, rarement plus grands). Le produit de convolution consiste en pratique à appliquer, en chaque pixel de l'image initiale, une somme des intensités de ses pixels voisins (dont la distance de voisinage est définie par la taille du masque) pondérée par les coefficients du masque.

Ainsi, pour tout pixel (u,v) de l'image initiale $\mathcal{I}$ non situé à ses bords[2], la convolution par le masque $H(2m+1, 2n+1)$ donne

$$\begin{aligned} Y(u,v) &= \mathcal{I}(u,v) * H(u,v) \\ Y(u,v) &= \sum_{k=-m}^{m} \sum_{l=-n}^{n} \mathcal{I}(u-k, v-l) \times H(k,l) \end{aligned}$$

► Filtres classiques

Il existe un certain nombre de filtres classiques très utilisés dans le traitement d'images. Nous distinguons principalement trois catégories, les filtres "passe-bas" (*moyenneur, médian*), les filtres "dérivateurs d'ordre 1" (*Roberts, Sobel, Prewitt, Canny*), et les filtres "dérivateurs d'ordre 2" (*Laplacian of gaussian*). Ils seront exposés en détail par la suite.

II.2.3 Histogramme d'une image

L'**histogramme** d'une image N&B est un outil de représentation qui permet de montrer la répartition des niveaux de gris dans l'image.

Dans le cas discret, la définition de l'histogramme est très simple et consiste à donner le nombre de pixels ayant le niveau de gris n :

$$\mathcal{H}_{\mathcal{I}}(n) = \sum_{u=1}^{U} \sum_{v=1}^{V} |\mathcal{I}(u,v) = n| \tag{II.8}$$

Dans le cas de l'exemple suivant (figure II.23), on voit apparaître clairement deux classes de pixels : les pixels plutôt clairs, qui sont présents principalement sur le masque du personnage, et les pixels plutôt sombres, partout ailleurs sur l'image.

On obtient ainsi la répartition des pixels en fonction de l'intensité lumineuse. Plusieurs types d'informations peuvent être obtenus à la lecture de cet histogramme : si une image est

[2] c'est-à-dire vérifiant les quatre conditions suivantes : $u \geq m$, $u \leq U - m$, $v \geq n$ et $v \leq V - n$.

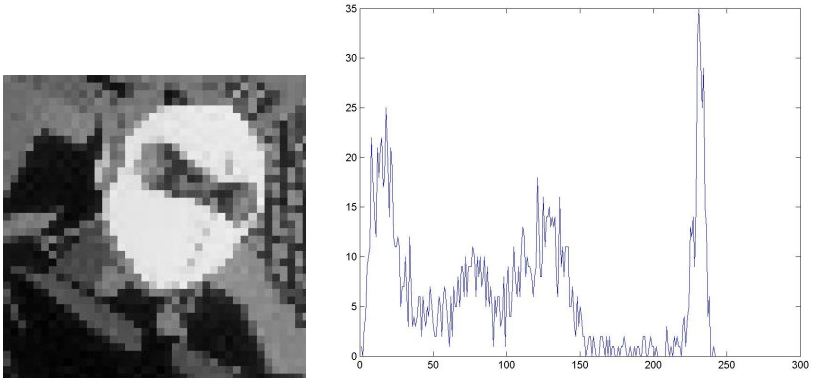


FIG. II.23 – Histogramme d'une image. L'image originale est de taille 40×40, pour 256 niveaux de gris.

sombre ou claire, s'il existe des classes de pixels permettant un début de segmentation, le nombre de niveaux de gris, *etc.*

II.2.4 Spectre colorimétrique d'une image

Le spectre colorimétrique d'une image correspond au sens classique du spectre en physique : une quantité d'énergie (ici, des pixels) attachée à une bande de fréquence de la lumière émise.

La représentation la plus adaptée à cela est la représentation I-T-S, et notamment la composante de teinte T. Le spectre colorimétrique d'une image peut être représenté par l'histogramme de la composante de teinte.

Il faut cependant noter qu'une image blanche n'est pas une image ou toutes les teintes sont représentées, mais une image de forte intensité. De la même façon, une image noire "sans énergie" peut cependant être représentée au niveau spectral.

II.2.5 Pré-traitements d'une image

Les dispositifs d'acquisition et de transmission d'images introduisent inéluctablement de nombreuses perturbations dans l'image qu'ils délivrent. Il est donc nécessaire, en général, d'effectuer une chaîne de pré-traitements avant de travailler sur ces images et d'exploiter de l'information utile.

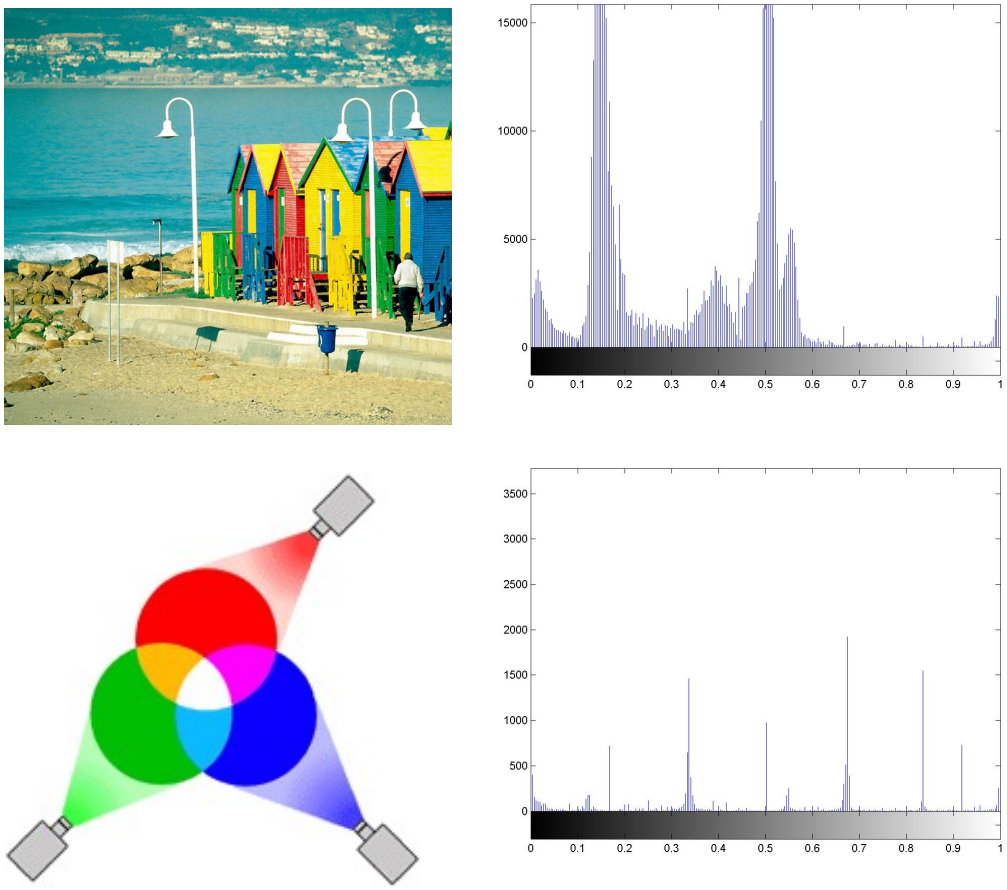


FIG. II.24 – Spectre colorimétrique d'images. L'abscisse du spectre est une graduation des couleurs (comprise entre 0 et 1), l'ordonnée représente le nombre de pixels attachés à chaque couleur.

► Expansion d'histogramme

Le principe d'une telle transformation est d'utiliser toute la palette lumineuse d'une image en niveaux de gris afin de la rendre plus "visible", si celle-ci a été sous-exposée ou sur-exposée par exemple.

Supposons une image et son histogramme associé, compris entre n_0 et n_1 (figure II.25). On constate qu'une partie du spectre est inexistante, de part une sous exposition lors de la prise de vue.

Afin de pouvoir visualiser au mieux cette image, il est nécessaire d'étaler le spectre sur toute la plage visible. Ainsi, les limites du spectre initial sont repoussées (linéairement ou non) à k_{min} (en général 0) et k_{max} (en général 255). Ainsi, dans le cas linéaire, les pixels d'intensité

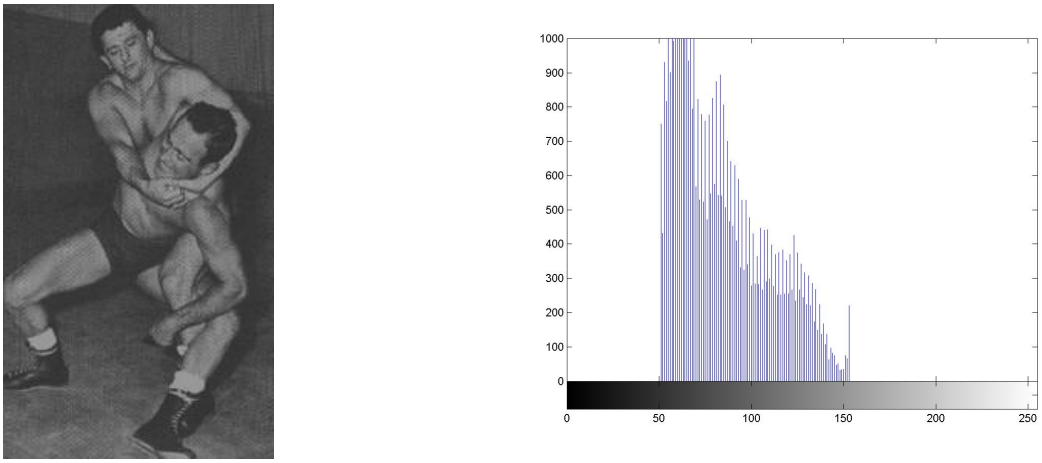


FIG. II.25 – Une image et son histogramme.

k quelconque seront ajustés à la nouvelle intensité $k' = f(k)$:

$$f(k) = \frac{k_{max}(k - k_0) - k_{min}(k - k_1)}{k_1 - k_0} \tag{II.9}$$

et en général,

$$f(k) = 255.\frac{k - k_0}{k_1 - k_0} \tag{II.10}$$

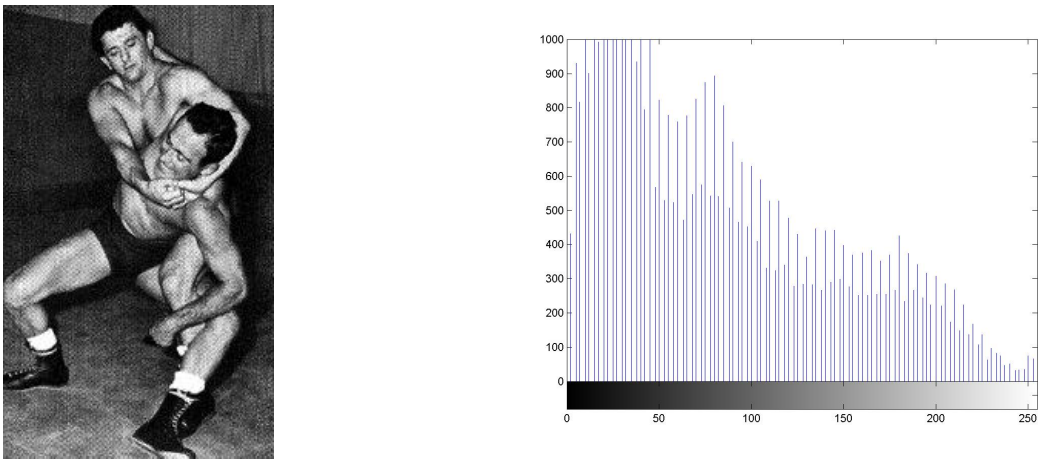


FIG. II.26 – Image rectifiée par expansion d'histogramme.

► Égalisation d'histogramme

Cette transformation consiste à rendre le plus "plat" possible l'histogramme, c'est-à-dire de rendre les niveaux de gris équiprobables dans l'intervalle $[f_{min}, f_{max}] = [0, 255]$.

Soit p_i le nombre de pixel d'intensité z_i dans l'histogramme initial, et q_i le nombre de pixel d'intensité z_i dans l'histogramme final. Puisque celui-ci est voulu uniforme, $q_i = q, \forall i$.

Le processus est itératif. Il s'agit dans un premier temps de trouver k_1 tel que :

$$\sum_{i=0}^{k_1-1} p_i \leq q < \sum_{i=0}^{k_1} p_i \tag{II.11}$$

Tous les pixels d'intensité $z_1, z_2, \ldots, z_{k_1-1}$ sont ramenés à z_1. Ensuite, pour la deuxième itération, il s'agit de trouver k_2 tel que :

$$\sum_{i=0}^{k_2-1} p_i \leq 2 \times q < \sum_{i=0}^{k_2} p_i \tag{II.12}$$

Les pixels d'intensité $z_{k_1}, \ldots, z_{k_2-1}$ sont ramenés à z_2. Et ainsi de suite. Un exemple illustratif est donné figure II.27.

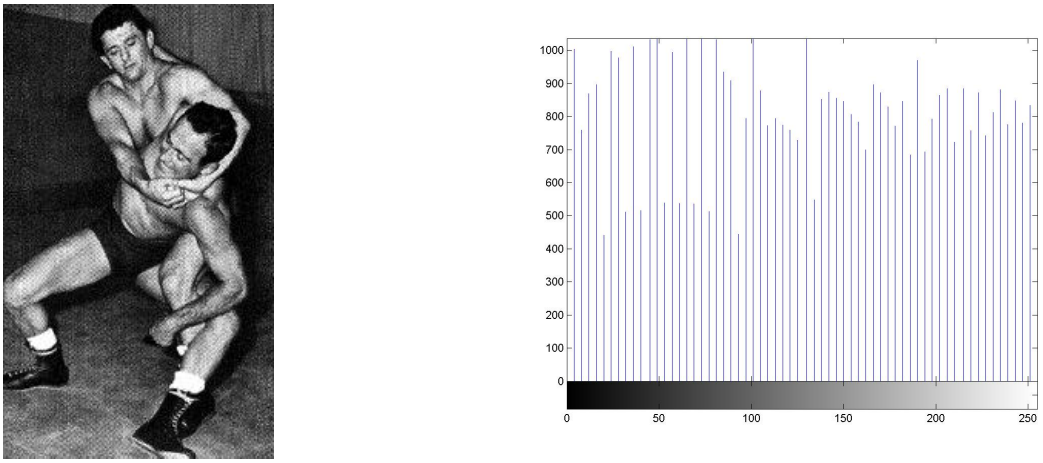


FIG. II.27 – Image rectifiée par égalisation d'histogramme.

On constate que l'histogramme n'est pas pleinement équiprobable, se qui s'explique par une distribution discrète des intensités des pixels dans l'image initiale.

► Réduction du bruit

Il existe plusieurs modélisations du bruit : sa nature plus ou moins impulsionnelle peut être décrite à l'aide de densités de probabilités de la forme :

$$f(a) = Ce^{-K|a|^\alpha} \tag{II.13}$$

pour laquelle on retrouve un bruit gaussien pour $\alpha = 2$, et un bruit exponentiel pour $\alpha = 1$. D'un point de vue matriciel, le bruit peut être considéré comme additif ($I' = I + B$), multiplicatif ($I = I.B$) ou convolutif ($I = I * B$).

La méthode la plus simple et la plus naturelle pour réduire le bruit, supposé aléatoire, centré et additif, est d'appliquer sur l'image un filtre passe-bas, c'est-à-dire un masque moyenneur (carré) :

$$H(u,v) = \frac{1}{U_M^2} \begin{pmatrix} 1 & 1 & \dots & 1 \\ 1 & 1 & \dots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \dots & 1 \end{pmatrix} \tag{II.14}$$

dont la taille influe sur le résultat final : plus la taille est grande, plus le lissage est important ce qui a pour effet de réduire les contrastes et de rendre la photo floue ; en revanche, si la taille du masque est trop petite, le filtrage risque de ne pas être très efficace. Il y a donc un compromis à trouver sur la taille de ce masque.

Le terme $\frac{1}{U_M^2}$ permet de garder constante l'intensité moyenne de l'image. De façon générale, le masque de convolution $\mathcal{H}$ doit vérifier :

$$\sum_{u=-U_M}^{U_M} \sum_{v=-V_M}^{V_M} H(u,v) = 1 \tag{II.15}$$

Une autre possibilité consiste à différencier les coefficients de la matrice. Pour un masque 3×3, on peut trouver par exemple

$$H(u,v) = \frac{1}{10} \begin{pmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{pmatrix} \tag{II.16}$$

ce qui permet de donner plus de poids au pixel courant.

Enfin, un dernier filtre, non linéaire celui-là, est souvent utilisé pour éviter l'influence de pixels trop clairs ou trop sombres : le filtre médian. Il affecte au pixel courant la valeur médiane des pixels dans une fenêtre donnée. Par exemple, si la valeur des pixels dans une fenêtre 5×5 est de la forme suivante :

$$\mathcal{I}(u,v) = \begin{bmatrix} \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & 55 & 59 & 61 & 69 & 71 & \dots \\ \dots & 59 & 255 & 255 & 70 & 73 & \dots \\ \dots & 61 & 63 & \mathbf{63} & 72 & 75 & \dots \\ \dots & 61 & 63 & 67 & 77 & 81 & \dots \\ \dots & 65 & 69 & 71 & 79 & 81 & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \end{bmatrix} \tag{II.17}$$

La valeur du pixel central est remplacée par la valeur médiane de la liste de 25 pixels : 69 alors que la valeur moyenne est égale à 83. On évite ainsi le piège de pixels parasites. Le filtre médian est particulièrement efficace pour éliminer le bruit du type *"poivre et sel"*.

II.2.6 Extraction de Contours

Les contours dans une image fournissent des informations essentielles sur le contenu de cette image. Un objet est souvent mieux "reconnu" à ses contours plutôt qu'à son contenu. De plus, les études physiologiques montrent que l'être humain y est très sensible.

Les contours sont des variations importantes d'intensité lumineuse ou de couleur dans l'image. Il est donc nécessaire de s'intéresser à leur dérivée spatiale.

► Généralités

Considérons une image continue quelconque en niveau de gris : $\mathcal{I}(x,y)$. On peut définir sa dérivée selon les deux directions x et y :

$$\nabla\mathcal{I}(x,y) = \frac{\partial\mathcal{I}(x,y)}{\partial x} + \frac{\partial\mathcal{I}(x,y)}{\partial y} = \nabla_x\mathcal{I}(x,y) + \nabla_y\mathcal{I}(x,y) \tag{II.18}$$

soit $\theta(x,y)$ l'angle de direction du gradient, on a :

$$\tan\left(\theta(x,y)\right) = \frac{\nabla_y\mathcal{I}(x,y)}{\nabla_x\mathcal{I}(x,y)} \tag{II.19}$$

En discret, la dérivation peut se traduire de différentes manières : soit en appliquant strictement la définition de la dérivée :

$$\nabla_u\mathcal{I}(u,v) = \frac{\partial\mathcal{I}(u,v)}{\partial u} = \mathcal{I}(u+1,v) - \mathcal{I}(u,v)$$

et

$$\nabla_v\mathcal{I}(u,v) = \frac{\partial\mathcal{I}(u,v)}{\partial v} = \mathcal{I}(u,v+1) - \mathcal{I}(u,v) \tag{II.20}$$

soit en "centrant" la dérivation autour du point courant :

$$\nabla_u\mathcal{I}(u,v) = \frac{\partial\mathcal{I}(u,v)}{\partial u} = \mathcal{I}(u+1,v) - \mathcal{I}(u-1,v)$$

et

$$\nabla_v \mathcal{I}(u,v) = \frac{\partial \mathcal{I}(u,v)}{\partial v} = \mathcal{I}(u,v+1) - \mathcal{I}(u,v-1) \tag{II.21}$$

avec toujours

$$\theta(u,v) = \arctan\left(\frac{\nabla_v \mathcal{I}(u,v)}{\nabla_u \mathcal{I}(u,v)}\right) \tag{II.22}$$

Ainsi, en composant les deux dérivées,

$$\nabla \mathcal{I}(u,v) = \mathcal{I}(u+1,v) - \mathcal{I}(u-1,v) + \mathcal{I}(u,v+1) - \mathcal{I}(u,v-1) \tag{II.23}$$

et l'on peut donc écrire que

$$\nabla \mathcal{I} = \mathcal{H} * \mathcal{I} \tag{II.24}$$

avec

$$\mathcal{H} = \begin{pmatrix} 0 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \tag{II.25}$$

Dans la pratique, cependant, il est préférable de décomposer la dérivation selon les deux axes de l'image :

$$\mathcal{H}_u = \begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix} \text{ et } \mathcal{H}_v = \begin{pmatrix} -1 & 0 & 1 \end{pmatrix} \tag{II.26}$$

On note qu'en général le gradient vertical est dirigé vers le haut :

$$\mathcal{H}_u = \begin{pmatrix} 1 \\ 0 \\ -1 \end{pmatrix} \text{ et } \mathcal{H}_v = \begin{pmatrix} -1 & 0 & 1 \end{pmatrix} \tag{II.27}$$

ce qui inverse spatialement la définition de la dérivée.

La dérivée d'une image peut être donc vue comme la convolution de l'image par des noyaux dérivatifs. La détection des contours se fait donc essentiellement par l'entremise de filtrages linéaires. Nous allons en voir quatre ici, très classiques : les filtres de Roberts, Prewitt, Sobel, suivis des filtres de deuxième ordre.

► Filtre de Roberts

Le filtre de Roberts est un filtre diagonal, issu de l'opérateur II.20, de la forme suivante :

$$\mathcal{H}_{\diagdown} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \text{ et } \mathcal{H}_{\diagup} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \tag{II.28}$$

Le filtre de Roberts est adapté aux contours diagonaux. Il est cependant très sensible aux bruits.

► Filtre de Prewitt

Le filtre de Prewitt se présente de la forme suivante :

$$\mathcal{H}_u = \frac{1}{3}\begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{pmatrix} \text{ et } \mathcal{H}_v = \frac{1}{3}\begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix} \tag{II.29}$$

On remarque que le gradient est ici orienté vers le haut.

► Filtre de Sobel

Le filtre de Sobel est une généralisation du filtre de Prewitt :

$$\mathcal{H}_u = \frac{1}{c+2}\begin{pmatrix} 1 & c & 1 \\ 0 & 0 & 0 \\ -1 & -c & -1 \end{pmatrix} \text{ et } \mathcal{H}_v = \frac{1}{c+2}\begin{pmatrix} -1 & 0 & 1 \\ -c & 0 & c \\ -1 & 0 & 1 \end{pmatrix} \tag{II.30}$$

avec en général $c = 2$.

Ces deux derniers filtres sont plus stables que le filtre de Roberts.

► Filtres du deuxième ordre

Le filtre laplacien est un filtre du deuxième ordre défini de la façon suivante :

$$\mathcal{L}(\mathcal{I}(x,y)) = \nabla^2(\mathcal{I}(x,y)) = \frac{\partial^2 \mathcal{I}}{\partial x^2} + \frac{\partial^2 \mathcal{I}}{\partial x^2} \tag{II.31}$$

qui se traduit, en terme de convolution, par les masques suivants :

$$\mathcal{H} = \begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix} \text{ ou } \mathcal{H} = \begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix} \tag{II.32}$$

Pour détecter des contours, il est nécessaire de détecter les passages par zéro de l'image filtrée. Pour cela, le plus simple est d'utiliser un autre filtre, le "Laplacien de Gaussienne" (LoG), qui permet de filtrer l'image dans un même temps. Soit $\mathcal{Y}(x,y)$ l'image convoluée et $g(x,y)$ une gaussienne d'écart-type σ,

$$\mathcal{Y}(x,y) = LoG\big(\mathcal{I}(x,y)\big) = \nabla^2\big(g(x,y) * \mathcal{I}(x,y)\big) \tag{II.33}$$

$$= \big(\nabla^2 g(x,y)\big) * \mathcal{I}(x,y) \tag{II.34}$$

avec

$$\nabla^2 g(x,y) = \big(\frac{x^2+y^2-2\sigma^2}{\sigma^4}\big).e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{II.35}$$

On obtient une forme de "chapeau mexicain" dont la traduction numérique, en terme de masque de convolution, est pour une matrice 5×5 de la forme :

$$\mathcal{H} = \begin{pmatrix} 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & -2 & -1 & 0 \\ -1 & -2 & 16 & -2 & -1 \\ 0 & -1 & -2 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 \end{pmatrix} \tag{II.36}$$

► Le filtre de Canny

Le détecteur de Canny est un détecteur gaussien qui optimise la localisation des dérivées et la qualité du signal final (ratio signal/bruit).

Considérons l'image initiale $\mathcal{I}(u,v)$ convoluée avec une gaussienne, $\mathcal{I}_g$, et deux opérateurs de dérivation simples $\mathcal{H}_u$ et $\mathcal{H}_v$:

$$\mathcal{H}_u(u,v) = \begin{pmatrix} -1 & 1 \\ -1 & 1 \end{pmatrix} \text{ et } \mathcal{H}_v(u,v) = \begin{pmatrix} 1 & 1 \\ -1 & -1 \end{pmatrix} \tag{II.37}$$

La convolution de cette nouvelle image avec ces deux masques dérivateurs permet d'en déduire la magnitude et l'angle de la dérivée :

$$\begin{aligned} \mathcal{M}(u,v) &= \sqrt{\mathcal{P}^2(u,v) + \mathcal{Q}^2(u,v)} \\ \theta(u,v) &= \arctan \frac{\mathcal{Q}(u,v)}{\mathcal{P}(u,v)} \end{aligned} \tag{II.38}$$

avec

$$\begin{aligned} \mathcal{P}(u,v) &= \mathcal{I}_g(u,v) * \mathcal{H}_u(u,v) \\ \mathcal{Q}(u,v) &= \mathcal{I}_g(u,v) * \mathcal{H}_v(u,v) \end{aligned} \tag{II.39}$$

L'idée maintenant est de ne garder que les maxima locaux de $\mathcal{M}$. Pour cela, la direction $\theta(u,v)$ du gradient est discrétisé selon les quatre directions (0 à 3) possibles dans une "8-connexité"

$$\zeta(u,v) = \text{Sector}\big(\theta(u,v)\big) \qquad \begin{pmatrix} 3 & 2 & 1 \\ 0 & & 0 \\ 1 & 2 & 3 \end{pmatrix} \tag{II.40}$$

Ensuite, en chaque point de la matrice de magnitude $\mathcal{M}$, on compare le pixel central avec ses deux voisins définis par la direction ζ. Si le pixel central est inférieur à l'un de ses deux voisin, il passe à zéro. Cette procédure permet d'affiner les contours jusqu'à obtenir un seul pixel d'épaisseur.

Ensuite, les valeurs maximales de $\mathcal{M}$ restantes sont seuillées. Trouver le bon seuil τ est difficile à déterminer, et bien souvent n'existe pas. Aussi un algorithme de double-seuillage est-il utilisé pour améliorer le résultat. Un premier seuil τ_1, assez bas, permet de conserver tous les contours, avec cependant des pixels parasites. Il fournit l'image $\mathcal{T}_1$. Un second seuil

τ_2, approximativement double du premier, permet d'éliminer les "faux" contours, mais élimine également de "vrais" contours. Il fournit l'image $\mathcal{T}_2$.

Le principe est de prendre l'image très seuillée, $\mathcal{T}_2$, puis lorsqu'un contour s'arrête, on regarde dans $\mathcal{T}_1$ si le contour existe et continue. Si c'est le cas, on valide le contour de $\mathcal{T}_1$. Et ainsi de suite.

Il en résulte en général une image finale de bonne qualité, qui permet d'extraire proprement les contours d'une scène dans une image.

II.2.7 Morphologie mathématique

C'est à partir de 1964 que G. Matheron invente la *Morphologie Mathématique* (MM) comme outil d'étude des champs aléatoires statiques. Plus tard, elle sera introduite dans le traitement d'image, notamment par J. Serra. Le terme de "morphologie" signifie littéralement "étude des formes", dont les bases mathématiques sont issues de la théorie ensembliste de Minkowski de 1903.

► Opérateurs de base

Il existe deux opérateurs basiques sur lesquels sont fondés les algorithmes de MM : l'*érosion*, ϵ, et la *dilatation*, δ. Ils utilisent tous deux un élément structurant, E, qui paramètre ces opérateurs morphologiques.

Définition 1 *L'***érosion** ϵ *et la* **dilatation** δ *d'une image binaire* $\mathcal{I}(u,v)$ *par un élément structurant* $E(x,y)$ *se définissent par :*

$$\epsilon_{E(x,y)}(\mathcal{I}) = \min_{E} \mathcal{I}(u+i, v+j) \tag{II.41}$$

$$\delta_{E(x,y)}(\mathcal{I}) = \max_{E} \mathcal{I}(u+i, v+j) \tag{II.42}$$

$\mathcal{I}_\epsilon = \epsilon_E(\mathcal{I})$ *et* $\mathcal{I}_\delta = \delta_E(\mathcal{I})$ *sont alors appelés respectivement* **érodé** *et* **dilaté** *de l'image* $\mathcal{I}$.

Une érosion aura tendance sur une image binaire à réduire la taille des gros objets (un objet est une forme connexe de pixels à 1), à éliminer de l'image les petit objets. Une dilatation aura quant à elle tendance à grossir les objets de l'image et à supprimer les "trous" à l'intérieur des objets (figure II.28).

Sur une image en niveau de gris, une érosion (*resp.* dilatation) aura tendance à assombrir (*resp.* éclaircir) l'image en créant des paliers de niveau de gris minimaux (*resp.* maximaux) localement. Dans les deux cas, un effet "pointilliste" apparaît à cause de ces paliers.

► Filtres morphologiques

À partir de ces deux opérateurs de base, il est possible de définir des *filtres morphologiques* qui permettent de garder la taille initiale des objets.

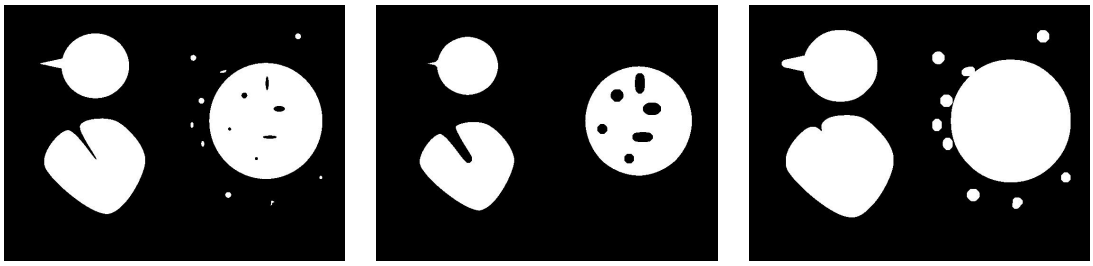


FIG. II.28 – À gauche, une image binaire avec des objets blancs ; au centre, l'érodé, à droite, le dilaté.

Définition 2 *L'***ouverture** ω *et la* **fermeture** ϕ *d'une image binaire* $\mathcal{I}$ *par un élément structurant* E *se définissent par :*

$$\omega_E(\mathcal{I}) = (\delta_E \circ \epsilon_E)(\mathcal{I}) \quad \text{(II.43)}$$
$$\phi_E(\mathcal{I}) = (\epsilon_E \circ \delta_E)(\mathcal{I}) \quad \text{(II.44)}$$

$\mathcal{I}_\omega = \omega_E(\mathcal{I})$ *et* $\mathcal{I}_\phi = \phi_E(\mathcal{I})$ *sont alors appelés respectivement* **ouvert** *et* **fermé** *de l'image* $\mathcal{I}$.

Une ouverture d'une image par un élément structurant est donc une érosion suivie d'une dilatation, alors qu'une fermeture est une dilatation suivie d'une érosion. Dans une image binaire (figure II.29), une ouverture aura tendance à éliminer les petits objets, tout en gardant la taille des gros objets. Une fermeture aura, quant à elle, tendance à éliminer les petits trous dans les objets.

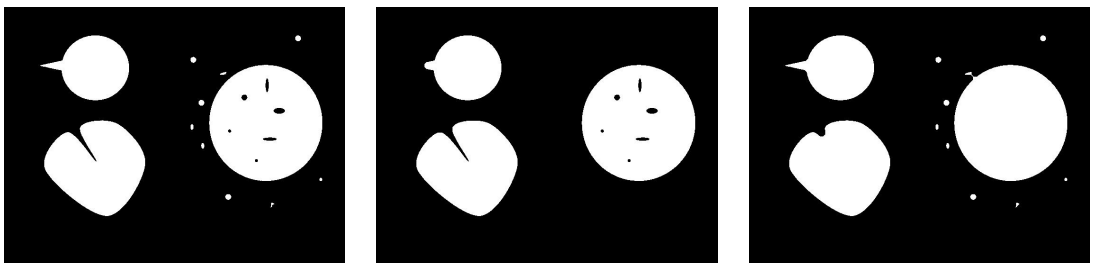


FIG. II.29 – À gauche, une image originale avec des objets blancs ; au centre, l'ouvert, à droite, le fermé.

Pour une image en niveaux de gris, une ouverture (*resp.* fermeture) a également tendance à assombrir (*resp.* éclaircir) l'image.

Ces filtres non linéaires peuvent servir à *nettoyer* l'image de petits parasites indésirables.

► Détection d'objets

Ces opérateurs permettent de détecter des formes particulières dans une image binaire. L'exemple de la figure II.30 est explicite. Soit trois objets dans l'image originale à gauche suivi de trois éléments structurants particuliers (non à l'échelle). Les trois images suivantes montrent l'érosion de la première image à l'aide respectivement des éléments structurants $n^{\circ}1$, 2 et 3.

FIG. II.30 – Formes filtrées à l'aide d'éléments structurants particuliers.

Une interprétation de l'érosion est qu'elle permet de détecter les formes qui peuvent enfermer entièrement l'élément structurant. Par exemple le rectangle vertical peut rentrer entièrement dans la première forme à gauche de l'image originale, alors qu'il ne s'intègre pas dans les deux autres formes. Les pixels blancs restant dans l'érodé sont les centres des éléments structurants qui s'intègrent dans les formes.

Il est ainsi possible de détecter des formes particulières en fonction de l'élément structurant choisi, et de qualifier l'information de présence de tel ou tel objet.

II.3 Algorithmes Utilisés dans nos Applications

Dans cette section nous détaillons les algorithmes d'analyse d'image que nous avons utilisés pour caractériser, à l'aide d'attributs, les images que le robot, dans notre application, prend de l'environnement.

L'analyse de chaque image s'est faite selon deux approches : l'approche *structurelle*, qui se base uniquement sur l'intensité de l'image, et l'approche *colorimétrique*, qui se base sur la couleur.

Le principe général de base est le suivant. Sur chaque image des *primitives* sont extraites. Ces primitives sont de l'information à l'état brut. À partir de ces primitives, on vérifie la présence ou non d'attributs dans l'image. Ces attributs ont été définis au préalable par l'utilisateur. Les attributs sont en général plus stables et de plus haut niveau que les primitives ; cependant, des primitives peuvent aussi être des attributs.

Pour l'analyse structurelle, les primitives seront des segments. Pour l'analyse colorimétrique, les primitives seront des pixels particuliers dans le domaine de représentation Intensité-teinte-saturation.

Après avoir essayé plusieurs librairies d'analyse d'image, nous avons jeté notre dévolu sur la librairie *OpenCV*[3] d'INTEL, de loin la plus complète et la plus ergonomique en C++. Plusieurs essais auparavant ont été menés sous MATLAB puis sous l'environnement *calife*, développé au LAAS, qui propose des bibliothèques d'analyse d'images et une interface de commande pour les projets menés au LAAS.

II.3.1 Analyse structurelle

L'analyse structurelle est basée sur la recherche de segments dans une image. L'étude détaillée de ces segments permettra de définir les attributs structurels de l'image. Un exemple est donné figure II.31, dans laquelle les segments de l'image issus des contours sont extraits.

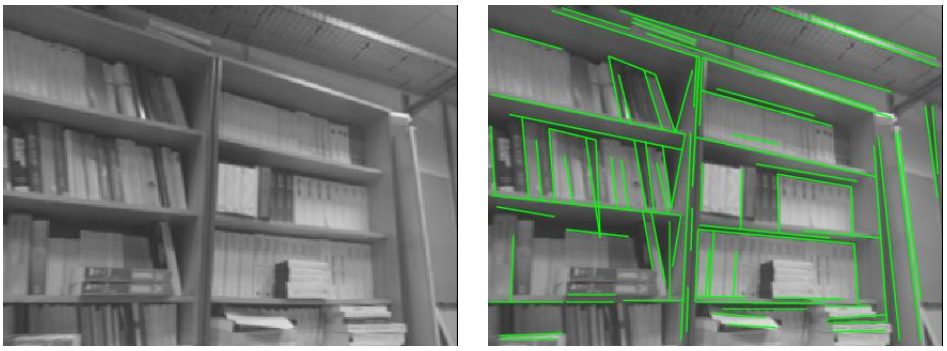


FIG. II.31 – Une bibliothèque au LAAS et les segments de contours extraits.

► Algorithme d'extraction de segments

Une fois l'image capturée par le robot, elle est transformée en image Noir&Blanc de façon classique (Eq. II.1, page 28), puis on lui applique un filtre de Canny afin de dégager les contours. Une fois les contours extraits, on utilise une approximation polygonale de type *Douglas-Peucker*, dont l'algorithme figure II.35 est détaillé ci-dessous. Enfin, les segments de l'image sont les segments des polygones d'approximation. L'algorithme synthétique d'extraction de segments est exposé figure II.32, et une illustration est proposée dans les figures II.33 et II.34.

► Algorithme de Douglas-Peucker

L'algorithme de Douglas-Peucker[4] [Douglas 73] est utilisé ici pour faire une approximation polygonale des contours. L'idée est la suivante : on recherche sur la courbe à approximer autant de nœuds nécessaires afin de garder une distance minimale entre la ligne de segments articulée par chaque nœud et la courbe en question. L'algorithme est présenté figure II.35 et

[3] la librairie est téléchargeable sous *sourceforge* : http ://sourceforge.net/projects/opencvlibrary/
[4] publié précédemment par Ramer un an plus tôt, en 1972

1. Transformer l'image couleur en image Noir&Blanc
2. Appliquer le filtre de Canny
3. Extraire les contours dans l'image
4. Faire une approximation polygonale (type *Douglas-Peucker*)
5. Extraire les segments des polygones résultants.

FIG. II.32 – Algorithme d'extraction de segments dans une image.

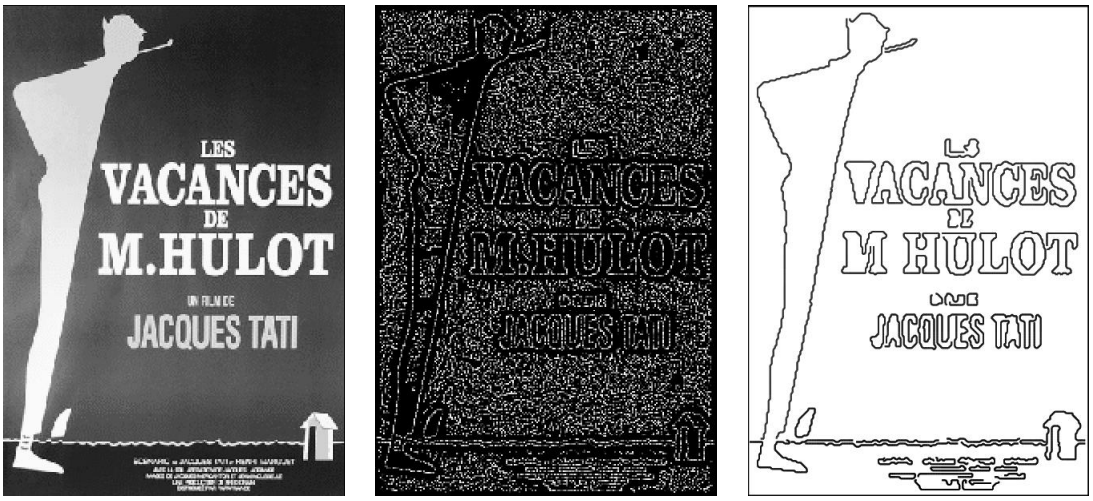


FIG. II.33 – Extraction de contours.

une illustration figure II.36.

Cet algorithme est paramétré par la distance maximale δ entre la courbe et son approximation. Plus cette distance est petite, plus l'approximation est fine, mais plus le nombre de nœuds augmente, avec évidemment le temps d'exécution (quadratique avec le nombre n de nœuds). L'algorithme peut également être paramétré par un nombre de nœuds maximal. Une simplification basée sur des zones de tolérance (*"hull"*) est proposée par [Hershberger 92] et permet de réduire le temps d'exécution en $n \log_2 n$.

Enfin, on peut noter également l'algorithme performant de Wall et Danielsson [Wall 84] pour l'approximation polygonale.

► Attributs associés aux segments

Notre système est entièrement paramétrable en terme de longueur et d'orientation du segment : On peut par exemple diviser les segments...

– en petits, moyens et grands segments, avec des bornes de longueur (en pixels) associées,

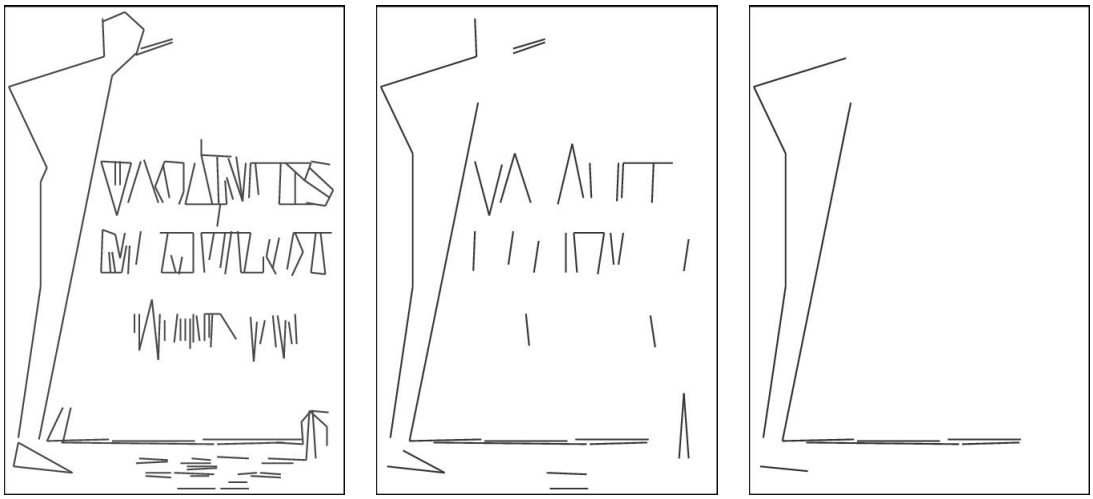


FIG. II.34 – Petits, moyens et grands segments de l'image.

Soit κ une courbe à approximer par une courbe polygonale λ, et une distance δ
1. Tracer une ligne entre les deux extrémités de κ et mettre les nœuds initiaux
2. Pour tout segment de λ entre deux nœuds consécutifs
  2.1. Chercher la distance maximale entre le segment et la courbe
  2.2. Si cette distance est supérieure à δ
    2.2.1. Insérer un nouveau nœud en lieu de distance maximale

---

FIG. II.35 – Algorithme de Douglas-Peucker.

– en N_θ orientations possibles,
– et, pour chaque catégorie, en petit, moyen et grand nombre, avec des quantités associées,
– *etc.*

On peut ainsi quantifier le nombre total de catégories de segments :

$$N_{seg} = (3 + N_\Theta) \times 3 \text{ attributs de segments} \tag{II.45}$$

II.3.2 Analyse colorimétrique

L'analyse colorimétrique étudie quant à elle les pixels de différentes *couleurs* du spectre visible. Le terme de "couleur" est pris ici, à l'exception du noir et du blanc, au sens physique du terme (longueur d'onde), et non au sens de la perception.

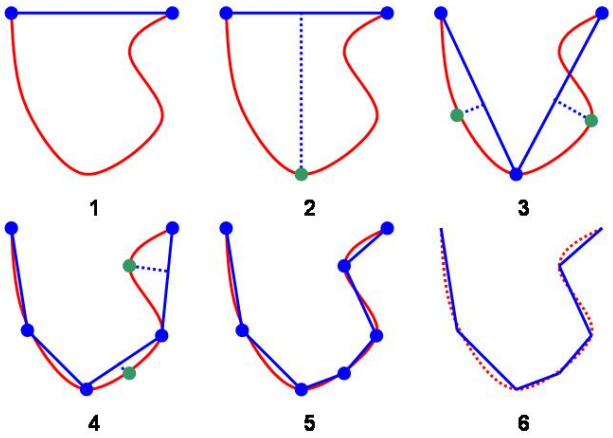


FIG. II.36 – Approximation de Douglas-Peucker. Les nœuds verts représentent les nœuds nouvellement créés.

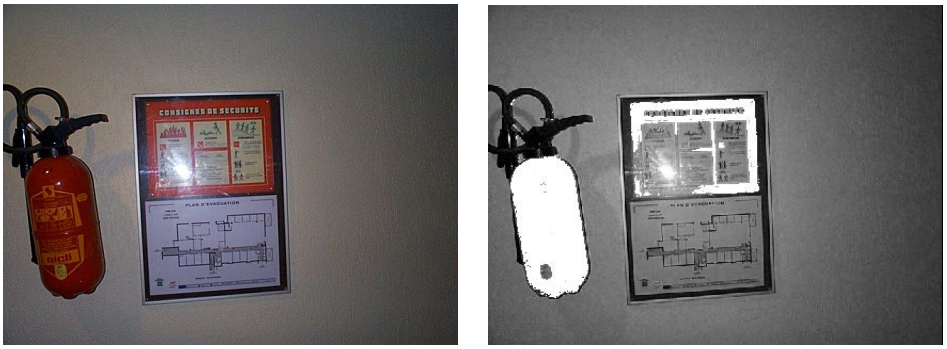


FIG. II.37 – Une image issue du LAAS dans laquelle seul le rouge ”pur” est extrait.

La représentation utilisée pour les couleurs est la représentation I-T-S. En effet, le *spectre colorimétrique*, que l'on peut approximer par l'histogramme de la teinte, permet d'avoir une représentation étalée des couleurs de l'image. L'histogramme est découpé en bandes de largeur (réglable) δF, à partir desquelles on définit un *taux de recouvrement* (réglable également) τ_r proportionnel à la largeur de chaque bande.

Une **bande de couleur** correspond donc à une bande de l'histogramme centrée autour d'une valeur F_0 et de largeur $\Delta F = \delta F + 2 \times (\tau_r * \delta F)$, le recouvrement dépassant de chaque côté de la bande centrale considérée. Ce taux de recouvrement permet de garder une certaine continuité entre les bandes “spectrales”.

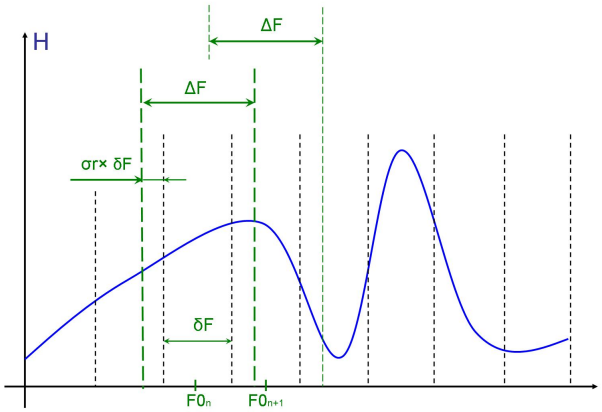


FIG. II.38 – Découpage de l'histogramme des teintes en bandes de couleurs.

Les pixels appartenant à chaque bande de couleur sont *validés* (et binarisés à 1) par les nécessités suivantes :
- une saturation minimale et maximale,
- et une intensité minimale et maximale.

L'image résultante est donc une image binaire correspondant aux pixels validés dans chaque bande de couleur. Elles sont ensuite "nettoyées" à l'aide d'un filtre morphologique (ouverture).

► Attributs associés aux couleurs

Pour chaque bande de couleur, six caractéristiques sont extraites, à savoir la présence...
- de peu de pixels,
- de beaucoup de pixels,
- d'énormément de pixels,
- d'un petit objet,
- d'un moyen objet,
- et/ou d'un grand objet.

dans l'image.

Un objet est un ensemble de pixels validés connexes. Ils sont détectés à partir d'une érosion avec comme élément structurant un petit, moyen et grand objet. Un exemple est donné figure II.39, dans laquelle le rouge "pur" est sélectionné et l'image nettoyée (au centre) à l'aide d'une ouverture suivie d'une fermeture. Enfin, une érosion permet de qualifier la présence d'un objet rouge dans l'image, puisque l'image érodée (à droite) est non vide.

FIG. II.39 – Une image issue du LAAS dans laquelle seul le rouge ”pur” est extrait et un objet est détecté grâce à une érosion.

Il y a donc au total :

$$N_{coul} = (N_{bandes} + 2) \times 6 \text{ attributs de couleur} \tag{II.46}$$

où le ’2’ représente les deux *“couleurs”* noir et blanc.

II.3.3 Structuration générale de l’information

Transformer des primitives en attributs demande souvent une discrétisation d’informations continues. Cette discrétisation est propre à l’application et à la taille des images. De plus, l’articulation entre attributs mesurés se fait par inclusion des grandeurs. Par exemple, un ”grand” objet renferme un ”moyen” objet, et lui-même un ”petit” objet. De la même façon, quand l’image renferme ”beaucoup” d’éléments quelconques, (comme des pixels bleus par exemple), alors elle renferme ”peu” de ces mêmes éléments. L’attribut correspondant est validé. Il n’est pas utile de généraliser dès maintenant sur la discrétisation, mais nous aborderons ce problème lors des expérimentations elles-mêmes.

Notons cependant qu’une discrétisation adaptative ne peut être envisagée, car l’apprentissage sera incrémental. Donc dès le début les seuils de discrétisation doivent être définis.

Une fois extraites toutes les informations contenues dans une image, la structuration générale des données se fait selon la figure II.40. À chaque image est associée la présence ou non de tel ou tel attribut : s’il y a beaucoup de segments orientés, s’il y a la présence d’un objet de couleur rouge, *etc.*

La liste des attributs a évolué au cours des manipulations. Elle sera précisée au moment voulu, pour chaque expérience. Il est à noter que notre système est ouvert, et accepte tout type d’algorithme d’analyse d’image, avec pour seule contrainte le temps d’exécution.

C’est pour cette contrainte que les attributs de texture n’ont pas été introduits dans notre application. Les algorithmes de segmentation texturelle étant encore lourds et peu efficace en environnement intérieur (où il y a en général “beaucoup” de classes), nous avons préféré dans un premier temps nous attacher à des primitives relativement simples et rapidement extractibles

		Attribut 1	Attribut 2	Attribut 3	Attribut 4		Attribut f		Attribut Nf
Site 1	Image 1.1	X	X						
	Image 1.2		X	X			X		
	Image 1.3								
			X						
	Image 1.n1								
Site 2	Image 2.1			X					
	Image 2.2			X	X	X		X	X
	Image 2.3					X			
	Image 2.n2								
...	...		X					X	
Site N	Image N.1								X
	Image N.2		X					X	
	Image N.3								
							X		
	Image N.nN								

FIG. II.40 – Structuration générale de l'information. Un site est un lieu particulier de l'environnement (une pièce par exemple), dans lequel un grand nombre d'images ont été prises par le robot. À chaque site est donc associé les images correspondantes, et à chaque image est associée la présence ou non de chaque attribut précédemment décrit.

(couleurs, contours*etc.*).

La question essentielle qui se pose alors est la suivante : comment gérer cette information ? Comment la synthétiser ? Comment extraire de l'information pertinente ?

C'est l'objectif de ce manuscrit, et en particulier des chapitres 4 et 5. Avant cela, nous nous attardons un peu sur les aspects "robotiques" de la navigation.

Chapitre III

Robotique et Navigation

Naviguer et se localiser dans un environnement est une tâche extrêmement difficile pour un robot mobile. Même si les capteurs proprioceptifs et extéroceptifs disponibles sur le marché sont de plus en plus performants et précis, l'intégration de toute l'information qu'ils délivrent dans un but de localisation reste incertaine et peu maîtrisée.

Ce chapitre rassemble, après une très brève présentation des robots utilisés au cours de cette thèse, les notions relatives à la localisation en robotique mobile autonome et propose sur le sujet un état de l'art non exhaustif. On y expose les notions de navigation, de modélisation de l'environnement (notamment topologique, pour notre application), de localisation, pour terminer vers une entité centrale dans notre application : les amers visuels.

III.1 Les Robots

III.1.1 Pekee à SUPAERO

Pekee (figure III.1) est un petit robot de la société Wany Robotics[1] qui permet de développer rapidement des algorithmes propres à la robotique mobile terrestre autonome.

Il dispose d'un ensemble de capteurs proprioceptifs et extéroceptifs (figure III.2) :

[1] http ://www.wanyrobotics.com

FIG. III.1 – Pekee.

- une ceinture de 15 capteurs infra-rouge,
- des odomètres sur chaque roue,
- deux gyroscopes (tangage et lacet),
- deux capteurs de température (interne et externe),
- un capteur de lumière,
- et un capteur de chocs.

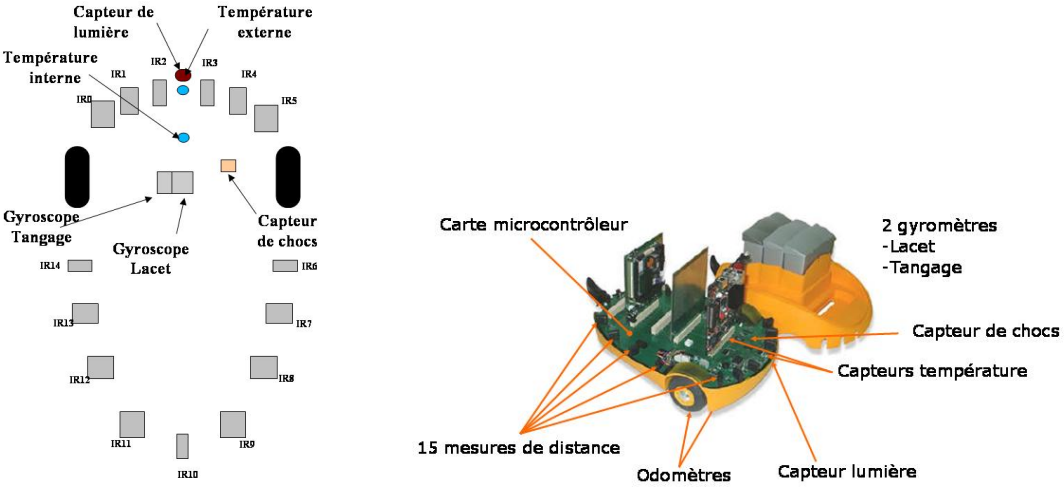


FIG. III.2 – Les capteurs de Pekee.

La communication avec un ordinateur se fait par l'entremise d'un réseau intranet. Sa commande est de "haut niveau", à savoir qu'on le commande en vitesse linéaire et en rotation. Il est de plus équipé d'une simple caméra couleur 140×180 très classique.

III.1.2 Diligent au LAAS

Diligent (figure III.3, à gauche) est un robot bien plus évolué et doté de capteurs beaucoup plus performants. Il est équipé de deux caméras montée sur un banc de type “pan-tilt”, de deux ceinture de capteurs ultra-sons, et de quatre roues articulées indépendamment lui permettant une complète “holonomie”.

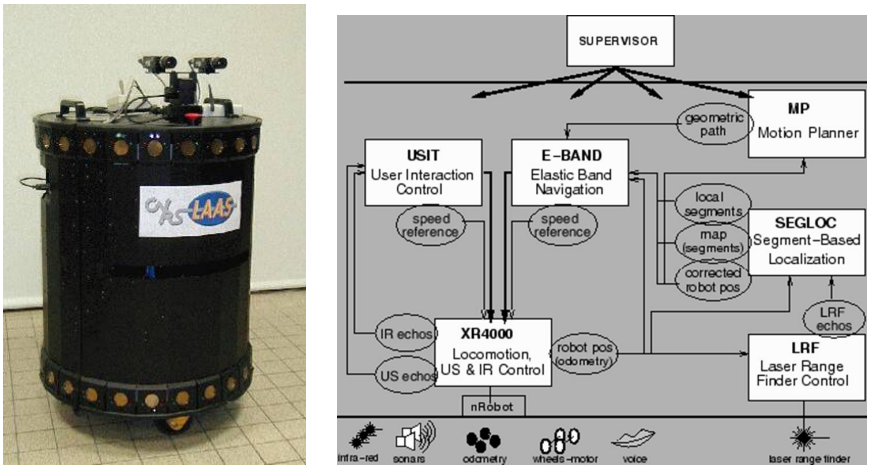


FIG. III.3 – Diligent et son architecture logicielle.

De plus, il est équipé d’une architecture logicielle modulaire (figure III.3, à droite), lui permettant de s’intégrer totalement dans l’ensemble des robots du LAAS-CNRS.

III.2 Navigation

La **navigation** est un terme général qui se définissait initialement comme l’*“ensemble des sciences et techniques du déplacement des navires”*. Cette définition à ensuite été étendue à d’autres types de véhicules comme les véhicules aériens par exemple. On remarque que ce terme, en revanche, n’est pas utilisé pour les véhicules terrestres de type automobile ou train. Une explication viendrait du fait que ces véhicules se déplacent sur les voies déjà tracées et balisées : une voiture sort rarement de la route, et un train encore moins de ses rails.

Le monde animal a beaucoup inspiré les recherches dans le domaine de la navigation ([Trullier 97, Collet 96, Veelaert 96, Dyer 96, Moller 97, Moller 98a, Moller 98b, Balakrishnan 97, Bianco 00, Nehmzow 95]. . .). Appliquée à la robotique mobile, cette définition peut se résumer à l’élaboration des techniques permettant de relier un point A à un point B. D’autres proposent des déclinaisons plus étendues [Trullier 97] : *“Navigation is the process of determining and maintaining a course or trajectory from one place to another. Processes for estimating one’s position with respect to the known world are fundamental to it. The known world is composed*

of the surfaces whose locations relative to one another are represented on a map". D'autres encore transforment cette définition en trois interrogations [Levitt 90] :

1. Where am I ?
2. Where are other places relative to me ?
3. How do I get to another place from there ?

On voit ainsi transparaître plusieurs notions relatives à la navigation. Les deux premières interrogations de Levitt et Lawton reflètent un problème de **localisation**. La localisation suppose donc des *éléments de repère* (que l'on appellera *"amers"* plus loin) qu'il faut reconnaître dans un premier temps, et éventuellement savoir positionner par rapport à soi dans un deuxième temps. Ils sont donc entièrement dépendants des capacités de perception du robot. La troisième interrogation reflète un problème de *planification de trajectoire*, notion que nous ne développerons pas ici.

Il est possible de distinguer les compétences de navigation "locale" (*local navigation skills*, où toute la scène "utile" est dans le champ de vision de l'animal) et les compétences de navigation "globale" (*way-finding skills*, sur une échelle plus grande) avant que [Trullier 97] (repris par [Filliat 01]) propose cinq grands types de navigation :

– l'*approche*, c'est-à-dire la capacité de se diriger vers un objet toujours visible ;
– le *guidage*, c'est-à-dire la capacité de se diriger vers un objet non toujours visible ;
– l'*action associée à un lieu*, c'est-à-dire la capacité d'atteindre un but depuis des positions pour lesquelles ce but ou des amers qui caractérisent le(s) lieu(x) sont visibles (à chaque lieu est associé un enchaînement d'actions) ;
– la navigation *topologique* ;
– la navigation *métrique*.

Les deux dernières catégories autorisent une navigation globale et permettent de rejoindre un but arbitraire au sein d'un environnement. Ils s'appuient pour cela sur un modèle interne du monde (carte) qui supporte une planification.

Ces éléments font donc référence implicitement à la notion de *carte*. Les questions qui se posent alors sont les suivantes : qu'est-ce qu'une carte ? Quelles sont leurs représentations ? Ces représentations sont-elles absolues ou relatives ? Comment les cartes sont-elles construites ?

III.3 Les Modèles de l'Environnement

Deux approches co-existent de manière complémentaire pour modéliser un environnement : l'approche métrique et l'approche topologique, chacune ayant leurs avantages et leurs inconvénients respectifs [Thrun 96, Thrun 98a].

III.3.1 Approches métriques

L'approche métrique consiste à construire -ou à modéliser- l'environnement de façon métrique, c'est-à-dire de définir des distances entre les éléments de l'environnement (entre le robot et un objet ou un mur, ou entre différents murs, *etc.*). Ces distances, ainsi que leur précision, dépendent directement des capteurs de mesure embarqués sur le robot.

Il existe plusieurs manières de construire une carte métrique : soit à l'aide d'une *grille d'occupation* (on initialise une grille vierge et l'on noircit les cases susceptibles d'être occupées), soit à l'aide de primitives simples, principalement des segments (on repère les segments type murs...), soit à l'aide de modélisations plus complexes.

La première solution est sans doute la plus populaire [Zwynsvoorde 00] car elle est facile à mettre en œuvre et est bien adaptée à l'utilisation de capteurs à ultrasons, simples et peu coûteux, comparativement à la deuxième solution qui nécessite un capteur de précision type laser.

Les récentes techniques en matière de cartographie métrique (principalement le SLAM (*Simultaneous Localization and Mapping* [Smith 87, Thrun 98b]) permettent de construire des cartes précises et donc de planifier des trajectoires dans un environnement assez important. Cependant, cette technique requiert des traitements de données capteur en chaque instant pour se localiser, et la complexité algorithmique gandit avec le nombre de capteurs au fur et à mesure des informations perçues [Hernandez 84].

Les cartes métriques locale permettent cependant au robot de se déplacer dans un environnement sans beaucoup d'incertitude. Elles permettent également de définir des trajectoires pour circuler et atteindre un objectif précis. Sans information métrique locale, le robot ne peut se déplacer.

Mais il est très difficile pour un robot de se déplacer seul, dans un environnement global, avec une telle connaissance de l'environnement. De plus, une carte métrique -même précise- à l'échelle d'un laboratoire par exemple serait inexploitable localement dans un bureau pour contourner un obstacle. Et une carte précise à l'échelle locale demanderait d'énormes ressources pour une carte à grande échelle. C'est pourquoi une autre modélisation a été proposée : la modélisation topologique.

III.3.2 Approches topologiques

Les modèles topologiques ou qualitatifs constituent une alternative attrayante [Zwynsvoorde 00] car ils sont plus compacts et permettent des descriptions plus intuitives et plus efficaces pour la planification de parcours, en intégrant une **sémantique** de l'environnement.

Le Laboratoire d'Informatique et de Robotique de SUPAERO peut se modéliser à l'aide de *nœuds* (cercles) et d'*arcs* (liaisons entre les nœuds) comme le montre la figure III.4.

La sémantique attachée à cette carte est la suivante :

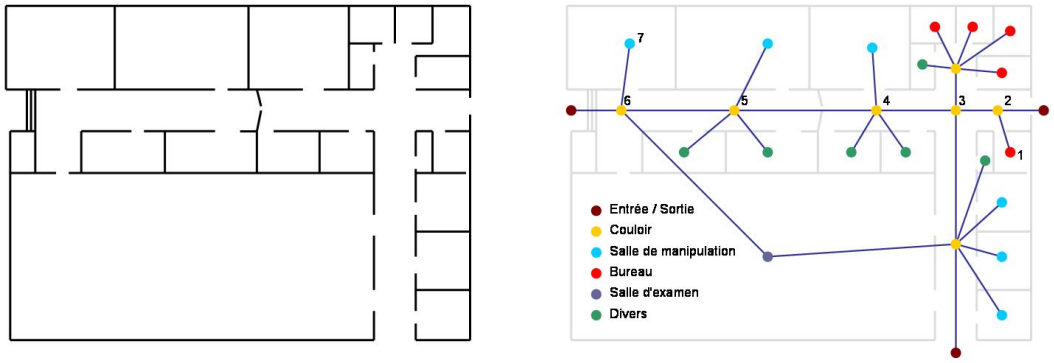


FIG. III.4 – Carte métrique (à gauche) et carte topologique (à droite) du LIA.

– les nœuds marrons représentent les entrées/sorties de la carte ;
– les nœuds jaunes représentent les zones de passage (couloir) ;
– les nœuds turquoises représentent les salles de manipulation ;
– les nœuds rouges représentent les bureaux ;
– le nœud gris représente la salle d'examen ;
– et les nœuds verts-gris représentent des salles diverses (placards...)

Il est possible également d'attacher une sémantique sur les arcs.

Il est ainsi très "facile" de naviguer dans le laboratoire, notamment de connaître son chemin pour aller d'un endroit à un autre. Par exemple, pour aller du bureau à la salle de manipulation, plusieurs chemins sont possibles mais le plus court est la chaîne 1-2-3-4-5-6-7 de la carte de la figure III.4. Cependant toute information métrique est perdue.

Mais il est possible d'enrichir cette représentation en associant à chaque arc une distance, un coût, une probabilité de passage, un histogramme [Ulrich 00], etc., et à chaque nœud une *étiquette* plus ou moins fournie : type de lieu, superficie, utilité...

Chacune de ces étiquettes intégrera, après apprentissage, un ou des amers caractérisant ce lieu. Ainsi le robot reconnaîtra l'endroit où il se trouve dans la carte topologique.

Il est important de remarquer qu'ici le graphe topologique, très intuitif, est calqué sur la géographie du lieu (un nœud = une pièce). Mais il est parfois utile de construire une carte d'amers (un nœud = un amer). Les représentations sont très variées.

On peut noter qu'il est possible de passer d'une représentation métrique à une représentation topologique succincte [Thrun 98c, Zwynsvoorde 00], mais l'inverse est impossible.

III.3.3 Approches mixtes

Ces deux représentations sont complémentaires. Une carte *idéale* pourrait être définie topologiquement à grande échelle et métriquement à petite échelle. Ainsi à chaque nœud du graphe topologique est associé, entre autres informations, une carte métrique locale [Thrun 98c, Thrun 98b]. Par exemple, une telle approche développée dans [Thrun 98c] permet de construire une carte métrique très précise à partir de corrections issues de la construction d'une carte topologique.

Il existe d'autres classifications de l'environnement, comme les environnements artificiels et naturels, ou bien (ce qui correspond en général à ce qui précède) structuré et non structuré. Cependant nous n'abordons pas cette catégorisation.

III.4 Localisation

III.4.1 L'inspiration biologique

Le besoin de se repérer dans l'espace chez l'animal a pour probable origine la survie de l'espèce, selon deux axes principaux : la recherche de la meilleure position possible par rapport à une source de nourriture donnée, et la migration bi-annuelle de certaines espèces vers des contrées plus accueillantes.

L'abeille est un très bon sujet de recherche pour le premier axe. Le biologiste autrichien Karl Von Frisch (Vienne, 1886 - Munich, 1982) fut le premier, dès 1915, à élucider le *langage* des abeilles pour la signalisation de sources de nourriture. Les butineuses qui reviennent à la ruche, après avoir délivré le butin sucré, *dansent* selon un code bien spécifique qui indique l'azimut et la distance de la source à ses collègues.

Il existe deux types de danses : la danse en "rond" qui indique une source à moins de 100 mètres, et la danse en "huit" ou "frétillante" au delà. Lors de cette danse, l'abeille exécute à chaque demi-cercle des segments orientés fournissant de l'information sur la distance (variation de vitesse de la danse), la fertilité (vivacité du frétillement) et la direction de la source : l'angle qu'elle fait par rapport à la verticale est identique à l'angle entre la source et l'azimut du soleil. On voit ainsi que le point de repère principal des abeilles est le soleil. Ce point de repère, bien que mobile, est stable dans le sens ou il est toujours visible par les abeilles, celles-ci étant sensibles aux rayons ultra-violets.

L'étude des oiseaux migrateurs[2] permet de mettre en évidence les aptitudes aux déplacements lointains. Là encore, la destination étant invisible (imperceptible) depuis le lieu de départ, il est nécessaire de savoir définir un cap et une durée de déplacement pour atteindre un objectif précis. Plusieurs modèles existent. Initialement, les travaux de G. Ising (1945) montrèrent l'importance du soleil comme repère visuel des animaux migrateurs. Kramer (1953) ensuite utilisa

[2]Bien que les oiseaux forment les voyages les plus longs, comme les sternes qui se déplacent entre l'Arctique et l'Antarctique, beaucoup d'autres espèces migrent comme les saumons (jusqu'à 3000 km !) ou les éléphants.

des projecteurs afin de simuler le soleil avec des étourneaux, et constata que ceux-ci faisaient varier, le long de la journée, l'angle entre leur déplacement et le projecteur fixe. Ceci suppose donc non seulement une mesure du temps au cours du déplacement, mais également la capacité de l'animal à retenir des informations relatives à ses positions initiale et courante. D'autres théories existent sur l'orientation des oiseaux nocturnes : la position des étoiles (Sauer, 1957), ou le champ magnétique terrestre (Wiltschko, 1976). Walcott, Gould et Kirschvink montrèrent l'existence de masses magnétiques microscopiques dans le cerveau des pigeons voyageurs.

III.4.2 Les différentes approches

Le problème de la localisation est fondamental en robotique mobile. On distingue ici trois approches différentes de ce vaste thème :
- localisation absolue et localisation relative ;
- suivi de position et localisation globale ;
- modèles markoviens de la localisation.

III.4.3 Localisation absolue et localisation relative

La localisation absolue nécessite des informations externes (type GPS, ou caméra dans une enceinte), appelées informations *extéroceptives*, pour lesquelles une modélisation préalable du monde est indispensable, ou internes (odomètres, gyromètres, accéléromètres, ...), appelées informations *proprioceptives*, dont l'erreur explose avec l'avancement du robot.

L'autre solution -en général la plus intéressante- reste la localisation relative par rapport à des amers précédemment définis. Différentes techniques de localisation (reconnaissance pour une approche topologique et positionnement pour une approche métrique) sont mises en œuvre dans les laboratoires : triangulation [Madsen 98], correspondances 3D, estimations à l'aide de filtres de Kalman, etc.

La localisation absolue, seule, n'est que très peu utilisée en robotique mobile. Elle n'a son intérêt qu'avec une connaissance globale de la position des amers, dont se servent les robots pour se repérer.

III.4.4 Suivi de position et localisation globale

Ainsi il existe une autre approche de la localisation [Filliat 01], fréquemment répandue, où l'on distingue les deux catégories suivantes : le *suivi de position*, c'est-à-dire la capacité de mettre à jour une position existante, et la *localisation globale*, qui consiste à retrouver une position sans qu'aucune estimation initiale ne soit fournie.

Dans tous les cas, la qualité des amers est importante, et certaines de leurs propriétés indispensables : en représentation topologique, l'amer doit être discriminant par rapport à son environnement et différent des autres amers (sauf si l'on considère une combinaison d'amers), et en représentation métrique "la géométrie" de l'amer ne doit permettre aucune ambiguïté possible (*perceptual aliasing*), permettant ainsi au robot de se positionner de façon sûre et

définitive.

La différence essentielle entre la localisation globale (absolue) avec amers (dans le cadre topologique par exemple) et le suivi de position vient de l'espace de définition des amers. En localisation globale, les amers doivent être globaux, c'est-à-dire discriminants les uns par rapport aux autres dans tout l'espace considéré. En revanche, en suivi de position, les amers doivent avoir une définition locale, et permettre de discriminer un nœud de ses voisins. Ainsi deux mêmes amers peuvent co-exister à condition d'être loin l'un de l'autre, et qu'aucune ambiguïté ne soit possible (on peut comparer ce phénomène -deux positions à partir d'une même perception- au problème du *perceptual aliasing*). La position est ainsi estimée de proche en proche.

Le gros inconvénient de l'approche locale (c'est-à-dire avec suivi de position) est qu'à l'initialisation, puisque plusieurs amers peuvent coexister, il n'est pas toujours possible de déterminer sa position de façon certaine. Il faudra alors une "chaîne" d'amers consécutifs pour -éventuellement- se positionner, ce qui suppose une perte de temps lors du déplacement du robot. En revanche, un grand avantage est la richesse des amers possibles, puisqu'ils ne sont contraints que par des spécificités géographiques.

Ceci nous mène naturellement vers des approches markoviennes de la localisation.

III.4.5 Modèles markoviens de la localisation

► Généralités

On définit mathématiquement une **chaîne de Markov** comme étant une suite stochastique vérifiant la propriété de Markov, soit, pour tout t,

$$\pi(q_t = s_i / q_{t-1} = s_j, q_{t-2} = s_k, \dots) = \pi(q_t = s_i / q_{t-1} = s_j) \tag{III.1}$$

avec
- $s_i, s_j, \dots$ les différents états possibles du système, et
- q_t l'état du système à l'instant t.

Ainsi, la probabilité d'être dans un état, conditionnée aux états antérieurs, ne dépend que de l'état précédent, et non plus de tous ses n états antérieurs.

► Localisation markovienne

Les modèles markoviens reposent donc sur des probabilités de transitions entre différents états d'un système, ou entre différents nœuds du graphe associé.

Supposons une carte associée à un graphe topologique (figure III.5).

Cette probabilité dépendra dans notre application de la présence ou non d'amers pour chaque nœud du graphe. La probabilité de se rendre dans un site voisin lorsque le robot perçoit un amer de ce site est proche de 1 (hors bruits de capteurs), et proche de 0 sinon. En pratique, ces probabilités dépendent de plusieurs facteurs : il est alors possible de délivrer non pas une

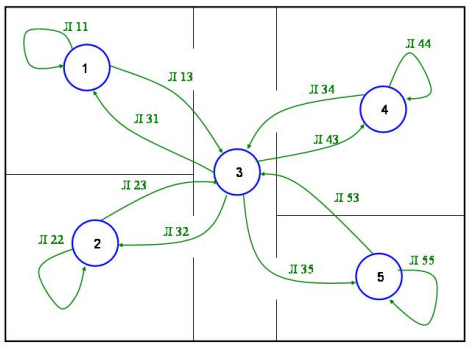


FIG. III.5 – Carte topologique et probabilités de transition entre salles.

localisation précise, mais une distribution de probabilités sur la localisation.

On retrouve cette approche markovienne aussi bien sur des modélisations métriques (grilles d'occupation) [Fox 98, Thrun 99] que topologiques [Shatkay 97, Theocharous 01].

III.5 Exploration et Cartographie

L'exploration d'un lieu se fait donc au fur et à mesure que le robot se déplace dans ce lieu. Il a plusieurs façons de se déplacer, du plus bas niveau (type déplacement au hasard et réflexes) au plus haut niveau (recherche d'indices -nos amers-, détection des sorties possibles, etc.) en passant par des niveaux intermédiaires (suivi de murs...) ce qui lui permet de construire son environnement (grilles d'occupation, segments, carte topologique...).

La cartographie est donc la construction d'une représentation interne de l'environnement. Cette construction interne se base sur le positionnement d'éléments distinctifs, comme les amers, sur une carte. La détermination de la position des ces amers sur une carte est le problème clé de la cartographie [Fox 99].

Le principal enjeu est alors de trouver ces amers stables et de les reconnaître ensuite, lorsque le robot repasse au même endroit.

La plupart des approches métriques se basent sur les points d'intérêt (comme dans [Jung 03], où les auteurs utilisent des groupes de points d'intérêt pour valider la corrélation). Cependant certains travaux se basent sur de la fusion de capteurs [Majumder 00], ou l'objectif est de construire une carte sous-marine à partir d'un véhicule sous-marin. Au niveau topologique, dans les travaux de [Ranganathan 01] par exemple, un nœud est créé lorsque, localement, le *meilleur* (meilleur rapport saillance/stabilité) amer est sélectionné. Cependant, la construction d'un nouveau nœud n'est pas obligatoirement corrélée avec un nouvel amer, mais peut

dépendre d'informations externes comme un pas de porte entre deux pièces par exemple.

III.6 Les Amers Visuels en Robotique

III.6.1 Définition étendue d'un amer

Comme défini dans le dictionnaire Hachette, un **amer**, terme maritime, est *tout point des côtes très visible (clocher, balise, etc.), porté sur une carte, servant de repère pour la navigation*. Mais dans la langue de Shakespeare (*"landmark"*), ce terme revêt un sens plus général, comme un bâtiment important, un lieu à visiter [Hayet 02a] voire même une date ou un objet important sur le plan historique par exemple (voir figure ci-contre : sur la banderolle est inscrit *"WWII submarine - National Historic Landmark"*). Le terme de *repère* (au sens du point de repère) comme traduction de "landmark" serait ainsi bien plus appropriée. Dans le monde de la robotique mobile, la définition la plus proche serait donc celle du *point de repère* au sens strict, que le robot n'a pas spécialement besoin de visiter, mais surtout de reconnaître. Ainsi trouve-t-on, dans la littérature consacrée, différentes définitions d'amers telles que "des signes distinctifs d'une image qui peuvent être aisément reconnus dans une seconde image obtenue d'un point de vue différent" [Knapek 00], ou plus simplement "un objet visuel identifiable dans un environnement" [Bianco 99]. Habituellement les amers ne sont pas introduits par une définition formelle mais plutôt à travers des propriétés spécifiques comme "facilement distinguable" (saillant) [Dodds 97] ou "localement unique". Dans un sens plus concret, un amer peut être un objet [Dudek 00], une couleur [Ulrich 00], des points d'intérêt [Schmid 97], *etc.*

L'utilisation d'amers visuels est nécessaire dans la reconnaissance d'images ou de lieux. Mais *qu'est-ce qu'un amer ?* Un amer est une entité de l'image qui se définit par tout ou partie des propriétés suivantes [Vandapel 00] :

- **discrimination** : afin de différencier facilement l'amer et son environnement ;
- **unicité** : afin qu'un amer ne soit pas confondu avec un autre ;
- **invariance** : l'amer doit être indépendant le plus possible des conditions externes (point de vue, conditions lumineuses...) ;
- **stabilité** : l'amer lui-même doit être le plus stable possible ;
- **précision** : notamment pour la triangulation, l'amer doit offrir des points précis ;
- et enfin "**interprétabilité**" : l'amer doit être facilement reconnaissable en lui-même.

Toutes ces propriétés ne sont pas nécessaires : ainsi, certains se limitent aux deux premières [Knapek 00] *'salient and distinctive'*. Par ailleurs, dans certains cas, l'unicité n'est pas indispensable si l'on considère non plus des amers en eux-mêmes mais des ensembles d'amers.

III.6.2 Différents types d'amers

Plusieurs classifications d'amers existent, notamment celle qui différencie les amers naturels des amers artificiels, mais nous introduisons ici une autre classification liée aux capacités de perception et de reconnaissance de la machine. Nous séparons les amers en trois catégories :

– les amers entièrement prédéfinis : une base d'images représentant des objets est introduite dans le robot ;
– les amers partiellement prédéfinis : ces amers potentiels ont une structure commune ;
– les amers non prédéfinis : aucune hypothèse n'est formulée *a priori* pour les amers potentiels.

Notre démarche s'inscrit dans cette dernière catégorie : le robot doit pouvoir choisir dynamiquement les amers les plus pertinents de façon autonome.

► Les Amers Définis

Ces amers correspondent en général à des objets courants (figure III.6). Un robot étant à l'heure actuelle incapable de lister les objets d'une scène, ces derniers ont été préalablement enregistrés dans une banque d'images [Jedynak 96, Dudek 00] que le robot *"n'a plus qu'à"* reconnaître.

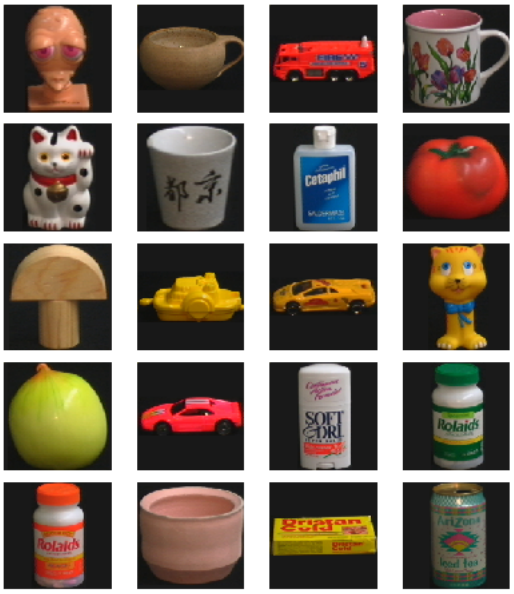


FIG. III.6 – Banque d'images servant d'amers dans un environnement.

Plusieurs méthodes ont été utilisées pour caractériser ces objets : les points d'intérêt [Schmid 97],

la couleur [Dodds 97, Ulrich 00, Hall 00], auxquels s'ajoutent des outils plus spécifiques : D. Lowe, qui propose le *SIFT : Scale Invariant feature Transform* [Lowe 99], le *PCA-based Methods* de G. Dudek et D. Jugessur [Dudek 00] (projection sur une base de vecteurs propres issus de la matrice de covariance [Cocquerez 95]) , tous deux basés sur des invariants locaux, ou l'utilisation d'arbres de classification par B. Jedynak et F. Fleuret [Jedynak 96], qui 'enregistrent' plusieurs vues d'un même objet, pour ne citer qu'eux.

► Les amers pré-définis

Les amers pré-définis ont tous, dès le départ, une particularité commune. Par exemple, au LAAS [Hayet 02b, Ranganathan 01, Hayet 00] et [Ayala 00] , des formes quadrangulaires planes (posters) servent au robot à présélectionner ses éventuels amers. Ensuite soit il les caractérise s'il ne les connaît pas, soit se repère s'il les connaît déjà. Il peut ainsi construire une carte topologique et se repérer dans son environnement.

Dans l'exemple suivant [Hayet 02b], les amers sont tous des formes quadrangulaires planes, type poster. Une fois repérés, ils sont analysés et caractérisés à l'aide de points d'intérêt (figure III.7).

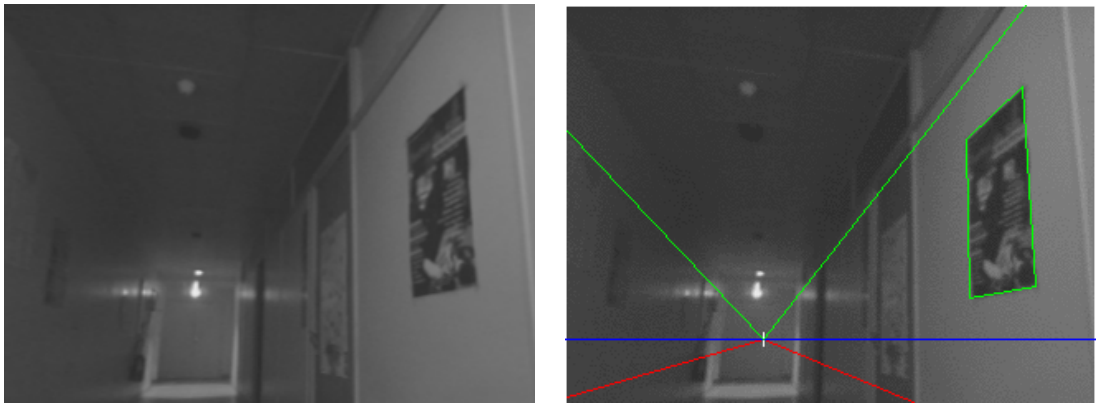


FIG. III.7 – Les formes quandrangulaires planes sont détectées et caractérisées.

► Les amers non pré-définis

L'idée ici, qui est l'une des idées centrales de la thèse, est que le ou les amers qui caractériseront une pièce ne seront aucunement pré-définis, c'est-à-dire que l'on ne cherchera pas délibérément tel ou tel type d'amers, telle ou telle structure géométrique, telle ou telle particularité : le robot ne sait pas par avance ce dont il va se servir comme amer.

Le robot n'a donc aucune connaissance préalable sur son environnement, et donc sera capable de s'adapter à tout environnement, structuré ou non.

Dans la littérature consacrée, cette “philosophie” se retrouve dans plusieurs approches du problème. En plus de la recherche et de la caractérisation de points d'intérêt, ou de l'étude plus globale d'images (au travers de transformations d'images type “analyse en composantes principales” ou “analyse en composantes indépendantes”), utilisant une quelconque distance (Mahalanobis, Hausdorff, ...), nous pouvons citer la construction de *cartes de saillance*.

Beaucoup de travaux ayant une telle approche sont d'inspiration biologique [Itti 98, Arleo 01, Gaussier 00].

III.6.3 L'inspiration biologique

Une grande tendance actuelle est la construction d'algorithmes bio-inspirés. En effet, la robotique mobile est à la recherche de nouveaux modèles pour la navigation, la reconnaissance de lieux, etc. La recherche en neuroscience, quant à elle, a besoin de valider ses modèles et la robotique mobile offre un support idéal d'expérimentation.

► Les cartes de saillance

Les primates ont une capacité à interpréter une scène très complexe en temps réel [Itti 98], malgré les limites dynamiques du réseau de neurones dédié à cette tâche. Il est donc très probable que la complexité des informations soit réduite avant analyse.

De nombreux modèles ont été proposés en ce sens et l'un d'entre eux, connu sous le nom de *théorie de fusion des attributs* (*feature integration théorie*), a conduit à une architecture d'inspiration biologique dans laquelle une scène est analysée au travers de ses différentes composantes. L'idée générale est la suivante [Itti 98] : l'image est décomposée en différentes cartes (figure III.8), via un certain nombre de filtres, chacune représentant telle ou telle spécificité à des échelles variables : les cartes de segments (verticaux, horizontaux, etc.) obtenues à l'aide des filtres de Gabor, les cartes de couleurs (rouge, vert, bleu), et la carte des intensités.

Un opérateur linéaire agissant simultanément sur les cartes d'attributs à différentes échelles permet de mettre en évidence les discontinuités spatiales locales, en s'inspirant du comportement des cellules de la rétine.

Un opérateur non linéaire de seuillage local permet de dégager les attributs dominant localement. Finalement, un opérateur de fusion plus ou moins complexe (combinaison linéaire, ajustement de la fusion par apprentissage supervisé, compétition des attributs) met en évidence les objets d'intérêt en s'inspirant des mécanismes biologiques de focalisation de l'attention.

► Les modélisations de l'hippocampe

Une autre approche est issue de la modélisation de l'hippocampe, situé dans la partie interne du lobe temporal, et qui contient les structures fonctionnelles de la mémoire.

C'est à partir de cette modélisation que P. Gaussier *et al.* [Gaussier 00] ont développé l'architecture `PerAc`, qui permet à un robot d'apprendre à reconnaître un signe (une flèche par

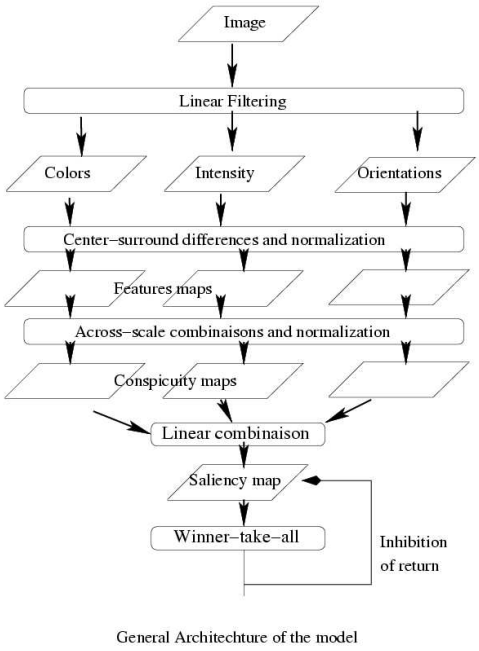


FIG. III.8 – Cartes de Saillance.

exemple) et de se déplacer en fonction de ce signe, avec l'aide d'un tuteur. Ainsi, après plusieurs séquences d'apprentissage, si le robot repère une flèche orientée vers la droite (*resp.* la gauche), il tournera à droite (*resp.* à gauche).

D'autres travaux sont en cours au Laboratoire de Physiologie de la Perception et de l'Action du Collège de France, sous la direction d'Alain Berthoz. La politique scientifique tend à favoriser de plus en plus l'interaction entre la robotique et les neurosciences. Un exemple-type excelent est la modélisation des cellules de lieux (*place cells*) dans l'hippocampe des rats, à l'origine de la navigation de ces animaux ([Arleo 98, Arleo 02, Arleo 01]...).

III.6.4 Notre approche du problème

L'idée générale est que le robot sache déterminer, par apprentissage, les amers les plus utiles, les plus robustes, les plus caractéristiques pour chaque pièce d'un environnement structuré, afin de les reconnaître par la suite et de se localiser efficacement.

Il est important de noter que la recherche d'amers se fait dans deux sens différents :

1. des amers comme *conjonction* d'attributs des différentes images d'une même pièce,
2. et des amers comme *disjonction* d'attributs entre deux pièces différentes.

En effet, par définition, un amer est caractéristique d'un et d'un seul lieu, et la recherche de simples propriétés communes à un ensemble de vues d'une même pièce n'est pas garant de la propriété d'unicité des-dits amers. Dans un premier temps, nous allons donc rechercher des *attributs* dans les images, que nous définirons ensuite, avant d'exposer l'algorithme général de notre approche.

La question qui se pose alors est la suivante : comment extraire des amers symboliques ? Comment articuler et combiner les attributs visuels ? Quels processus d'apprentissage permettent-ils, à partir de ces attributs, de caractériser chaque pièce d'un environnement structuré ?

Le prochain chapitre expose quelques techniques d'apprentissage les plus courantes afin d'exploiter ces attributs dans un processus de reconnaissance de lieux à partir d'images naturelles et met en lumière les avantages et inconvénients respectifs de chaque technique en fonction de nos hypothèses initiales.

Chapitre IV

Apprentissage, Analyse de Données et Classification

"Jadis, on appelait pédagogue l'esclave qui conduisait à l'école l'enfant noble. (...) Le petit quitte la maison de famille ; sortie : deuxième naissance. Tout apprentissage exige ce voyage avec l'autre et vers l'altérité. Pendant ce passage, bien des choses changent." Michel Serres, Le Tiers-Instruit

La terme d'*"apprentissage"* apparu au XIV^e se définissait comme l'*action d'apprendre un métier*. Il s'agissait donc d'une notion bien définie dans le contexte du travail essentiellement manuel de l'époque (d'où le terme d'"apprenti", par exemple). La définition s'est étendue et généralisée ensuite, par analogie, à *l'acquisition des connaissances* dans son ensemble (savoir, école ...), et aujourd'hui cette définition s'est adaptée aux contextes de chaque discipline.

En psychologie[1], par exemple, l'apprentissage se définit comme un *"processus d'effet plus ou moins durable par lequel des comportements nouveaux sont acquis ou des comportements déjà présents sont modifiés en interaction avec le milieu ou l'environnement"*. En médecine, il s'agit d'un *"processus par lequel une personne construit sa motricité ou élargit son champ de connaissances"*. En zoologie, il s'agit d'un *"processus cognitif qui permet à un animal, à partir de son expérience passée, d'assimiler l'organisation de son environnement et les conséquences*

[1] Toutes les définitions qui suivent dans ce paragraphe sont issues de l'Office de la Langue Française.

de ses propres actions, et d'accomplir l'autorégulation de ses comportements en fonction de cette assimilation". Enfin, pour finir sur une définition plus "exotique", dans le monde de la publicité, l'apprentissage est vu comme la *"modification du comportement d'un consommateur à la suite d'un stimulus publicitaire"*. Que ce soit pour l'homme, l'animal ou le consommateur, les apprentissages sont très similaires...

Quoi qu'il en soit, ces définitions plus actuelles montrent le glissement qui s'est opéré aux cours des dernières décennies, de la simple acquisition des connaissances vers la modification du comportement (habituation, voire conditionnement, comme le montrent les travaux de Pavlov). L'apprentissage renvoie à l'opposition entre l'inné et l'acquis. Il dépasse cette opposition en proposant un schéma dynamique d'acquisition.

D'un point de vue informatique, l'apprentissage est vu comme la capacité d'une machine à modifier elle-même sa base de connaissance afin d'améliorer les résultats qu'elle propose ou d'agir de façon appropriée. On entre ainsi dans le domaine de l'*intelligence artificielle*, qui distingue la machine intelligente du simple calculateur ou du simple "disque dur".

Il existe plusieurs lignes directrices pour orienter les recherches vers l'apprentissage de machines : aide à la décision, compréhension de systèmes complexes, compréhension de l'espèce humaine, éternelle amélioration des conditions de vie, progrès, *etc.* L'apprentissage s'intéresse en général à plusieurs domaines particuliers avec leurs applications respectives (comportement, stratégie, navigation, discrimination, catégorisation, adaptation, *etc.*), et dans lesquelles on cherche à répondre à un objectif bien précis.

Ce chapitre à pour objectif, dans un premier temps, de présenter des notions générales sur l'apprentissage pour situer notre approche : apprentissage supervisé, apprentissage non supervisé, et apprentisage par renforcement. Notre approche se situe dans le cadre de l'apprentissage supervisé.

Ensuite sont présentées quelques notions générales d'analyse de données, avant d'introduire succinctement différentes techniques de classification supervisée : les hyper-plans séparateurs, les support vector machines, les réseaux connexionnistes, les k plus proches voisins, les arbre de décision, et enfin, la hiérarchie de concepts, que nous avons choisie pour la suite de notre travail.

IV.1 Généralités sur l'Apprentissage

L"apprentissage" au sens large est un domaine si vaste qu'on ne peut le synthétiser de façon convenable dans un chapitre comme celui-ci.

Comment le définir ? D'une manière générale [Dutech 98], *l'apprentissage est la capacité à faire mieux la prochaine fois*. [Dutech 98] et [Pastor 95] donnent tous deux des significations de *"faire mieux"*.

Une machine sera appelée "intelligente" à partir du moment ou elle construit son propre modèle du monde et agit en fonction d'un objectif précis et de circonstances non anticipées, comme pour la planification. Elle doit pour cela posséder tous les éléments nécessaires à son apprentissage, à savoir,

1. un support de mémoire : table de correspondance entrées / sorties, réseau de neurones, *etc.* ;
2. et un mécanisme de programmation ou de modification de cette mémoire, en fonction de ses entrées (perceptions du monde) et de ses sorties (actions, décisions...).

Ces deux éléments sont nécessaires à tout apprentissage artificiel, et en général sont interdépendants l'un de l'autre. Le schéma général en apprentissage est présenté figure IV.1.

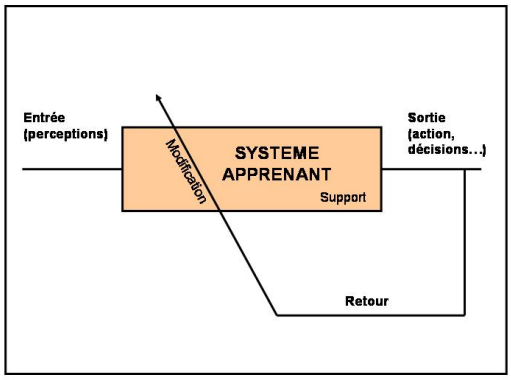


FIG. IV.1 – Schéma général de l'apprentissage.

Parmi les différentes formes d'apprentissage (l'apprentissage de symboles, la classification, les techniques connexionnistes, les algorithmes génétiques, l'apprentissage par renforcement, le méta-apprentissage, *etc.* [Shavlik 90]), on distingue trois modes d'apprentissage bien différents :

– l'apprentissage supervisé ;
– l'apprentissage non supervisé ;

– et l'apprentissage par renforcement.

Nous présentons ici succinctement ces trois apprentissages dans des applications particulièrement adaptées (un simulateur pour un apprentissage supervisé, et un contrôleur pour l'apprentissage par renforcement).

IV.1.1 L'apprentissage supervisé

Un apprentissage supervisé suppose que soit fourni à la machine la sortie désirée en fonction de l'entrée (et si le système est récursif en fonction des entrées précédentes). La simulation d'un système réel $\mathcal{S}$ par un système apprenant permet d'illustrer ce type d'apprentissage (figure IV.2).

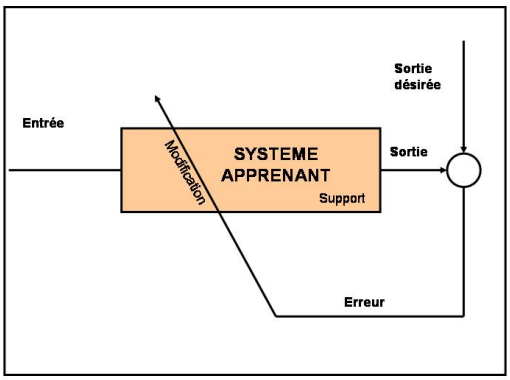


FIG. IV.2 – Apprentissage supervisé : la sortie du système apprenant est comparée à la sortie désirée, et l'erreur permet de modifier le support d'apprentissage : si l'erreur est nulle, les sorties sont identiques, donc le système est bien programmé.

Une organisation particulière d'apprentissage supervisé s'articule autour de deux machines, l'une "instructeur", l'autre "apprenti". Le dialogue qui s'instaure entre ces deux machines est alors sous forme de questions-réponses [Angluin 88, Sallantin]. La machine "apprenti" pose deux types de questions (d'appartenance : *"est-ce que f(x) est vrai"*, ou d'équivalence : *"est-ce que f=g"*) et la machine "instructeur" doit répondre en conséquence (respectivement soit par *"vrai"* ou *"faux"*, soit par *"vrai"* ou en donnant un conte-exemple). Le résultat d'une telle organisation est la suivante : l'apprenti parvient à résoudre un problème avec un nombre fini d'erreurs s'il possède les même capacités que l'instructeur et si celui-ci a déjà résolu le problème. Ceci est vrai même si l'instructeur est "malicieux" (il trompe l'apprenti un nombre fini de fois) et "faillible" (il se tait un nombre fini de fois) [Angluin 97]. De plus, ce type d'apprentissage converge à la limite : l'apprenti ne se trompera qu'un nombre fini de fois après un nombre fini d'exemples. D'autres caractéristiques relatifs à ces résultats sont analysées par Sallantin dans [Sallantin].

Les techniques d'apprentissage supervisé plus "classiques" seront développées longuement par la suite dans ce chapitre.

IV.1.2 L'apprentissage non supervisé

L'apprentissage non supervisé (figure IV.3) suppose donc qu'il n'y a pas de retour d'information venant de l'environnement. L'intervention humaine, en cours d'apprentissage, est donc inexistante. Elle valide cependant le résultat final.

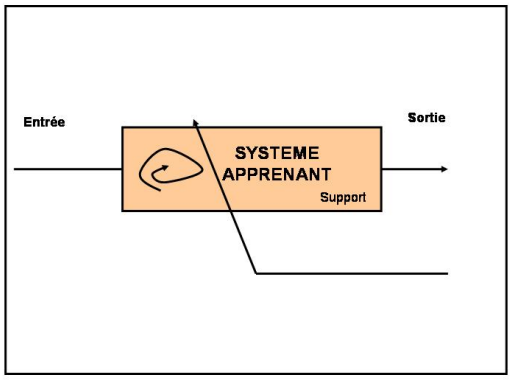


FIG. IV.3 – Apprentissage non supervisé : il n'y a pas de retour d'information de l'environnement.

Ce type d'apprentissage peut se décliner principalement sous ces différentes formes (figure IV.4) :

– l'estimation de fonction de densité ou de fonction de probabilité. C'est la forme la plus générale d'apprentissage non-supervisé ;

– la découverte de classes *naturelles*, par regroupement d'individus plus ou moins similaires, ou *"clustering"* (par exemple, l'algorithme des K-moyennes) ;

– et l'apprentissage de variétés de faible dimension, permettant de représenter les données de façon synthétique (par exemple, l'analyse en composantes principales / indépendantes).

▶ Algorithme des K-moyennes

L'algorithme des "K-moyennes" est un algorithme très simple qui permet de regrouper des données en classes (clustering).

Un exemple illustratif est montré figure IV.6. Il existe des variantes de cet algorithme [Theiler 97, Leeser 98], mais elles sont toutes issues de l'algorithme initial présenté figure IV.5.

Cet algorithme converge toujours. Initialement, le nombre K de centroïdes est soit fixé par l'opérateur, soit incrémentalement augmenté jusqu'à ce que toutes les distances entre les cen-

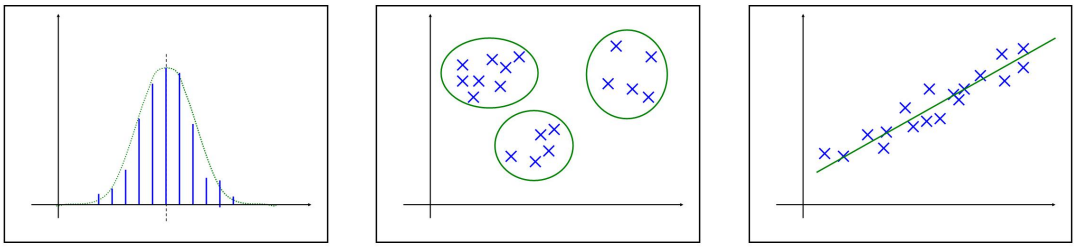


FIG. IV.4 – Trois formes d'apprentissage non supervisé. À gauche, à partir du nombre d'éléments en fonction de la valeur d'une donnée précise, il est possible de déterminer le type de distribution (ici, gaussienne) et les caractéristiques associées (μ, σ) ; au centre, un cas typique de regroupement de données (clustering) ; à droite, une représentation simplifiée des données autour d'une droite (analyse en composantes principales).

---

1. Placer K centroïdes au hasard
2. Tant que les centroïdes bougent
  2.1. Attacher chaque donnée au centroïde le plus proche
  2.1. Déplacer chaque centroïde au centre de gravité des données qui y sont attachées

---

FIG. IV.5 – Algorithme des K-Moyennes.

troïdes et "leurs" données soient inférieures à une distance maximale donnée.

On constate enfin que cet algorithme peut dépendre d'un grand nombre de paramètres en fonction de l'application : sans entrer dans le détail, citons la notion de distance et sa mesure, le nombre de classes, la validation terminale, l'initialisation...

► Analyse en composantes principales

L'extraction de caractéristiques [Deco 96] est l'un des principaux objectifs de l'apprentissage non supervisé. Une première approche, maintenant classique, est l'utilisation d'un outil statistique appelé ACP (Analyse en Composantes Principales), dont le but est de trouver un système de M vecteurs orthogonaux ($M < N = dim(E)$) sur lequel la variance des données projetés est maximale. Ces M vecteurs, appelés *first Principal Components* (par opposition aux $N - M$ *second Principal Components*) définissent les directions de variance maximal.

Le calcul se détermine à l'aide du théorème suivant :

Théorème 1 *La $k^{ème}$ composante principale du vecteur d'entrée x est le vecteur propre v_k normalisé correspondant à la valeur propre λ_k de la matrice de covariance Q_x, avec des*

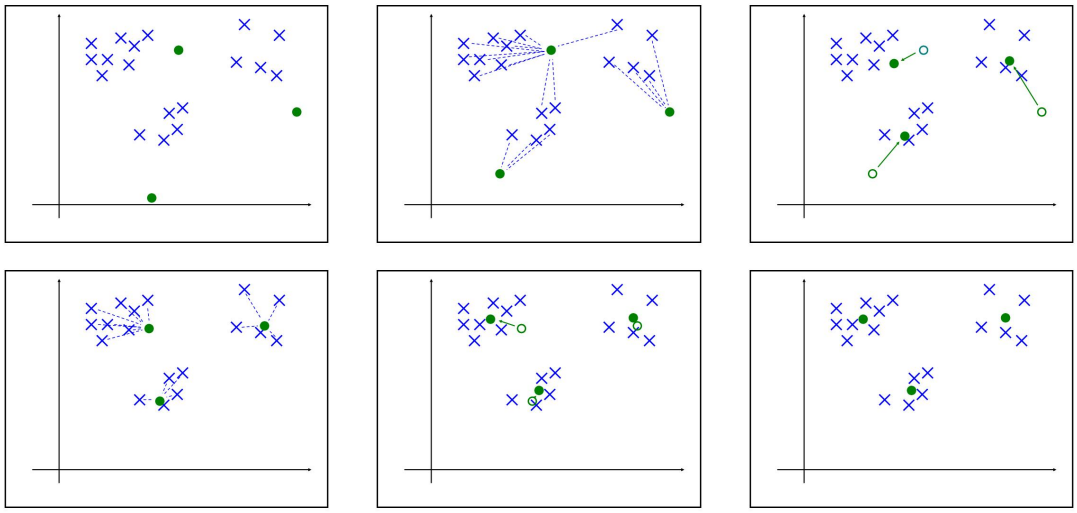


FIG. IV.6 – Illustration de l'algorithme des K-Moyennes. Initialement, des centroïdes sont positionnés au hasard, et à chaque donnée est attaché le centroïde le plus proche ; ensuite, le centroïde se déplace au centre de gravité de "ses" données, puis de nouveau à chaque donnée est attaché le centroïde le plus proche, et ainsi de suite jusqu'à ce que les centroïdes ne bougent plus.

valeurs propres ordonnées $\lambda_1 > \lambda_2 > \cdots > \lambda_N$.

L'ensemble de ces vecteurs orthogonaux forment une "représentation" synthétique des données comme le montre la figure IV.7, à gauche.

► Analyse en composantes indépendantes

Une généralisation non linéaire de cette méthode, beaucoup plus puissante mais de mise en œuvre plus délicate [Hérault 94], a été introduite, à l'origine, pour comprendre le fonctionnement du cerveau et du système nerveux, puis pour répondre à des problèmes spécifiques de traitement du signal [Deco 96]. Appelée ACI (Analyse en Composantes Indépendantes), elle a permis d'obtenir de meilleurs résultats dans des applications de compression d'information et de séparation de sources [Hérault 94]. Cette théorie est développée dans le livre de G. Deco et D. Obradovic [Deco 96].

Contrairement à l'ACP, les vecteurs de la base ne sont plus obligatoirement perpendiculaires. Cette technique permet donc de traiter les données issues de plusieurs sources indépendantes [Hyvarinen 01].

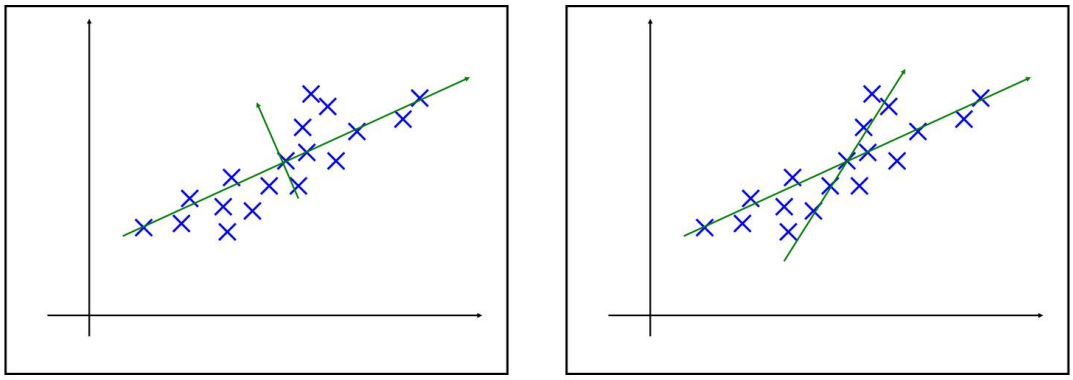


FIG. IV.7 – Illustration d'une analyse en composantes principales (à gauche) et d'une analyse en composantes indépendantes (à droite).

► Cartes auto-organisatrices (Kohonen)

Les cartes auto-organisatrices, ou cartes de Kohonen [Kohonen 82], ont été proposées par Kohonen en 1982 [Kohonen 82]. Elles consistent en un ensemble de neurones structurés sur lequel on définit une notion de voisinage : chaque neurone influe sur ses proches voisins. On introduit ainsi une topologie dans le réseau.

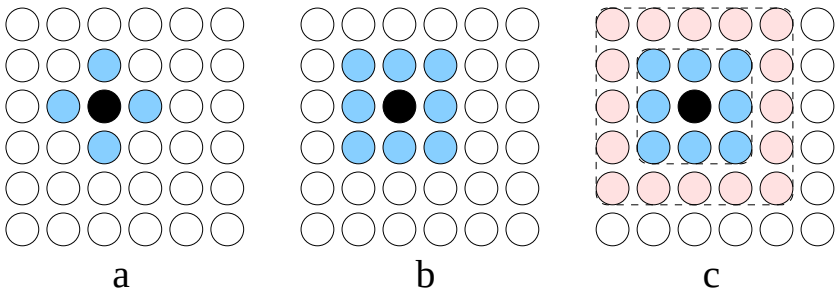


FIG. IV.8 – Cartes auto-organisatrices de Kohonen.

La figure IV.8 montre trois types de voisinage possibles (4-connexité à gauche, 8-connexité au centre), et à droite deux voisinages ($\mathcal{V}_1$ en bleu, $\mathcal{V}_2$ en rose) définis autour du neurone considéré.

L'idée de cet apprentissage est la suivante. Les neurones (vecteurs $\vec{q_{ij}}$ à N composantes, N étant la dimension de l'espace d'entrée, $I \times J$ la taille du réseau) de la carte sont initialisés aléatoirement. Lorsqu'un élément $\vec{x}$ de la base d'apprentissage est sélectionné, on recherche le neurone le plus proche (à l'aide de la distance de Mahalanobis ou autre), puis sont mises à jour

les composantes des neurones dans le voisinage du neurone sélectionné.

Cette mise à jour se fait sur deux "géographies" :
– pour le neurone le plus proche et ses voisins : $\vec{q_{ij}}(t+1) = \vec{q_{ij}}(t) + \alpha_t \cdot (\vec{x}(t+1) - \vec{q_{ij}}(t))$;
– pour les autres, rien ne change : $\vec{q_{ij}}(t+1) = \vec{q_{ij}}(t)$.

On peut remarquer que le coefficient d'apprentissage (ou taux d'apprentissage, ou pas d'adaptation) α dépend du temps. En effet, au début, le réseau sera très sensible aux vecteurs d'entrée $\vec{x}$, puis avec le temps le réseau se figera pour se stabiliser. En général, on pose $\alpha_t = 1/t$. Notons que s'il n'y a pas de voisin, on obtient une version adaptative des k-moyennes appelée "quantification vectorielle adaptative" (*LVQ : "Learning Vector Quantization"*).

Un autre facteur évoluant au cours du temps est le voisinage. En effet, au début, le voisinage sera plutôt large afin de mettre à jour un grand nombre de neurones. À terme, seul le neurone le plus proche sera mis à jour. De la même façon, on peut poser $r_t = r_0(1 - t/T)$, r étant le rayon du voisinage $\mathcal{V}$.

Les cartes permettent donc de dégager, une fois l'apprentissage avancé, des zones géographiques spécialisées dans des familles de vecteurs d'entrée, permettant ainsi une classification non supervisée. Ce modèle a connu dans les années 90 diverses variantes pour la classification non supervisée puis supervisée (apprentissage compétitif flou, mélange de gaussiennes, quantification adaptative vectorielle). L'algorithme simplifié de Kohonen est présenté figure IV.9.

---

1. Choix du nombre de neurones et de la topologie du réseau ; initialisation aléatoire des poids ; initialisation du voisinage et du pas d'adaptation
2. Tirage aléatoire d'une entrée dans la base d'apprentissage
3. Recherche du neurone le plus proche
4. Modification des poids des neurones voisins
5. Calcul de l'étendue du voisinage et du pas d'adaptation
6. Retour à l'étape 2 sauf si le test d'arrêt est vérifié

FIG. IV.9 – Algorithme simplifié de Kohonen.

IV.1.3 L'apprentissage par renforcement

► Généralités

Introduit par R.S. Sutton en 1984 [Sutton 84], l'apprentissage par renforcement (figure IV.10) est à mi-chemin entre l'apprentissage supervisé et l'apprentissage non supervisé. Il utilise une information particulière, le **signal de renforcement**, noté en général r, qualifiant l'ac-

tion entreprise.

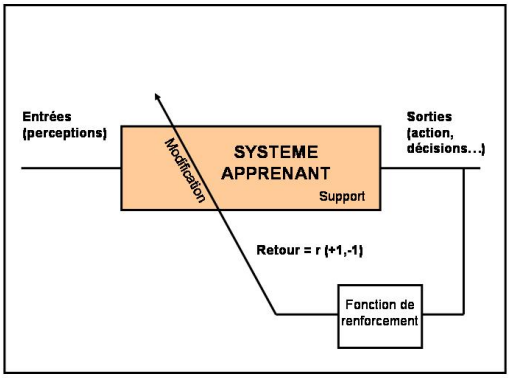


FIG. IV.10 – Apprentissage par renforcement.

L'idée d'un tel apprentissage est la suivante : l'agent, c'est-à-dire l'entité qui doit apprendre d'un côté, et agir "correctement" de l'autre, fait des expériences. Si celles-ci sont concluantes pour mener vers l'objectif final, le signal de renforcement sera alors égal à 1 ; inversement, s'il s'éloigne de son objectif, le signal sera égal à -1. Ce signal de renforcement, à travers la **fonction de renforcement**, joue donc un rôle de critique vis-à-vis de l'agent, sans pour autant lui donner des instructions (ce qui aboutirait à un apprentissage supervisé).

Pour illustrer ceci (figure IV.11), prenons le cas de la commande du pendule inversé. Le but est de trouver l'action a (ou commande c) optimale (ici, la force : $a = F$) en fonction de l'état du système (décrit par les deux paramètres Θ et $\dot{\Theta}$) afin de maintenir le pendule droit sur son socle, et ce malgré d'éventuelles perturbations. Il est possible de définir comme fonction de renforcement la fonction suivante :

$$r = \begin{cases} 1 & \text{si } (|\Theta| < \Theta_0) \text{ et } (|\dot{\Theta}| < |\dot{\Theta}_0|) ; \\ -1 & \text{sinon.} \end{cases}$$

Ainsi, on renforcera les actions qui mèneront vers l'objectif final, et inversement on "dé-renforcera" les actions non efficaces.

L'apprentissage par renforcement est donc un apprentissage par "essais et erreurs", très proche du fonctionnement humain. On n'apprend pas à un enfant qui essaie de marcher à activer tels et tels muscles pour avancer (si tant est que l'on soit nous même capable de connaître ces muscles !), mais on le soutient et le réconforte lorsqu'il réussit. La

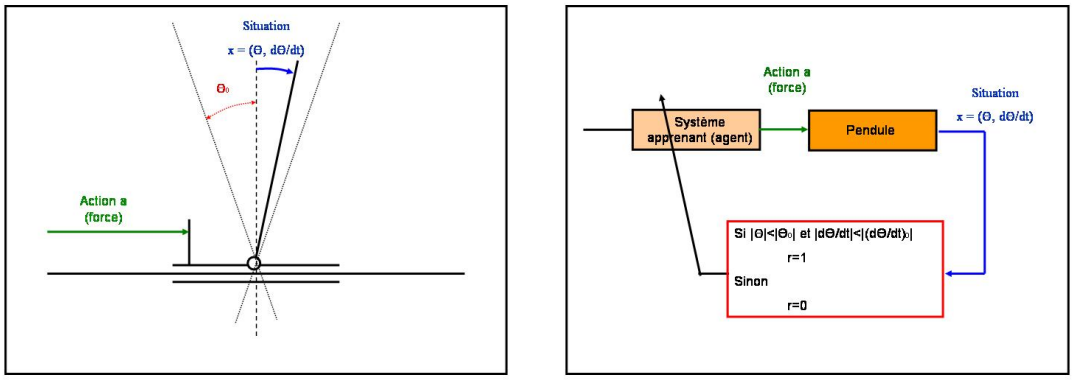


FIG. IV.11 – Apprentissage par renforcement. Si le pendule est en position verticale et à faible vitesse, le renforcement est positif ; sinon, il est négatif.

motivation et le jeu sont donc des moteurs puissants pour se perfectionner et développer ses capacités psychomotrices.

► Un algorithme particulier : le $Q - Learning$

L'algorithme $Q - learning$ (Quality-learning), plus simple que le AHC-learning (Adaptive Heuristic Critic-learning), construit une fonction Q, appelée fonction d'évaluation, qui représente la prévision sur la somme des renforcements que l'agent espère recevoir en exécutant l'action a à partir d'une situation x, et prend en compte la prédiction passée à un pas de temps. En effet, il est utile pour l'agent de connaître les conséquences de sa future action, afin de savoir s'il doit l'exécuter ou non.

► Fonction d'évaluation Q

En d'autres termes, $Q(x, a)$ donne une "idée" des conséquences qu'introduirait une action a dans une situation x définie. Plus la valeur de Q est grande, plus cette action sera bénéfique. Ainsi, dans une situation x donnée, on cherchera l'action a qui maximise la valeur de Q afin d'avoir l'action optimale :

$$a = \underset{b \in A}{\operatorname{argmax}}\, Q(x, b) \tag{IV.1}$$

Dans ce cas, a représente l'action de plus grande utilité à exécuter à partir de la situation x à un instant donné, A étant l'ensemble des actions possibles stockées en mémoire.

► Fonction de mise à jour de Q

La fonction de mise à jour a pour but de réactualiser la fonction d'évaluation après chaque expérience. On en déduit, pour chacune d'elle à l'instant t, l'équation suivante :

$$Q(x_t, a)_{nouveau} = Q(x_t, a)_{ancien} + \beta(r + \gamma.max_{b \in A} Q(x_{t+1}, b) - Q(x_t, a)_{ancien}) \quad \text{(IV.2)}$$

L'erreur calculée $\Delta = r + \gamma.max_{b \in A} Q(x_{t+1}, b) - Q(x_t, a)_{ancien}$ met à jour la fonction d'évaluation Q, et permet de propager dans le temps les valeurs de renforcement à travers la chaîne d'actions réalisées.

Le paramètre $\beta \in [0, 1]$ est un paramètre de *vitesse* de mise à jour. Plus β est proche de 1, plus Q est sensible aux dernières expériences ; plus β est proche de 0, moins l'influence des dernières expériences est importante. Ce paramètre caractérise donc une *profondeur de mémoire* prise en compte lors de la mise à jour de Q.

► **Algorithme général du** $Q - Learning$

L'algorithme général du $Q - Learning$ est décrit figure IV.12.

---

1. Initialiser la mémoire interne
2. À chaque instant t
  2.1. Observer l'état du monde
  2.2. Choisir une action a_t selon une heuristique choisie (au hasard, qui maximise Q, *etc.*)
  2.3. Éxécuter a_t
  2.4. Observer la nouvelle situation x_{t+1}
  2.5. Mettre à jour la fonction Q

FIG. IV.12 – Algorithme du Q-learning.

On peut noter, dans les travaux de Ricordeau [Ricordeau 04], que cette fonction Q peut être représentée par un treillis de Galois, au travers de la notion de "Q-concept-learning".

► **Convergence**

La convergence du Q-learning, contrairement au AHC-learning, a été démontrée sous certaines conditions par Watkins et Dayan en 1992.

IV.2 Analyse de Données

Nous présentons ici une introduction à l'analyse de données et des outils d'évaluation des classifieurs. Nous nous sommes largement inspiré des ouvrages suivants : [Mitchell 97, Govaert 03]

IV.2.1 Analyse de données et KDD

Née au début des années 90, dans le but d'exploiter les énormes bases de données disponibles dans les entreprises, l'"analyse de données" (*"Data mining"* en anglais), ou "fouille de données", est une composante importante de l'"extraction de connaissances dans une base de données" (*"Knowledge Discovery in Databases"* ou *"KDD"* en anglais). Le KDD est le processus complet qui comprend :

- la sélection,
- le pré-traitement,
- la transformation,
- l'analyse elle-même (*"data mining"*),
- et l'évaluation

des données. L'analyse des données est le cœur du processus, même s'il ne représente qu'entre 15% et 25% du processus global du KDD.

IV.2.2 Analyse prédictive et analyse descriptive

L'analyse des données peut se diviser en deux catégories : l'analyse *prédictive* et l'analyse *descriptive*. L'analyse prédictive consiste à trouver des *séparateurs* pour faire de la classification et de la prédiction (arbres de décision, classification bayésienne, SVM...) alors que l'analyse descriptive consiste à trouver des similarités entre les données d'une classe (*clustering*, apprentissage par associations de règle...) Pour résumer, on peut affirmer que l'analyse prédictive cherche à séparer les individus (au sens large) d'une population donnée, alors que l'analyse descriptive cherche à regrouper ces individus (figure IV.13).

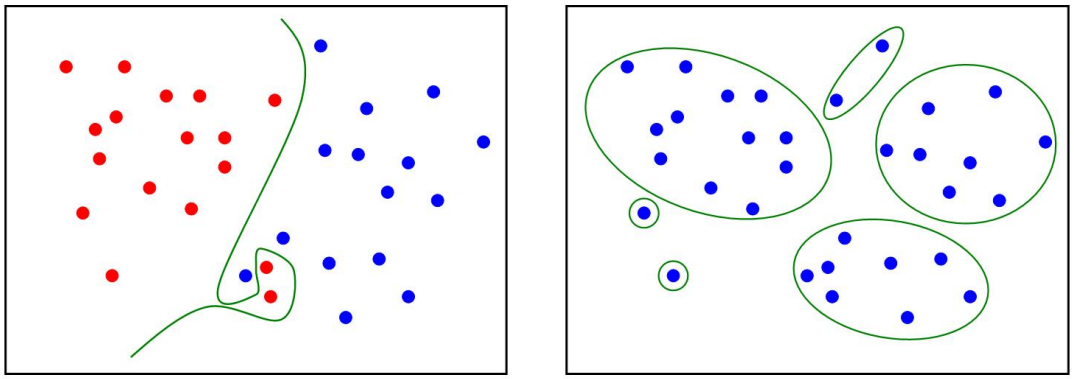


FIG. IV.13 – Analyse prédictive (à gauche) et analyse descriptive (à droite).

Nous nous intéressons ici à l'analyse prédictive. Ci-après sont exposés des outils permettant de classer des données selon ce point de vue. Avant cela, nous présentons en quelques mots un outil d'évaluation d'un classifieur.

IV.2.3 Matrice de confusion

L'évaluation de l'analyse se fait dans un premier temps au travers d'une **matrice de confusion** (on trouve les termes de **"confusion matrix"** ou **"contingency table"** en anglais) qui regroupe les résultats d'un classifieur. Prenons l'exemple d'une population regroupant deux types de données, les positives (rouges) et les négatives (bleues), et deux classifieurs linéaires (dont la formalisation sera donnée par la suite).

La matrice de confusion (figure IV.14) regroupe, à partir d'un échantillon d'évaluation (qui peut être la base d'apprentissage ou une base de test), les données positives classifiées positives, appelées "Vrais positifs" (ou *"True Positive"* -TP- en anglais), les données négatives classifiées négatives, appelées "Vrais Négatifs" (ou *"True Negative"* -TN), les données positives classifiées négatives, appelées "Faux Négatifs" (*"False Negative"* -FN), et enfin les données négatives classifiées positives, appelées "Faux Positifs" (*"False Positive"* -FP).

	Données classées positives	Données classées négatives
Données positives	**Vrais positifs - TP**	**Faux Négatifs - FN**
Données négatives	**Faux Positifs - FP**	**Vrais Négatifs - TN**

FIG. IV.14 – Matrice de confusion.

Ainsi, dans l'exemple de la figure IV.15, les deux classifieurs C_1 et C_2 sont de nature très différente : alors que C_1 cherche à diviser le jeu de données *"naturellement"*, en fonction d'un vide qui sépare au mieux les données, C_2 cherche avant tout à classer correctement les données positives.

IV.2.4 Le "ROC Space"

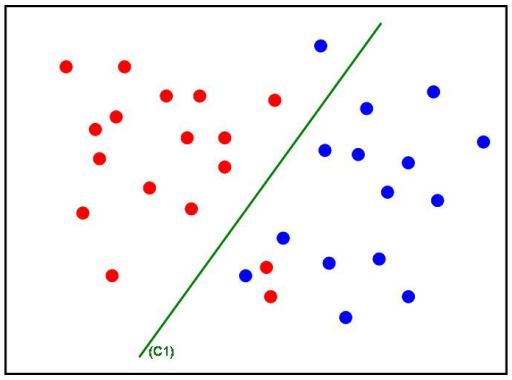
Le "ROC Space" [Provost 01] (*"Receiver Operating Characteristic"*, terme d'origine médical) est un espace de représentation des matrices de confusion. En abscisse, on représente le taux de "faux positifs" FP_r, c'est-à-dire le nombre d'exemples négatifs classés positifs sur le nombre d'exemples négatifs :

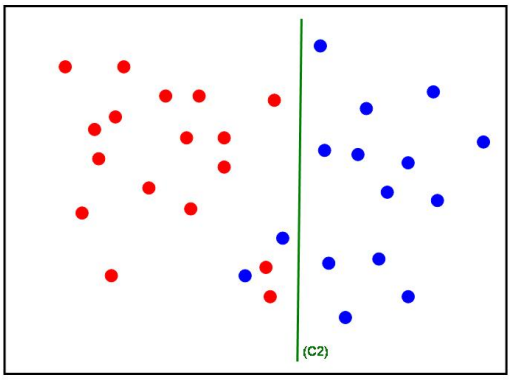
$$FP_r = \frac{FP}{FP + TN} \tag{IV.3}$$

et en ordonnée on représente le taux de "vrais positifs" TP_r, c'est-à-dire le nombre d'exemples positifs classés positifs sur le nombre d'exemples positifs :

$$FP_r = \frac{TP}{TP + FN} \tag{IV.4}$$

Dans notre exemple,





Classifieur C1			
	Prévus positifs	Prévus négatifs	
Positifs	**15**	**2**	17
Négatifs	**1**	**14**	15
	16	16	32

Classifieur C2			
	Prévus positifs	Prévus négatifs	
Positifs	**17**	**0**	17
Négatifs	**2**	**13**	15
	19	13	32

FIG. IV.15 – Matrices de confusion des classifieurs C_1 et C_2.

$$TP_r(C_1) = \frac{15}{17} = 88,2\% \quad \text{(IV.5)}$$

$$FP_r(C_1) = \frac{1}{15} = 6,67\% \quad \text{(IV.6)}$$

$$TP_r(C_2) = \frac{17}{17} = 100\% \quad \text{(IV.7)}$$

$$FP_r(C_2) = \frac{2}{15} = 13,3\% \quad \text{(IV.8)}$$

Ainsi avec ces données il est possible de représenter les caractéristiques du classifieur dans cet espace de représentation $(x = FP_r, y = TP_r)$, figure IV.16.

Plus un classifieur est situé en haut à gauche, plus il est performant. Ensuite, tout est question de compromis entre un classifieur qui classifie peu de données, tout en ne faisant que peu d'erreurs, et un classifieur qui classifie beaucoup, avec plus d'erreur. Seuls le type d'application et les contraintes initiales permettent de répondre au problème.

IV.3 Apprentissage et Classification Supervisée

Nous nous intéressons maintenant à un aspect particulier de l'apprentissage : la *classification*. Il existe plusieurs définitions similaires de ce terme : *former des classes dont on attend qu'elles soient cohérentes et bien isolées* [Govaert 03], *acte cognitif ou procédure permettant d'affecter à un objet la famille à laquelle il appartient, autrement dit de le reconnaître*

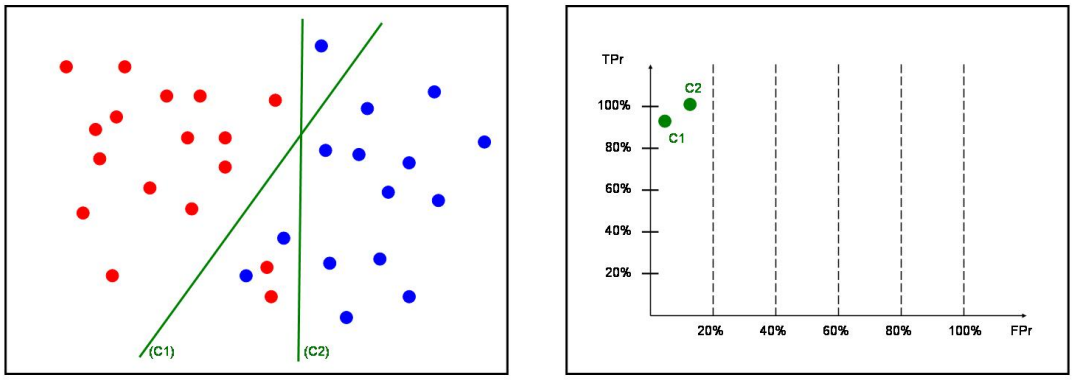


FIG. IV.16 – ROC Space.

[Cornuéjols 02], *etc.*

La classification existe pour les trois modes d'apprentissage : en mode supervisé (on affecte la classe à chaque donnée du jeu d'apprentissage), en mode non supervisé (clustering), et en apprentissage par renforcement, si la classification répond à un objectif qualifiable par la machine.

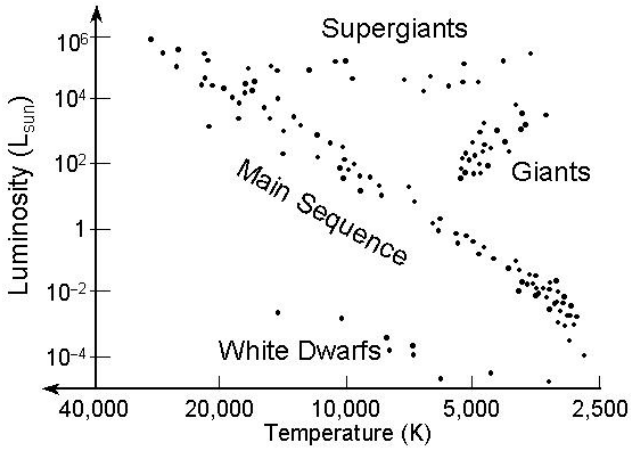


FIG. IV.17 – Diagramme de Hertzsprung-Russel. Ce diagramme de classification des étoiles fut élaboré indépendamment par l'astronome danois Eljnar Hertzsprung et l'astronome américain Henry Russel, en 1912.

Nous présentons succinctement dans cette section différentes méthodes de classification supervisée, domaine d'apprentissage principal de notre application. Nous nous sommes libre-

ment inspirés des ouvrages suivants : [Mitchell 97, Cornuéjols 02].

L'idée générale de la classification consiste donc à trouver, à partir d'un jeu de données appelé *base d'apprentissage*, une **règle de classification** vérifiée autant que possible par tous les éléments de la base d'apprentissage, et validée par un maximum d'éléments d'une *base de test*.

IV.3.1 Hyperplans séparateurs

Supposons dans une base d'apprentissage un ensemble $X = (x)_{1...N}$ de N données de degré d réparties en deux classes q_1 (rouge) et q_2 (bleu), comme le montre la figure IV.18.

De la même façon que précédemment, on étiquette "positif" les éléments de classe q_1 et "négatif" les éléments de classe q_2. On note alors respectivement $u = 1$ et $u = -1$.

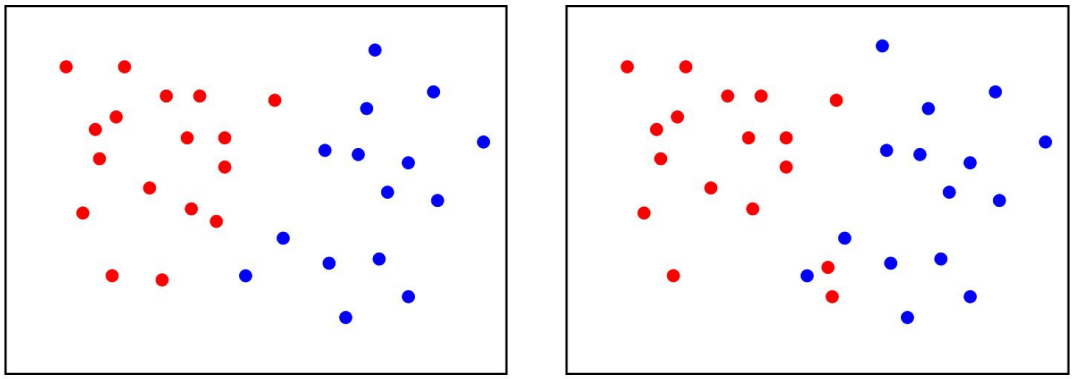


FIG. IV.18 – Deux jeux de données initiales, séparées en deux classes.

► Idée générale et définitions

L'idée la plus simple pour trouver une classification est de chercher une droite dans le plan qui sépare les deux classes (figure IV.19).

Dans $\mathbf{R}^d$, l'équation générale d'un hyperplan $\mathcal{H}$ est de la forme suivante :

$$\mathcal{H} : w_0 + w^T.x = 0 \tag{IV.9}$$

avec w un vecteur de dimension d et w_0 un scalaire. Dorénavant, on posera $h(x) = w_0 + w^T.x$.

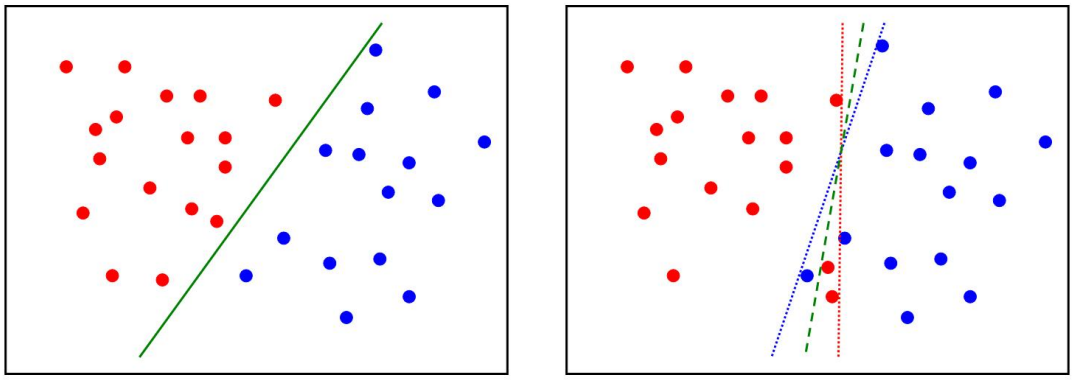


FIG. IV.19 – Droites séparatrices sur les jeux de données initiales.

Soit à considérer les deux propositions suivantes :

$$\forall x \in q_1, h(x) = w_0 + w^T.x \geq 0 \quad \text{(IV.10)}$$
$$\forall x \in q_2, h(x) = w_0 + w^T.x \leq 0 \quad \text{(IV.11)}$$

Définition 3 *Si* $\exists(w, w_0) \in (\boldsymbol{R}^d, \boldsymbol{R})$ *tel que les propositions ci-dessus soient vérifiées, alors on dit que les classes sont* **séparables**.

Définition 4 *On appelle* **hyperplan séparateur** *un hyperplan qui sépare parfaitement deux classes séparables d'une base de données.*

Ainsi le premier exemple admet un hyperplan séparateur car les deux classes sont séparables. Dans le second exemple, les droites séparatrices *maximales* admissibles par chaque classe se croisent entre deux de leurs éléments ; il n'existe donc pas d'hyperplan séparateur, car les classes ne sont pas séparables.

► Règle de classification

Lorsqu'apparaît un nouvel objet x^* à classer, il suffit de regarder sa position pour déterminer sa classe : s'il est à gauche, alors l'objet est rouge ; s'il est à droite, alors l'objet est bleu (figure IV.20).

Il est alors possible d'établir une **règle de classification** qui s'exprimerait dans nos exemples de la façon suivante : *je cherche la droite séparatrice optimale, et je décrète que ce qui est à gauche de cette droite est rouge, ce qui est à droite est bleu.*

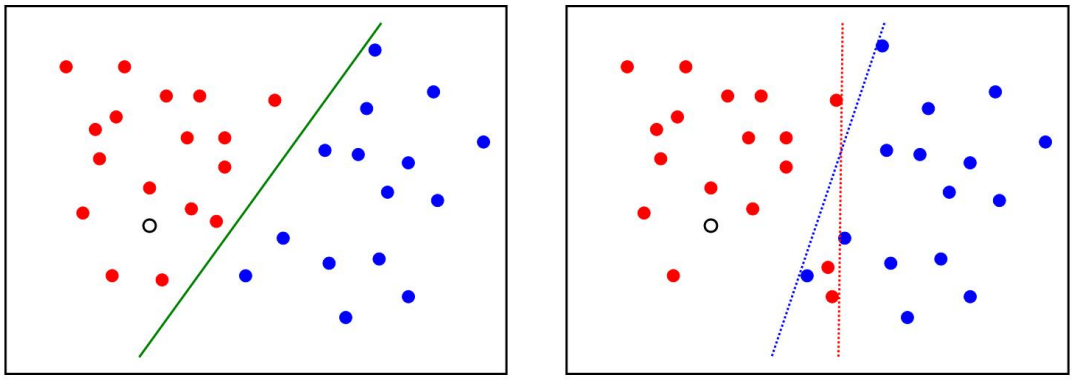


FIG. IV.20 – Classification d'un nouvel élément.

Mathématiquement, établir la règle de classification consiste à déterminer les paramètres a_0 et a tels que les équations IV.10 et IV.11 soient *au mieux* vérifiées, puis à poser pour tout nouvel élément x^* :

$$\text{Si} \quad h(x) = w_0 + w^T.x^* \geq 0, \text{ alors } x^* \in q_1(u(x) = 1) \tag{IV.12}$$

$$\text{Si} \quad h(x) = w_0 + a^T.w^* \leq 0, \text{ alors } x^* \in q_2(u(x) = -1) \tag{IV.13}$$

Remarque 1 *Il n'est pas nécessaire que les classes soient séparables pour établir une règle de classification. On parle alors non plus d'hyperplan séparateur, mais d'hyperplan* **optimal**.

Dans le second exemple, on peut établir par exemple comme règle de classification : *"ce qui est à gauche des deux droites maximales admissibles par chaque classe est rouge, ce qui est à droite des deux droites maximales admissibles par chaque classe est bleu, sinon ce n'est pas classé"*. Le nouvel objet du second exemple serait alors rouge.

► Marge et hyperplan optimal

Dans le cas où les classes sont linéairement séparables, il existe en général une infinité d'hyperplans séparateurs. Il est donc intéressant de rechercher l'hyperplan optimal.

Cet hyperplan optimal correspond à une marge maximale $\hat{\mathcal{M}}$ (figure IV.21). La distance entre un point x et l'hyperplan $h(x) = w_0 + w^T.x$ étant égale à $|h(x)|/||w||$, maximiser cette distance (marge) revient donc à trouver (w_0, w) définis par :

$$\underset{w,w_0}{\text{argmax}} \min\{||x - x_i|| : x \in \mathbf{R}^d, h(x) = 0, i = 1 \ldots N\} \tag{IV.14}$$

Dans ce cas, en normalisant w et w_0, on peut poser $h(x) = 1$ pour l'exemple positif le plus proche (plus tard appelé "vecteur de support"), et $h(x) = -1$ pour l'exemple négatif le plus

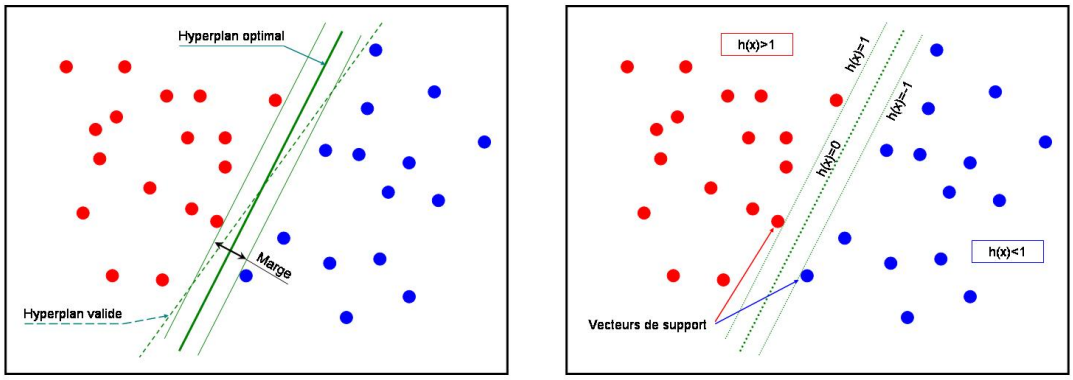


FIG. IV.21 – Hyperplan séparateur et marge.

proche. Dans ce cas,

$$\hat{\mathcal{M}} = \frac{2}{||w||} \tag{IV.15}$$

► Algorithmes de recherche de l'hyperplan optimal

Il existe plusieurs méthodes pour rechercher un hyperplan convenable. Nous allons en parcourir trois : la méthode de Fisher (la plus ancienne), l'algorithme du percetron, qui ne converge pas si les éléments ne sont pas linéairement séparables, et une approche itérative, l'algorithme de Ho et Kashyap).

► Méthode de Fisher

L'algorithme de Fisher est un algorithme particulier qui travaille non plus dans l'espace vectoriel initial de dimension d, mais sur une droite $\mathcal{D}$ de cet espace sur laquelle se projettent tous les vecteurs x de la base d'apprentissage.

Cette droite est construite afin de discriminer, de façon optimale, les projetés des vecteurs d'apprentissage sur cette droite.

L'hyperplan séparateur est alors l'hyperplan perpendiculaire à cette droite passant par le point de discrimination maximale. Un exemple est montré figure IV.22.

► Algorithme du perceptron

L'algorithme du perceptron est un algorithme itératif issu des premières approches connexionnistes (voir plus loin), au début des années 60.

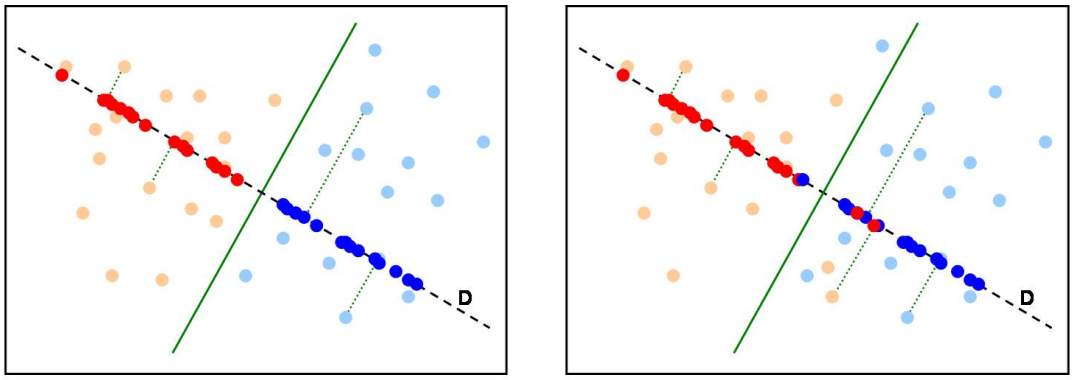


FIG. IV.22 – Méthode de Fisher.

L'itération est portée sur le vecteur a vu précédemment et évolue en fonction des éléments d'entrée et de leur classe. À chaque nouvel élément x de la base d'apprentissage, le vecteur est testé par l'hyperplan $w_{(t)}$ courant. Si celui-ci classe l'élément correctement, l'hyperplan n'est pas modifié. Si celui se trompe, alors on considère deux cas :

$$\text{si } x \in q_1, \text{ alors } w_{(t+1)} := w_{(t)} + \alpha x \tag{IV.16}$$

$$\text{si } x \in q_2, \text{ alors } w_{(t+1)} := w_{(t)} - \alpha x \tag{IV.17}$$

L'algorithme du perceptron (cité dans [Cornuéjols 02]) est donné figure IV.23. Il est important de noter que l'algorithme du perceptron ne converge pas si les éléments ne sont pas linéairement séparables.

▶ Algorithme de Ho et Kashyap

Considérons les vecteurs W, M et B avec $W^T = (w_0, w^T)$ les paramètres de l'hyperplan, M l'ensemble des éléments associés à leur classe, et B un vecteur positif de $\mathbf{R}^N$. Chercher un hyperplan séparateur revient à rechercher le vecteur W tel que :

$$W^T.M = B^T \tag{IV.18}$$

Dans la pratique, rechercher un hyperplan optimal revient à minimiser une fonction, comme par exemple la fonction :

$$J(W, B) = ||W^T.M - B^T||^2 \tag{IV.19}$$

S'il existe un hyperplan séparateur, M est inversible et on obtient $J(W, B) = 0$. Dans le cas contraire, il faudra minimiser cette fonction. On utilise alors la *pseudo-inverse* $M^+ =$

1. Choisir w_0 et α positifs quelconques
2. t=0
3. Tant que $t \leq t_{max}$
  3.1. Prendre au hasard un élément x de la base d'apprentissage
  3.2. Si x est bien classé par le classifieur courant
    3.2.1. Alors $w_{(t+1)} := w_{(t)}$
  3.3. Sinon
    3.3.1. Si $x \in q_1$
      3.3.1.1. Alors $w_{(t+1)} := w_{(t)} + \alpha x$
    3.3.2. Si $x \in q_2$
      3.3.2.1. Alors $w_{(t+1)} := w_{(t)} - \alpha x$
  3.4. $t = t + 1$

FIG. IV.23 – Algorithme du perceptron.

$M^T.(M.M^T)^{-1}$ de M issue du raisonnement suivant :

$$W^T.M = B^T \quad \text{(IV.20)}$$
$$W^T.M.M^T = B^T.M^T \quad \text{(IV.21)}$$
$$W^T.(M.M^T).(M.M^T)^{-1} = B^T.M^T.(M.M^T)^{-1} \quad \text{(IV.22)}$$
$$W^T = B^T.M^T.(M.M^T)^{-1} \quad \text{(IV.23)}$$
$$W^T = B^T.M^+ \quad \text{(IV.24)}$$

L'algorithme itératif proposé par Ho et Kashyap est celui présenté figure IV.24. α est un coefficient qui fixe la vitesse de convergence de l'algorithme. $\lfloor V \rfloor$ est le vecteur V pour lequel les composantes négatives ont été remplacées par 0.

1. Choisir B_0 et α positifs quelconques
2. t=0
1. Tant que $t \leq t_{max}$ ou $J(W, B) \geq \epsilon$
  3.1. $W^T_{(t)} := B^T_{(t)}.M^+$
  3.2. $B^T_{(t+1)} := B^T_{(t)} + \alpha.\lfloor W^T_{(t)}.M - B^T_{(t)} \rfloor$
  3.3. $t = t + 1$

FIG. IV.24 – Algorithme itératif de Ho et Kashyap.

La valeur de α entre dans un compromis classique en automatique : plus elle sera petite,

plus l'algorithme convergera lentement, et il risquera de converger vers un minimum local. En revanche, une grande valeur de α risquera d'entraîner une instabilité, ce qui peut empêcher toute convergence de l'algorithme.

IV.3.2 Les Support Vector Machine

Dans la plupart des cas, les classes d'objets ne sont pas linéairement séparables. L'idée alors est de trouver un nouvel espace de représentation des données (autre que celui des descripteurs) dans lequel les classes seront linéairement séparables. C'est la philosophie des *Support Vector Machine* [Vapnik 96] (on trouve parfois le terme francisé de *Support à Vaste Marges*). Le terme de SVM regroupe en fait une famille de méthodes permettant d'atteindre cet objectif.

Sans entrer dans le détail des procédures, ce n'est pas l'objet de ce manuscrit, nous exposons ici une vue d'ensemble de cette technique récente.

Lorsque les deux classes sont séparables, on a vu que rechercher l'hyperplan optimal revient à chercher le séparateur de plus vaste marge (figure IV.21). Résoudre ce problème revient à :

- minimiser $||w||^2$,
- sous les contraintes $\forall i = 1 \dots N, u_i.h(x_i) \geq 1$.

Ce problème est un problème d'optimisation quadratique sous contraintes linéaires d'inégalités. Dans le cas où il existe une solution (classes linéairement séparables), le théorème de Khun-Tucker permet, aux travers des paramètres de Lagrange-Khun-Tucker, de trouver la solution par la recherche de paramètres α_i tels que :

$$\begin{cases} \forall i = 1 \dots N, \alpha_i \geq 0 \\ \sum_{i=1}^{N} \alpha_i.u_i = 0 \\ \max_\alpha \{\sum_{i=1}^{N} \alpha_i - 1/2 \sum_{i,j=1}^{N} \alpha_i \alpha_j u_i u_j (x_i.x_j)\} \end{cases}$$

L'hyperplan solution peut alors s'écrire de la façon suivante :

$$h(x) = (w^*.x) + w_0^* = \sum i = 1^N \alpha_i^* u_i (x.x_i) + w_0^* \tag{IV.25}$$

avec les α_i solution du système précédent.

Soit Φ une transformation non linéaire qui permet de passer de l'espace de description initial vers un espace de redescription. Le problème d'optimisation s'écrit alors :

$$\begin{cases} \forall i = 1 \dots N, \alpha_i \geq 0 \\ \sum_{i=1}^{N} \alpha_i.u_i = 0 \\ \max_\alpha \{\sum_{i=1}^{N} \alpha_i - 1/2 \sum_{i,j=1}^{N} \alpha_i \alpha_j u_i u_j \langle \Phi(x_i).\Phi(x_j) \rangle\} \end{cases}$$

L'hyperplan solution peut alors s'écrire de la façon suivante :

$$h(x) = (w^*.x) + w_0^* = \sum i = 1^N \alpha_i^* u_i \langle \Phi(x_i).\Phi(x_j) \rangle + w_0^* \tag{IV.26}$$

avec les α_i solution du système précédent.

Or les produits scalaires deviennent impossible à calculer pour des grandes dimensions, c'est pourquoi on utilise des fonctions bilinéaires symétriques positives, appelées *fonctions noyau* $K(x, y)$, qui correspodent à un produit scalaire $\langle \Phi(x).\Phi(y) \rangle$. Le nouveau système s'écrit alors :

$$\begin{cases} \forall i = 1 \dots N, \alpha_i \geq 0 \\ \sum_{i=1}^{N} \alpha_i.u_i = 0 \\ \max_\alpha \{ \sum_{i=1}^{N} \alpha_i - 1/2 \sum_{i,j=1}^{N} \alpha_i \alpha_j u_i u_j K(x_i, x_j) \} \end{cases}$$

L'hyperplan solution peut alors s'écrire de la façon suivante :

$$h(x) = (w^*.x) + w_0^* = \sum i = 1^N \alpha_i^* u_i K(x_i, x_j) + w_0^* \tag{IV.27}$$

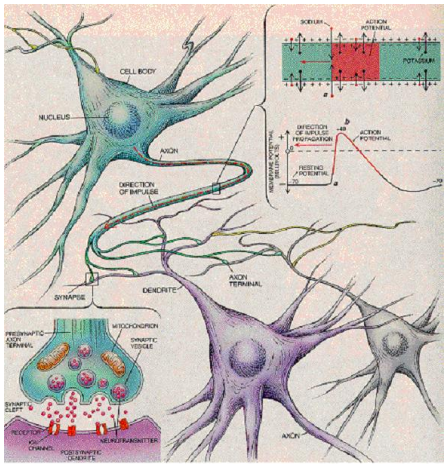
avec les α_i solution du système précédent.

Sans aller plus loin (de nombreux ouvrages complets traitent directement du sujet, notamment celui de Cristianini et Shawe-Taylor [Cristianini 00]), il faut savoir qu'il existe plusieurs types de fonctions noyaux (polynomiaux,sigmoïdaux,...), et qu'il est possible d'en construire des plus complexes à partir de fonctions noyaux simples.

IV.3.3 Réseaux connexionnistes

► Historique des réseaux de neurones

Dans les années 40 naquit aux États-Unis la "cybernétique", du grec "kubernetike", terme introduit par Norbert Wiener, qui eut son doctorat à 18 ans à Harvard et son premier poste de mathématique à 25 ans au MIT, *"which has given me the encouragement to work and the freedom to think"*, en opposition avec Harvard. Son livre, "Cybernetics or Control and Communication in the Animal and the Machine", publié en 1948, introduisit le projet de la cybernétique, à savoir l'unification des notions de communication et de contrôle dans les machines, les êtres vivants et les sociétés. Cependant, les travaux de Wiener restaient dans des échelles macroscopiques.



Les premiers travaux de modélisation de la cellule nerveuse élémentaire sont dus à un neuropsychiatre, Warren Mac Culloch, et à un mathématicien, Walter Pitts, qui en 1943 introduisirent les "neurones formels" [Culloch 43]. Suivirent les travaux de Donald Hebb sur les mécanismes d'apprentissage, à savoir l'élaboration de la première règle empirique pour la modification des poids synaptiques [Hebb 49].

Durant les années 50 et 60, le connexionnisme connut, dans le cadre de l'intelligence artificielle, un

essor considérable, avec notamment le premier modèle de réseau de neurones constitué, “SMART”, suivi, en 1965, de “Dendral” pour la chimie et “Prospector” en 1976 pour la géologie. Entre temps, en 1957, le “Perceptron” fut présenté par Franck Rosenblatt [Rosenblatt 62] en vue de comprendre la mémoire humaine, puis l’“adaline”, conçu par Bernards Widrow [Widrow 60], qui reprend le principe des circuits électriques pour l’analyse du signal. Cependant, en 1961, Minsky et Paper mirent en évidence les limitations de ces approches, notamment le caractère générique de la non séparabilité linéaire.

Enfin, depuis les années 80, un retour en force a lieu dans le domaine, notamment grâce aux travaux de John J. Hopfield [Hopfield 82], qui proposa une solution au problème du voyageur de commerce, de Tuevo Kohonen avec ses cartes auto-organisatrices [Kohonen 82], et de Crick et Waltson (prix Nobel pour la découverte de la structure en double hélice de l’ADN), qui utilisèrent des Perceptrons multi-couches dans leurs travaux.

Aujourd’hui, les recherches dans le domaine sont beaucoup plus “bio-inspirés”, et utilisent des modèles de neurones bien plus proches des neurones réels, les *neurones à spike* [Thorpe 96, Gerstner 02], que nous n’aborderons pas ici.

▶ Du neurone biologique au neurone formel

Les neurones , comme nous l’avons vu précédemment, sont des unités cellulaires de relais et d’interprétation de signaux situées dans le système nerveux et cérébral de l’animal. Il en existe approximativement 25 milliards à la naissance chez l’être humain. Ils sont composés d’un corps cellulaire appelé le soma, où se situent le noyau et toute la machinerie cellulaire, et d’extrémités telles que les dendrites et l’axone. Il existe plusieurs types de neurones biologiques (figure II.7, page 20), bien que tous soient composés des mêmes éléments essentiels pour fonctionner.

D’un point de vue fonctionnel, lorsque des facteurs externes tels que des hormones, des sens, *et caetera*, viennent en contact avec les récepteurs situés sur la surface des dendrites du neurone, l’énergie physico-chimique interne du soma, apportée par chaque dendrite, est modifiée, et celle-ci devient active au delà d’un certain seuil. Dans ce cas, l’énergie est évacuée par son axone et elle va exciter ainsi d’autres cellules avec qui elle est connectée (en moyenne, un neurone est connecté à 10^4 neurones voisins).

Le premier modèle (figure IV.25) de neurone fut le neurone de MacCulloch et Pitts [Culloch 43], en 1943. Les entrées venant d’autres neurones ou des entrées du système sont pondérées par $w_1 \dots w_n$, appelés **poids synaptiques** ou **efficacités synaptiques**, et le soma est composé de deux parties : une première partie qui additionne les entrées pondérées, et une seconde partie validant l’excitation du neurone.

Ainsi, pour ce modèle, en considérant un seuil s appelé **seuil de décharge** du neurone,

$$y = (\sum_i w_i x_i > S) \tag{IV.28}$$

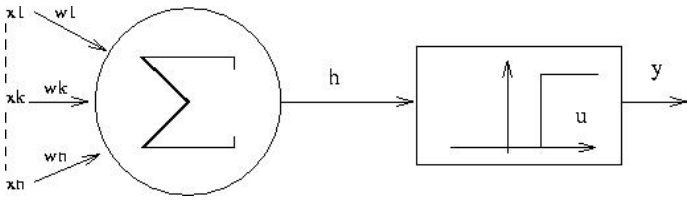


FIG. IV.25 – Modèle de McCulloch et Pitts.

Le modèle de MacCulloch et Pitts peut être généralisé de plusieurs manières :

$$y = f(\sum_i w_i x_i) \tag{IV.29}$$

avec pour f, appelée **fonction de transfert** ou **fonction d'activation** du neurone, plusieurs formes possibles (figure IV.26) : seuil, linéaire (par morceaux), sigmoïdale, gaussienne, *etc.*

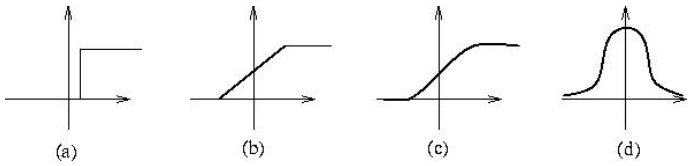


FIG. IV.26 – Différentes fonctions d'activation d'un neurone.

► Le perceptron de Rosenblatt

Le perceptron de Rosenblatt (figure IV.27) a pour origine le fonctionnement de la rétine et de la perception visuelle. L'idée est de baser l'apprentissage sur la correction d'erreur, c'est-à-dire de modifier les poids w_i en fonction des réponses à des tests d'entrée. L'algorithme a été décrit précédemment, figure IV.23.

Un exemple simple est l'exemple du OU (figure IV.28, à gauche). Supposons que l'on veuille faire apprendre la fonction OU entre deux entrées, x_1 et x_2. Cette fonction est modélisable par la structure de gauche. En effet, cette structure, en supposant comme fonction d'activation une fonction seuil S, est modélisable mathématiquement de la façon suivante :

$$y = (w_1.x_1 + w_2.x_2 > S) \tag{IV.30}$$

Ainsi, l'algorithme convergera par exemple vers la droite séparatrice D de la figure IV.28, telle que $w_1 = 1, w_2 = 1$, et $S = 0.5$.

Il en serait de même pour la fonction ET, avec cependant un seuil $S = 1,5$. En revanche, on ne peut modéliser un OU exclusif, ce modèle étant efficace uniquement dans le cas de données linéairement séparables. Il est nécessaire d'améliorer le modèle (avec le perceptron multi-couches).

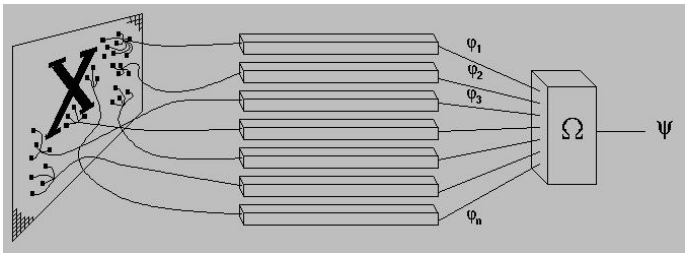


FIG. IV.27 – Perceptron de Rosenblatt.

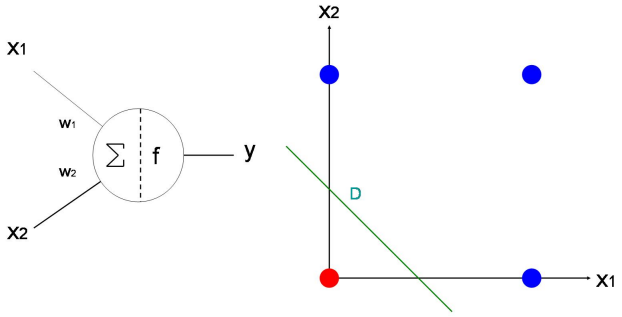


FIG. IV.28 – Perceptron pour le OU.

► L'Adaline de Widrow

Afin de sortir du modèle binaire "tout ou rien", l'Adaline (ADAptative LINear Element) est une modélisation introduite par Widrow dans les années 50, plus simple que le modèle de MacCulloch et Pitts puisque la fonction de transfert se réduit à l'identité (avec seuil) en sortie.

L'Adaline est un classifieur des moindres carrés de marge $\beta > 0$ qui minimise l'erreur quadratique moyenne :

$$\mathcal{E}(w) = \frac{1}{N} \sum_{i=1}^{N} ||y_i - f(x_i, w)||^2 \tag{IV.31}$$

où w est un paramètre d'apprentissage.

Ainsi, alors que le perceptron peut ne pas converger en cas de non séparabilité linéaire des données, l'Adaline convergera toujours vers une solution en général unique, tout en acceptant certaines erreurs de classification.

► Le perceptron multi-couches

L'idée est alors d'introduire une ou plusieurs couches cachées, entre la couche d'entrée du réseau de neurones et la couche de sortie (figure IV.29). Dans ce modèle, chaque neurone d'une couche est relié à tous les neurones des couches voisines.

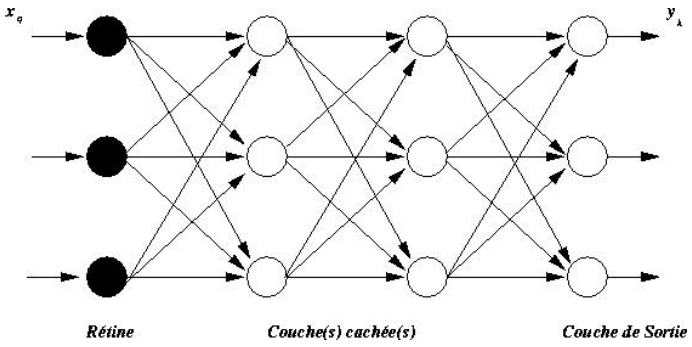


FIG. IV.29 – Perceptron multi-couches.

Les couches cachées permettent de faire des opérations intermédiaires avant d'atteindre l'objectif final. Ainsi, pour le OU exclusif (figure IV.30), il est possible de le modéliser simplement à l'aide d'une couche cachée composée de deux neurones : l'un codant le OU, l'autre le ET, et la sortie doit valider le OU sauf si le ET est actif.

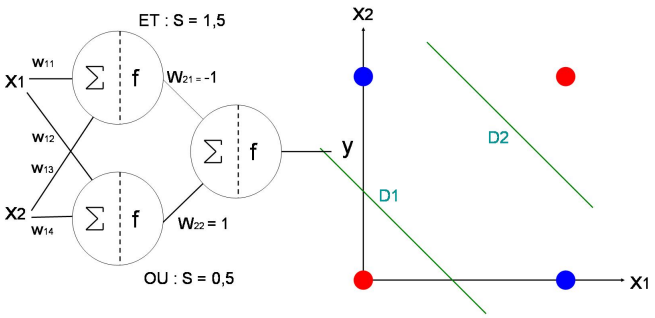


FIG. IV.30 – Perceptron pour le OU exclusif.

Ainsi, $w_{11} = w_{12} = w_{13} = w_{14} = 1$ (droites D_1 et D_2) valident le OU, et $w_{22} = -1$ et $w_{22} = 1$ interdisent le ET. En dehors du neurone du ET, tous les seuils sont à $0,5$.

L'algorithme initial de l'apprentissage du perceptron est un algorithme classique de descente du gradient. Cependant, signalons qu'un algorithme plus efficace consiste à propager

l'erreur d'une couche à l'autre dans le sens inverse de la propagation des activations : l'algorithme de rétropropagation du gradient de l'erreur.

► Le réseau de Hopfield

La renaissance des réseaux de neurones est liée aux travaux de à John J. Hopfield, qui en 1982 proposa un modèle à connexions complètes symétriques : tous les neurones sont connectés les uns les autres, avec le même poids reciproque $w_{ij} = w_{ji}$.

Le réseau de Hopfield est un réseau :
– à N neurones dont l'activité est comprise entre 0 et 1,
– de connexion complète,
– de poids de connexions w_{ij} relatifs à la connexion du neurone N_i au neurone N_j,
– de poids de connexions symétriques : $w_{ij} = w_{ji}$
– et d'auto-connexion nulle : $x_{ii} = 0$.
Ce réseau a permis de modéliser le fonctionnement associatif de la mémoire humaine - et animale-, type Pavlov, tel que exprimé par Donald O. Hebb en 1949. Cette loi peut être modélisée par les équations suivantes :

$$w_{ij}(t+1) = w_{ij}(t) + \partial w_{ij}(t) \quad \text{(IV.32)}$$
$$\partial w_{ij}(t) = x_i(t).x_j(t) \quad \text{(IV.33)}$$

où $\partial w_{ij}(t)$ est la coactivité qui est modélisée comme étant le produit des activités post et présynaptique, et $x_i(t)$ est l'activité du neurone i à l'instant t.

Le modèle de Hopfield est aujourd'hui dépassé. L'hypothèse de symétrie est peu plausible biologiquement et sa capacité est limitée. Cependant, son intérêt est d'introduire des connexions récurrentes qui mènent à un modèle neuronal dynamique.

IV.3.4 Les k plus proches voisins

La classification par k plus proches voisins ($k - ppv$) est très simple à mettre en œuvre : il s'agit de donner une classe à un élément nouveau en fonction de la classe de ses k voisins (il faut donc pour cela qu'une métrique soit définie).

Ainsi, pour la classification $1 - ppv$, la classe du nouvel objet est la classe du plus proche voisin. Il n'y a pas d'ambiguïté possible. Par contre, pour plusieurs voisins, s'ils sont de classe différente, la décision peut suivre plusieurs stratégies :
– affecter la classe de la classe majoritaire chez les k voisins,
– pondérer l'affectation de classe avec la distance : plus un voisin est éloigné, plus son influence est moindre,
– *etc.*

Dans l'exemple suivant (figure IV.31), à gauche, la classification est simple quelque soit le nombre de voisins choisis : le nouvel objet est rouge. À droite, par contre, tout dépend du nombre de voisins choisis et de l'heuristique de classification.

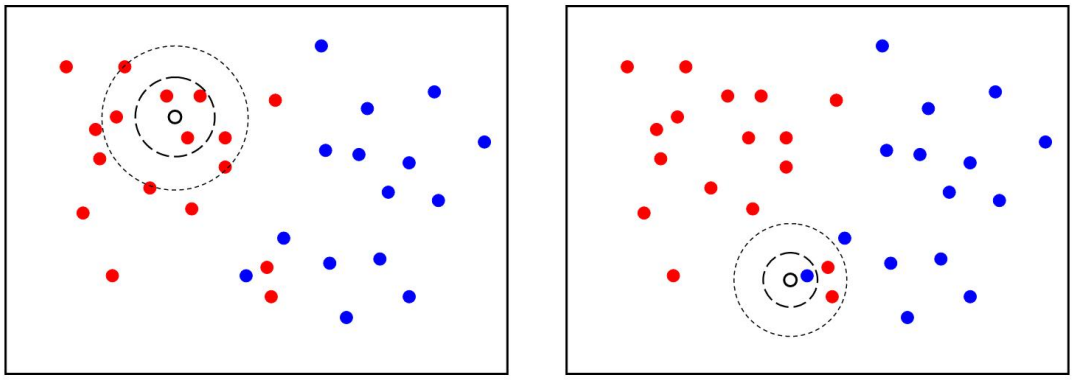


FIG. IV.31 – Classification avec les $k - ppv$. À gauche, aucune ambiguïté possible. À droite, tout dépend du nombre de voisins choisis et de l'heuristique de classification.

Pour $k = 1$, le nouvel objet est bleu. Pour $k = 3$, si les trois voisins ont le même *poids*, alors le nouvel objet est rouge. Si, par contre, le poids est pondéré par l'inverse de la distance, alors le nouvel objet peut être bleu.

IV.3.5 Arbres de décision

Le formalisme des arbres de décision permet de classifier un nouvel objet en testant ses attributs les uns à la suite des autres. La motivation initiale est issue du monde des sciences naturelles pour la classification des êtres vivants.

Sur une population initiale d'exemples, une *question* (ou *sélecteur*) est posée à chaque exemple afin de couper cette population initiale en deux parties. La question porte sur un attribut. Si les attributs sont binaires (figure IV.33), alors la question est : *l'attribut λ est-il présent dans l'exemple ¿ '*. Si les attributs sont continus (figure IV.32), alors la question est *la valeur de l'attribut λ est-elle supérieure à une valeur λ_0 ?"*. Et ainsi de suite jusqu'à ce que tous les exemples initiaux soient décrits entièrement.

La règle de classification est ensuite la suivante :tout à nouvel objet, une succession de questions (tests) lui est soumis, définies à paritr de ses réponses successives (cheminement dans l'arbre).

L'avantage des arbres de décision sont qu'ils sont simples et faciles à mettre en œuvre. De plus, l'approche symbolique permet une classification compréhensible par l'homme. En revanche, ils sont *fragiles* (si une erreur d'aiguillage à lieu pour une raison quelconque, le résultat sera faux), et la question à poser à chaque nœud n'est pas toujours évidente à établir.

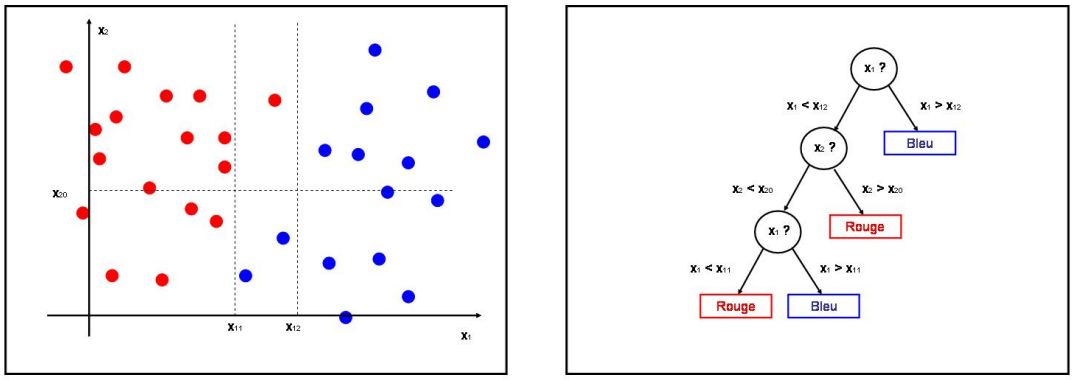


FIG. IV.32 – Classification par arbre de décision de données continues. Ici, les valeurs x_{11}, x_{12} et x_{20} sont fixées par l'expert.

IV.3.6 Hiérarchie de concepts

L'analyse formelle de concepts est un domaine qui déborde sur l'apprentissage symbolique et l'analyse de données. Alors que les arbres de décision hiérarchisent artificiellement les attributs en fonction de l'heuristique choisie (sélection au hasard, sélection qui coupe la classe d'apprentissage en deux parties les plus égales, *etc.*), l'idée est de hiérarchiser les concepts, c'est-à-dire les attributs associés aux objets qui les ont décrit. C'est le cœur de l'analyse formelle de concepts.

Les réseaux de neurones, comme les support vector machine, ne permettent pas un apprentissage symbolique. De plus, il n'y a pas d'hypothèse à formuler sur l'initialisation du support d'apprentissage, comme pour un réseau de neurones, non plus sur la structure de ce réseau. Enfin, le processus ne peut être purement incrémental, et l'apprentissage doit se poursuivre après l'expérimentation.

En conclusion, le formalisme des treillis de Galois (ou treillis de concepts) exposé chapitre suivant permet un apprentissage

– symbolique : il utilise les attributs visuels extraits des images que le robot capture de l'environnement ;
– incrémental : l'apprentissage se fait au fur et à mesure que le robot explore son environnement ;
– souple : une mise à jour du treillis est très simple à intégrer ;
– déterministe : un même treillis est construit à partir d'un même contexte, sans considération de méthodes d'initialisation ;
– et modulaire : on peut utiliser un grand treillis décrivant tout l'environnement, ou bien plusieurs petits treillis locaux ; on introduit ainsi dans le système d'apprentissage la topologie des lieux.

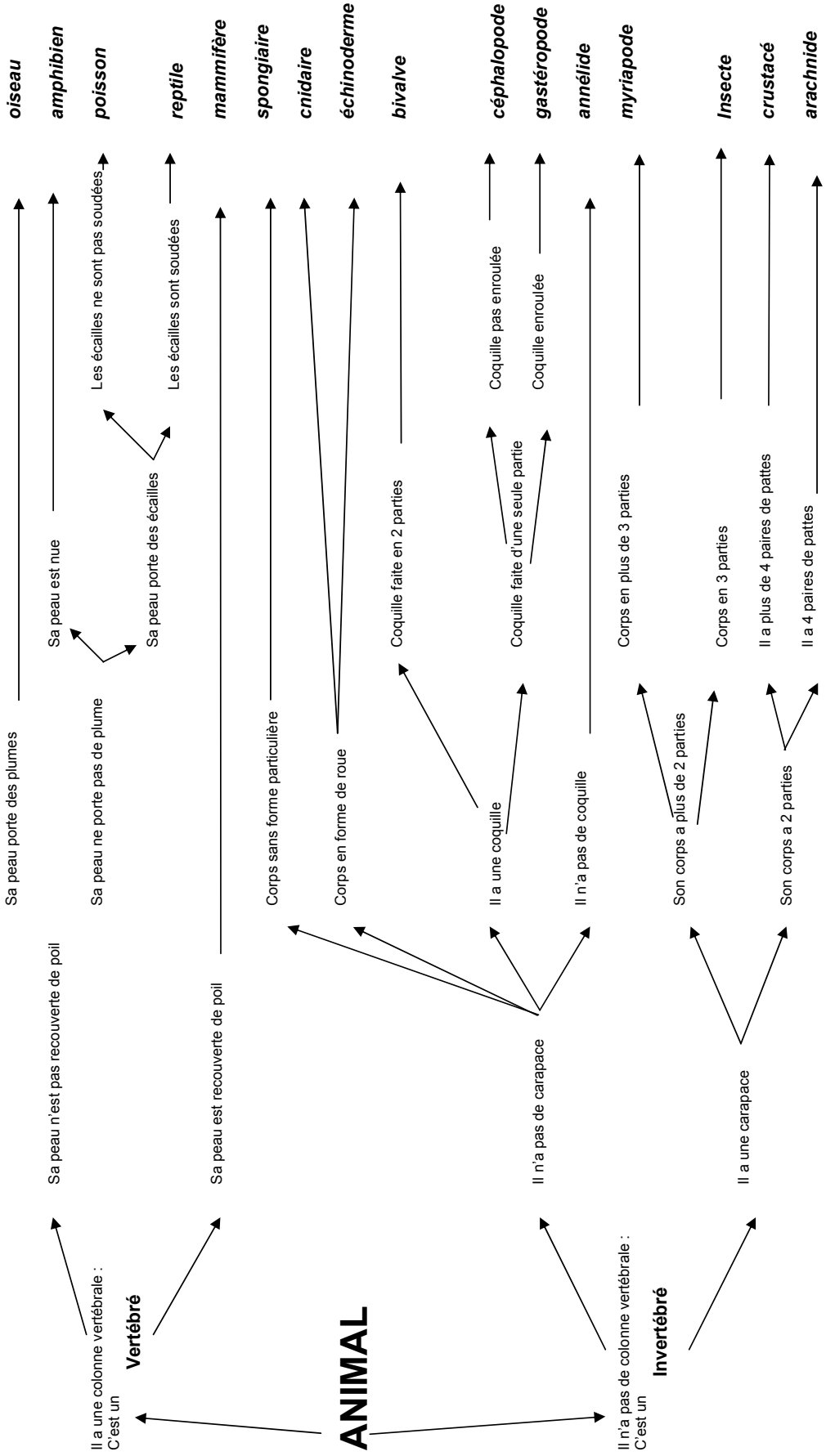


FIG. IV.33 – Classification des Animaux.

Chapitre V

Les Treillis de Galois

Les treillis de Galois ont été largement utilisés en Intelligence Artificielle ces vingt dernières années. Cette théorie a été développée sous le nom d'*Analyse Formelle de Concepts*. Plusieurs algorithmes constructifs ont été mis au point depuis lors [Kuznetsov 01], et certaines applications concrètes sont apparues récemment, notamment en apprentissage artificiel [Liquière 98], fouille de données [Stumme 02], ou dans le domaine aéronautique [Chaudron 03] ou immunologique [Duquenne 03]. Plus anciennement, on peut citer des travaux en biologie [Nguifo 94] ou dans la navigation et la récupération de données [Carpineto 96].

Les complexités des algorithmes de construction de treillis (ou plus précisément de recherche de concepts) étant très importantes, les applications étaient jusqu'à présent plutôt rares et limitées mais les machines actuelles permettent d'envisager des applications de plus grande ampleur.

Ce chapitre est assez formel, à l'image des publications liées aux treillis de Galois. Nous nous sommes attachés cependant, à chaque fois que cela était possible, à donner un exemple simple afin d'illustrer les différentes notions développées dans ce chapitre. L'exemple courant comporte comme objets les planètes de notre système solaire, associées à leurs attributs potentiels ; dans notre application, les objets seront les images capturées par le robot, associées aux attributs présentés précédemment.

Dans ce chapitre, nous présentons donc la formalisation des treillis de Galois, suivie dans

un premier temps par les algorithmes les plus courants de construction de treillis, et dans un deuxième temps des systèmes généraux à base de treillis. Suivent une section relative à la classification par des treillis avec les algorithmes que nous avons développés, une section relative aux aspects probabilistes des treillis que nous avons définis, avant d'exposer une particularité de notre application, les treillis locaux.

V.1 Formalisation

Les notions les plus importantes pour le formalisme des treillis de Galois sont les notions d'**objets**, d'**attributs**, de **contexte**, de **concept** et de **treillis**, que nous allons développer dans ce chapitre. Les deux ouvrages de référence sur le sujet sont, dans l'ordre chronologique, le livre de M. Barbut & B. Monjardet, *Ordre et Classification* [Barbut 70], et plus récemment le livre de B. Ganter & R. Wille, *Formal Concept Analysis* [Ganter 99].

V.1.1 Définitions

Les définitions données ici sont très détaillées afin de permettre à un "non-mathématicien" de comprendre leur sens à partir de peu de connaissances. Un expert pourra aller directement à l'essentiel.

Un exemple courant (figure V.1) illustre ces définitions tout au long du chapitre. Les neuf planètes de notre système solaire possèdent trois fonctions : la taille, la distance par rapport au soleil et la présence de lune autour d'elles.

Les fonctions sont ensuites discrétisées, en général artificiellement par un expert. La discrétisation définit alors les attributs. Dans l'exemple courant, par exemple, la taille a été discrétisée en trois catégories. Il y a en tout sept attributs.

EXEMPLE DES PLANÈTES							
	Taille			**Distance**		**Présence d'une lune**	
	Petite	Moyenne	Grande	Proche	Loin	Lune	Pas de lune
Mercure	√			√			√
Vénus	√			√			√
Terre	√			√		√	
Mars	√			√		√	
Jupiter			√		√	√	
Saturne			√		√	√	
Uranus		√			√	√	
Neptune		√			√	√	
Pluton	√				√	√	

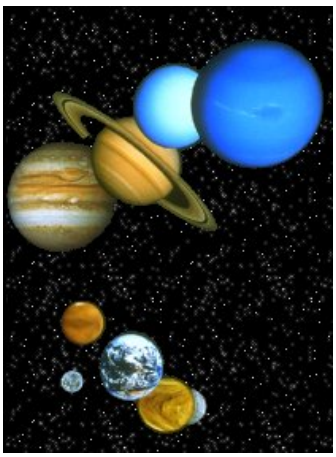
FIG. V.1 – L'exemple des *"planètes"*

Il est à noter que cet exemple est particulier, en ce sens que chaque planète possède un même nombre d'attributs (trois) et que ces attributs ne sont pas indépendants entre eux (une planète ne peut en même temps posséder et ne pas posséder de lune). Cependant, il reste très intéressant -car simple- pour exposer ce formalisme.

► Contexte

Dans le cadre de l'apprentissage à partir d'exemples, un *contexte* est défini par les exemples et leurs propriétés présentes ou non. Dans le langage de l'analyse formelle de concepts, on utilise les termes d'*objets*, d'*attributs* et de *relation binaire* entre les deux ensembles. Ainsi,

Définition 5 *Un* **contexte** $\mathcal{K}$ *est un triplet* $(\mathcal{O}, \mathcal{F}, \zeta)$ *dans lequel* $\mathcal{O}$ *est un ensemble d'*__objets__*,* $\mathcal{F}$ *un ensemble d'*__attributs__*, et* ζ *une* **relation binaire** *entre* $\mathcal{O}$ *et* $\mathcal{F}$*.*



Dans l'exemple des *"planètes"*,
– {Mercure, Vénus, Terre, Mars, Jupiter, Saturne, Uranus, Neptune, Pluton} est l'ensemble des objets $\mathcal{O}$,
– {Petite, Moyenne, Grande, Proche, Loin, Lune, Pas de lune} est l'ensemble des attributs $\mathcal{F}$,
– et la relation binaire ζ traduit la présence ou non d'un attribut pour chaque planète : $\zeta(o, f) = 1 \Leftrightarrow$ l'attribut f est un attribut de la planète o.

Par la suite, il est nécessaire d'introduire la notion de *contexte propre* afin de simplifier les écritures futures dans le cadre de particularités problématiques :

Définition 6 *Soit* $\mathcal{K} = (\mathcal{O}, \mathcal{F}, \zeta)$ *un contexte. Si*
– *il n'y a pas d'objet ayant tous les attributs,*
– *et il n'y a pas d'attribut commun à tous les objets,*
alors le contexte sera défini comme **propre**.

Cette définition servira par la suite, notamment pour la classification d'objets. En effet, si le contexte n'est pas propre, et en particulier si un objet possède tous les attributs, il ne sera pas possible de faire de la classification. Les classes étant définies par l'inclusion d'attributs, aucune classe ne pourrait se définir avec un tel objet.

► Dualité

Définition 7 *Étant donné un contexte* $\mathcal{K} = (\mathcal{O}, \mathcal{F}, \zeta)$*, il est possible de définir deux applications, l'une de* $\mathcal{O} \longrightarrow \mathcal{F}$*, l'autre de* $\mathcal{F} \longrightarrow \mathcal{O}$*, et utilisant la même notation (')*, *par les formulations suivantes :*

$$\forall \mathcal{A} \subset \mathcal{O}, \mathcal{A}' = \{f \in \mathcal{F} \mid \forall o \in \mathcal{A}, \zeta(o, f) = 1\} \tag{V.1}$$
$$\forall \mathcal{B} \subset \mathcal{F}, \mathcal{B}' = \{o \in \mathcal{O} \mid \forall f \in \mathcal{B}, \zeta(o, f) = 1\} \tag{V.2}$$

$\mathcal{A}'$ est appelé **dual** *de $\mathcal{A}$, et $\mathcal{B}'$ est appelé* **dual** *de $\mathcal{B}$.*

Ainsi, le dual $\mathcal{A}'$ d'un ensemble $\mathcal{A}$ d'objets est l'ensemble des attributs présents dans tous les objets de $\mathcal{A}$, et le dual $\mathcal{B}'$ d'un ensemble $\mathcal{B}$ d'attributs est l'nsemble des objets qui possèdent tous les attributs de $\mathcal{B}$.

On a alors les propriétés suivantes :

Propriété 1 *$\mathcal{A}_i \subseteq \mathcal{A}_j \Rightarrow \mathcal{A}'_j \subseteq \mathcal{A}'_i$ et $\mathcal{B}_i \subseteq \mathcal{B}_j \Rightarrow \mathcal{B}'_j \subseteq \mathcal{B}'_i$*

Démonstration 1 *Soit $f \in \mathcal{A}'_j$. On a alors $\forall o \in \mathcal{A}_j, \zeta(o, f) = 1$. Or $\mathcal{A}_i \subseteq \mathcal{A}_j$ donc $\forall o \in \mathcal{A}_i, \zeta(o, f) = 1$. Donc $f \in \mathcal{A}'_i$, donc $\mathcal{A}'_j \subseteq \mathcal{A}'_i$.*

Propriété 2 *$\mathcal{A} \subseteq \mathcal{A}''$ et $\mathcal{B} \subseteq \mathcal{B}''$.*

Démonstration 2 *Soit $o \in \mathcal{A}$. si $\zeta(o, f) = 1$, alors $f \in \mathcal{A}'$. Soit $f \in \mathcal{A}'$. Si $\zeta(o, f) = 1$, alors $o \in \mathcal{A}''$.*

Propriété 3 *$\mathcal{A}' = \mathcal{A}'''$ et $\mathcal{B}' = \mathcal{B}'''$.*

Démonstration 3 *La propriété 2 nous dit que $\mathcal{A}' \subseteq \mathcal{A}'''$. Ensuite, la propriété 1 nous permet d'affirmer que si $\mathcal{A} \subseteq \mathcal{A}''$, alors $\mathcal{A}''' \subseteq \mathcal{A}'$. Donc $\mathcal{A}' = \mathcal{A}'''$.*

Propriété 4 *$\mathcal{A} \subseteq \mathcal{B}' \Leftrightarrow \mathcal{B} \subseteq \mathcal{A}'$.*

Démonstration 4 *Si $\mathcal{A} \subseteq \mathcal{B}'$, alors $\mathcal{B}'' \subseteq \mathcal{A}'$. Or $\mathcal{B} \subseteq \mathcal{B}''$, donc $\mathcal{B} \subseteq \mathcal{A}'$. En tenant le raisonnement dual, on obtient $\mathcal{A} \subseteq \mathcal{B}'$.*

Propriété 5 *$(\mathcal{A}_i \cup \mathcal{A}_j)' = \mathcal{A}'_i \cap \mathcal{A}'_j$.*

Démonstration 5 *$\mathcal{A}_i \subseteq (\mathcal{A}_i \cup \mathcal{A}_j)$ donc $(\mathcal{A}_i \cup \mathcal{A}_j)' \subseteq \mathcal{A}'_i$. De même, $(\mathcal{A}_i \cup \mathcal{A}_j)' \subseteq \mathcal{A}'_j$. Donc $(\mathcal{A}_i \cup \mathcal{A}_j)' \subseteq \mathcal{A}'_i \cap \mathcal{A}'_j$. Posons maintenant $\mathcal{B}_k = \mathcal{A}'_i \cap \mathcal{A}'_j$. On a donc $\mathcal{B}_k \subseteq \mathcal{A}'_i$ et $\mathcal{B}_k \subseteq \mathcal{A}'_j$, et donc $\mathcal{A}''_i \subseteq \mathcal{B}'_k$ et $\mathcal{A}''_j \subseteq \mathcal{B}'_k$. Ensuite, on en déduit que $\mathcal{A}_i \subseteq \mathcal{B}'_k$ et $\mathcal{A}_j \subseteq \mathcal{B}'_k$ et donc $(\mathcal{A}_i \cup \mathcal{A}_j) \subseteq \mathcal{B}'_k$. Pour finir, $\mathcal{B}_k \subseteq \mathcal{B}''_k \subseteq (\mathcal{A}_i \cup \mathcal{A}_j)'$, et donc $(\mathcal{A}'_i \cap \mathcal{A}'_j) \subseteq (\mathcal{A}_i \cup \mathcal{A}_j)'$.*

Prenons comme sous-ensemble de planètes l'ensemble {Mercure, Vénus, Terre, Mars}. Son dual est l'ensemble des attributs communs à ces planètes. Ainsi,

$$\{\text{Mercure, Vénus, Terre, Mars}\}\text{'} = \{\text{Petite, Proche}\}$$

De la même façon,

$$\{\text{Uranus, Neptune, Pluton}\}\text{'} = \{\text{Loin, Lune}\}$$

Réciproquement, prenons comme sous-ensemble d'attributs l'ensemble {Petite, Proche}. Son dual est l'ensemble des planètes possédant ces attributs. Ainsi,

$$\{\text{Petite, Proche}\}' = \{\text{Mercure, Vénus, Terre, Mars}\}$$

Et de la même façon,

$$\{\text{Loin, Lune}\}' = \{\text{Jupiter, Saturne, Uranus, Neptune, Pluton}\}$$

Remarque 2 *Le deuxième exemple montre la non involutivité de la dualité. Il n'y a pas toujours bi-dualité entre deux ensembles donnés. Ceci est à la base de la notion de concept.*

► Concept

Définition 8 *Étant donné un contexte $\mathcal{K} = (\mathcal{O}, \mathcal{F}, \zeta)$ et deux sous-ensembles $\mathcal{A} \subseteq \mathcal{O}$, $\mathcal{B} \subseteq \mathcal{F}$, la paire $\mathcal{C} = (\mathcal{A}, \mathcal{B})$ est appelée* **concept** *si et seulement si $\mathcal{A}' = \mathcal{B}$ et $\mathcal{B}' = \mathcal{A}$.*

Ainsi, un concept est une association *particulière* entre un sous-ensemble d'objets et un sous-ensemble d'attributs. Cette particularité est en fait la bi-dualité telle que définie précédemment : à un ensemble d'objets ($\mathcal{A}$) on associe les attributs ($\mathcal{B}$) communs (au sens de l'intersection) à tous ses éléments, et si tous les objets ayant $\mathcal{B}$ comme attributs communs sont dans l'ensemble $\mathcal{A}$, alors $\mathcal{C} = (\mathcal{A}, \mathcal{B})$ est un concept.

Ainsi, dans l'exemple des *"planètes"*, l'association

$$(\{\text{Mercure, Vénus, Terre, Mars}\}, \{\text{Petite, Proche}\})$$

est un concept, alors que l'association

$$(\{\text{Uranus, Neptune, Pluton}\}, \{\text{Loin, Lune}\})$$

n'est pas un concept.

Définition 9 *Soit $\mathcal{C} = (\mathcal{A}, \mathcal{B})$ un concept quelconque. $\mathcal{A}$ est appelé* **extension** *du concept $\mathcal{C}$ et $\mathcal{B}$* **intension** *(ou* **description***) du concept $\mathcal{C}$.*

Remarque 3 *Un concept peut être défini uniquement par son intension (ou par son extension - bien que peu usité).*

Dans ce cas, le concept se "résume" à un sous-ensemble d'attributs et peut s'écrire de la façon suivante : $\mathcal{C} = (\mathcal{B}', \mathcal{B})$.

Remarque 4 *Un couple quelconque $(\mathcal{B}', \mathcal{B})$ n'est pas toujours un concept, la dualité n'étant définie que dans un seul sens.*

Théorème 2 *Un couple quelconque $(\mathcal{B}', \mathcal{B})$ est un concept si et seulement si $\mathcal{B}'' = \mathcal{B}$.*

Théorème 3 *Tout couple $(\mathcal{B}', \mathcal{B}'')$ ou $(\mathcal{A}'', \mathcal{A}')$ est un concept.*

Les démonstrations sont évidentes. Ce dernier théorème servira par la suite.

▶ Treillis

Pour définir proprement un treillis, nous avons besoin des définitions suivantes (voir par exemple [Dubreil 64]) :

Définition 10 *Un* **ensemble ordonné** *$\mathcal{L}$ est un ensemble pour lequel on a défini une relation d'ordre, c'est-à-dire une relation binaire $\preccurlyeq$:*
- *réflexive : $\forall \mathcal{C} \in \mathcal{L}, \mathcal{C} \preccurlyeq \mathcal{C}$,*
- *transitive : $\mathcal{C}_i \preccurlyeq \mathcal{C}_j$ et $\mathcal{C}_j \preccurlyeq \mathcal{C}_k \Rightarrow \mathcal{C}_i \preccurlyeq \mathcal{C}_k$,*
- *et antisymétrique (ou propre) : $\mathcal{C}_i \preccurlyeq \mathcal{C}_j$ & $\mathcal{C}_j \preccurlyeq \mathcal{C}_i \Rightarrow \mathcal{C}_i = \mathcal{C}_j$.*

Si tous les éléments de $\mathcal{L}$ sont comparables, c'est-à-dire si $\forall(\mathcal{C}_i, \mathcal{C}_j)$, $\mathcal{C}_i \preccurlyeq \mathcal{C}_j$ ou $\mathcal{C}_j \preccurlyeq \mathcal{C}_i$, alors l'ensemble est **totalement ordonné** et on parle alors de **chaîne**. Dans le cas contraire, s'il existe deux éléments $(\mathcal{C}_i, \mathcal{C}_j)$ non comparables (on note en général $\mathcal{C}_i \| \mathcal{C}_j$), alors l'ensemble est **partiellement ordonné**. La figure V.2 illustre ces différents ensembles.

Définition 11 *On appelle* **élément majorant** *(resp.* **élément minorant***) d'un sous-ensemble $\mathcal{A}$ de $\mathcal{L}$ un élément $\hat{\mathcal{C}}_{\mathcal{A}}$ (resp. $\check{\mathcal{C}}_{\mathcal{A}}$) de $\mathcal{L}$ tel que $\forall \mathcal{C} \in \mathcal{A}$, $\mathcal{C} \preccurlyeq \hat{\mathcal{C}}_{\mathcal{A}}$ (resp. $\check{\mathcal{C}}_{\mathcal{A}} \preccurlyeq \mathcal{C}$).*

À partir de ces éléments, il est maintenant possible de définir le treillis :

Définition 12 *Un* **treillis** *est un ensemble partiellement ordonné dans lequel deux éléments quelconques admettent toujours un plus petit majorant (que nous appellerons borne supérieure ou* **BS***) et un plus grand minorant (que nous appellerons borne inférieure ou* **BI***).*

Remarque 5 *Il est possible de définir un* **sup demi-treillis** *(resp.* **inf demi-treillis***) à partir uniquement de la définition de la borne supérieure (resp. borne inférieure).*

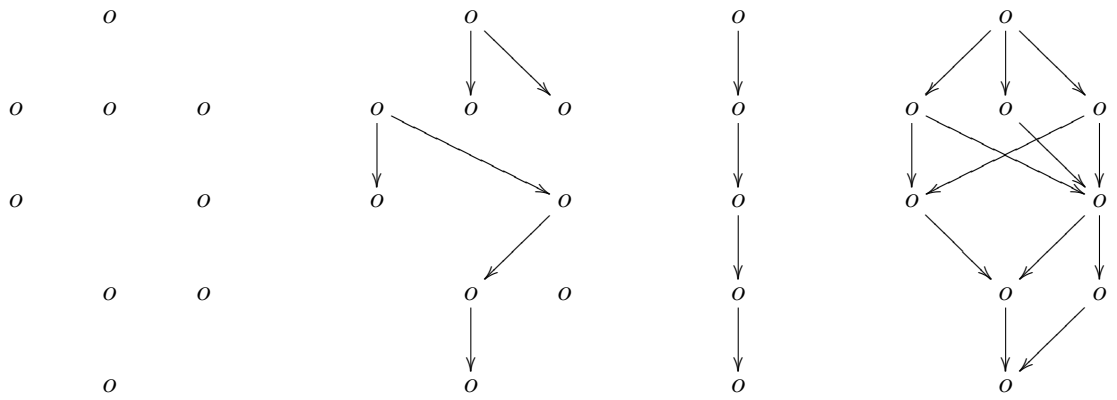


FIG. V.2 – Différents types d'ensembles. À gauche, aucun élément n'est comparable (*anti-chaîne*) ; au centre-gauche, seuls certains éléments sont comparables (*ensemble partiellement ordonné*) ; au centre-droit, tous les éléments sont comparables (*chaîne*) et à droite, un treillis.

L'exemple suivant est proche de notre application. Soit $\mathcal{F}$ un ensemble quelconque, l'ensemble des parties de $\mathcal{F}$: $E = \mathcal{P}(\mathcal{F})$ est ordonné par l'inclusion des ensembles. Dans ce cas, la relation d'ordre est notée $\subseteq$.

► Treillis de Galois

Un treillis de Galois (ou treillis de concepts) est un treillis dont les nœuds correspondent à des concepts, issus d'un contexte donné. Il faut donc définir dans un premier temps une relation d'ordre sur les concepts.

Définition 13 *Les concepts sont hiérarchisés par l'inclusion des intensions :*

$$(\mathcal{A}_i, \mathcal{B}_i) \preccurlyeq (\mathcal{A}_j, \mathcal{B}_j) \Leftrightarrow \mathcal{B}_j \subseteq \mathcal{B}_i \tag{V.3}$$

Ainsi un concept $\mathcal{C}_i$ sera *supérieur* à un concept $\mathcal{C}_j$ si son intension (son ensemble d'attributs) est inclus dans celui de $\mathcal{C}_j$. On peut remarquer également que plus un concept est *élevé*, plus son intension est de cardinalité petite. L'ensemble des concepts est donc partiellement ordonné. À partir de cette définition, il est possible d'identifier la borne supérieure et la borne inférieure de deux concepts.

Théorème 4 *Soit $\mathcal{C}_i = (\mathcal{A}_i, \mathcal{B}_i)$ et $\mathcal{C}_j = (\mathcal{A}_j, \mathcal{B}_j)$ deux concepts quelconques. La borne supérieure (ou* **supremum***) $\hat{\mathcal{C}}_{ij}$ de ($\mathcal{C}_i$,$\mathcal{C}_j$) est un concept et est donné par la formule suivante :*

$$\hat{\mathcal{C}}_{ij} = \mathcal{C}_i \vee \mathcal{C}_j = ((\mathcal{A}_i \cup \mathcal{A}_j)'', \mathcal{B}_i \cap \mathcal{B}_j) \tag{V.4}$$

et la borne inférieure (ou **infimum***)* $\check{\mathcal{C}}_{ij}$ *de (*$\mathcal{C}_i$*,*$\mathcal{C}_j$*) est un concept et est donné par la formule suivante :*

$$\check{\mathcal{C}}_{ij} = \mathcal{C}_i \wedge \mathcal{C}_j = (\mathcal{A}_i \cap \mathcal{A}_j, (\mathcal{B}_i \cup \mathcal{B}_j)'') \tag{V.5}$$

Démonstration 6 *(pour la BS) Soit* $\hat{\mathcal{C}}_{ij} = (\hat{\mathcal{A}}_{ij}, \hat{\mathcal{B}}_{ij})$ *la BS des concepts* $\mathcal{C}_i$ *et* $\mathcal{C}_j$ *considérés. On a donc, par définition,* $\mathcal{C}_i \preccurlyeq \mathcal{C}_{ij}$ *et* $\mathcal{C}_j \preccurlyeq \mathcal{C}_{ij}$*. On en déduit que* $\mathcal{B}_{ij} \subseteq \mathcal{B}_i$ *et* $\mathcal{B}_{ij} \subseteq \mathcal{B}_j$*. Or le plus grand ensemble (la BS est définie comme le plus petit des majorants, donc comme le majorant de plus grande intension) commun à deux ensembles quelconques étant l'intersection de ces derniers, on en déduit que l'intension de la BS des deux concepts est égale à l'intersection des intensions des concepts. Pour terminer, il faut s'assurer que* $\hat{\mathcal{C}}_{ij}$ *est un concept.* $(\mathcal{A}_i \cup \mathcal{A}_j)' = \mathcal{A}_i' \cap \mathcal{A}_j' = \mathcal{B}_i \cap \mathcal{B}_j$*. On peut donc écrire* $\hat{\mathcal{C}}_{ij}$ *sous la forme* $\hat{\mathcal{C}}_{ij} = ((\mathcal{A}_i \cup \mathcal{A}_j)'', (\mathcal{A}_i \cup \mathcal{A}_j)')$*. Donc* $\hat{\mathcal{C}}_{ij}$ *est un concept.*

Remarque 6 *Il est possible de généraliser ces notions à tout ensemble de concepts : soit* $\{\mathcal{C}_i = (\mathcal{A}_i, \mathcal{B}_i)\}_{i \in \mathcal{I}}$ *un ensemble de concepts issus d'un contexte donné, alors*

$$\hat{\mathcal{C}}_i = \bigvee_{i \in \mathcal{I}} \mathcal{C}_i = ((\bigcup_{i \in \mathcal{I}} \mathcal{A}_i)'', \bigcap_{i \in \mathcal{I}} \mathcal{B}_i) \tag{V.6}$$

et

$$\check{\mathcal{C}}_i = \bigwedge_{i \in \mathcal{I}} \mathcal{C}_i = (\bigcap_{i \in \mathcal{I}} \mathcal{A}_i, (\bigcup_{i \in \mathcal{I}} \mathcal{B}_i)'') \tag{V.7}$$

Si l'on considère l'ensemble $\mathcal{L}(\mathcal{K})$ de tous les concepts d'un contexte $\mathcal{K}$ donné, il existe parmi eux un *élément maximal* (*resp.* un *élément minimal*) défini comme le supremum (*resp. infimum*) de l'ensemble des concepts. L'élément maximal est de la forme suivante :

$$(\{\textit{tous les objets}\}, \{\textit{attributs communs à tous les objets}\}) \tag{V.8}$$

et l'élément minimal est de la forme suivante :

$$(\{\textit{objets ayant tous les attributs}\}, \{\textit{tous les attributs}\}) \tag{V.9}$$

Dans le cadre d'un contexte propre, l'élément maximal est de la forme suivante :

$$(\{\textit{tous les objets}\}, \emptyset) \tag{V.10}$$

et l'élément minimal est de la forme suivante :

$$(\emptyset, \{\textit{tous les attributs}\}) \tag{V.11}$$

Définition 14 *Soit* $\mathcal{K}$ *un contexte donné. Le quadruplet* $(\mathcal{L}(\mathcal{K}), \subseteq, \wedge, \vee)$ *(ensemble des concepts ordonnés par l'inclusion des intensions, associés aux opérateurs d'infimum et de supremum) est appelé* **treillis de Galois** *ou* **treillis de concepts***.*

#	Concept
I	({*Mercure, Vénus, Terre, Mars, Jupiter, Saturne, Uranus, Neptune, Pluton*},∅)
II	({*Mercure, Vénus, Terre, Mars, Pluton*},{*Petite*})
III	({*Terre, Mars, Jupiter, Saturne, Uranus, Neptune, Pluton*},{*Lune*})
IV	({*Mercure, Vénus, Terre, Mars*},{*Petite, Proche*})
V	({*Terre, Mars, Pluton*},{*Petite, Lune*})
VI	({*Jupiter, Saturne, Uranus, Neptune, Pluton*},{*Loin, Lune*})
VII	({*Mercure, Vénus*},{*Petite, Proche, Pas de lune*})
$VIII$	({*Terre, Mars*},{*Petite, Proche, Lune*})
IX	({*Pluton*},{*Petite, Loin, Lune*})
X	({*Uranus, Neptune*},{*Moyenne, Loin, Lune*})
XI	({*Jupiter, Saturne*},{*Grande, Loin, Lune*})
XII	(∅,{*Petite, Moyenne, Grande, Proche, Loin, Lune, Pas de lune*})

FIG. V.3 – Concepts du contexte des *"planètes"*

Dans l'exemple des *"planètes"*, on distingue 12 concepts au total (figure V.3), énumérés ici dans l'ordre croissant du cardinal de leur intension.

L'information contenue dans un contexte (issu en général d'une base d'apprentissage) est synthétisée sous forme de concepts. La hiérarchie est intrinsèque aux concepts, par inclusion de leur intension. Il est possible alors de les représenter graphiquement sous forme de treillis classique.

V.1.2 Représentations graphiques

▶ Diagramme de Hasse

Les treillis de Galois sont représentés, comme tout ensemble (partiellement) ordonné, par un **diagramme de Hasse**. Les nœuds sont les concepts du contexte. Le treillis est représenté figure V.4.

▶ Treillis d'héritage

Une autre représentation existe, plus synthétique, car elle évite les redondances d'information dans les concepts. Il s'agit du *treillis d'héritage* [Godin 95], dans lequel les nœuds ne sont plus des concepts, mais des *h-concepts*, qui ne comportent que des éléments nouveaux par

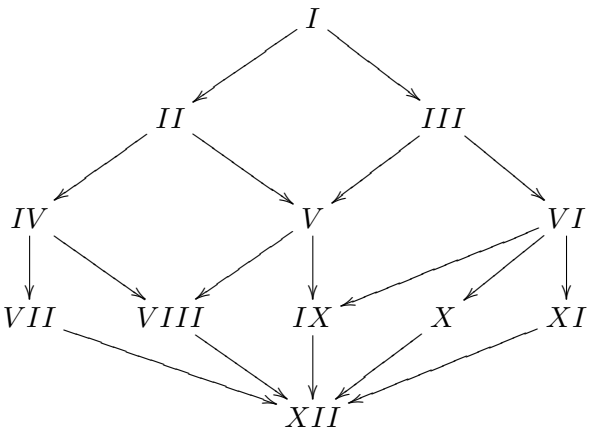


FIG. V.4 – Treillis de Galois relatif au contexte des *"planètes"*. Les nœuds représentent des concepts, le arcs représentent l'inclusion des intentions.

rapport aux ascendants et descendants. Formalisons ces h-concepts.

Définition 15 *Soit $\mathcal{L}$ un treillis de Galois et $\mathcal{C}$ un nœud de ce treillis. Le* **h-concept** $\mathcal{C}^h = (\mathcal{A}^h, \mathcal{B}^h)$ *correspondant au concept $\mathcal{C} = (\mathcal{A}, \mathcal{B})$ est la paire (nouveaux objets, nouveaux attributs), les nouveaux objets (resp. nouveaux attributs) étant définis par rapport aux descendants (resp. ascendants) du concept $\mathcal{C}$. $\mathcal{A}^h$ est alors appelé* **h-extension**, *et $\mathcal{B}^h$* **h-intension**.

Les concepts étant hiérarchisés par l'inclusion des attributs, un nouvel attribut est un attribut présent dans un concepts et non présent dans dans aucun des concepts ascendants.

Remarque 7 *En général, un h-concept n'est pas un concept.*

Définition 16 *Soit $\mathcal{L}$ un treillis de Galois. Le* **treillis d'héritage** $\mathcal{L}^h$ *est le treillis $\mathcal{L}$ pour lequel les concepts ont été remplacés par les h-concepts correspondants.*

Les h-concepts du contexte des *planètes* sont représentés figure V.5, et le treillis d'héritage (figure V.6) a la même forme que le treillis initial. Par exemple, le concept IV a comme intension les attributs $\{$*Petite, Proche*$\}$, or *Petite* est déjà présent dans son ascendant II. Il reste donc *Proche* dans l'intension du h-concept correspondant IV^h. De la même façon, le concept V a comme intension les attributs $\{$*Petite, Lune*$\}$, déjà présents dans les concepts ascendants II et III. Le h-concept correspondant V^h a donc $\emptyset$ comme intension. Le raisonnement est similaire avec les extensions comparées aux concepts descendants.

#	h-Concept
I^h	$(\emptyset,\emptyset)$
II^h	$(\emptyset,\{Petite\})$
III^h	$(\emptyset,\{Lune\})$
IV^h	$(\emptyset,\{Proche\})$
V^h	$(\emptyset,\emptyset)$
VI^h	$(\emptyset,\{Loin\})$
VII^h	$(\{Mercure, Vénus\},\{Pas de lune\})$
$VIII^h$	$(\{Terre, Mars\},\emptyset)$
IX^h	$(\{Pluton\},\emptyset)$
X^h	$(\{Uranus, Neptune\},\{Moyenne\})$
XI^h	$(\{Jupiter, Saturne\},\{Grande\})$
XII^h	$(\emptyset,\emptyset)$

FIG. V.5 – H-Concepts du contexte des *"planètes"*

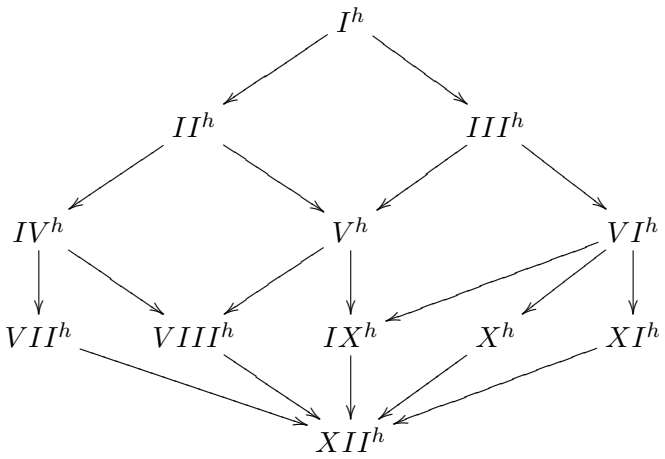


FIG. V.6 – Treillis d'héritage relatif au contexte des *"planètes"*

L'avantage de cette nouvelle représentation est la diminution considérable de la quantité d'information dans le treillis. L'inconvénient principal est qu'elle nécessite de garder le treillis construit, donc la hiérarchie entre concepts, alors que cette hiérarchie était implicite avec le

treillis de Galois.

Le treillis d'héritage n'est *a priori* qu'une représentation graphique. Nous avons ici étendu cette notion et défini, comme treillis d'héritage, un treillis de h-concepts. Ce sont donc de véritables treillis.

V.1.3 Induction de règles

La hiérarchie des concepts permet d'établir des *règles*, sur les attributs, à partir d'une base d'apprentissage. En effet, si un ensemble d'attributs $\mathcal{B}_i$ est inclus dans un deuxième ensemble $\mathcal{B}_j$, alors on peut en déduire que :

$$(\mathcal{B}_i \subseteq \mathcal{B}_j) \Longrightarrow (\mathcal{B}_j \Rightarrow \mathcal{B}_i) \tag{V.12}$$

À partir de cette remarque, il suffit de *"remonter"* le treillis pour en déduire un certain nombre de règles. Ainsi,

$$XII \rightarrow VII \rightarrow IV \rightarrow II \rightarrow I$$
$$XII \rightarrow VIII \rightarrow V \rightarrow III \rightarrow I$$
$$XII \rightarrow XI \rightarrow VI \rightarrow III \rightarrow I$$
$$\dots$$

Le sens de ces implications est plus lisible à partir du treillis d'héritage. Ainsi,

$$\begin{aligned}
(XII^h \rightarrow VII^h \rightarrow IV^h \rightarrow II^h \rightarrow I^h) &\Longrightarrow (\emptyset \Rightarrow \textit{Pas de lune} \Rightarrow \textit{Proche} \Rightarrow \textit{Petite} \Rightarrow \emptyset)\\
(XII^h \rightarrow VIII^h \rightarrow V^h \rightarrow III^h \rightarrow I^h) &\Longrightarrow (\emptyset \Rightarrow \emptyset \Rightarrow \emptyset \Rightarrow \emptyset \Rightarrow \emptyset)\\
(XII^h \rightarrow XI^h \rightarrow VI^h \rightarrow III^h \rightarrow I^h) &\Longrightarrow (\emptyset \Rightarrow \textit{Grande} \Rightarrow \textit{Loin} \Rightarrow \textit{Lune} \Rightarrow \emptyset)\\
&\dots
\end{aligned}$$

En généralisant à partir de notre base d'apprentissage, on peut en conclure que

- si une planète n'a pas de lune, alors elle est proche du soleil ;
- si une planète est proche du soleil, alors elle est petite ;
- si une planète est grande, alors elle est loin du soleil ;
- si une planète est loin du soleil, alors elle possède au moins une lune ;
- *etc.*

Puis, par transitivité,

- si une planète n'a pas de lune, alors elle est petite ;
- si une planète est grande, alors elle possède au moins une lune ;
- *etc.*

On voit qu'il est ainsi possible, à partir d'une base d'apprentissage, de générer des règles valables sur cette base, puis éventuellement généralisables si cette base est jugée *suffisamment* représentative (la traduction en règles peut poser problème en cas de données manquantes).

V.2 Algorithmes de Construction de Treillis

Les algorithmes de construction de treillis permettent de construire, à partir d'un contexte, le treillis de Galois correspondant. Il existe deux types d'algorithmes de construction de treillis de Galois -il s'agit en général d'algorithmes de recherche de concepts : les algorithmes incrémentaux et les algorithmes non incrémentaux.

V.2.1 Algorithmes incrémentaux

Les quatre algorithmes incrémentaux présentés ici (*Norris, Godin, Oosthuizen et Carpineto & Romano*) sont des algorithmes d'efficacité variable, développés en général dans le cadre d'un système plus global à base de treillis (voir paragraphe V.3).

Les algorithmes incrémentaux construisent le treillis au fur et à mesure qu'un nouvel objet arrive (dans notre application, les objects seront les images que le robot capture, ils arriveront donc de façon incrémentale). Ils comprennent donc une phase d'initialisation, suivie, à chaque nouvel objet entrant, d'une phase de mise à jour. Nous reprenons ici ces algorithmes afin de pouvoir les programmer facilement. Pour plus de détails sur chacun d'eux, se reporter à [Kuznetsov 01] et [Njamen 00].

► Algorithme de *Norris*

L'algorithme de Norris (figure V.7) s'initialise simplement par un ensemble vide ($\mathcal{L} = \emptyset$). Il s'est avéré beaucoup plus rapide que les précédents, c'est pour cette raison qu'il a ensuite été choisi pour notre application.

Cette version de l'algorithme de Norris impose la gestion de la liste des objets, ce qui prend du temps et de l'espace mémoire. Aussi proposons-nous par la suite une version modifiée ne gérant que la liste des attributs, comme le proposait l'algorithme de Carpineto & Romano.

► Algorithme de *Godin*

L'algorithme de Godin [Godin 91] est un algorithme basé sur la notion de *nœud générateur*. Un nœud $(\mathcal{A}_i, \mathcal{B}_i)$ est dit générateur pour le nouveau nœud $(\mathcal{A}_j, \mathcal{B}_j)$ si et seulement si $(\mathcal{A}_i, \mathcal{B}_i)$ est une borne supérieure de $\left\{\{(\mathcal{A}_h, \mathcal{B}_h) \in \mathcal{L} \text{ tel que } \mathcal{B}_j = \mathcal{B}_h \cap o'\right\}$ [Njamen 00], avec $\mathcal{L}$ le treillis courant, o le nouvel objet et o' ses attributs. L'algorithme est présenté figure V.8.

► Algorithme de *Oosthuizen*

L'algorithme d'*Oosthuizen* [Oosthuizen 91] s'est révélé légèrement plus rapide que le précédent pour des contextes aléatoires, tout en restant assez simple à programmer. De la même façon que l'algorithme de *Carpineto & Romano*, il ne gère que les attributs des concepts. Il est présenté figure V.9.

Norris()
1. $\mathcal{L} = \emptyset$
2. Pour chaque objet $o \in \mathcal{O}$
2.1. Ajouter$(o, \mathcal{L})$

Ajouter$(o, \mathcal{L})$
1. Pour chaque concept $(\mathcal{A}, \mathcal{B}) \in \mathcal{L}$
1.1. Si $\mathcal{B} \subseteq \{o\}'$
1.1.1. $\mathcal{A} := \mathcal{A} \cup \{o\}$
1.2. Sinon
1.2.1. $\mathcal{D} := \mathcal{B} \cap \{o\}'$
1.2.2. Si $\{\ h \mid (h \in \mathcal{O} \backslash \mathcal{A})$ & (h a déjà été ajouté) & $(\mathcal{D} \subseteq \{o\}')\ \} = \emptyset$
1.2.2.1. $\mathcal{L} := \mathcal{L} \cup \{(\mathcal{A} \cup \{o\}, \mathcal{D})\}$
2. Si $\{\ h \mid (h \in \mathcal{O})$ & (h a déjà été ajouté) & $(\{o\}' \subseteq \{h\}')\ \} = \emptyset$
2.1. $\mathcal{L} := \mathcal{L} \cup (\{o\}, \{o\}')$

FIG. V.7 – Algorithme de *Norris*.

Godin()
1. $\mathcal{L} = \emptyset$
2. Pour chaque objet $o \in \mathcal{O}$
2.1. Ajouter$(o, \mathcal{L})$

Ajouter$(o, \mathcal{L})$
1. $\mathcal{L}^\dagger = \emptyset$
2. Pour tout $(\mathcal{A}_i, \mathcal{B}_i) \in \mathcal{L}$
2.1. Si $(\mathcal{A}_i, \mathcal{B}_i)$ est un nœud générateur
2.1.1. Alors $\mathcal{L}^\dagger := \mathcal{L}^\dagger \cup (\mathcal{A}_i, \mathcal{B}_i) \cup \{\mathcal{A}_i \cup o, \mathcal{B}_i \cap o'\}$
2.2. Sinon
2.2.1. Si $\mathcal{B}_i \not\subseteq o'$
2.2.1.1. Alors $\mathcal{L}^\dagger := \mathcal{L}^\dagger \cup (\mathcal{A}_i, \mathcal{B}_i)$
2.2.2. Sinon
2.2.2.1. $\mathcal{L}^\dagger := \mathcal{L}^\dagger \cup (\mathcal{A}_i \cup o, \mathcal{B}_i)$
3. $\mathcal{L} := \mathcal{L}^\dagger$

FIG. V.8 – Algorithme de *Godin* (91).

► Algorithme de *Carpineto & Romano*

L'algorithme de *Carpineto & Romano* [Carpineto 93] est très simple à programmer. Pour un nouvel objet o, l'ensemble de ses attributs o' est comparé avec les intensions des concepts

Oosthuizen()
1. $\mathcal{L} = \emptyset$
2. Pour chaque objet $o \in \mathcal{O}$
  2.1. Ajouter$(o, \mathcal{L})$

Ajouter$(o, \mathcal{L})$
1. $\mathcal{L}^\dagger = \mathcal{L} \cup o'$
2. Pour chaque concept $\mathcal{B}_i \in \mathcal{L}$
  2.1. Si $o' \cap \mathcal{B}_i \neq \emptyset$
    2.1.1. $\mathcal{L}^\dagger := \mathcal{L}^\dagger \cup (o' \cap \mathcal{B}_i)$
3. $\mathcal{L} = \mathcal{L}^\dagger$

FIG. V.9 – Algorithme d'*Oosthuizen*.

déjà présents dans le treillis. L'intersection entre ces deux ensembles d'attributs génère, si elle n'est pas déjà présente ou incluse dans un des concepts déjà présents, un nouveau nœud (concept). Cet algorithme, tel qu'il est décrit dans [Njamen 00], est présenté figure V.10. Il ne gère que les listes d'attributs, aussi chaque concept est défini uniquement par son intension $\mathcal{B}$.

Carpineto & Romano()
1. Initialiser $\mathcal{L}$ avec $\mathcal{B}_0 = (11 \ldots 1)$
2. Pour chaque objet $o \in \mathcal{O}$
  2.1. Ajouter$(o, \mathcal{L})$

Ajouter$(o, \mathcal{L})$
1. $\mathcal{L}^\dagger = \mathcal{L}$
2. Pour chaque concept $\mathcal{B}_i \in \mathcal{L}$
  2.1. Si $(o' \cap \mathcal{B}_i) \neq \emptyset$
    2.1.1. INSERER := VRAI
    2.1.2. Pour tout $\mathcal{B}_j \succcurlyeq \mathcal{B}_i$
      2.1.2.1. Si $(o' \cap \mathcal{B}_i) \subseteq \mathcal{B}_j$
        2.1.2.1.1. INSERER := FAUX
    2.1.3. si $(o' \cap \mathcal{B}_i) = \mathcal{B}_i$
      2.1.3.1. INSERER := FAUX
    2.1.4. Si INSERER = VRAI
      2.1.4.1. $\mathcal{L}^\dagger := \mathcal{L}^\dagger \cup (o' \cap \mathcal{B}_i)$
3. $\mathcal{L} = \mathcal{L}^\dagger$

FIG. V.10 – Algorithme de *Carpineto & Romano*.

V.2.2 Algorithmes non incrémentaux

Les trois algorithmes non incrémentaux les plus couramment utilisés sont les algorithmes de *Chein*, *Ganter* et *Bordat*. Dans notre application, le processus étant purement incrémental, nous ne nous sommes pas attardés sur ces algorithmes, ne pouvant *a priori* concurrencer les algorithmes incrémentaux, les données étant générées naturellement en flux.

V.2.3 Complexité des algorithmes

Plusieurs travaux se sont attachés à comparer les algorithmes entre eux en termes de complexité algorithmique et de temps de calcul. L'article le plus récent sur le sujet est l'article de Kuznetsov et Obiedkov [Kuznetsov 02]. Cependant, il ne traite pas de tous les algorithmes, c'est pourquoi nous faisons référence à la thèse de P. Njiwoua Njamen [Njamen 00].

Le tableau figure V.11 présente les algorithmes et leurs complexités algorithmiques calculées par [Kuznetsov 02] et [Njamen 00]. Cette complexité dépend essentiellement du nombre d'objets $m = |\mathcal{O}|$, du nombre d'attributs $n = |\mathcal{F}|$, du nombre de concepts dans le treillis $l = |\mathcal{L}|$, de la densité du contexte (nombre de connexions de Galois sur le nombre maximum, à savoir $\frac{\sum_{o,f} \zeta(o,f)}{|\mathcal{O}| \times |\mathcal{F}|}$), et bien entendu de la répartition des attributs. Ce dernier facteur étant difficilement modélisable, la complexité se calcule dans le pire cas.

Algorithme	Incrémental	Complexité [Kuznetsov 02]	Complexité [Njamen 00]
Godin	√	$O(l^4)$	$O(2^{2m}.n)$
Carpineto & Romano	√		$O(3^m.2^n.n)$
Oosthuizen	√		$O(2^n.n^4)$
Norris	√	$O(m^2.n.l)$	
Chein		$O(m^2.n.l)$	$O(2^m.min(m,n)^3.(m+n))$
Ganter		$O(m^2.n.l)$	$O(2^m.n.m^2)$
Bordat		$O(m.n^2.l)$	$O(2^{min(m,n)}.n^3.m^2.min(m,n))$

FIG. V.11 – Complexité des algorithmes incrémentaux de construction de treillis.

Dans notre application, le premier algorithme implémenté a été l'algorithme de Carpineto & Romano, car il était simple à mettre en œuvre. Ensuite, toujours parmi les algorithmes incrémentaux, l'algorithme de Norris s'est avéré plus rapide que l'algorithme de Carpineto & Romano, ce qui se vérifie dans cette comparaison, pour des valeurs de m et n très grandes.

V.3 Systèmes à Base de Treillis de Galois

Ici sont présentés succinctement, dans l'ordre chronologique de leur apparition, quelques systèmes d'apprentissage basés sur des treillis de Galois.

Le système GRAND (GRAph-based iNDuction) [Oosthuizen 88] a été introduit en 1988 par Oosthuizen, et s'adapte à la classification supervisée ou non supervisée. Dans ce système, la classe d'un objet est un attribut comme les autres. L'algorithme utilisé pour ce système est l'algorithme qui porte son nom.

Le système LEGAL (LEarning with GAlois Lattice) [Liquière 90, Nguifo 93], introduit par M. Liquière et E. Mephu Nguifo, se particularise par le fait qu'il ne construit pas entièrement le treillis. La connaissance apprise est représentée par un sup-demi treillis, et utilise l'algorithme de Carpineto & Romano pour la construction du treillis. L'ensemble initial des objets est séparé en deux parties : les exemples (ou exemples positifs) et les contre-exemples (ou exemples négatifs) : on a donc $\mathcal{O} = \mathcal{O}^+ + \mathcal{O}^-$. Considérant une *hypothèse* (ou *régularité*) comme une conjonction d'attributs (intension de concept), elle peut être définie comme *valide* si elle vérifie au moins α exemples positifs, *pseudo-cohérente* si elle rejette au moins β exemples négatifs, et *pertinente* si elle est valide et pseudo-cohérente. Les valeurs de α et β sont définies par l'utilisateur. Seuls les nœuds de régularité valide sont insérés dans le treillis. L'objectif de LEGAL est de déterminer l'ensemble $\mathcal{P}$ des régularités pertinentes qui caractérisent les exemples positifs.

Le Système GALOIS [Carpineto 93] a été introduit par Carpineto & Romano en 1993. L'ensemble des données est séparé en deux parties égales, une première partie permettant la construction du treillis de Galois, la seconde partie permettant de tester le classifieur. L'idée générale est basée sur la similarité entre un nouvel objet à classer et les concepts du treillis une fois construit. Cette similarité repose sur le nombre de concepts dont l'intension est égale ou plus générale (incluse) dans l'ensemble des attributs de l'objet.

RULEARNER [Sahami 95] utilise l'algorithme d'Oosthuizen pour la construction du treillis à partir d'une base d'exemples. Il définit des nœuds positifs si le pourcentage d'exemples négatifs couverts par le nœud est inférieur à un seuil donné.

GRAAL [Liquière 98] étend les treillis de Galois aux objets structurés. Ce système permet ainsi de généraliser des opérateurs pour des graphes étiquetés.

IGLUE [Nguifo 98], enfin, fait suite à LEGAL et combine l'utilisation du treillis et l'algorithme du plus proche voisin. Il utilise une fonction d'entropie pour sélectionner les nœuds pertinents. IGLUE sélectionne ainsi, contrairement à LEGAL, tous les sous-ensembles d'exemples significatifs.

V.4 Treillis de Galois et Classification

Le but de notre approche étant dans un premier temps de faire de la classification supervisée, nous avons enrichi les algorithmes existant afin de leur permettre de classifier lors de la recherche et de la mise à jour des concepts avec les algorithmes incrémentaux.

L'hypothèse de travail est que la base d'apprentissage (ou le contexte) est partitionnée, c'est-à-dire que tout objet o a une et une seule classe $q = q(o)$. Dans notre application, en

effet, toute image (objet) issu de l'environnement appartient à un site (classe) et à un seul. L'ensemble des N classes est noté $\mathcal{Q}$.

Il existe plusieurs manières de généraliser les classifications existantes. En effet, jusqu'à présent, seule une classification binaire à été utilisée (par exemple chez [Sahami 95]) et la généralisation à plusieurs classes, à notre connaissance, n'est jamais abordée.

Étant donné un contexte multi-classes partitionné, nous pouvons définir trois types de classification :

1. la classification *"chacun pour soi"*, pour laquelle on considère toutes les classes,
2. la classification *"seul contre tous"*, pour laquelle la classe considérée est notée positive et toutes les autres négatives,
3. et la classification *"topologique"*, pour laquelle la classe considérée est notée positive et les classes connexes au sens topologique (donc géographique dans notre application) sont notées négatives.

Nous n'avons considéré dans un premier temps que la première classifications, qui permet facilement ensuite de considérer les deux autres. Cependant, la troisième classification sera utile lors de l'introduction de *treillis locaux*, dans la section V.6.

V.4.1 Notion de concept classé

Définition 17 *Soit $\mathcal{K}$ un contexte propre et $\mathcal{L}$ l'ensemble des concepts associés. Un concept $\mathcal{C} \in \mathcal{L}$ est de classe q si et seulement si tous les objets de son extension sont de classe q. Le concept est alors dit* **classé**.

Ainsi, à chaque concept une classe q est associée, en fonction de la classe des objets. Un concept qui n'a pas de classe est de classe nulle : $q(\mathcal{C}) = 0$. Dans l'exemple des *planètes*, si l'on considère deux classes, (*Mercure, Vénus, Terre, Mars*) pour la première, (*Jupiter, Saturne, Uranus, Neptune, Pluton*) pour la seconde, les concepts seront classés en trois catégories : ceux de la classe 1, ceux de la classe 2, et ceux qui non pas de classe (ou de classe 0). Le résultat est donné figure V.12.

Pour les trois algorithmes incrémentaux présentés ci-avant, l'aspect *"multi-classes"* a été introduit afin d'associer incrémentalement, dans chaque algorithme, les concepts à leur classe. Par abus de language, puisque tous les concepts sont représentés par leur intension, nous pourrons appeler "concept" une intension de concept.

V.4.2 Généralités sur les algorithmes de classification

Tous les algorithmes incrémentaux présentés ici ont en commun, à partir d'un nouvel objet o de classe $q(o)$, la comparaison de ses attributs o' avec un nœud courant $\mathcal{B}_i$ de classe $q(\mathcal{B}_i)$. Le nœud créé à partir de cette comparaison est l'intersection des attributs, que l'on notera $\mathcal{D}$: $\mathcal{D} = \mathcal{B}_i \cap o'$, de classe $q(\mathcal{D})$ (figure V.13).

Trois cas sont possibles :

#	Classe	Concept
I	-	($\{$*Mercure, Vénus, Terre, Mars, Jupiter, Saturne, Uranus, Neptune, Pluton*$\}$,$\emptyset$)
II	-	($\{$*Mercure, Vénus, Terre, Mars, Pluton*$\}$,$\{$*Petite*$\}$)
III	-	($\{$*Terre, Mars, Jupiter, Saturne, Uranus, Neptune, Pluton*$\}$,$\{$*Lune*$\}$)
IV	1	($\{$*Mercure, Vénus, Terre, Mars*$\}$,$\{$*Petite, Proche*$\}$)
V	-	($\{$*Terre, Mars, Pluton*$\}$,$\{$*Petite, Lune*$\}$)
VI	2	($\{$*Jupiter, Saturne, Uranus, Neptune, Pluton*$\}$,$\{$*Loin, Lune*$\}$)
VII	1	($\{$*Mercure, Vénus*$\}$,$\{$*Petite, Proche, Pas de lune*$\}$)
$VIII$	1	($\{$*Terre, Mars*$\}$,$\{$*Petite, Proche, Lune*$\}$)
IX	2	($\{$*Pluton*$\}$,$\{$*Petite, Loin, Lune*$\}$)
X	2	($\{$*Uranus, Neptune*$\}$,$\{$*Moyenne, Loin, Lune*$\}$)
XI	2	($\{$*Jupiter, Saturne*$\}$,$\{$*Grande, Loin, Lune*$\}$)
XII	-	($\emptyset$,$\{$*Petite, Moyenne, Grande, Proche, Loin, Lune, Pas de lune*$\}$)

FIG. V.12 – Classe des concepts

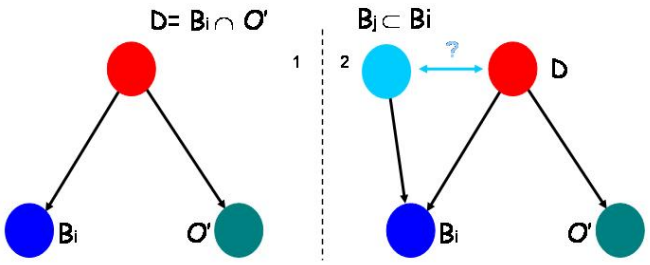


FIG. V.13 – Création d'un nouveau nœud

1. si $q(o) = q(\mathcal{B}_i)$, alors $q(\mathcal{D}) = q(o)$;
2. si $\mathcal{B}_i = \mathcal{B}_0$ (concept initial), alors $q(\mathcal{D}) = q(o)$;
3. sinon, $q(\mathcal{D}) = 0$.

$q = 0$ signifie que le nœud est non classé.

Reste la comparaison, pour la mise à jour de la classe des nœuds du treillis, avec les majorants de $\mathcal{B}_i$. Soit $\mathcal{B}_j$ un majorant ; quatre cas sont possibles (figure V.14) :

1. si $\mathcal{B}_j \subset \mathcal{D} \subset \mathcal{B}_i$, alors
   – si $q(\mathcal{D}) = 0$, alors $q(\mathcal{B}_j) = 0$;
2. si $\mathcal{D} \subset \mathcal{B}_j \subset \mathcal{B}_i$, alors rien ne change ;
3. si $\mathcal{D} = \mathcal{B}_j \subset \mathcal{B}_i$, alors
   – si $q(\mathcal{D}) = 0$, alors $q(\mathcal{B}_j) = 0$;
4. et enfin si $\mathcal{D}$ et $\mathcal{B}_j$ sont incomparables, alors rien ne change.

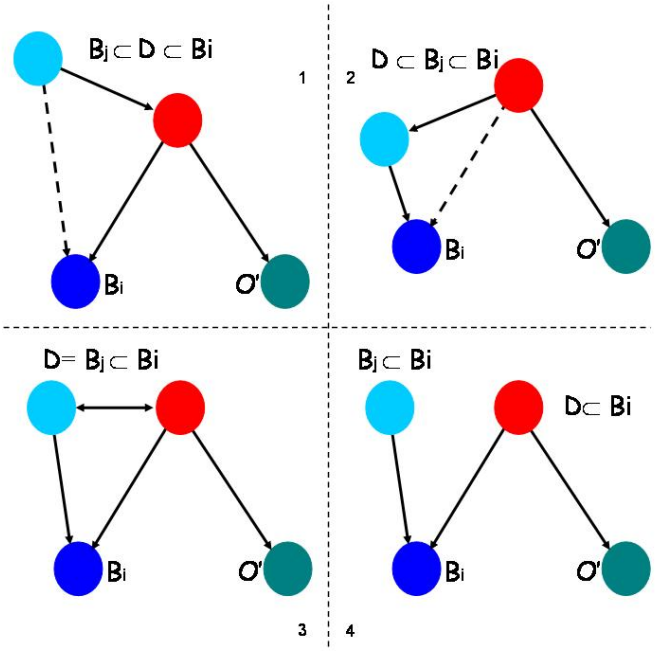


FIG. V.14 – Quatre cas pour la classification.

Ensuite, on doit considérer les deux cas particuliers suivants (figure V.15) : $\mathcal{D} = \mathcal{B}_i$ et $\mathcal{D} = o'$. Dans le premier cas, si $q(o) \neq q(\mathcal{B}_i)$, alors il faut changer la classe de $\mathcal{B}_i$: $q(\mathcal{B}_i) = 0$. Dans le second cas, en revanche, la classe de $\mathcal{B}_i$ ne change pas.

Enfin, si le nouveau nœud crée n'est pas comparable avec un nœud du treillis, alors la classe du nouveau nœud est la classe de l'objet dont il est issu.

V.4.3 Algorithme de *Carpineto & Romano multi-classes*

Les principes précédents ont été intégrés dans l'algorithme de Carpineto & Romano (figure V.16).

On peut remarquer que le premier cas ($\mathcal{B}_j \subset \mathcal{D} \subset \mathcal{B}_i$) n'a pas à être traité dans l'algorithme à chaque étape, il le sera lorsque le nœud $\mathcal{B}_j$ sera le nœud courant $\mathcal{B}_i$.

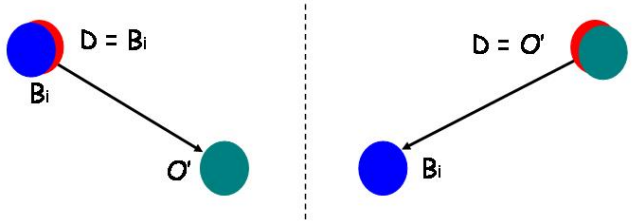


FIG. V.15 – Deux cas particuliers.

Carpineto & Romano()
1. Initialiser $\mathcal{L}$ avec $\mathcal{B}_0 = (11\ldots1)$
2. Pour chaque objet $o \in \mathcal{O}$
  2.1. Ajouter$(o, \mathcal{L})$

Ajouter$(o, \mathcal{L})$
1. $\mathcal{L}^\dagger = \mathcal{L}$
2. Pour chaque concept $\mathcal{B}_i \in \mathcal{L}$
  2.1. Si $\mathcal{D} = (o' \cap \mathcal{B}_i) \neq \emptyset$
    2.1.1. INSERER := VRAI
    2.1.1b. Si $q(o) = q(\mathcal{B}_i)$
      2.1.1b.1. $q(\mathcal{D}) := q(o)$
    2.1.1t. Sinon
      2.1.1t.1. $q(\mathcal{D}) := 0$
    2.1.2. Pour tout $\mathcal{B}_j$ borne sup de $\mathcal{B}_i$
      2.1.2.1. Si $(o' \cap \mathcal{B}_i) \subseteq \mathcal{B}_j$
        2.1.2.1.1. INSERER := FAUX
    2.1.3. si $\mathcal{D} = (o' \cap \mathcal{B}_i) = \mathcal{B}_i$
      2.1.3.1. INSERER := FAUX
      2.1.3.1b. $q(\mathcal{B}_i) = q(\mathcal{D})$
    2.1.4. Si INSERER = VRAI
      2.1.4.1. $\mathcal{L}^\dagger := \mathcal{L}^\dagger \cup (o' \cap \mathcal{B}_i)$
3. $\mathcal{L} = \mathcal{L}^\dagger$

FIG. V.16 – Algorithme de *Carpineto & Romano* multi-classes.

V.4.4 Modification de l'algorithme de *Norris* : Algorithme de *Norris-A*

Afin de minimiser l'espace mémoire et de ne gérer que la liste d'attributs dans l'algorithme de Norris, nous proposons ici une version modifiée de cet algorithme, appelé algorithme de

Norris-A (*A* pour *A*ttributs).

L'initialisation est identique à précédemment, et ce pour toutes les versions proposées ici. Dès à présent, un concept n'est représenté que par son intension, à savoir sa liste d'attributs. $\mathcal{L}$ est alors une liste d'intensions $\mathcal{B}$ hiérarchisées par l'inclusion.

Il est facile de gérer les expressions " $h \mid (h \in \mathcal{O})$", "$h$ a déjà été ajouté" et "$\mathcal{D} \subseteq \{o\}'$" ; gérer $h \in \mathcal{O}\backslash\mathcal{A}$ l'est moins. Nous tenons le raisonnement suivant :

$$h \in \mathcal{A} \quad \Rightarrow \quad \{h\} \subseteq \mathcal{A} \tag{V.13}$$
$$h \in \mathcal{A} \quad \Rightarrow \quad \mathcal{A}' \subseteq \{h\}' \tag{V.14}$$

Or, puisque h est un élément déjà présent et que $\mathcal{A}$ est l'intension d'un concept,

$$h \notin \mathcal{A} \quad \Rightarrow \quad \mathcal{A}' \not\subseteq \{h\}' \tag{V.15}$$
$$h \notin \mathcal{A} \quad \Rightarrow \quad \mathcal{B} \not\subseteq \{h\}' \tag{V.16}$$

On en déduit que :

$$(h \in \mathcal{O}\backslash\mathcal{A}) \quad \Leftrightarrow \quad (h \in \mathcal{O}) \,\&\, (h \notin \mathcal{A}) \tag{V.17}$$
$$(h \in \mathcal{O}\backslash\mathcal{A}) \quad \Leftrightarrow \quad (h \in \mathcal{O}) \,\&\, (\mathcal{B} \not\subseteq \{h\}') \tag{V.18}$$

Ainsi à la ligne 1.2.2., "$(h \in \mathcal{O}\backslash\mathcal{A})$" est remplacé par "$(h \in \mathcal{O})$ & $(\mathcal{B} \not\subseteq \{h\}')$". On en déduit l'algorithme de *Norris-A* (figure V.17), "A" pour "Attributs".

Ajouter$(o, \mathcal{L})$
1. Pour chaque concept $\mathcal{C} = (\mathcal{B}', \mathcal{B}) \in \mathcal{L}$
  1.1. Si $\mathcal{B} \subseteq \{o\}'$
    1.1.1. $-$
  1.2. Sinon
    1.2.1. $\mathcal{D} := \mathcal{B} \cap \{o\}'$
    1.2.2. Si $\{\ h \mid (h \in \mathcal{O})$ & $(\mathcal{B} \not\subseteq \{h\}')$ & (h a déjà été ajouté) & $(\mathcal{D} \subseteq \{o\}')\ \} = \emptyset$
      1.2.2.1. $\mathcal{L} := \mathcal{L} \cup \{(\mathcal{D}', \mathcal{D})\}$
2. Si $\{\ h \mid (h \in \mathcal{O})$ & (h a déjà été ajouté) & $(\{o\}' \subseteq \{h\}')\ \} = \emptyset$
  2.1. $\mathcal{L} := \mathcal{L} \cup (\{o\}, \{o\}')$

FIG. V.17 – Algorithme de Norris-A.

V.4.5 Algorithme de *Norris-A multi-classes*

Quelques remarques -similaires à précédemment- avant d'exposer l'algorithme :

– lorsqu'un concept "intègre" un nouvel objet (ligne 1.1.1.), si ces deux éléments sont de même classe, alors le concept reste de la classe en question. Sinon, il devient classe nulle ;
– de la même façon, l'intersection (ligne 1.2.2.1.) entre un concept et un objet est de classe nulle si les deux éléments sont de classes différentes. Sinon, il est de la classe commune ;
– lorsqu'un concept est créé (section 2.1.), il est de la classe du nouvel objet.

On en déduit l'algorithme de Norris-A-MC (figure V.18).

Ajouter$(o, \mathcal{L})$
1. Pour chaque concept $\mathcal{C} = (\mathcal{B}', \mathcal{B}) \in \mathcal{L}$
  1.1. Si $\mathcal{B} \subseteq \{o\}'$
    1.1.1. $-$
    1.1.2. Si $q(o) \neq q(\mathcal{C})$
      1.1.2.1. $q(\mathcal{C}) = 0$
  1.2. Sinon
    1.2.1. $\mathcal{D} := \mathcal{B} \cap \{o\}'$
    1.2.2. Si $\{\ h \mid (h \in \mathcal{O})\ \&\ (\mathcal{B} \not\subseteq \{h\}')\ \&$ (h a déjà été ajouté) $\&\ (\mathcal{D} \subseteq \{o\}')\ \} = \emptyset$
      1.2.2.1. $\mathcal{L} := \mathcal{L} \cup \{(\mathcal{D}', \mathcal{D})\}$
      1.2.2.2. Si $q(o) \neq q(\mathcal{C})$
        1.2.2.2.1. $q(\mathcal{D}) = 0$
      1.2.2.3. Sinon
        1.2.2.3.1. $q(\mathcal{D}) = q(o)$
2. Si $\{\ h \mid (h \in \mathcal{O})\ \&$ (h a déjà été ajouté) $\&\ (\{o\}' \subseteq \{h\}')\ \} = \emptyset$
  2.1. $\mathcal{L} := \mathcal{L} \cup (\{o\}, \{o\}')$
  2.2. $q((\{o\}, \{o\}')) = q(o)$

FIG. V.18 – Algorithme de Norris-A-MC

V.4.6 Classer à partir d'un treillis d'héritage

Le treillis d'héritage (section V.1.2) permet de diminuer considérablement la quantité d'informations à stocker dans les concepts, en retirant l'information redondante. Cependant, il est nécessaire de garder stockée, d'une façon ou d'une autre, la hiérarchie entre les h-concepts.

Les concepts construits uniquement à partir de redondance d'information formeront des h-concepts vides, et pourtant ils sont tout autant porteurs d'information de par leur hiérarchie et leur position dans le treillis d'héritage. La classe des h-concepts, qui doit être identique à

celle des concepts correspondant, n'est pas évidente à extraire à partir uniquement du treillis d'héritage.

Nous exposons ici une méthode qui permet de déterminer la classe des h-concepts uniquement à partir du treillis d'héritage.

Considérant les h-concepts relatifs à une classe q, c'est-à-dire les h-concepts dont les h-extensions sont des objets de la classe q uniquement. Il faut ensuite sélectionner les ascendants et les descendants de tous ces h-concepts. Enfin, il faut éliminer ceux qui ont au moins un h-concept relatif à un autre site, parmi leurs descendants. Un exemple illustratif est donné figure V.19.

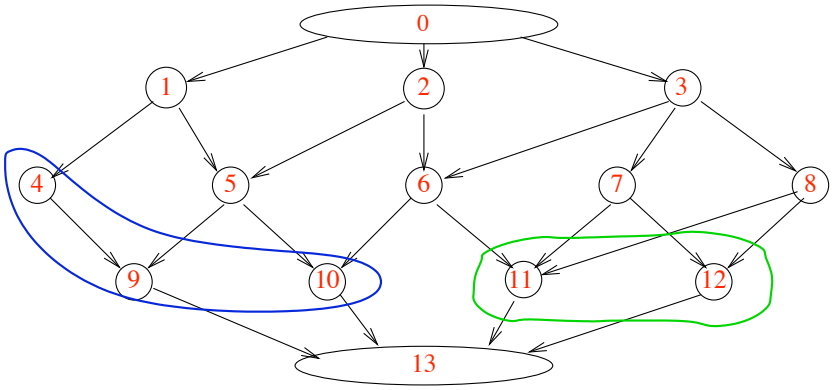


FIG. V.19 – Exemple illustratif de la méthode de classification à partir du treillis d'héritage. Considérons deux classes et les h-conceps relatifs à chaque classe (par exemple, h-concepts n°4, 9 et 10 pour la première, n°11 et 12 pour la seconde). Parmi les ascendants et descendants des h-concepts relatifs à la classe 1, seuls les h-concept n°1, 4, 5, 9 et 10 n'ont pas de descendants dans la classe 2. Ce sont donc des h-concepts de classe 1. De même, parmi les ascendants et descendants des h-concepts relatifs à la classe 2, seuls les h-concept n°7, 8, 11, 12 n'ont pas de descendants dans la classe 2. Ce sont donc des h-concepts de classe 2.

Remarque 8 *Un h-concept peut être classé même si celui-ci est vide.*

V.4.7 Classer à partir d'un *sur-contexte*

L'idée développée par [Oosthuizen 88] est d'introduire la classe comme attribut parmi les autres. Il est donc nécessaire de construire un *sur-contexte*, défini comme l'association du contexte initial $\mathcal{K}$ et du contexte $\mathcal{K}_q$ ayant comme attributs les classes des objets. L'exemple courant est montré figure V.20.

	Taille			Distance		Présence d'une lune		Classe	
	Petite	Moyenne	Grande	Proche	Loin	Lune	Pas de lune	q_1	q_2
Mercure	√			√			√	√	
Vénus	√			√			√	√	
Terre	√			√		√		√	
Mars	√			√		√		√	
Jupiter			√		√	√			√
Saturne			√		√	√			√
Uranus		√			√	√			√
Neptune		√			√	√			√
Pluton	√				√	√			√

FIG. V.20 – Sur-contexte de l'exemple des *"planètes"*

Le treillis correspondant change structurellement par l'introduction de nouveaux nœuds (notamment lorsque deux objets identiques -en termes d'attributs- sont de classes différentes) et par la modification de nœuds existants (intension). Ensuite, il est facile de classer les contextes :

Définition 18 *Un concept $\mathcal{C}$ issu d'un sur-contexte $\mathcal{K} + \mathcal{K}_q$ est de classe q si et seulement si q est présente comme attribut dans l'intension de $\mathcal{C}$:*

$$\mathcal{C} = (\mathcal{A}, \mathcal{B}) \in q \Leftrightarrow \begin{cases} q \in \mathcal{B} \\ \forall q^* \neq q, q^* \notin \mathcal{B} \end{cases} \tag{V.19}$$

avec $\mathcal{K}_q = (\mathcal{O}, \mathcal{Q}, \zeta_q)$ le contexte construit à partir des mêmes objets et de leur classe, ζ_q étant l'application associant à chaque objet sa classe.

Remarque 9 *L'introduction du sur-contexte peut changer le treillis initial, notamment lorsque deux objets identiques (en termes d'attributs) sont de classes différentes.*

Remarque 10 *La notion de sur-contexte est généralisable car chaque attribut est potentiellement créateur d'une classe.*

V.5 Aspects Probabilistes

Jusqu'à présent, l'attribution d'une classe à un concept était binaire, une classe lui étant attribuée ou non. Un concept ne pouvait être de plusieurs classes, mais en revanche pouvait n'être d'aucune classe.

De tels concepts classés sont en pratique très peu nombreux. L'idée que nous avons alors mise en place, pour enrichir l'information sur le concept, est d'y associer non plus une simple appartenance à une classe, mais une **probabilité associée** à chaque classe.

Soit $\mathcal{B} \in \mathcal{F}$ un ensemble d'attributs quelconque et q une classe spécifiée. L'objectif ici est donc, *"voyant"* $\mathcal{B}$, de connaître la probabilité $\pi(q|\mathcal{B})$ d'être dans la classe q. La règle de Bayes nous permet d'écrire :

$$\pi(q|\mathcal{B}) = \frac{\pi(\mathcal{B} \text{ et } q)}{\pi(\mathcal{B})} \tag{V.20}$$

la probabilité étant empirique sur la base d'apprentissage, on en déduit que :

$$\pi(q|\mathcal{B}) = \frac{card(\mathcal{B}' \text{ et } q)}{card(\mathcal{B}')} \tag{V.21}$$

or, dans le cadre d'un concept, les probabilités sont intrinsèquement lisibles dans la cardinalité de l'extension du concept :

$$\pi(q|\mathcal{B}) = \frac{card(\mathcal{A} \text{ et } q)}{card(\mathcal{A})} \tag{V.22}$$

De cette manière, pour tout concept du treillis, il est possible de définir les probabilités d'appartenance de $\mathcal{B}$ à chaque classe du contexte.

Remarque 11 *Si* $\mathcal{C} = (\mathcal{A}, \mathcal{B})$ *est un concept classé dans la classe* $q_{\mathcal{C}}$*, on a* $\pi(q_{\mathcal{C}}|\mathcal{B}) = 1$ *et* $\forall q \neq q_{\mathcal{C}}, \pi(q_{\mathcal{C}}|\mathcal{B}) = 0$.

Par exemple, dans le cadre des *"planètes"*, il est possible pour chaque concept de déterminer les probabilités d'appartenance à chaque classe définie précédemment (figure V.21).

Concept	$\pi(q=1)$	$\pi(q=2)$
I^{π}	-	-
II^{π}	$4/5$	$1/5$
III^{π}	$2/7$	$5/7$
IV^{π}	1	0
V^{π}	2/3	1/3
VI^{π}	0	1
VII^{π}	1	0
$VIII^{\pi}$	1	0
IX^{π}	0	1
X^{π}	0	1
XI^{π}	0	1
XII^{π}	-	-

FIG. V.21 – Probabilité d'Appartenance à chaque Classe.

L'algorithme de Norris nécessite *a priori* la gestion des objets afin de déterminer les probabilités associées à chaque concept. Or l'algorithme de Norris-A utilisé ne gère que les attributs. Nous avons donc développé l'"algorithme de Norris-A probabiliste" (figure V.22) permettant la gestion non pas des objets, mais du nombre d'objets de chaque classe.

Ajouter$(o, \mathcal{L})$
1. Pour chaque concept $\mathcal{C} = (\mathcal{B}', \mathcal{B}) \in \mathcal{L}$
  1.1. Si $\mathcal{B} \subseteq \{o\}'$
    1.1.1. $-$
    1.1.2. Si $q(o) \neq q(\mathcal{C})$
      1.1.2.1. $q(\mathcal{C}) = 0$
    1.1.3. $\mathcal{K}_{\mathcal{C}}(q(o)) = \mathcal{K}_{\mathcal{C}}(q(o)) + 1$
  1.2. Sinon
    1.2.1. $\mathcal{D} := \mathcal{B} \cap \{o\}'$
    1.2.2. Si $\{\ h \mid (h \in \mathcal{O})\ \&\ (\mathcal{B} \not\subseteq \{h\}')\ \&$ (h a déjà été ajouté) $\&\ (\mathcal{D} \subseteq \{o\}')\ \} = \emptyset$
      1.2.2.1. $\mathcal{L} := \mathcal{L} \cup \{(\mathcal{D}', \mathcal{D})\}$
      1.2.2.2. Si $q(o) \neq q(\mathcal{C})$
        1.2.2.2.1. $q(\mathcal{D}) = 0$
      1.2.2.3. Sinon
        1.2.2.3.1. $q(\mathcal{D}) = q(o)$
    1.2.3. $\forall q, \mathcal{K}_{\mathcal{D}}(q) = \mathcal{K}_{\mathcal{C}}(q)$
    1.2.4. $\mathcal{K}_{\mathcal{D}}(q(o)) = 1$
2. Si $\{\ h \mid (h \in \mathcal{O})\ \&$ (h a déjà été ajouté) $\&\ (\{o\}' \subseteq \{h\}')\ \} = \emptyset$
  2.1. $\mathcal{L} := \mathcal{L} \cup (\{o\}, \{o\}')$
  2.2. $q((\{o\}, \{o\}')) = q(o)$
  2.3. $\mathcal{K}_{(\{o\},\{o\}')}(q(o)) = 1$

FIG. V.22 – Algorithme de Norris-A Probabiliste

Ensuite, pour tout concept $\mathcal{C} = (\mathcal{B}', \mathcal{B})$ du treillis, le calcul de la probabilité $\pi(\theta|\mathcal{C})$ pour une classe θ se fait de la façon suivante :

$$\pi(\theta|\mathcal{C}) = \frac{\mathcal{K}_{\mathcal{C}}(\theta)}{\sum_{\theta \in \Theta} \mathcal{K}_{\mathcal{C}}(\theta)} \tag{V.23}$$

V.6 Treillis de Galois Locaux

La taille (nombre de concepts) d'un (seul) treillis, que l'on pourrait qualifier de *"global"* ou de *général*, relatif à un contexte entier, étant exponentielle avec le nombre d'objets, d'attributs et de concepts, l'idée que nous avons développée consiste à décentraliser ce treillis, en introduisant ainsi des **treillis locaux** attachés à chaque classe, et par la suite à chaque nœud

d'un graphe topologique.

De cette manière, on remplace un gros treillis correspondant au contexte global par plusieurs treillis de taille beaucoup plus petite.

V.6.1 Treillis local

Étant donné un contexte multi-classes et partitionné, une première idée est d'attacher à chaque classe un treillis, construit uniquement à partir des objets de la classe considérée. Il est alors possible de formaliser les treillis locaux de la façon suivante.

► Formalisation

Définition 19 *Le* **contexte général** *ou* **contexte global** $\mathcal{K}$ *est le contexte (partitionné) relatif à l'ensemble des objets* $\mathcal{O}$*, l'ensemble des attributs* $\mathcal{F}$*, et l'application* $\zeta : \mathcal{O} \times \mathcal{F} \rightarrow \{0, 1\}$ *; on a alors* $\mathcal{K} = (\mathcal{O}, \mathcal{F}, \zeta)$.

Définition 20 *On appelle* **application locale** ζ_q *l'application* ζ *restreinte aux objets* $\mathcal{O}_q$ *de la classe* q.

Définition 21 *Soit* $\mathcal{K}$ *un contexte multi-classes, on appelle* **contexte local** $\mathcal{K}_q$ *de la classe* q *le contexte* $\mathcal{K}_q = (\mathcal{O}_q, \mathcal{F}, \zeta_q)$.

Remarque 12 *Le contexte étant partitionné en* N *classes, on a alors :*

$$\mathcal{K} = \bigoplus_{q=1...N} \mathcal{K}_q \tag{V.24}$$

où $\bigoplus$ est l'opérateur de somme directe. Une fois le contexte local défini, il est simple de définir les concepts locaux et treillis locaux :

Définition 22 *On appelle* **concept local** $\mathcal{C}_q$ *tout concept issu du contexte local* $\mathcal{K}_q$.

Définition 23 *On appelle* **treillis local** $\mathcal{L}_q$ *le treillis issu du contexte local* $\mathcal{K}_q$.

► Exemple

Cette décentralisation des treillis permet de réduire considérablement la complexité globale de notre algorithme, en créant ainsi N treillis locaux attachés aux N classes de notre contexte partitionné.

Ainsi, dans l'exemple des *planètes*, si l'on considère les deux mêmes classes que précédemment, (*Mercure, Vénus, Terre, Mars*) pour la première, (*Jupiter, Saturne, Uranus, Nepture, Pluton*) pour la seconde, il est possible d'en déduire les contextes locaux $\mathcal{K}_1$ et $\mathcal{K}_2$ (figures V.23 et V.24) puis d'en déduire, pour chacun d'entre eux, les concepts associés (figures V.25 et V.26), et les treillis associés (figure V.27).

	Taille			**Distance**		**Présence d'une lune**	
	Petite	Moyenne	Grande	Proche	Loin	Lune	Pas de lune
Mercure	√			√			√
Vénus	√			√			√
Terre	√			√		√	
Mars	√			√		√	

FIG. V.23 – Contexte local relatif à la classe 1 : $\mathcal{K}_1$

	Taille			**Distance**		**Présence d'une lune**	
	Petite	Moyenne	Grande	Proche	Loin	Lune	Pas de lune
Jupiter			√		√	√	
Saturne			√		√	√	
Uranus		√			√	√	
Neptune		√			√	√	
Pluton	√				√	√	

FIG. V.24 – Contexte local relatif à la classe 2 : $\mathcal{K}_2$

#	**Concept**
I_1	({*Mercure, Vénus, Terre, Mars*},{*Petite, Proche*})
II_1	({*Mercure, Vénus*},{*Petite, Proche, Pas de lune*})
III_1	({*Terre, Mars*},{*Petite, Proche, Lune*})
IV_1	($\emptyset$,{*Petite, Moyenne, Grande, Proche, Loin, Lune, Pas de lune*})

FIG. V.25 – Concepts locaux relatifs à la classe 1

► Extension

Cependant, comme nous le verrons par la suite, les contextes locaux vont être modifiés par la sélection préalable d'attributs. Nous définissons donc :

#	Concept
I_2	({*Jupiter, Saturne, Uranus, Neptune, Pluton*},{*Loin, Lune*})
II_2	({*Pluton*},{*Petite, Loin, Lune*})
III_2	({*Uranus, Neptune*},{*Moyenne, Loin, Lune*})
IV_2	({*Jupiter, Saturne*},{*Grande, Loin, Lune*})
V_2	($\emptyset$,{*Petite, Moyenne, Grande, Proche, Loin, Lune, Pas de lune*})

FIG. V.26 – Concepts locaux relatifs à la classe 2

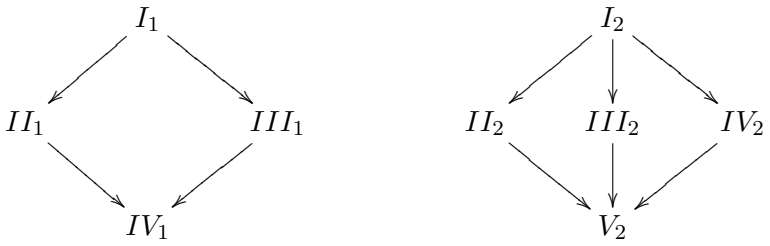


FIG. V.27 – Treillis de Galois relatifs à chaque classe : $\mathcal{L}_1$ à gauche, $\mathcal{L}_2$ à droite.

Définition 24 *Soit $\mathcal{K}_q$ un contexte local et ζ_q une modification de l'application locale ζ_q. On appelle* **contexte local relatif** *le contexte $\mathcal{K}_q^{'} = (\mathcal{Q}_q, \mathcal{F}, \zeta_q^{'})$.*

De la même façon que précédemment, il est possible de définir les concepts locaux relatifs et treillis locaux relatifs :

Définition 25 *On appelle* **concept local relatif** *$\mathcal{C}_q^{'}$ tout concept issu du contexte local relatif $\mathcal{K}_q^{'}$.*

Définition 26 *On appelle* **treillis local relatif** *$\mathcal{L}_q^{'}$ le treillis local issu du contexte local relatif $\mathcal{K}_q^{'}$.*

V.6.2 Treillis local de voisinage

Jusqu'à présent, nous avons défini des contextes locaux uniquement basés sur les classes elles-mêmes. On peut alors appeler de tels treillis des *treillis locaux d'ordre 0*. Or, dans un contexte topologique, où il existe des connexités entre les classes, il est possible de définir des *treillis locaux d'ordre N* construits à partir du contexte d'un nœud donné et des contextes des nœuds voisins (voisinage défini topologiquement à l'ordre N).

Ainsi, un treillis local d'ordre 1 prendra en considération la classe q et ses voisins immédiats, un treillis local d'ordre 2 prendra en compte la classe q et ses voisins à un ordre 2, et ainsi de suite.

► Formalisation

Définition 27 *On appelle* **application locale de voisinage à l'ordre** V $\zeta_{q+\mathcal{V}^V}$ *l'application* ζ *restreinte aux objets* $\mathcal{O}_{q+\mathcal{V}^V}$ *de la classe* q *et de ses voisins* $\mathcal{V}^V$ *à l'ordre* V.

Définition 28 *Soit* $\mathcal{K}$ *un contexte multi-classes, on appelle* **contexte local de voisinage à l'ordre** V $\mathcal{K}_{q+\mathcal{V}^V}$ *de la classe* q *le contexte* $\mathcal{K}_{q+\mathcal{V}^V} = (\mathcal{O}_{q+\mathcal{V}^V}, \mathcal{F}, \zeta_{q+\mathcal{V}^V})$.

Définition 29 *On appelle* **concept local** $\mathcal{C}_{q+\mathcal{V}^V}$ *tout concept issu du contexte local* $\mathcal{K}_{q+\mathcal{V}^V}$.

Définition 30 *On appelle* **treillis local** $\mathcal{L}_{q+\mathcal{V}^V}$ *le treillis issu du contexte local* $\mathcal{K}_{q+\mathcal{V}^V}$.

► Exemple

Reprenons l'exemple des *planètes* pour lesquelles on définit cinq classes, associées à une topologie comme le montre la figure V.28.

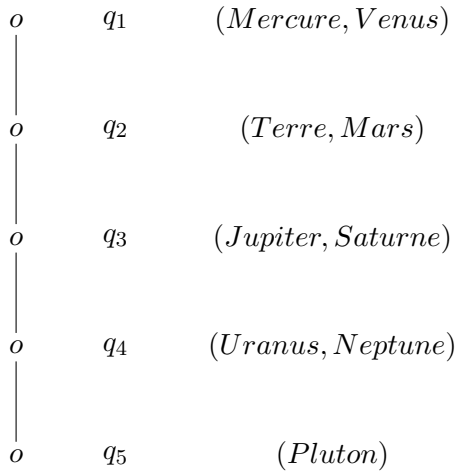


FIG. V.28 – Les planètes, auxquelles sont associées cinq classes et une topologie

Dans ce cas, on peut définir le contexte local de voisinage d'ordre 1 associé à la classe q_2 le contexte restreint aux planètes Mercure, Vénus, Terre, Mars, Jupiter et Saturne. Pour le contexte local de voisinage d'ordre 2 associé à la classe q_2 on ajoutera la classe q_4, c'est-à-dire les planètes Uranus et Neptune.

V.7 Conclusion

Dans ce chapitre, nous avons exposé dans un premier temps le formalisme mathématique des treillis de Galois, les algorithmes incrémentaux de construction, et les systèmes à base de treillis. Dans un deuxième temps, nous avons introduit la notion de classification au sein des treillis, notamment différents "outils" (algorithmes multi-classes, treillis d'héritage, treillis locaux...) basés sur les treillis de Galois pour la classification supervisée (éventuellement topologique).

La formalisation de tels outils est assez succincte, et demanderait à être fouillée de façon plus approfondie. Dans le cadre de cette thèse, en vue d'une application robotique, nous nous sommes attachés à développer et implémenter ces outils expérimentalement.

Le prochain chapitre permet d'établir le lien entre les concepts d'un treillis et les amers visuels relatifs à un environnement donné. Ils présente la façon dont ces outils sont utilisés pour la localisation du robot mobile autonome.

Chapitre VI

Localisation Topologique, Amers Visuels et Treillis de Galois

La localisation dans un environnement plus ou moins structuré, comme nous l'avons vu, se fait soit de façon métrique, par rapport à des murs, des portes, *etc.*, soit de façon topologique, par rapport à des amers (visuels). Les deux approches sont complémentaires. Cependant nous ne nous intéressons ici qu'à l'aspect topologique de cette localisation.

Ce chapitre fait le lien, dans cette optique, entre la robotique, les amers visuels et les treillis de Galois. On y définit les amers à partir des concepts, puis on expose les stratégies mises en œuvre pour permettre au robot de se localiser dans son environnement.

On distingue ici trois types de stratégies : la stratégie d'apprentissage, qui consiste à sélectionner les amers visuels, la stratégie de test, qui consiste à rechercher dans une nouvelle image ces amers visuels pour permettre au robot de se localiser, et la stratégie de décision, qui consiste à prendre une décision sur la localisation en fonction des amers trouvés. Cette stratégie de décision peut s'accompagner ou non d'une mise à jour du treillis, peut s'inscrire ou non dans un processus de vision active. Chaque cas est envisagé.

Ensuite, quatre différentes techniques de localisation sont exposées dans ce chapitre. La première technique consiste à construire un treillis général à partir de toutes les images appartenant aux différents sites de l'environnement, capturées lors de la phase de test, et de leurs attributs respectifs. Les amers visuels sont alors extraits du treillis général. Or, dans la pratique, lors de la phase de test, peu d'amers sont trouvés par le robot. Aussi avons-nous enrichi la

notion d'amers à l'aide de probabilités associées à chaque site.

Ces deux premières techniques ont pour inconvenient de construire un seul treillis pour tout l'environnement. La troisième technique consiste donc à décentraliser notre approche et à construire des treillis locaux, attachés à chaque site de l'environnement. Une quatrième technique, enfin, est abordée ici : il s'agit d'introduire dans les treillis un aspect markovien dans la localisation.

VI.1 De la Robotique aux Treillis de Galois

L'environnement est supposé structuré, à savoir qu'il existe des sites et des passages entre ces sites. Les sites sont en général des salles mais pas nécessairement.

D'un point de vue topologique, il y a plusieurs manières de modéliser l'environnement. Nous nous tenons ici à une modélisation classique, à savoir qu'un site est associé à un nœud, et un passage entre deux sites est associé à un arc du graphe topologique. Cette modélisation permet de modéliser les salles simplement par un nœud. De plus, les arcs seront à terme *facilement* détectables à l'aide de capteurs de proximité.

Le robot capture, lors de la phase d'apprentissage, des images issues de chaque site. Ainsi à chaque site est associé un grand nombre d'images. Chaque image étant analysée selon la procédure décrite dans la section II.3, il lui est associé un certain nombre d'attributs, et l'information peut être structurée comme le montre la figure II.40 (page 53) du deuxième chapitre. Les images correspondent donc à des objets pour les treillis de Galois, et les attributs de l'image correspondent aux attributs du contexte.

L'ensemble des images issues d'un même site sont regroupées dans une classe d'image. Une classe correspond donc à un site, et donc à un nœud du graphe topologique.

Ainsi, la mise en correspondance entre le monde de l'image et le monde des treillis de Galois se fait de la façon suivante :

Monde de la robotique	$\leftrightarrow$	Monde des treillis de Galois
Image	$\leftrightarrow$	Objet
Attribut d'une image	$\leftrightarrow$	Attribut
Table de correspondance	$\leftrightarrow$	Application ζ
Images, attributs et table	$\leftrightarrow$	Contexte
Site	$\leftrightarrow$	Classe

Dans la suite de ce mémoire, à chaque terme d'*image* on pourra entendre *objet*, et réciproquement ; à chaque terme de *site* on pourra entendre *classe*, et réciproquement ; *etc.*

VI.2 Amers Visuels et Concepts

Nous prenons ici comme hypothèse que le contexte est propre et partitionné en sites ou classes.

VI.2.1 Définitions d'un amer

Comme nous l'avons cité précédemment, un amer est *"quelque chose qui permet de se repérer dans un environnement"*. Pour se repérer, il est nécessaire de *voir* quelque chose qui soit propre à chaque site. Supposons que la phase d'apprentissage soit terminée. Le robot dispose donc d'un contexte propre (aucune image ne possède toutes les propriétés) et partitionné en classes. Supposons également que le treillis de Galois correspondant soit construit. Les amers visuels seront définis de la manière suivante :

Définition 31 *Un amer visuel d'un site q est une combinaison de caractéristiques présente dans certaines images du site q et absente de toute autre image issue d'un autre site $q^* \neq q$.*

Cette définition nous permet d'introduire le lien entre un amer visuel et un concept issu du treillis précédemment construit :

Définition 32 *Soit un contexte $\mathcal{K}$, et un sous-ensemble quelconque d'objets $\mathcal{A} \subset \mathcal{O}$. Une combinaison d'attributs $\mathcal{B} \subset \mathcal{F}$ est un* **amer** *de $\mathcal{A}$ si et seulement si :*
- *$\mathcal{B}'' = \mathcal{B}$,*
- *et $\mathcal{B}' \subseteq \mathcal{A}$.*

La première condition $\mathcal{B}'' = \mathcal{B}$ nous assure, au travers des concepts, de rechercher les combinaisons optimales (et minimales) d'attributs et de minimiser ainsi le nombre de combinaisons à rechercher dans les images. Un amer est donc l'intension d'un concept particulier. La seconde condition nous assure que ces combinaisons appartiennent bien (et uniquement) à certaines images du site concerné.

Ainsi, dans l'exemple des *planètes*, si l'on considère les deux même sites que dans le chapitre précédent, $\{$*Mercure, Vénus, Terre, Mars*$\}$ pour le premier, $\{$*Jupiter, Saturne, Uranus, Neptune, Pluton*$\}$ pour le second, il est possible d'en déduire les amers de chacun d'entre eux, figure VI.1.

Théorème 5 *Soit $\mathcal{K}$ un contexte propre et partitionné. Un amer de la classe q est l'intension d'un concept de classe q.*

Démonstration 7 *Soit $\mathcal{B}_q$ un amer de la classe q. Alors $\mathcal{B}_q'' = \mathcal{B}_q$ donc $\mathcal{B}_q$ est l'intension d'un concept. De plus, $\mathcal{B}_q' \subset q$ donc le concept est de classe q. En sens inverse, si $\mathcal{B}_q$ est l'intension d'un concept de classe q, alors $\mathcal{B}_q'' = \mathcal{B}_q$ et puisque tous les objets du concept sont de classe q, alors $\mathcal{B}_q$ est un amer de la classe q.*

Site	Amers	Concept associé
1	*Petite* et *Proche*	IV
	Petite et *Proche* et *Pas de lune*	VII
	Petite et *Proche* et *Lune*	$VIII$
2	*Loin* et *Lune*	VI
	Petite et *Loin* et *Lune*	IX
	Moyenne et *Loin* et *Lune*	X
	Grande et *Loin* et *Lune*	XI

FIG. VI.1 – Amers de chaque site.

VI.2.2 Couverture

La notion de couverture est relativement intuitive. Elle permet de savoir si un ensemble d'amers permet de reconnaître toutes les images d'un site.

Définition 33 *Soit $\{\mathcal{B}_{q,i}\}_{i=1...N_q}$ les N_q amers d'un site q. Le site est* **couvert** *par ses amers si et seulement si*

$$\bigcup_{i=1...N_q} \mathcal{B}'_{q,i} = q \tag{VI.1}$$

Ainsi, toujours avec le même exemple, l'amer $\{$*Petite* et *Proche*$\}$ couvre le premier site, ainsi que l'association des amers $\{$*Petite* et *Proche* et *Pas de lune*$\}$ et $\{$*Petite* et *Proche* et *Lune*$\}$.

VI.2.3 Amer plein

Un amer plein d'un site s est un amer qui permet de couvrir un site à lui tout seul.

Définition 34 *Soit $\mathcal{B}_q$ un amer du site q. $\mathcal{B}_q$ est un* **amer plein** *si et seulement si $\mathcal{B}'_q = q$.*

Il existe deux amers pleins dans notre exemple, un pour chaque site. En effet, puisque $\{Petite, Proche\}' = \{Mercure, Venus, Terre, Mars\}$, alors $\{$*Petite* et *Proche*$\}$ est un amer plein du premier site. En tenant le même raisonnement pour le second site, puisque $\{Loin, Lune\}' = \{Jupiter, Saturne, Uranus, Neptune, Pluton\}$ alors $\{$*Loin* et *Lune*$\}$ est un amer plein de ce site.

Remarque 13 *Dans le cas où il existe un amer plein dans un site q, la couverture de ce site est évidente.*

VI.2.4 Amer maximal

Lorsque deux concepts sont des amers d'une même classe, et que ceux-ci sont comparables, alors il est utile de ne conserver que le concept le plus général, celui-ci étant impliqué par l'autre. En effet, soient $\mathcal{B}_q^1$ et $\mathcal{B}_q^2$ deux amers d'un ensemble d'images $\mathcal{A}$,

$$\begin{aligned}(\mathcal{B}_q^1 \succcurlyeq \mathcal{B}_q^2) &\implies (\mathcal{B}_q^2 \Rightarrow \mathcal{B}_q^1) \\ &\implies \mathcal{B}_q^1 \text{ sera présent quelque soit } \mathcal{B}_q^1 \text{ ou } \mathcal{B}_q^2 \\ &\implies \mathcal{B}_q^1 \text{ est plus général que } \mathcal{B}_q^2\end{aligned}$$

On peut ainsi établir la définition suivante :

Définition 35 *Un* **amer maximal** *est un amer d'intension minimale.*

Propriété 6 *Les amers pleins sont des amers maximaux. La réciproque est fausse.*

VI.3 Recherche d'un Amer dans une Image

Afin de localiser une image, c'est-à-dire de connaître le site dont elle est issue, il est nécessaire de rechercher les amers présents dans cette image.

Un amer est une combinaison d'attributs. Si cette combinaison est incluse dans les attributs de l'image, alors l'amer est présent dans l'image. De façon formelle, on écrit :

Définition 36 *Soit $\mathcal{I}$ une image et $\mathcal{B}_\mathcal{I}$ les attributs présents dans l'image. Soit $\mathcal{B}_q$ un amer du site q. L'amer $\mathcal{B}_q$ est présent dans l'image $\mathcal{I}$ si et seulement si $\mathcal{B}_q \subseteq \mathcal{B}_\mathcal{I}$.*

Ainsi la présence d'un amer dans une image se définit-elle par l'inclusion des attributs.

VI.4 Phases et Stratégies Associées de Localisation du Robot à l'aide d'Amers Visuels

Dans notre application, il existe trois phases de fonctionnement, et pour chacune d'elles une ou plusieurs stratégies bien définies :

1. la première phase est la phase d'apprentissage. La stratégie correspondante consiste à extraire les caractéristiques de chaque image, à construire le ou les treillis de Galois et à extraire les amers (globaux ou locaux) visuels ;
2. la deuxième phase est la phase de test, qui consiste à localiser l'image en cherchant des amers ;
3. la troisième phase consiste à prendre une décision sur la localisation, soit en fonction d'une simple image, soit en fonction d'un flux d'images.

De plus, d'autres stratégies peuvent intervenir, et nous en présentons deux ici : la première consiste à mettre à jour le treillis lors de la phase de test, et la seconde, propre à notre application et à la robotique mobile autonome, consiste à intégrer la *vision active* dans notre processus décisionnel.

En fin de chapitre (section VI.9, page 160) est présentée une synthèse des différentes stratégies d'apprentissage et de localisation.

VI.4.1 Phase d'apprentissage

La première phase est la **phase d'apprentissage**, dont l'objectif est la recherche d'amers visuels pour chaque site de l'environnement (figure VI.2).

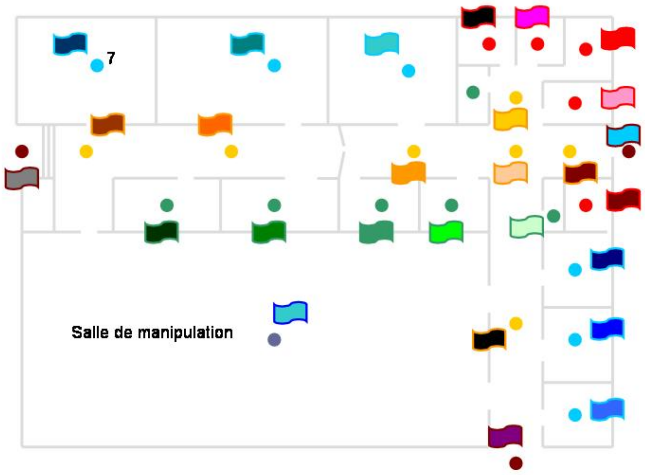


FIG. VI.2 – Amers attachés à chaque site.

Il s'agit donc, dans un premier temps, d'analyser les images du site, d'en extraire les attributs présents, puis de construire le treillis global (en approche globale) ou les treillis locaux (en approche locale) correspondant. Cette stratégie d'apprentissage est décrite figure VI.3.

L'approche globale consiste à construire un treillis global, avec toutes les images de la base d'apprentissage, et d'en extraire des amers globaux. L'approche locale consiste à construire des treillis locaux, avec les images issues d'un site et des sites voisins, et d'en extraire des amers locaux. Cette dernière approche ne permet pas un processus entièrement incrémental. Ces processus sont détaillés par la suite (section VI.6 pour l'approche globale et section VI.7 pour l'approche locale).

Lors de la phase d'apprentissage, le robot est déplacé (il n'est pas encore pourvu d'algorithmes de déplacements évolués) d'un site à l'autre. Pour observer au mieux son environne-

I. Apprentissage()
I.1. Pour chaque nouvelle image d'apprentissage
  I.1.1. Extraire les primitives présentes dans cette image
  I.1.2. Déterminer la présence ou non des attributs dans l'image
  I.1.3. Si on est en approche globale,
    I.1.3.1. Mettre à jour le treillis de Galois
    I.1.3.2. Extraire les amers visuels du treillis
  I.1.4. Si on est en approche locale,
    I.1.4.1. Mettre à jour les treillis locaux d'ordre 0
    I.1.4.2. ou Mettre à jour les *"mapping"* locaux d'ordre 0
I.2. En fin d'apprentissage, si on est en approche locale,
  I.2.1. Construire les treillis locaux
  I.2.2. Extraire les amers visuels de chaque treillis

FIG. VI.3 – Phase d'apprentissage.

ment, le robot tourne régulièrement sur lui-même (figure VI.4).

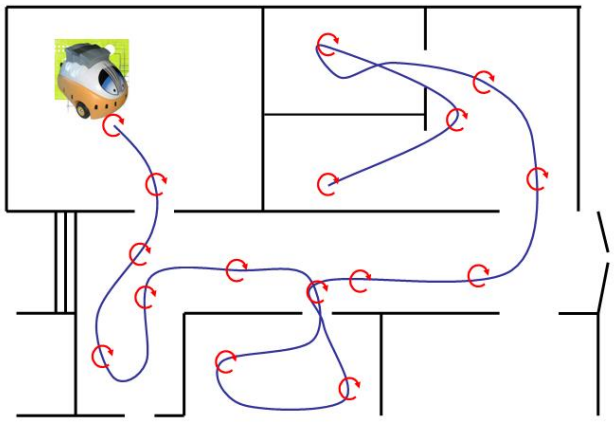


FIG. VI.4 – Déplacements du robot lors de l'apprentissage.

Les images sont prises à la volée. Le robot fait donc son apprentissage sur une vision partielle de l'environnement, et peut manquer certains attributs, donc certains amers.

VI.4.2 Critère d'arrêt de fin d'apprentissage

La phase d'apprentissage est une phase délicate en ce sens où il n'existe pas, *a priori*, de critère d'arrêt à l'apprentissage. Comment savoir si un apprentissage a été *correctement* effectué ? Comment juger qu'une base d'apprentissage est *assez* représentative ?

Notre démarche permet de répondre au problème et de donner un critère d'arrêt à l'apprentissage. Lors de la phase d'apprentissage, le robot effectue une phase de test pour localiser la nouvelle image, et compare le résultat avec le site réel courant. Si les sites correspondent, ou si le robot ne propose aucun site, l'apprentissage se déroule normalement.

Or si les amers trouvés ne correspondent pas au site donné, on peut en conclure que l'apprentissage n'est pas suffisant. Les phases de test, de décision et de mise à jour corrigeront éventuellement les dernières erreurs (d'un point de vue théorique, pour éviter toute erreur, il faudrait arrêter l'apprentissage après un nombre infini d'images à analyser... ce qui nous ramènerait à un cas classique d'apprentissage). Dans la pratique, la phase de mise à jour permet de palier ce problème.

L'algorithme général est présenté figure VI.5.

I. ApprentissageAvecCritereDArret()
I.1. Pour chaque nouvelle image d'apprentissage
  I.1.1. Test()
  I.1.2a. S'il existe des amers appartenant à un autre site, ou s'il n'y a pas d'amers présents, l'apprentissage doit continuer
  I.1.2b. Si, *depuis longtemps*, il n'y a plus d'amers d'autres sites, l'apprentissage peut s'arrêter pour le site en question
  I.1.3. Si on est en approche globale,
    I.1.3.1. Mettre à jour le treillis de Galois
    I.1.3.2. Extraire les amers visuels du treillis
  I.1.4. Si on est en approche locale,
    I.1.4.1. Mettre à jour les treillis locaux d'ordre 0
    I.1.4.2. ou Mettre à jour les *"mapping"* locaux d'ordre 0
I.2. En fin d'apprentissage, si on est en approche locale,
  I.2.1. Construire les treillis locaux
  I.2.2. Extraire les amers visuels de chaque treillis

FIG. VI.5 – Phase d'apprentissage avec critère de fin d'apprentissage.

VI.4.3 Phase de test

La deuxième phase est une phase de test. Son objectif est de localiser l'image. Le principe est simple : pour localiser toute nouvelle image, il faut et il suffit de rechercher des amers dans cette image (figure VI.6). Ensuite trois cas sont possibles :

– soit il n'y a aucun amer, alors on recherche des amers probabilistes (section VI.5),

– soit les amers présents appartiennent à un seul et même site, alors l'image est localisée dans le site en question,

– soit il existe au moins deux amers appartenant à deux sites différents, alors l'image n'est pas localisée.

L'algorithme correspondant est présenté figure VI.7.

FIG. VI.6 – Recherche d'amers dans l'image.

Dans le cas où aucun amer ne serait présent dans l'image, nous introduisons une *approche probabiliste* qui permet de chercher dans l'image des *amers probabilistes*. Cette technique est présentée section VI.5.

VI.4.4 Phase de décision

La troisième phase consiste à prendre une décision sur la localisation du robot. Prenons le cas le plus simple : le robot se localise en fonction de chaque image. Ainsi, si l'image est localisée dans le site q, alors le robot est situé dans le site q.

Ce processus dans la pratique est très risqué, et une image seule ne permet pas à un robot de se localiser de façon sure. Aussi est-il plus robuste d'utiliser plusieurs images successives.

II. Test()
II.1. Pour chaque nouvelle image de test
II.1.1. Extraire les primitives présentes dans cette image
II.1.2. Déterminer la présence ou non des attributs dans l'image
II.1.3. Rechercher la présence d'amers dans cette image
II.1.4. S'il n'y a pas d'amers, alors on recherche des amers probabilistes
II.1.5. S'il n'y a que des amers appartenant à un seul et même site, alors l'image est localisée dans le site en question
II.1.6. S'il y a au moins deux amers appartenant à deux sites différents, alors l'image n'est pas localisée

FIG. VI.7 – Phase de test.

Pour chaque image, le robot effectue une recherche d'amers. Si une image ne possède que des amers issus d'un seul et même site, l'image est située (correctement ou non). On incrémente ensuite le nombre d'images situées pour chaque site.

La décision dépend du nombre d'images situées dans chaque site. Si un *grand nombre* d'images sont situées dans un site spécifique et *peu* d'images dans tous les autres sites, alors le robot peut considérer qu'il est situé dans le site en question. Il faut pour cela tenir compte du ratio suivant :

$$r = \frac{\text{Nombre d'images situées dans un site } q}{\sum_{q^* \neq q} \text{Nombre d'images situées dans un site } q^*} \tag{VI.2}$$

La décision est alors la suivante : si ce ratio est supérieur à un seuil donné (proche de 1, en général 0,9), alors le robot est situé de façon *certaine*. Sinon, le robot ne peut prendre de décision et l'apprentissage doit être poursuivi. Si le robot ne peut se localiser au delà d'un certain *temps de test*, le processus s'arrête.

Afin de rendre plus robuste la prise de décision et de trouver de nouvelles images de test, il est nécessaire de donner au robot la "liberté" de se mouvoir seul dans son environnement. On parle alors de *vision active*.

VI.4.5 Intégration de la vision active

Les robots que nous utilisons sont des robots mobiles, et sont maîtres de leur déplacements. Ils peuvent donc intégrer ces déplacements dans leur prise de décision. L'idée est simple : le robot est situé dans un site, et s'il ne trouve pas d'amer ou s'il trouve des amers de plusieurs sites, alors il peut se mouvoir pour rechercher d'autres amers et ainsi prendre une décision sur une image *a posteriori*, puis remettre à jour le treillis de concepts. Ce processus est décrit dans

III. Décision()
III.1. Tant que le robot n'est pas localisé de façon *certaine*
III.1.1. Localiser chaque nouvelle image de test
III.1.2. Si le temps de test est supérieur à un temps donné
III.1.1.1. Le robot n'est pas localisé
III.1.1.2. Reprendre l'apprentissage
III.1.1.3. Sortie du programme
III.2. Le robot est localisé

FIG. VI.8 – Phase de décision.

l'algorithme de la figure VI.9.

III. DécisionAvecVisionActive()
III.1. Tant que le robot n'est pas localisé de façon *certaine*
III.1.1. Bouger dans l'environnement afin de trouver une nouvelle image de test
III.1.2. Si le temps de test est supérieur à un temps donné
III.1.1.1. Le robot n'est pas localisé
III.1.1.2. Reprendre l'apprentissage
III.1.1.3. Sortie du programme
III.2. Le robot est localisé

FIG. VI.9 – Phase de décision avec vision active.

La stratégie de vision active consiste à laisser le robot chercher de nouvelles images tant qu'il n'est pas situé de façon certaine. Le robot tire un nombre n entre 0 et 1 selon une probabilité uniforme. La vitesse de rotation du robot Ω est fonction de cette probabilité (figure VI.10). Sa vitesse linéaire v, en revanche, est proportionnelle au temps. On a alors pour les vitesses du robot :

$$\begin{cases} \Omega = (n - \frac{1}{4})\Omega_{max} \\ v = k.t \end{cases} \tag{VI.3}$$

Ω_{max} représente la vitesse de rotation maximale du robot. Ainsi, si $n = 1/4$, alors le robot ne tourne pas ; si $n = 0$, il tourne à gauche avec une vitesse $\Omega = -\Omega_{max}/4$; et si $n = 1$, il tourne à droite.

De cette manière, même si le mouvement est un peu chaotique, la tendance globale du déplacement du robot est de tourner sur la droite. La trajectoire ressemble alors à une coquille

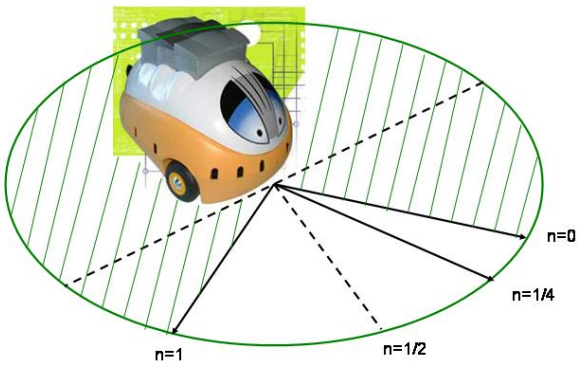


FIG. VI.10 – Orientation du robot lors de la recherche d'amers.

d'escargot (figure VI.11).

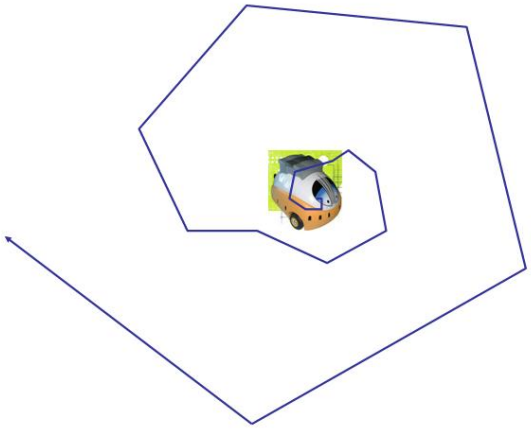


FIG. VI.11 – Mouvement du robot lors de la recherche d'amers.

Dans certains cas il est possible d'améliorer l'apprentissage à la suite de la phase de décision : on parle alors de localisation avec mise à jour du treillis.

VI.4.6 Localisation avec mise à jour du treillis

Dans l'exemple précédent, s'il existe dans un même site de test des amers appartenant à plusieurs sites différents, il est nécessaire de mettre à jour le treillis... à condition d'être relativement sûr de la localisation. En effet, si la localisation s'avère défectueuse, on risquerait de perdre beaucoup d'amers et de renforcer de fausses informations.

Une stratégie est alors de garder un historique des images et de leur localisation présumée (vrai ou fausse). Dans ce cas, si une image, mal localisée ou qui possède plusieurs amers de plusieurs sites, est "noyée" dans un flux d'images situées de façon certaine (en décision finale), sachant que notre robot ne change pas de site en cours, il sera raisonnable de penser que cette image est située au même endroit que ses suivantes et précédentes.

Cette stratégie demandant une dépendance "historique" entre le test et la décision, il est nécessaire de présenter un algorithme global, figure VI.12. Cette stratégie a pour principal objectif la mise à jour du treillis -et donc des amers- lors de la phase de test. Cependant, l'inconvénient est de devoir conserver le treillis en entier, et non plus seulement les amers visuels.

I. Apprentissage()
II. Test()
III. Décision()
idem.

IV. Mise à Jour()
IV.1. Pour chaque image de test précédemment capturée
  IV.1.1. Étiqueter cette image de la classe finale choisie
  IV.1.2. Mettre à jour le treillis
IV.2. Rechercher les nouveaux amers pour chaque classe

FIG. VI.12 – Stratégie de Localisation avec Mise à Jour du Treillis.

VI.5 Localisation d'une Image à l'aide d'Amers Probabilistes

Revenons plus en détail ici sur la phase de test et la localisation d'une image. Nous savons qu'en pratique, notamment pour une approche globale (voir ci-dessous), les amers peuvent être relativement rares, et situer une image peut être difficile. De plus, la localisation à l'aide d'amers implique, pour toute nouvelle image, une réponse binaire sur la présence ou non d'amers visuels : si tous les amers présents appartiennent à un seul et même site, alors l'image est située ; sinon, aucune décision n'est prise quelque soit le nombre d'amers relatifs à chaque site.

Cependant, si l'on observe -par exemple- une combinaison particulière d'attributs qui serait présente...
- 99% du temps dans des images issues d'un site θ_1,
- et 1% du temps dans des images issues d'une deuxième site θ_2,

ne pourrait-on conclure quant à une probabilité de localisation de l'image en question ?

C'est l'intérêt des amers probabilistes.

L'utilisation d'amers probabilistes est directement liée à la notion de probabilité associée définie au chapitre précédent. Le treillis dont dispose le robot pour localiser une image permet de lui fournir une information plus riche que précédemment, à savoir une probabilité d'appartenance à telle ou telle classe, une probabilité d'être situé dans tel ou tel site.

VI.5.1 Amer probabiliste

Cette information est contenue dans tous les concepts du treillis. Aussi un amer probabiliste peut-il être pris au sens très large, à savoir que toute intension de concept est un amer probabiliste.

Définition 37 *Un* **amer probabiliste** *est une intension de concept munie des probabilités associées pour chaque classe.*

Dans ces conditions, comment sélectionner un amer parmi les autres ? Comment hiérarchiser les amers pour une classe donnée ?

VI.5.2 Similarité entre deux ensembles d'attributs

Choisir et hiérarchiser des concepts dans un but bien précis demande une technique de comparaison basée sur une métrique quelconque. Il faut donc définir une *similarité* entre concepts, et plus précisément entre deux ensembles d'attributs. Distances et similarités ont été largement utilisées dans les arbres de décision et les arbres quaternaires (voir [Kodratoff 88, Rukoz 02]), ou sur de simples primitives [Foggia 99].

Nous avons donc cherché une similarité assez simple entre deux ensembles d'attributs. Dans la littérature il existe une infinité de définitions, plus ou moins propres à chaque domaine. Alors que [Tversky 77] propose un *"feature-based contrasted model"* dans lequel les attributs communs tendent à faire croître la similarité, et les différences d'attributs à la faire diminuer, [Lin 98] donne une définition propre de la similarité : *"la similarité entre A et B est mesurée par le ratio entre la quantité d'information nécessaire à établir ce qu'il y a de commun entre A et B et l'information qui décrit entièrement A et B"*. Puis il propose la formule suivante :

$$\mathcal{S}(A,B) = \frac{logP(common(A,B))}{logP(description(A,B))} \tag{VI.4}$$

où P est une probabilité donnée. À partir de ces considérations, nous nous sommes donnés une définition simple de la similarité entre deux ensembles d'attributs :

Définition 38 *Soient $\mathcal{B}_1$ et $\mathcal{B}_2$ deux ensembles d'attributs. La similarité $\mathcal{S}$ entre ces deux ensembles est donnée par la formule suivante :*

$$\mathcal{S}(\mathcal{B}_1, \mathcal{B}2) = \frac{card(\mathcal{B}_1 \cap \mathcal{B}_2)}{card(\mathcal{B}_1 \cup \mathcal{B}_2)} \tag{VI.5}$$

VI.5.3 Similarité entre une image et un concept

Dans notre application, lors de la phase de test (localisation du robot), il ne s'agit pas de calculer la similarité entre deux ensembles d'attributs quelconques, mais plutôt entre un concept du treillis et les attributs de la nouvelle image à classer, à localiser. Nous définissons donc la similarité entre une image et un concept par la similarité entre les attributs de l'image et l'intension du concept :

Définition 39 *Soit $\mathcal{I}$ une image à classer, $\mathcal{B}_\mathcal{I}$ la combinaison d'attributs présente dans $\mathcal{I}$, et $\mathcal{C} = (\mathcal{A}, \mathcal{B})$ un concept du treillis. La* **similarité** *entre l'image et le concept est définie de la façon suivante :*

$$\mathcal{S}(\mathcal{I}, \mathcal{C}) = \frac{card(\mathcal{B}_\mathcal{I} \cap \mathcal{B})}{card(\mathcal{B}_\mathcal{I} \cup \mathcal{B})} \tag{VI.6}$$

VI.5.4 Similarité pondérée entre une image et un concept

Dans la plupart des cas, il est préférable de sélectionner les amers les plus généraux possible, c'est-à-dire les amers de plus petite intension en terme de cardinalité. C'est pourquoi il est possible de pondérer la similarité afin de privilégier les concepts les plus généraux.

On peut donc définir la similarité par la formule suivante :

$$\mathcal{S}(\mathcal{I}, \mathcal{C}) = \frac{1}{card(\mathcal{B})} . \frac{card(\mathcal{B}_\mathcal{I} \cap \mathcal{B})}{card(\mathcal{B}_\mathcal{I} \cup \mathcal{B})} \tag{VI.7}$$

ou bien,

$$\mathcal{S}(\mathcal{I}, \mathcal{C}) = e^{-card(\mathcal{B})} . \frac{card(\mathcal{B}_\mathcal{I} \cap \mathcal{B})}{card(\mathcal{B}_\mathcal{I} \cup \mathcal{B})} \tag{VI.8}$$

VI.5.5 Amer probabiliste maximal

Enfin, pour rechercher l'amer probabiliste maximal, et le valider, il faut rechercher dans un premier temps le concept le plus similaire :

Définition 40 *Soit $\mathcal{B}_\mathcal{I}$ les attributs d'une image à classer, le concept le plus similaire à l'image $\mathcal{I}$ dans $\mathcal{L}$ est le concept $\mathcal{C}^*$ tel que :*

$$\mathcal{C}^* = \underset{\mathcal{C} \in \mathcal{L}}{\operatorname{argmax}} \, \mathcal{S}(\mathcal{I}, \mathcal{C}) \tag{VI.9}$$

Dans un deuxième temps, il est nécessaire d'extraire la classe maximale de ce concept :

Définition 41 *Soit $\mathcal{C} = (\mathcal{A}, \mathcal{B})$ un concept probabiliste. Sa classe maximale $q^*_{\mathcal{C}}$ est la classe définie par :*

$$q^*_{\mathcal{C}} = \underset{q}{\operatorname{argmax}}\, p(q|\mathcal{B}) \tag{VI.10}$$

On obtient ainsi la classe la plus probable du concept le plus similaire. Enfin, dans un troisième et dernier temps, le concept maximal sera validé si et seulement si il vérifie une dernière contrainte sur sa localisation. Soit T un seuil quelconque,

$$\mathcal{C}^* \text{ est validé } \Leftrightarrow \pi(q^*_{\mathcal{C}^*}) > T \tag{VI.11}$$

VI.5.6 Nouvelle phase de test

La phase de test, dans le cadre d'un treillis probabiliste, consiste à rechercher tout d'abord le concept le plus similaire dans le treillis, puis à trouver la classe maximale de ce concept, et enfin à la valider ou non.

Dans la pratique, cependant, les amers probabilistes viendront en complément des amers "classiques". L'algorithme utilisé est alors celui présenté figure VI.13. Cette stratégie recherche dans un premier temps les amers classiques dans la nouvelle image, et s'il n'y a pas d'amer alors les amers probabilistes sont recherchés et éventuellement validés.

II. TestProba()
II.1. Pour chaque nouvelle image de test
II.1.1. Extraire les primitives présentes dans cette image
II.1.2. Déterminer la présence ou non des attributs dans l'image
II.1.3. Rechercher la présence d'amers dans cette image
II.1.4. S'il n'y a pas d'amers,
II.1.4.1. Rechercher le concept le plus similaire
II.1.4.2. Rechercher la classe de probabilité maximale de ce concept $q^*_{\mathcal{C}^*}$
II.1.4.3. S'il existe un site tel que sa probabilité d'appartenance est supérieure à un seuil, alors l'image est située dans ce site
II.1.5. S'il n'y a que des amers appartenant à un seul et même site, alors l'image est localisée dans le site en question
II.1.6. S'il y a au moins deux amers appartenant à deux sites différents, alors l'image n'est pas localisée

---

FIG. VI.13 – Phase de test à l'aide d'amers probabilistes.

VI.5.7 Discussion

L'idée initiale d'utiliser des treillis probabilistes était d'exploiter de l'information cachée dans le treillis. L'utilisation des amers "classiques" ne laisse pas une grande souplesse dans la localisation d'images par des amers visuels.

Une conséquence directe de ce manque de souplesse apparaît dans le taux de réponse faible que nous donne l'approche classique. L'utilisation d'amers probabilistes permet d'augmenter ce taux de réponse de près de 20% dans les essais que nous avons menés (voir chapitre suivant), avec un seuil $T = 0,75$ fixé arbitrairement.

Ensuite, bien entendu, il est possible d'associer cet algorithme avec les algorithmes de mise à jour du treillis ou de vision active.

Le principal inconvénient de l'approche probabiliste est que le treillis doit être conservé entièrement, afin de pouvoir rechercher le concept le plus similaire à toute nouvelle image à localiser. Dans le cas des amers globaux, s'il n'y a pas de volonté de mise à jour du treillis, seuls les amers sont enregistrés, et le treillis peut être détruit.

VI.6 Localisation à l'aide d'Amers Globaux

Dans le cadre de notre application, le robot dispose d'un contexte global (prenant en compte toutes les images), partitionné (toutes les images appartiennent à un et un seul site défini) et propre (aucune image ne possède tous les attributs, aucun attribut n'est présent dans toutes les images -ce deuxième cas est en fait beaucoup moins problématique).

Les amers issus du treillis global sont des amers globaux, qui n'appartiennent qu'à un seul site. En d'autres termes, chaque site possède des amers qui lui sont propres, et il ne peut y avoir de confusion entre plusieurs sites à partir de deux amers identiques.

VI.6.1 Définition d'un amer de site

Soit $\mathcal{K} = (\mathcal{O}, \mathcal{F}, \zeta)$ ce contexte et q une classe (site) de ce contexte. En reprenant la définition précédente, on a :

Définition 42 *$\mathcal{B}_q$ est un* **amer global du site** *q si et seulement si :*
- $\mathcal{B}''_q = \mathcal{B}_q$,
- *et* $\mathcal{B}'_q \subset q$.

VI.6.2 Phase d'apprentissage

Dans le cadre de l'approche globale, la phase d'apprentissage est la plus simple : le treillis global est construit incrémentalement à partir des attributs des images successives captées par le robot. Les amers visuels extraits à partir du treillis sont des amers globaux.

VI.6.3 Phases de test

La phase de test consiste à rechercher les amers dans chaque image. Il est cependant possible, au cours d'une trajectoire de test, de valider des amers topologiquement.

Considérant une topologie donnée, le robot ne peut *a priori* sauter d'un site q à un autre site q' non connexes. Aussi, une nouvelle localisation d'image sera validée si et seulement si le nouveau site est connexe avec le site précédent.

VI.6.4 Phase de décision

Le processus est identique à ce qui a été précédemment écrit, quelque soit la stratégie utilisée (localisation à partir d'une image ou d'un flux d'images).

VI.6.5 Algorithme relatif à l'approche globale

L'algorithme général utilisé dans nos expériences pour localiser un robot dans un environnement topologique est présenté figure VI.14. Il utilise la vision active pour rechercher des amers. Il n'y a pas de mise à jour du treillis dans un premier temps.

VI.6.6 Discussion

Le processus d'apprentissage à l'aide d'un seul et même treillis permet de localiser le robot dans tous les sites de l'environnement. En effet, les amers étant uniques, il ne peut -sauf erreur d'apprentissage- y avoir d'ambiguïté sur la localisation à partir d'amers visuels.

De plus, l'algorithme d'apprentissage est entièrement incrémental, et permet un processus temps réel tout au long de l'expérience.

Cependant, l'approche globale suppose la construction d'un treillis à l'aide de toutes les images de l'environnement. Il peut donc être très gros, et la mise à jour du treillis peut par moment s'avérer difficile et long.

C'est pourquoi nous avons introduit les treillis locaux dans notre processus.

VI.7 Localisation à l'aide d'Amers Locaux

Dans une application réelle où le robot capture 1000 images environ, décrites à l'aide de 153 attributs, le treillis général peut contenir jusqu'à 200.000 concepts, ce qui représente un espace mémoire très important. Aussi l'idée a été de distribuer ce treillis en *treillis locaux* propres

I. Apprentissage()
I.1. Pour chaque nouvelle image d'apprentissage
I.1.1. Extraire les primitives présentes dans cette image
I.1.2. Déterminer la présence ou non des attributs dans l'image
I.1.3. Mettre à jour le treillis de Galois
I.1.4. Extraire les amers visuels du treillis

II. Test()
II.1. Pour chaque nouvelle image de test
II.1.1. Extraire les primitives présentes dans cette image
II.1.2. Déterminer la présence ou non des attributs dans l'image
II.1.3. Rechercher la présence d'amers dans cette image
II.1.4. S'il n'y a pas d'amer, alors on recherche des amers probabilistes
II.1.5. S'il n'y a que des amers appartenant à un seul et même site, alors l'image est localisée dans le site en question
II.1.6. S'il y a au moins deux amers appartenant à deux sites différents, alors l'image n'est pas localisée

III. DécisionAvecVisionActive()
III.1. Tant que le robot n'est pas localisé de façon certaine
III.1.1. Bouger dans l'environnement afin de trouver une nouvelle image de test
III.1.2. Si le temps de test est supérieur à un temps donné
III.1.1.1. Le robot n'est pas localisé
III.1.1.2. Reprendre l'apprentissage
III.1.1.3. Sortie du programme
III.2. Le robot est localisé

FIG. VI.14 – Algorithme relatif à l'approche globale.

à chaque site.

L'introduction des amers locaux dans un processus de localisation répond ainsi à un quadruple objectif. Le premier d'entre eux est de soulager la machine d'un treillis global, en créant ainsi des structures locales, plus petites, permettant de distribuer l'information sur chaque nœud du graphe topologique.

Le deuxième objectif était d'avoir une lisibilité des treillis plus facile et plus rapide. En effet, l'extraction d'information est bien plus simple dans de petites structures que dans de plus grandes.

Le troisième objectif était de diminuer le temps de construction du treillis. En effet, avec des contextes plus petits, de par la complexité exponentielle des algorithmes, les temps de calcul se trouveront bien plus petits que pour un seul et gros contexte.

Enfin, le quatrième et dernier objectif était d'augmenter le nombre des amers potentiels. En effet, un amer doit être *unique* mais il peut être *localement unique* s'il n'y a pas d'ambiguïté sur la localisation du robot avant de voir cet amer. Un extincteur rouge à un endroit précis d'un bâtiment sera un amer si je suis déjà plus ou moins à cet endroit, même s'il existe d'autres extincteurs rouges ailleurs dans ce même bâtiment.

VI.7.1 Contexte topologique

Le contexte particulier de notre application est le contexte topologique. Le robot évolue en effet dans un espace qu'il se représente sous forme de graphe topologique, contenant des nœuds et des arcs. Chaque nœud représente un site particulier, chaque arc un passage entre deux sites.

La figure VI.15 représente une représentation topologique du LIA, le Laboratoire d'Informatique et d'Automatique de SUPAERO.

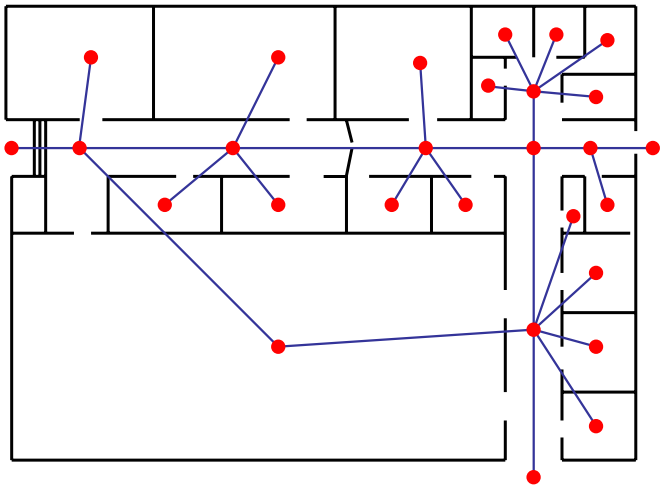


FIG. VI.15 – Carte métrique et topologique du LIA de SUPAERO

VI.7.2 Treillis local d'ordre 0

Le treillis local d'ordre 0 est un treillis de voisinage 0. Comme nous l'avons vu au chapitre précédent, il s'agit de construire, pour chaque nœud du graphe topologique, un treillis local

dont le contexte ne comporte que des images du site considéré.

Ce cas est assez particulier. En effet il ne permet plus de séparer les sites, à partir du moment où les treillis perdent toute notion de topologie et où chaque nœud est indépendant des autres. C'est pourquoi nous introduisons ici les treillis de voisinage ou treillis locaux d'ordre V.

VI.7.3 Treillis local d'ordre V

Dans cette approche, aucun attribut n'est éliminé du contexte qui reste tel que défini à l'origine. Cependant, nous construisons, associé à chaque site, un treillis de voisinage d'ordre V (V est la profondeur du voisinage au sens topologique du terme). Ces treillis permettront de sélectionner les amers propres à chaque site, relativement à son voisinage. La figure VI.16 illustre dans l'ordre les voisinages d'ordre 1 et d'ordre 2.

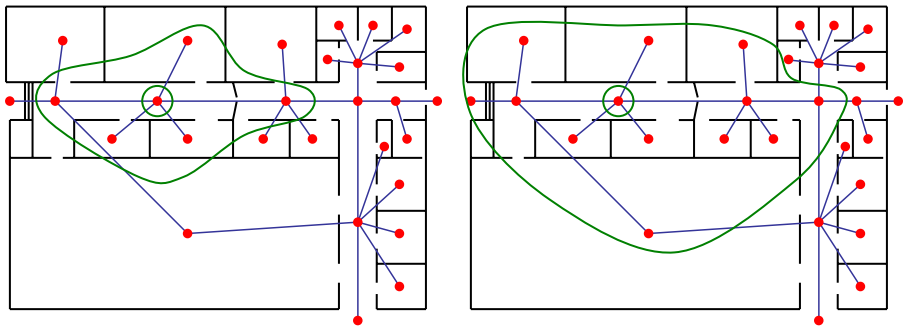


FIG. VI.16 – Topologie et Voisinage d'ordre 1 (à gauche) et d'ordre 2 (à droite).

Une fois les N treillis locaux construits, pour N sites, l'extraction des amers permet d'identifier -et donc de différencier- un site q de son voisin. Le robot est alors en mesure de détecter **ses changements de sites** et de basculer d'un site à l'autre, donc d'un treillis à l'autre. La figure VI.17 illustre ce changement de site.

VI.7.4 Construction du treillis de voisinage

Dans le cas de figure d'une construction topologique, deux configurations sont possibles : soit la carte topologique est précédemment connue, et l'on peut garder un processus entièrement incrémental, soit la carte topologique se construit au fur et à mesure de l'expérience, et le processus devient partiellement incrémental.

Pour illustrer ceci, prenons l'exemple d'un robot qui découvre son environnement. Au premier nœud, le robot construit son treillis. Lors de la première transition, une fois entré dans le deuxième nœud, il continue de construire le treillis attaché au premier nœud car il est connexe au nouveau nœud. Or pour construire le treillis du deuxième nœud, il est nécessaire de connaître

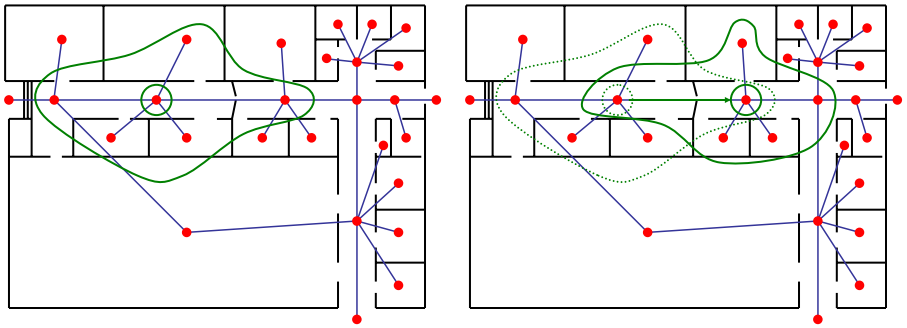


FIG. VI.17 – Topologie et Changement de Site.

les images du premier nœud, et notamment les images passées. Il est donc impossible de garder un processus entièrement incrémental, sauf à connaître par avance la carte topologique. Ce qui n'est pas le cas dans notre application.

Il existe alors deux manières de construire un treillis local de voisinage :
– à l'aide des contextes locaux (enregistrés lors de l'apprentissage),
– et d'une façon plus originale, à l'aide des treillis locaux d'ordre 0 (pouvant être construits, quant à eux, incrémentalement lors de l'apprentissage).

En effet, considérant un nœud quelconque du graphe topologique, il est possible de construire un treillis local de voisinage en fusionnant le treillis local d'ordre 0 considéré avec les treillis locaux des voisins. Cette fusion se fait à l'aide de l'algorithme ASSEMBLY [Valtchev 02], qui initialement était prévue dans une stratégie du type "divide and conquer", en construisant un gros treillis à partir de deux plus petits.

Il est donc possible, en fusionnant les treillis locaux d'ordre 0 d'un nœud quelconque et de ses voisins à l'ordre V au sens topologique, de construire le treillis local d'ordre V du nœud considéré.

VI.7.5 Phase d'apprentissage

Lors de la phase d'apprentissage, le robot peut soit construire et mettre à jour un treillis local de voisinage d'ordre 0 (avec uniquement les images attachées au site courant), soit enregistrer les *"mapping"* locaux d'ordre 0 dans le but de construire des treillis locaux d'ordre 1.

VI.7.6 Phase de test

Lors de la phase de test, lorsque le robot se situe dans un site spécifique, il a à disposition le treillis de voisinage d'ordre 1 (ou plus) attaché au nœud correspondant. Lorsqu'il se déplace,

il lui suffit de rechercher des amers dans le treillis relatif au nœud précédemment trouvé.

Il faut noter qu'au départ, il faut indiquer au robot le nœud dans lequel le robot se situe.

VI.7.7 Phase de décision

Elle est identique à précédemment.

VI.7.8 Algorithme relatif à l'approche locale

L'algorithme général développé dans notre expérience lors de l'approche locale est donné figure VI.18. Les hypothèse sont les mêmes que précédemment.

VI.7.9 Discussion

Le pricipal inconvénient de cette approche est que le robot peut ne pas se localiser de façon absolue dès le départ. En effet, plusieurs amers identiques peuvent coexister dans un même environnement. Il est cependant possible de déterminer la position à l'aide d'une chaîne d'amers qui peut se révéler unique.

Autre inconvénient important, si le robot lors de l'apprentissage ne crée pas un lien topologique entre deux sites, il ne pourra déceler lors du test d'un site à l'autre. En effet, il ne peut considérer comme voisins deux sites qui ne l'étaient pas lors de l'apprentissage.

En revanche, les avantages sont nombreux. Le premier d'entre eux est la gestion de nombreux petits treillis au lieu d'un grand. Deuxièmement, le nombre d'amers disponibles pour le robot est bien plus important, puisqu'il peut en attribuer à plusieurs endroits biens séparés et non connexes.

Ensuite, les treillis sont bien plus petits donc manipulables. Il est donc facile et rapide d'extraire des amers, et éventuellement de les mettre à jour. De plus, ils prennent peu de place mémoire.

Enfin, contrairement à l'approche globale bornée par le nombre d'amers possibles, l'approche décentralisée permet -théoriquement- de décrire un environnement de taille infinie, puisqu'il lui est possible d'utiliser le même amer pour plusieurs sites différents et non connexes.

Nous introduisons enfin une approche un peu particulière, qui n'a pu être validée expérimentalement mais qui présente un intérêt théorique que nous nous attacherons à appliquer par la suite : l'approche markovienne.

VI.8 Approche Markovienne

La localisation topologique d'un robot mobile s'inscrit de fait dans un processus markovien : la localisation du robot dépend de sa localisation précédente, et non de sa trajectoire

I. Apprentissage()
I.1. Pour chaque nouvelle image d'apprentissage
I.1.1. Extraire les primitives présentes dans cette image
I.1.2. Déterminer la présence ou non des attributs dans l'image
I.1.3. Mettre à jour les treillis locaux d'ordre 0
I.1.4. ou Mettre à jour les *"mapping"* locaux d'ordre 0
I.2. En fin d'apprentissage
I.2.1. Construire les treillis locaux
I.2.2. Extraire les amers visuels locaux de chaque treillis

II. Test()
II.1. Pour chaque nouvelle image de test
II.1.1. Extraire les primitives présentes dans cette image
II.1.2. Déterminer la présence ou non des attributs dans l'image
II.1.3. Rechercher la présence d'amers dans cette image
II.1.4. S'il n'y a pas d'amer, alors on recherche des amers probabilistes
II.1.5. S'il n'y a que des amers appartenant à un seul et même site, alors l'image est localisée dans le site en question
II.1.6. S'il y a au moins deux amers appartenant à deux sites différents, alors l'image n'est pas localisée

III. DécisionAvecVisionActive()
III.1. Tant que le robot n'est pas localisé de façon certaine
III.1.1. Bouger dans l'environnement afin de trouver une nouvelle image de test
III.1.2. Si le temps de test est supérieur à un temps donné
III.1.1.1. Le robot n'est pas localisé
III.1.1.2. Reprendre l'apprentissage
III.1.1.3. Sortie du programme
III.2. Le robot est localisé

FIG. VI.18 – Algorithme relatif à l'approche globale.

passée.

Un processus markovien (voir précédemment) permet de définir une probabilité sur un état (dans notre application, un site) conditionnée uniquement à l'état précédent, toute influence de la trajectoire passée étant supposée nulle. Ceci suppose donc que le robot ne possède pas de stratégies de déplacement pré-programmées le long de trajectoires pré-définies ou calculées.

Cette probabilité dépend dans l'absolu de plusieurs facteurs (bruits de capteurs, capacités

du robot à se déplacer, position du robot relativement à la zone de transition, *etc.*) Dans la pratique, nous basons cette probabilité sur les amers :

– si un amer de la pièce voisine est observé, alors la transition est validée et le robot change de site ;

– si un amer de la pièce courante est observé, le robot est toujours sur le même site ;

– si aucun amer n'est observé, aucune conclusion n'est possible ;

– enfin, si plusieurs amers de plusieurs sites sont observés, les treillis probabilistes sont utilisés pour calculer cette transition.

VI.8.1 Chaîne de Markov et treillis de voisinage

Supposons que le robot se situe sur un site bien défini. Sa probabilité de rester sur ce site ou de se retrouver dans un site voisin dépend du treillis de voisinage probabiliste associé au site en question.

La représentation topologique associée au treillis de voisinage d'ordre 1 peut donc être décrite par un processus markovien. En effet, la probabilité pour le robot de se retrouver dans une pièce voisine -ou de rester dans la même pièce- est indépendante du chemin parcouru précédemment.

En dehors de toute stratégie de navigation, les probabilités de transitions sont empiriques et dépendront du nombre d'amers visuels et de leur occurrence. Elles devront être en revanche indépendantes de la trajectoire d'apprentissage du robot. Pour cela, il faudra supposer que le robot adopte une trajectoire ergodique.

VI.8.2 Treillis de Galois markovien

Alors que le cadre précédent des chaînes de Markov n'était qu'une simple *lecture* markovienne du processus de localisation, nous proposons ici d'inscrire, à la source, c'est-à-dire dans le contexte d'apprentissage, l'information de classe de type markovien en introduisant un sur-contexte, dans lequel est indiqué, comme simple attribut, la classe de l'image courante et la classe de l'image précédente.

Ce sur-contexte, construit à partir du contexte $\mathcal{K}$, est donc de la forme $\mathcal{K} + \mathcal{K}_q^0 + \mathcal{K}_q^{-1}$ avec $\mathcal{K}_q^0 = (\mathcal{O}, \mathcal{Q}, \zeta_q^0)$ le contexte de classe de l'image courante et $\mathcal{K}_q^{-1} = (\mathcal{O}, \mathcal{Q}, \zeta_q^{-1})$ le contexte de classe de l'image précédente.

On définit ainsi le *treillis de Galois markovien* :

Définition 43 *Le treillis issu du sur-contexte $\mathcal{K} + \mathcal{K}_q^0 + \mathcal{K}_q^{-1}$ est appelé* **treillis de Galois markovien**.

Il est intéressant de s'arrêter sur certains concepts de ce treillis. Comme nous l'avons vu au chapitre précédent, les concepts classés (amers) de tels contextes sont les concepts n'ayant qu'une classe dans leur intension. Or ici une information supplémentaire apparaît, la classe de

l'image précédente. Il est donc possible de distinguer des *amers d'entrée* propres à chaque site voisin, d'où vient le robot lors de son exploration.

VI.9 Synthèse

Ici est présentée une synthèse regroupant les stratégies relatives à la phase d'apprentissage et les stratégies relatives à la phase de test et de décision.

La figure VI.19 expose les stratégies d'apprentissage. Chaque nouvelle image est analysée et les attributs sont extraits. Ensuite deux stratégies possibles : soit on adopte une approche globale, le treillis global est alors construit incrémentalement et les amers visuels sont extraits pour chaque site ; soit on adopte une approche locale, on a alors deux voies possibles :

– soit on construit les treillis locaux d'ordre 0, attachés à chaque nœud, et le treillis local de voisinage d'ordre 1 se construit par la fusion des treillis locaux d'ordre 0 en fonction des connexités du graphe topologique ;

– soit on enregistre les *"mapping"* locaux et l'on construit, *a posteriori* également, les treillis locaux d'ordre 1 à partir des *"mapping"* locaux connexes.

Ensuite, les amers visuels (globaux ou locaux) sont extraits pour chaque site. Enfin, la carte topologique se construit en fonction du site de la nouvelle image.

La phase de localisation (figure VI.20), qui réunit la phase de test (localisation des images) et la phase de décision (localisation du robot), commence par les mêmes processus : chaque nouvelle image est analysée et ses attributs sont extraits. Ensuite, on recherche dans cette image des amers visuels (soit du treillis global, soit du treillis local de voisinage d'ordre 1 ou plus). Chaque image peut alors être localisée (éventuellement avec une validation topologique). Le robot se localise alors soit sur chaque image, soit par souci de robustesse sur un flux d'images. Un processus de vision active permet au robot de continuer à chercher des amers ou à s'arrêter s'il estime être localisé de façon certaine.

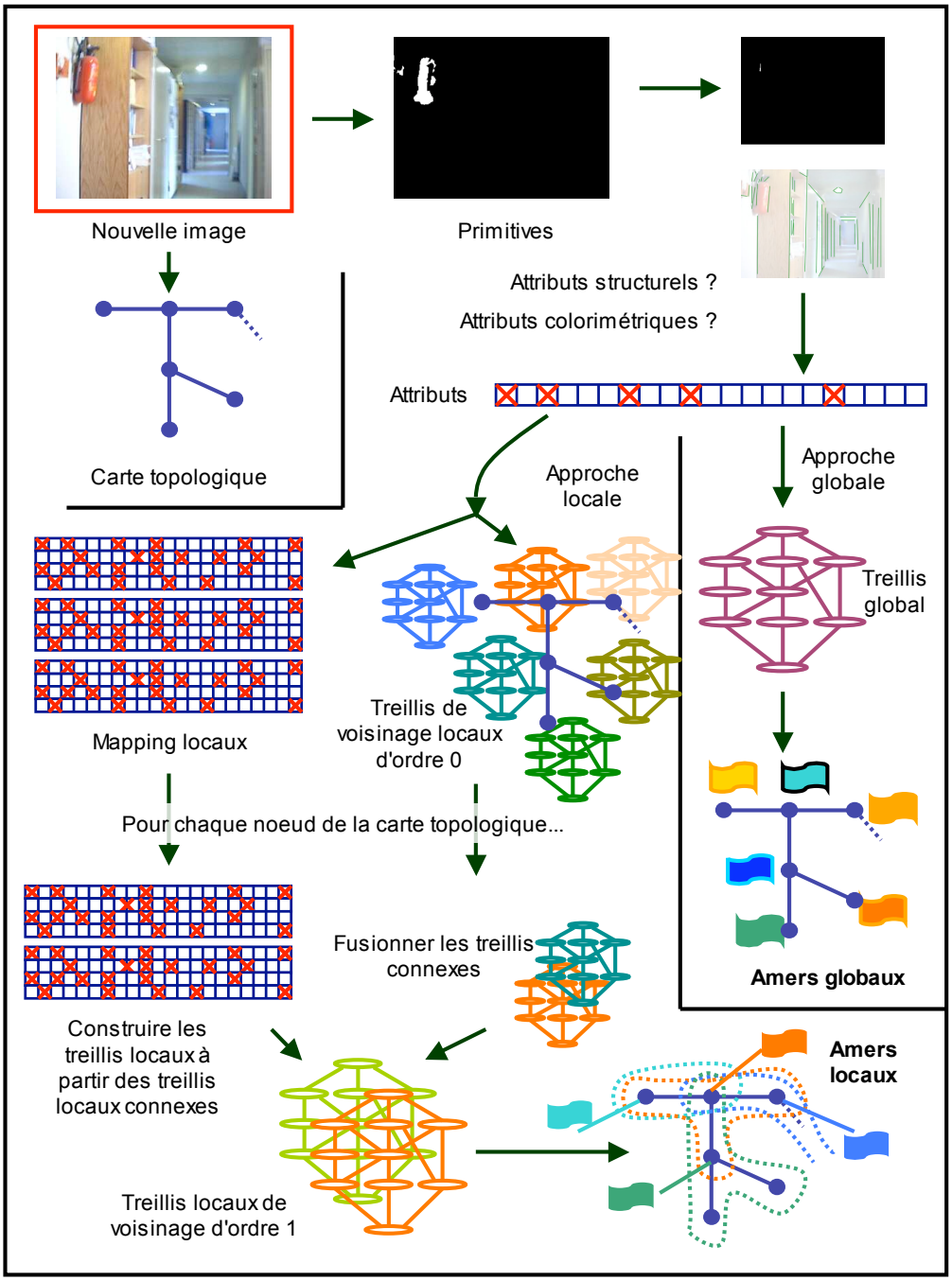


FIG. VI.19 – Les différentes stratégies de la phase d'apprentissage.

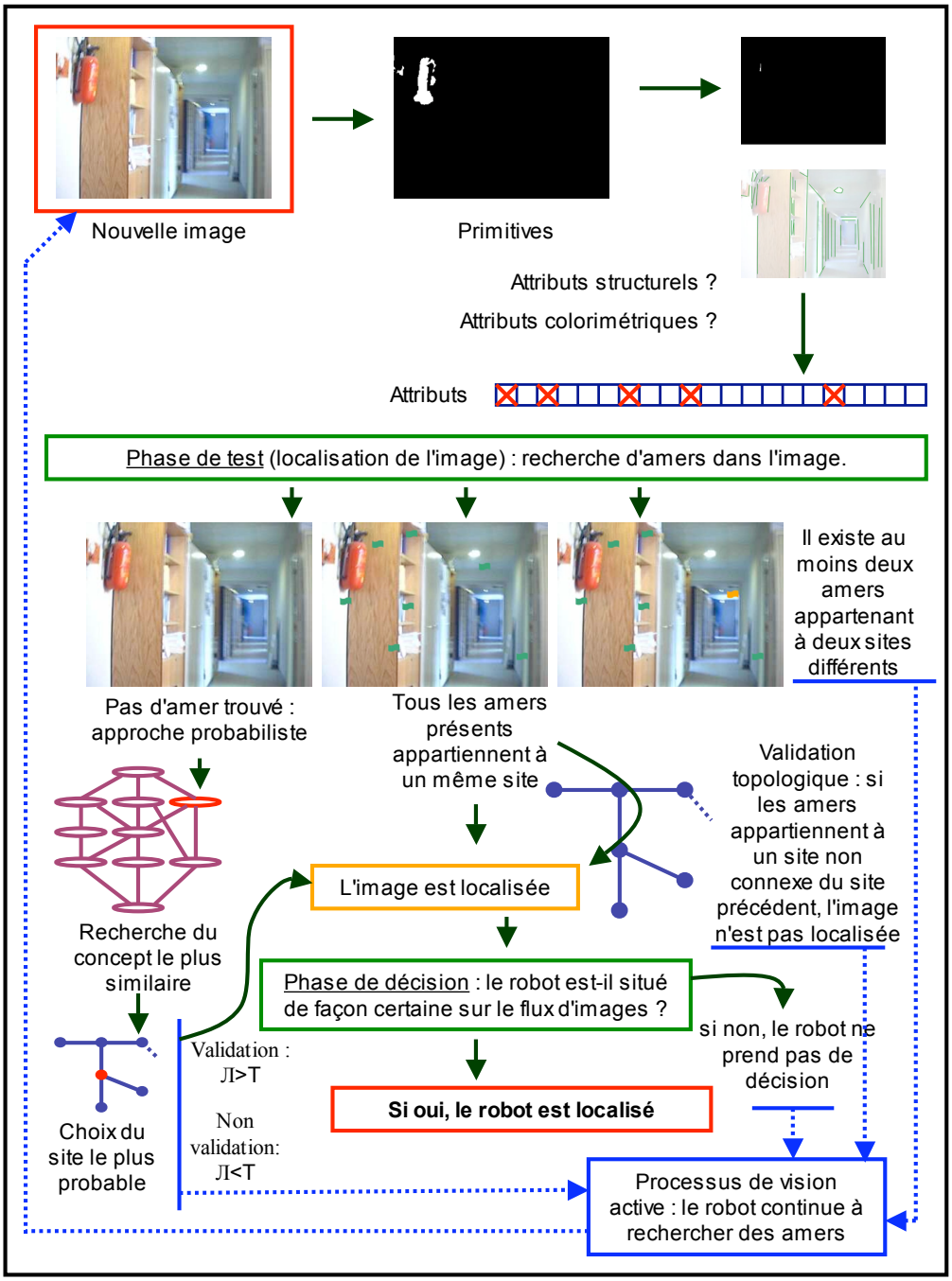


FIG. VI.20 – Les différentes stratégies de la phase de localisation.

Chapitre VII

Résultats Expérimentaux

Plusieurs expérimentations ont été faites dans les deux laboratoires. Les premières ont eu lieu au LAAS, et sont relatives à de petits contextes. En effet, le premier algorithme de construction de treillis utilisé fut celui de Carpineto & Romano, qui s'avéra plutôt lent dès lors que les expérimentations augmentaient en taille.

Les expérimentations suivantes ont eu lieu à SUPAERO avec des contextes plus grands. Le principal algorithme utilisé fut alors celui de Norris, et plus précisément celui de Norris-A, qui, dans l'état des expérimentations, nous convint parfaitement.

Ce chapitre est divisé en plusieurs sections que l'on peut regrouper en deux grandes parties. La première partie valide l'apport des treillis de Galois pour de la classification d'images à l'aide d'amers visuels. Une comparaison avec des réseaux de neurones et d'autres classifieurs "classiques" est menée, et l'apport des amers probabilistes est quantifié juste après.

La deuxième partie est consacrée à une application réelle au Laboratoire d'Informatique et d'Automatique de SUPAERO. La première approche utilisée est l'approche globale, suivie de l'approche locale. Dans les deux cas, les résultats sont donnés et analysés pour notre application.

VII.1 Classification d'Images et Localisation

La première utilisation de nos algorithmes fut appliquée à de la classification d'images, en dehors de tout aspect robotique. Ensuite un jeu d'images plus important issu du déplacement

de Diligent dans les couloirs de LAAS a été utilisé pour perfectionner notre classifieur. Cette dernière expérimentation a donné lieu à une comparaison avec d'autres classifieurs.

VII.1.1 Une première expérimentation

Cette première expérimentation était relative à un tout petit contexte : 15 images pour la base d'apprentissage, issues de 4 pièces différentes, et contenant potentiellement 50 attributs :

– peu de pixels (> 50), des pixels (> 300), beaucoup de pixels de rouge (> 3000), vert, bleu, cyan, magenta, jaune ($3 \times 6 = 18$ attributs) ; petit ($> 15 \times 15$), moyen ($> 50 \times 50$) ou gros ($> 75 \times 75$) objets R-V-B-C-M-J (18) ;

– un grand nombre (> 50), un très grand nombre (> 100) de grands (> 75 pixels) segments horizontaux, verticaux, ou de direction quelconque (2×3), un grand nombre, un très grand nombre de segment horizontaux, verticaux ou de même orientation ($2 + 2 + 2 = 6$) ; un grand nombre, un très grand nombre de segments de même taille (2).

▶ Phase d'apprentissage

La table de correspondance a été remplie et le treillis correspondant construit (algorithme de Carpineto & Romano [Carpineto 96]), figure VII.1.

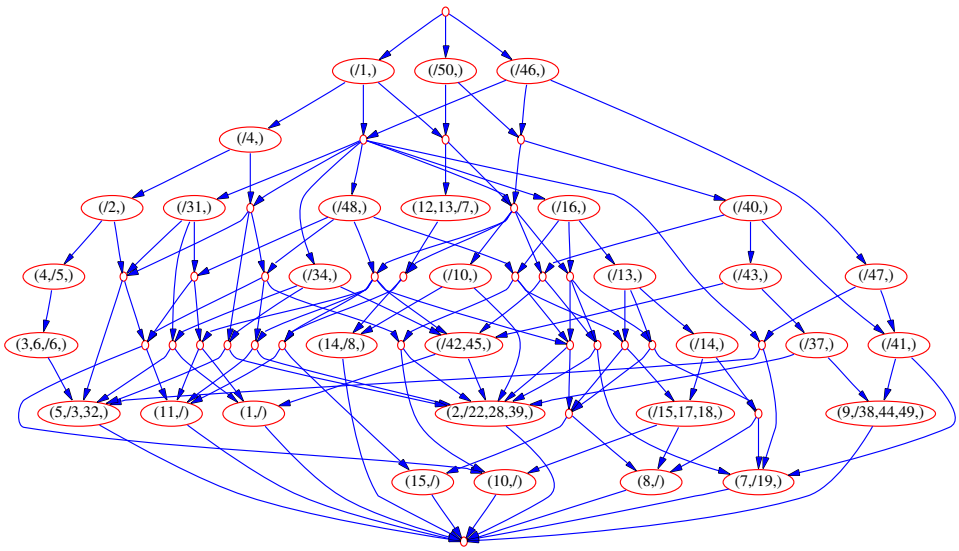


FIG. VII.1 – Treillis d'héritage global correspondant à notre application.

Le treillis représenté ici est le treillis d'héritage du contexte. Pour chaque ensemble d'images, les concepts correspondants ont été trouvés et l'on obtient (table VII.1 et figure VII.2) :

Lieu	**Amers pleins**	**Amers classiques**	**Amers maximaux**
Site n^o1	{#42.#45}	{#22.#28.#39}	{#42.#45}
Site n^o2	{#5}	{#6}, {#3.#32}	{#5}
Site n^o3	$\emptyset$	{#19}, {#41}, {#38.#44.#49}, {#15.#17.#18}, {#14}	{#41}, {#14}
Site n^o4	{#7}	{#8}	{#7}

TAB. VII.1 – Les différents amers

– pour le premier ensemble (images 1,2), {#42 et #45} est un amer plein et {#22.#28.#39} un amer partiel ; {#42 et #45} (*un grand nombre* et *un très grand nombre de segments horizontaux*) est un amer maximal de couverture totale ;
– pour le deuxième ensemble (images 3,4,5,6), {#5} est un amer plein et {#3.#32} et {#6} sont des amers partiels ; {#5} (*objet rouge moyen*) est l'amer maximum qui couvre ce site ;
– pour le troisième ensemble (images 7,8,9,10), pas d'amer plein mais quatre amers partiels : {#19}, {#41}, {#38.#44.#49}, {#15.#17.#18} et {#14}. Parmi ces amers partiel, l'amer {#14} couvre les amers {#19} et {#15.#17.#18}, et l'amer {#41} couvre les amers {#19} et {#38.#44.#49}. Ces amers ({#14} : *peu de pixels bleus* et {#41} : *un grand nombre de segments verticaux*) sont donc maximaux et l'on peut montrer également qu'ils forment une couverture totale ;
– et pour le dernier ensemble (images 11,12,13,14,15), {#7} est un amer plein et {#8} un amer partiel. {#7} (*des pixels verts*) est un amer maximal qui couvre ce site.

Il est *a priori* inutile de retenir tous les amers possibles. Aussi, s'il existe des amers pleins dans une classe, on éliminera les amers partiels. Dans le cas contraire, parmi les amers partiels, on ne retiendra que les amers partiels maximaux.

► Phase de test

Pour la phase de test, 20 nouvelles images ont été prises dans les même quatre pièces du laboratoire. 18 ont été correctement situées (c'est-à-dire qu'elles ne contenaient que des amers appartenant à la pièce correspondante), les deux dernières ne permettaient aucune conclusion.

► Treillis locaux

Afin de simplifier l'extraction d'amers, des treillis locaux ont été constitués à partir des contextes locaux (figure VII.3).

On constate que les treillis locaux sont bien plus lisibles, notamment pour l'extraction des "amers" maximaux. Cependant, ici, le fait de retrouver les mêmes amers relève du hasard, et

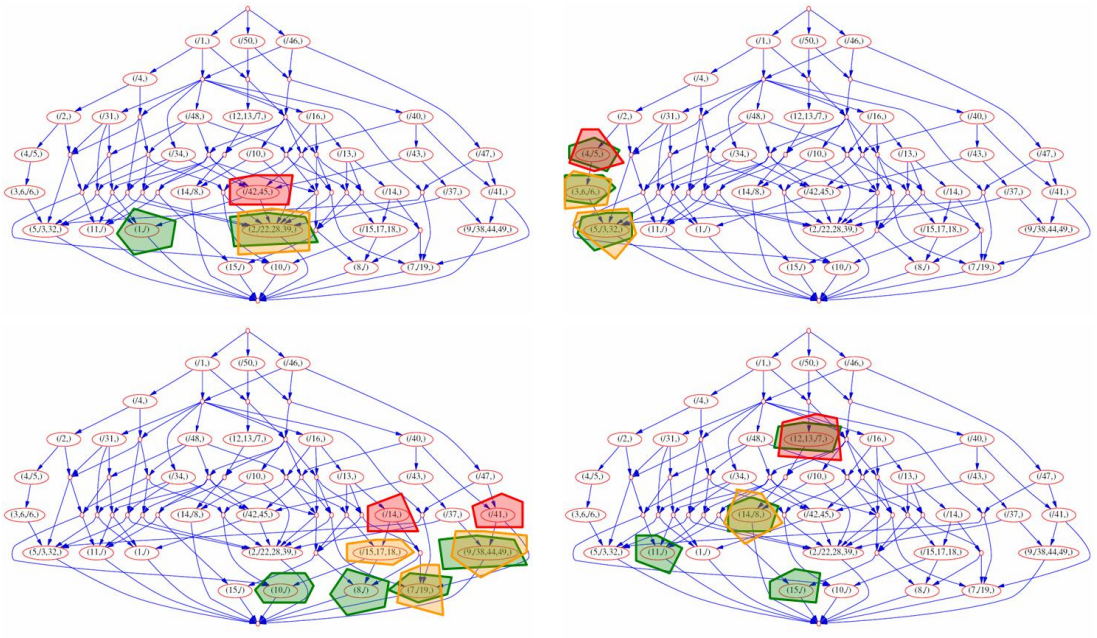


FIG. VII.2 – Sélection des amers. En vert sont les concepts relatifs à chaque classe ; en orange sont les concepts non maximaux et en rouge sont les concepts maximaux de chaque classe.

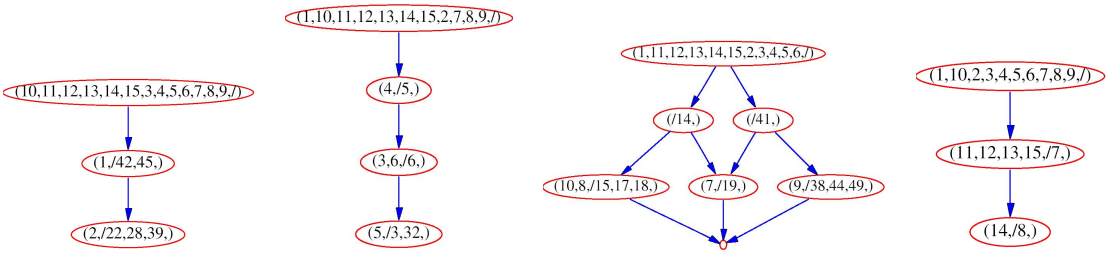


FIG. VII.3 – Treillis locaux.

dans la pratique il y a peu de chance que cette méthode permette d'extraire de bons amers, dans le sens où les amers partagés entre plusieurs classes ne sont plus différenciés.

VII.1.2 Expérimentation dans le couloir du LAAS

Une expérimentation de plus grande échelle a été menée dans les couloir du LAAS sur le robot Diligent, équipé d'une caméra couleur type "webcam". Le plan du lieu et les trajectoires approximatives sont exposés figure VII.4.

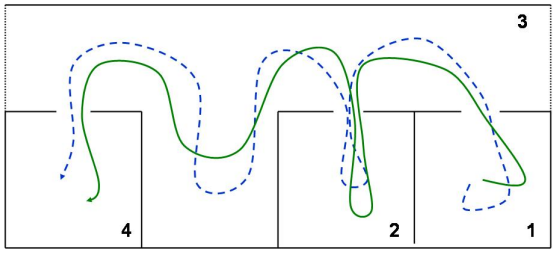


FIG. VII.4 – Plan d'expérimentation et trajectoires approximatives.

► Phase d'apprentissage

Le robot se déplace dans quatre pièces du laboratoire et acquiert les images (177 pour la phase de test) dans un flux continu (2 Hz). Le nombre d'attributs potentiels a été étendu à 66, en y incluant des contrastes couleurs et des informations de luminosité. Le robot obtient donc une table de 177×66 dont le treillis, qui comprend 5265 concepts, est construit en huit secondes sur une machine AMD Athlon 2400+. L'algorithme utilisé pour cette expérimentation est l'algorithme de Norris, plus rapide que le précédent.

Au total, 883 amers ont été extraits et 42 amers maximaux ont été retenus (tableau VII.2).

Lieu	**Amers pleins**	**Amers classiques**	**Amers maximaux**
Site n^o1	0	194	9
Site n^o2	0	316	8
Site n^o3	0	291	17
Site n^o4	0	82	8
Total	**0**	**883**	**42**

TAB. VII.2 – Extraction d'amers.

► Phase de test

Pendant la phase de test, 32 images différentes sont issues de la première pièce. Le robot cherche dans chacune d'elles des amers enregistrés lors de l'apprentissage. Pour le premier site, 14 images ne contiennent pas d'amer, une image contient deux amers ambigüs (un de la place 1, un de la place 3), une image contient un amer d'une autre pièce (une fausse réponse),

et 16 images contiennent uniquement des amers relatifs à cette pièce. On en déduit un taux de réponse de 53,1% (17/32), un taux de bonne réponse de 50% (16/32) dans l'absolu et de 94,1% (16/17) parmi les réponses, et enfin un taux d'erreur absolu de 3,13% et relatif de 5,88%. Tous les résultats sont regroupés tableau VII.3 (NI : nombre d'images ; NR : nombre de réponses ; NBR : nombre de bonnes réponses ; NFR : nombre de fausses réponses.)

Lieu	**NI**	**NR**	**NBR**	**NFR**
Site n^o1	32	17 (53,1%)	16 (50%)	1 (3,12%)
Site n^o2	50	13 (26%)	12 (24%)	1 (2%)
Site n^o3	31	10 (32,3%)	10 (32,3%)	0 (0%)
Site n^o4	38	20 (52,6%)	19 (50%)	1 (2,63%)
Total	**151**	**60 (39,8%)**	**57 (37,7%)**	**3 (1,99%)**

TAB. VII.3 – Résultats obtenus avec des amers classiques issus du treillis de Galois.

Le détail des résultats est fourni par le tableau suivant (figure VII.4).

Site réel	Sites choisis par le classifieur			
	Site n^o1	Site n^o2	Site n^o3	Site n^o4
Site n^o1	16	0	0	1
Site n^o2	0	12	1	0
Site n^o3	0	0	10	0
Site n^o4	1	0	0	19

TAB. VII.4 – Résultats détaillés obtenus avec des amers classiques issus du treillis de Galois.

Il est à noter que la phase de test s'effectue sur des images différentes de la phase d'apprentissage. Sur la base de test, le taux de bonnes réponses relatif est évidemment de 100%, mais le taux de réponses absolu est plus faible (88%, 43,1%, 85,7% et 54,8% respectivement pour chaque pièce). Ceci est dû essentiellement au fait qu'une image qui contient de l'information *a priori* n'en contient plus *a posteriori* , celle-ci étant commune à plusieurs pièces.

VII.2 Comparaison avec un Réseau de Neurones

VII.2.1 Choix du réseau

Afin de comparer ces résultats, un réseau de neurones a été mis en œuvre sous MATLAB composé de 66 neurones dans la couche d'entrée (correspondants aux 66 attributs), 66 neurones dans la deuxième et quatre neurones dans la couche de sortie (correspondants aux quatre pièces de l'environnement).

La fonction d'apprentissage est une descente de gradient avec rétro-propagation (`traingda`), avec une fonction de transfert de type tangente-sigmoïde pour chaque couche.

Le réseau ainsi construit est "optimal", dans le sens où de très nombreuses configurations différentes ont été testées afin d'obtenir le meilleur résultat possible. Les algorithmes de type *Levenberg-Marquardt* ou *régulation bayésienne*, plus efficaces en général, échouent ici principalement à cause d'un nombre d'entrées trop important.

VII.2.2 Résultats

Á chaque image d'entrée le réseau fournit une réponse qui correspond, à une normalisation près, à la probabilité d'appartenance à chaque pièce. Après convergence de l'apprentissage (correspondant à 700 "époques" environ), le meilleur taux de réponse obtenu est de 5% sur la base d'apprentissage, et de 30% sur la base de test. De plus, une très grande variabilité caractérise ces résultats : ces chiffres optimum ne sont donnés par le réseau qu'une fois sur dix environ.

Afin d'avoir une comparaison plus rigoureuse sur le plan scientifique, on donne la possibilité au réseau, comme dans notre technique, de ne pas répondre à une entrée. Dans la pratique, si la probabilité maximale d'être dans une pièce est, avec un seuil donné, supérieure à la probabilité d'être dans tout autre pièce, le résultat est validé. Ce seuil est calculé de façon adaptative afin d'avoir le même taux de "non-réponse". Ainsi, le réseau ne répond que s'il est relativement sûr de sa réponse. Les résultats présentés tableau VII.5, bien qu'optimaux pour un réseau de neurones, montrent une efficacité bien moins importante que l'approche que nous proposons ici.

Lieu	**NI**	**NR**	**NBR**	**NFR**
Site #1	32	17 (53,1%)	17 (53,1%)	0 (0%)
Site #2	50	17 (26%)	17 (26%)	0 (0%)
Site #3	31	14 (45,2%)	10 (32,2%)	4 (12,9%)
Site #4	38	12 (31,6%)	10 (26,3%)	2 (5,26%)
Total	**151**	**60 (39,8%)**	**54 (35,8%)**	**6 (3,98%)**

TAB. VII.5 – Résultats avec un réseau de neurones optimisé.

Le détail des résultats est fourni dans le tableau suivant (figure VII.6).

VII.2.3 Analyse

Avant toute chose, pour cette expérimentation, on constate que le nombre d'erreurs du classifieur à base de treillis est deux fois moins important que le classifieur à base de neurones.

Site réel	Sites choisis par le classifieur			
	Site n^o1	Site n^o2	Site n^o3	Site n^o4
Site n^o1	17	0	0	0
Site n^o2	0	17	0	0
Site n^o3	2	1	10	1
Site n^o4	0	0	2	10

TAB. VII.6 – Résultats détaillés avec un réseau de neurones optimisé.

La principale raison vient du grand nombre d'attributs en entrée du système. Le réseau de neurones ne peut utiliser d'algorithmes avancés de type "Levenberg-Marquardt" pour un grand nombre d'entrées, alors que le treillis extrait des amers quelque soit le nombre d'attributs des images.

Pour aller plus loin dans l'analyse, nous comparons ces classifieurs dans un espace de visualisation classique du *data mining*, le ROC space. La première étape est de construire les matrices de confusion dans lesquelles sont regroupés les résultats des deux classifieurs (tables VII.5 et VII.6).

Site 1	Total	Images classées site 1	Images non classées site 1
Images issues du site 1	32	16	1
Images non issues du site 1	119	1	42
Site 2	Total	Images classées site 2	Images non classées site 2
Images issues du site 2	50	12	1
Images non issues du site 2	101	0	47
Site 3	Total	Images classées site 3	Images non classées site 3
Images issues du site 3	31	10	0
Images non issues du site 3	120	1	49
Site 4	Total	Images classées site 4	Images non classées site 4
Images issues du site 4	38	19	1
Images non issues du site 4	113	1	39

FIG. VII.5 – Matrice de confusion du classifieur à base de treillis.

Ensuite, il s'agit de déterminer, pour chaque site, le taux de vrais positifs et le taux de faux positifs (Tables VII.7 et VII.8).

Enfin, il est possible de visualiser ces classifieurs dans l'espace de représentation "ROC Space" : pour chaque site, figure VII.9, ou en moyenne, figure VII.10.

On constate, avec un taux de vrais positifs quasi identique, que le taux de faux positifs du classifieur à neurones est deux fois plus important que celui du classifieur à treillis. De plus,

Site 1	Total	Images classées site 1	Images non classées site 1
Images issues du site 1	32	17	0
Images non issues du site 1	119	2	41
Site 2	Total	Images classées site 2	Images non classées site 2
Images issues du site 2	50	17	0
Images non issues du site 2	101	1	42
Site 3	Total	Images classées site 3	Images non classées site 3
Images issues du site 3	31	10	4
Images non issues du site 3	120	2	44
Site 4	Total	Images classées site 4	Images non classées site 4
Images issues du site 4	38	10	2
Images non issues du site 4	113	1	47

FIG. VII.6 – Matrice de confusion du classifieur à base de neurones.

Site	**Taux de vrais positifs**		**Taux de faux positifs**	
Site 1	16/32	50%	1/119	0,84%
Site 2	12/50	24%	0/101	0%
Site 3	10/31	32,3%	1/120	0,83%
Site 4	19/38	50%	1/113	0,88%
Total	57/151	37,7%	3/453	0,66%

FIG. VII.7 – Taux de vrais et faux positifs du classifieur à base de treillis.

Site	**Taux de vrais positifs**		**Taux de faux positifs**	
Site 1	17/32	53,1%	2/119	1,68%
Site 2	17/50	34%	1/101	0,99%
Site 3	10/31	32,3%	2/120	1,67%
Site 4	10/38	26,3%	1/113	0,88%
Total	54/151	35,8%	6/453	1,32%

FIG. VII.8 – Taux de vrais et faux positifs du classifieur à base de neurones.

malgré des classifications sans erreur pour les sites 1 et 2, il n'est pas un site ou le classifieur à neurones soit mieux positionné dans le "ROC Space" que le classifieur à treillis. La raison essentielle est que ce dernier fait moins d'erreurs en moyenne.

VII.3 Comparaison avec d'autres classifieurs

Les résultats sont comparés ici avec deux autres classifieurs classiques en analyse de données ($k$ppv) et traitement d'images (histogrammes). Dans le cas des histogrammes, le taux de non

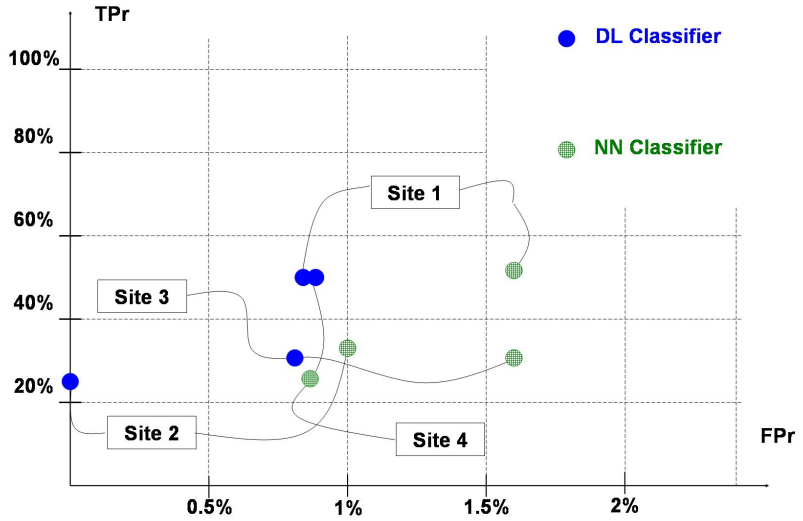


FIG. VII.9 – Représentation des classifieurs à base de treillis (*"DL Classifier"*) et à base de neurones (*"NN Classifier"*).

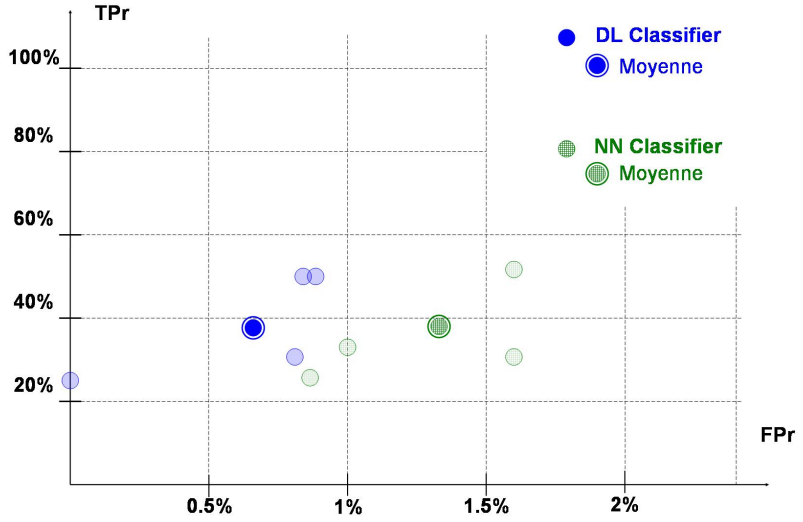


FIG. VII.10 – Représentation de la moyenne des classifieurs.

réponse est conservé. Dans le cas des $k$ppv, la distance choisie ne permet pas d'obtenir un taux de réponse de 63, leu plus proche étant de 78.

VII.3.1 Les k plus proches voisins

Ce classifieur se base sur l'algorithme des k plus proche voisins. La phase d'apprentissage consiste à enregistrer la liste des attributs des images de la base d'apprentissage. La phase de test consiste à comparer les attributs de la nouvelle image avec ceux des images de la base de test. Cette comparaison se fait par l'entremise d'une distance, définie, pour deux listes d'attributs quelconques, de la façon suivante :

$$\mathcal{D}(\mathcal{B}_1, \mathcal{B}_2) = \sum_{f=1}^{card(\mathcal{F})} (\mathcal{B}_1(f) \neq \mathcal{B}_2(f)) \tag{VII.1}$$

Cette distance consiste donc à compter le nombre d'attributs différents entre deux listes. Les résultats de cette analyse avec k=1 (qui donne de meilleurs résultats que k=2 ou k=3) sont donnés figure VII.7.

Site réel	Sites choisis par le classifieur			
	Site n^o1	Site n^o2	Site n^o3	Site n^o4
Site n^o1	10	2	0	0
Site n^o2	0	26	2	1
Site n^o3	0	12	2	1
Site n^o4	0	5	0	17

TAB. VII.7 – Résultats détaillés avec k=1.

On en déduit les matrices de confusion (figure VII.8) et les taux de vrais positifs et faux positifs (figure VII.11) pour chaque site.

1	Site n^o1	¬Site n^o1	2	Site n^o2	¬Site n^o2
Site n^o1	10	2	Site n^o2	26	3
¬Site n^o1	0	66	¬Site n^o2	19	30
3	Site n^o3	¬Site n^o3	4	Site n^o4	¬Site n^o4
Site n^o3	2	13	Site n^o4	17	5
¬Site n^o3	2	61	¬Site n^o4	2	54

TAB. VII.8 – Matrice de confusion avec k=1.

Ce classifieur est comparé au classifieur à base de treillis dans le "ROC space" figures VII.12 et VII.13

Site	Taux de vrais positifs		Taux de faux positifs	
Site 1	10/32	31,2%	0/119	0%
Site 2	26/50	52%	19/101	18,8%
Site 3	2/31	6,5%	2/120	1,7%
Site 4	17/38	44,7%	2/113	1,77%
Total	55/151	36,4%	23/453	5,1%

FIG. VII.11 – Taux de vrais et faux positifs avec k=1.

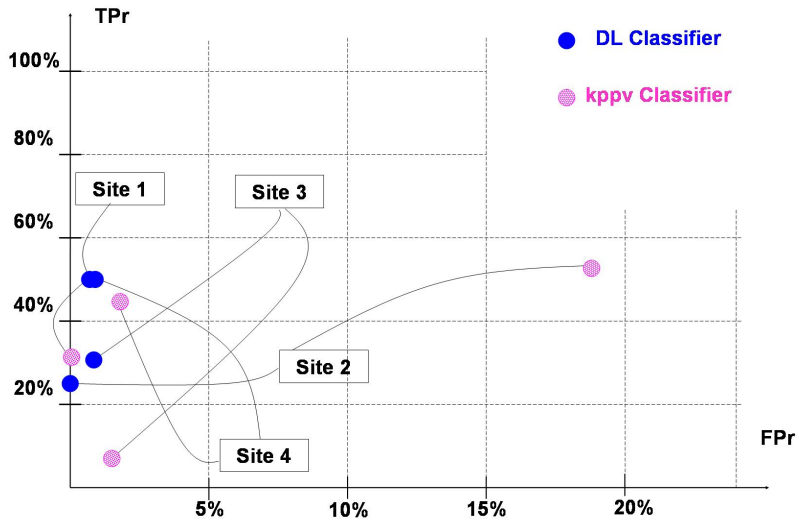


FIG. VII.12 – Représentation des classifieurs à base de treillis (*"DL Classifier"*) et par les $k$ppv (*"kppv Classifier"*).

VII.3.2 Les histogrammes

L'utilisation des histogrammes permet de travailler sur les images elle-même sans extraire au préalable des attributs particuliers dans l'image, sinon un histogramme.

L'histogramme en question est l'histogramme normalisé de la teinte. C'est l'histogramme le plus stable *a priori* pour décrire au mieux une image couleur. On oublie donc ici toute idée de structure dans l'image.

La phase d'apprentissage consiste à enregistrer les histogrammes de toutes les images. La phase de test consiste à comparer des distances entre les deux histogrammes. Nous avons défini une distance simple :

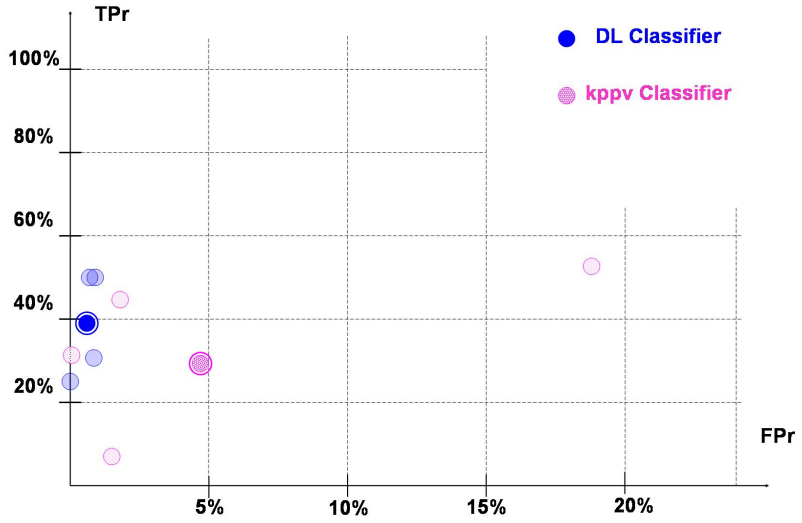


FIG. VII.13 – Représentation de la moyenne des classifieurs.

$$\mathcal{D}(\mathcal{H}_1, \mathcal{H}_2) = \sum_{n=0}^{N=255} |\mathcal{H}_1(n) - \mathcal{H}_2(n)| \tag{VII.2}$$

Les résultats sont fournis tableau VII.9 et la matrice de confusion tableau VII.10.

Site réel	Sites choisis par le classifieur			
	Site n^o1	Site n^o2	Site n^o3	Site n^o4
Site n^o1	2	0	0	0
Site n^o2	7	27	5	2
Site n^o3	1	1	8	1
Site n^o4	1	0	1	7

TAB. VII.9 – Résultats détaillés par comparaisons d'histogrammes.

On en déduit les taux de vrais positifs et les taux de faux positifs (figure VII.14), puis on représente l'ensemble dans le "ROC space" (figures VII.15 et VII.16, avec une échelle des abscisses différente de précédemment).

On constate que non seulement le taux TP_r est plus faible pour le classifieur à l'aide d'histogrammes, mais également que son erreur FP_r explose dans cette comparaison.

1	Site n^o1	¬Site n^o1	2	Site n^o2	¬Site n^o2
Site n^o1	2	0	Site n^o2	27	14
¬Site n^o1	9	52	¬Site n^o2	1	21
3	Site n^o3	¬Site n^o3	4	Site n^o4	¬Site n^o4
Site n^o3	8	3	Site n^o4	7	2
¬Site n^o3	6	46	¬Site n^o4	3	51

TAB. VII.10 – Matrice de confusion du classifieur à base d'histogrammes.

Site	**Taux de vrais positifs**		**Taux de faux positifs**	
Site 1	2/32	6,3%	0/119	0%
Site 2	27/50	54%	14/101	13,9%
Site 3	8/31	25,8%	3/120	2,5%
Site 4	7/38	18,4%	2/113	1,77%
Total	44/151	29,1%	19/453	4,19%

FIG. VII.14 – Taux de vrais et faux positifs du classifieur à base d'histogrammes.

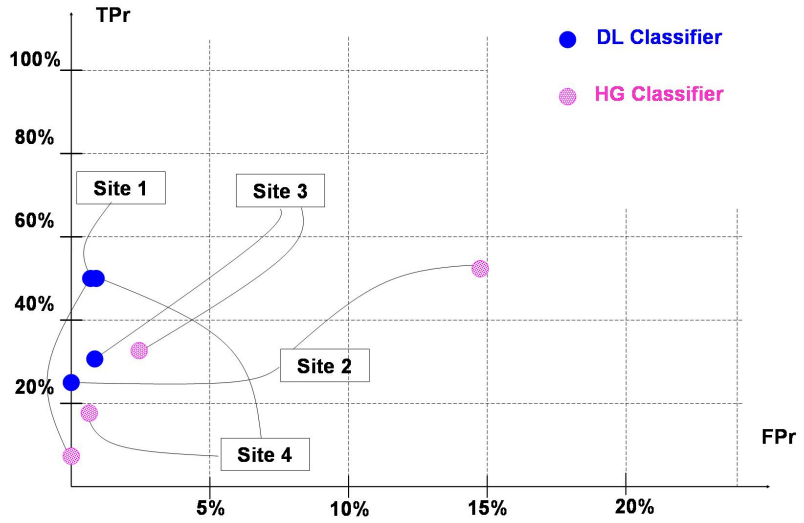


FIG. VII.15 – Représentation des classifieurs à base de treillis (*"DL Classifier"*) et à base d'histogrammes (*"HG Classifier"*).

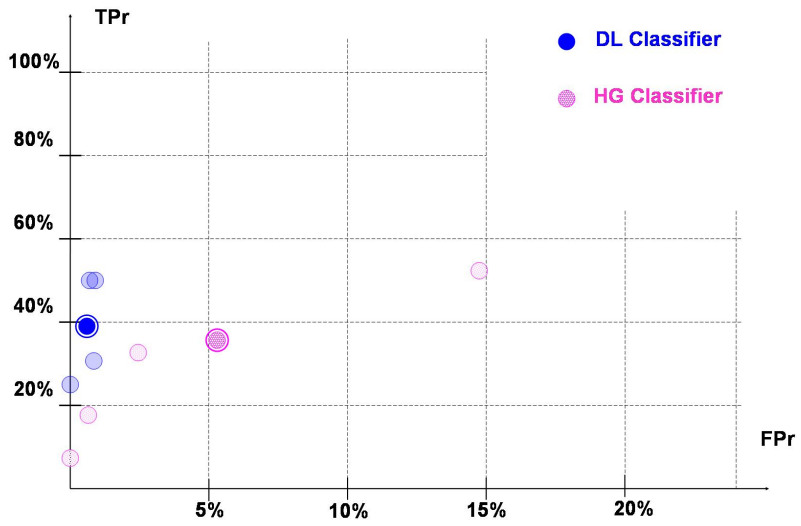


FIG. VII.16 – Représentation de la moyenne des classifieurs.

VII.4 Utilisation d'amers probabilistes

Les treillis probabilistes ont été introduits afin d'augmenter le nombre de réponses du classifieur. L'idée est la suivante : si le robot ne sait pas localiser une image, il cherche le concept le plus similaire et regarde sa probabilité d'appartenance à un site. Si celle-ci est supérieure à un seuil T (que nous avons fixé arbitrairement $T = 0,75$), alors le robot classe l'image dans le site en question. Les résultats sur le même jeu de test ont été les suivants (figure VII.11).

Place	NI	NR	NBR	NFR
Place #1	32	25 (78,1%)	24 (75%)	1 (3,13%)
Place #2	50	14 (28%)	12 (24%)	2 (4%)
Place #3	31	12 (38,7%)	12 (38,7%)	0 (0%)
Place #4	38	22 (57,9%)	21 (55,3%)	1 (2,63%)
Total	**151**	**73 (48,3%)**	**69 (45,7%)**	**4 (2,65%)**

TAB. VII.11 – Résultats avec des amers probabilistes.

Le détail des résultats est fourni par le tableau suivant (figure VII.12).

Les résultats sont conformes à nos attentes : 13 images de mieux ont été retenues (9,2 points de mieux, soit près de 19% d'images situées en plus), avec un taux d'erreur légèrement supérieur (2,65% au lieu de 1,99%).

Site réel	Sites choisis par le classifieur			
	Site n^o1	Site n^o2	Site n^o3	Site n^o4
Site n^o1	24	0	0	1
Site n^o2	1	12	1	0
Site n^o3	0	0	12	0
Site n^o4	1	0	0	21

TAB. VII.12 – Résultats détaillés avec des amers probabilistes.

La visualisation dans le "ROC Space" (figure VII.18) permet de tirer ensuite les conclusions suivantes : la première est que l'on garde le même ratio TP_r/FP_r, les moyennes étant alignées sur le même axe partant de l'origine ; ensuite, on reste dans une zone "intéressante" pour un classifieur permettant un taux de non-réponse ; enfin, le résultat reste bien meilleur que le classifieur à neurones.

Site	**Taux de vrais positifs**		**Taux de faux positifs**	
Site 1	24/32	75%	1/119	0,84%
Site 2	12/50	24%	2/101	1,98%
Site 3	12/31	38,7%	0/120	0%
Site 4	22/38	57,9%	1/113	0,88%
Total	70/151	46,3%	4/453	0,88%

FIG. VII.17 – Taux de vrais et faux positifs du classifieur probabiliste.

VII.5 Application Robotique : Apprentissage d'une partie du LIA de SUPAERO

Le robot *Pekee*, muni d'une caméra CCD couleur (140×180 pixels), est utilisé dans cette section afin de valider des résultats expérimentaux sur un robot réel.

L'environnement est une sous-partie du LIA. Il comporte sept sites distincts. Le robot, lors de la phase de test, est téléguidé pour explorer l'environnement. Il capture un flux d'images correspondant à environ 12 images par seconde. Ce flux dépend de l'encombrement de l'intranet. Toute l'analyse d'images et l'apprentissage se fait sur un ordinateur déporté.

Plusieurs expériences ont été menées, paramétrées selon un grand nombre de variables liées à l'image : la valeur des seuils dans la recherche des attributs, le nombre de bandes de couleurs, *etc.* Nous reproduisons ici la plus stable.

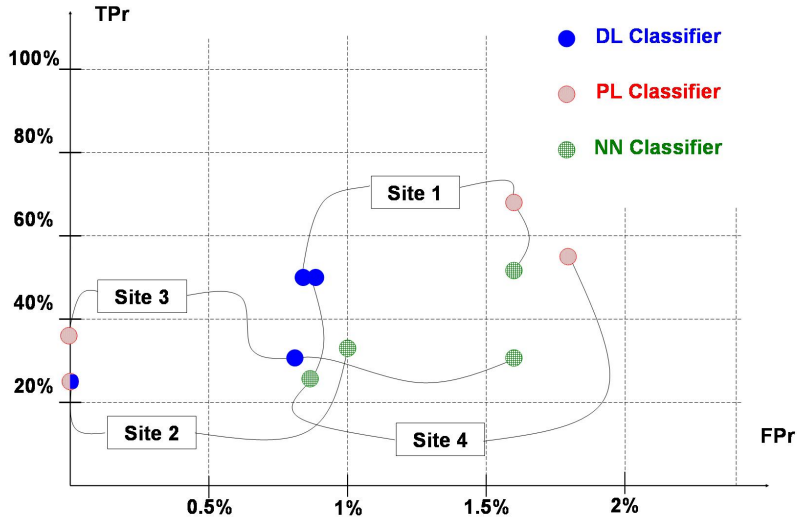


FIG. VII.18 – Représentation des classifieurs à base de treillis (*"DL Classifier"*), à base d'amers probabilistes (*"PL Classifier"*), et à base de neurones (*"NN Classifier"*).

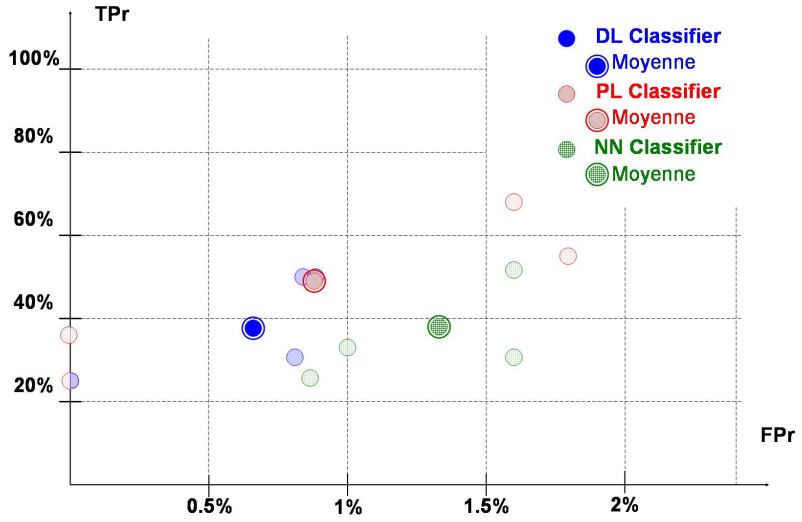


FIG. VII.19 – Représentation de la moyenne des classifieurs.

VII.5.1 Une interface pour la supervision

Une interface en VisualC++ a été développée afin de permettre une manipulation aisée du robot (figure VII.20).

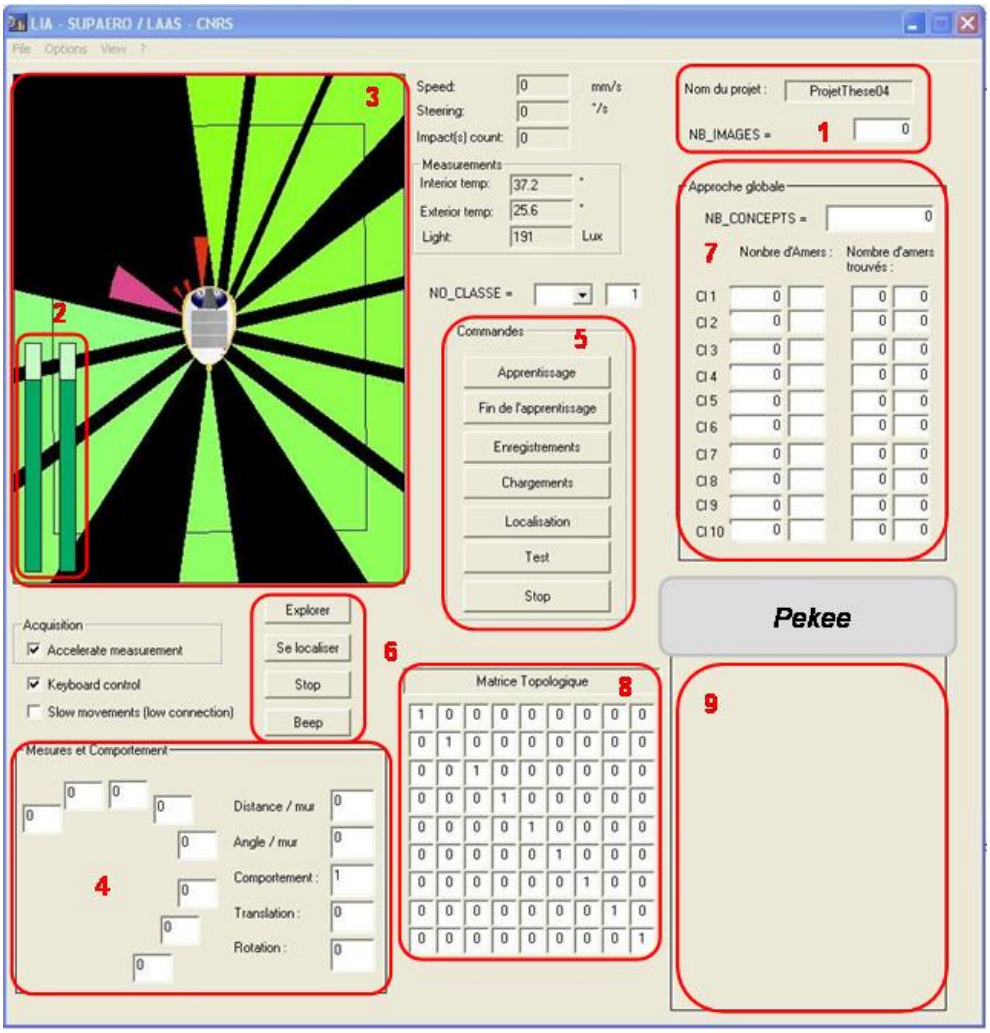


FIG. VII.20 – Une interface pour la supervision.

Cette interface comprend plusieurs zones :
– la zone 1 affiche le nom du projet et le nombre d'images en phase d'apprentissage ;
– la zone 2 montre le niveau de charge du robot de la base (à gauche) et du PC embarqué (à droite) ;
– la zone 3 montre *Pekee* dans son environnement, avec les lectures de distance fournies

par sa ceinture de capteurs infrarouges ;
– la zone 4 permet de connaître la valeur de ces distances -sur le flan droit- et certains paramètres intrinsèques (vitesse de rotation, vitesse de translation, type de comportement) ;
– les zones 5 et 6 sont des zones de contrôle du système (voir plus loin) ;
– La zone 7 est une zone d'affichage de données (nombre de concepts, nombre d'amers maximaux, nombre d'amers trouvés, et nombre cumulé d'amers trouvés) ;
– la zone 8 est la matrice topologique de l'environnement ;
– et la zone 9 n'est pas utilisée pour le moment.

Les boutons de contrôle sont les suivants :
– *"Apprentissage"* lance le processus d'apprentissage ;
– *"Fin de l'apprentissage"* le termine ;
– *"Enregistrements"* permet l'enregistrement des données, à savoir le(s) treilli(s), le(s) "mapping", les amers, et la carte topologique sous forme de fichiers ;
– *"Chargements"* permet le chargement complet de données enregistrées ;
– *"Localisation"* recherche les amers en temps réel dans chaque image ;
– *"Test"* lance une procédure de test quelconque ;
– *"Stop"* arrête tous les processus en cours ;
– *"Explorer"* lance un algorithme d'exploration de l'espace (voir plus loin) ;
– *"Se localiser"* permet au robot de se localiser en recherchant des amers, dans un processus de vision active ;
– *"Stop"* arrête tous les processus en cours ;
– et *"Beep"* fait sonner le robot.

VII.5.2 Les attributs utilisés

Deux types d'attributs on été utilisés dans cette application : des attributs colorimétriques et des attributs structurels. D'autres types d'attributs de plus haut niveau ont été testés, entre autre basés sur de la texture, mais la contrainte temps-réel ne permettait pas d'approfondir cette voie particulière. Les attributs utilisés sont donc ceux décrits section II.3, page 46.

► Attributs colorimétriques

Pour les aspects colorimétriques, la teinte a été divisée en 18 bandes se recouvrant chacune de 5% (pour chaque bord) les unes sur les autres. Une image est associée à chaque bande. Y sont ajoutées une image de noir (intensité inférieure à un seuil égal à 30) et une image de blanc (intensité supérieure à un seuil égal à 225).

Pour les 20 images, six attributs (voir page 51) sont détectés :
– le nombre de pixel est supérieur à 10,
– le nombre de pixel est supérieur à 500,
– le nombre de pixel est supérieur à 5000,
– s'il renferme un petit objet de taille 5×5,
– s'il renferme un moyen objet de taille 15×15,

– s'il renferme un grand objet de taille 30×30.

► Attributs structurels

D'un point de vue structurel, les segments issus de l'image sont analysés (voir page 47) et treize attributs sont détectés :
– un nombre de petits (inférieur à 20 pixels) segments supérieur à 50,
– un nombre de petits (inférieur à 20 pixels) segments supérieur à 100,
– un nombre de segments de même taille (à 5 pixels près) supérieur à 5,
– un nombre de segments de même taille (à 5 pixels près) supérieur à 15,
– un nombre de segments de même direction (à 9 degrés près) supérieur à 5,
– un nombre de segments de même direction (à 9 degrés près) supérieur à 15,
– un nombre de grands (supérieur à 50 pixels) segments supérieur à 5,
– un nombre de grands (supérieur à 50 pixels) segments supérieur à 15,
– un nombre de grands (supérieur à 50 pixels) segments verticaux supérieur à 5,
– un nombre de grands (supérieur à 50 pixels) segments verticaux supérieur à 15,
– un nombre de segments verticaux supérieur à 20,
– un nombre de segments verticaux supérieur à 50,
– et un nombre de contours supérieur à 50.

Le premier d'entre eux, à savoir la présence de petits segments dans une image, est caractéristique par exemple d'une plante verte (notemment s'il est associé à la bande de couleur "vert"). Le processus est illustré figure VII.21.

Notons que ces attributs sont relativement robustes (bien qu'encore loin de l'être fortement), bien plus que les primitives. Les seuils donnés ont été fixés en fonction de notre connaissance, il sera intéressant par la suite d'envisager un seuil adaptatif, selon la fréquence d'un attribut.

L'invariance angulaire est géré au niveau du nombre de segments de même orientation. Quelque soit cet angle, s'il y a plusieurs segments orientés dans le même sens, l'attributs correspondant sera activé. De même, l'invariance de distance est géré au niveau du nombre de segments ayant la même taille, quelque soit cette taille. Dans la pratique, par exemple, une bibliothèque se définit à partir d'un certain nombre de segments ayant la même taille et la même orientation. Ceci se vérifie quelque soit la position du robot relativement à la bibliothèque.

La robestesse est cependant très dépendante du matériel utilisé. La qualité moyenne de la caméra sur *Pekee* entraîne cependant du bruit sur les attributs détectés.

VII.6 Approche Globale

Dans cette section sont présentées les expérimentations relatives à l'approche globale. Le robot doit se repérer dans une partie du laboratoire.

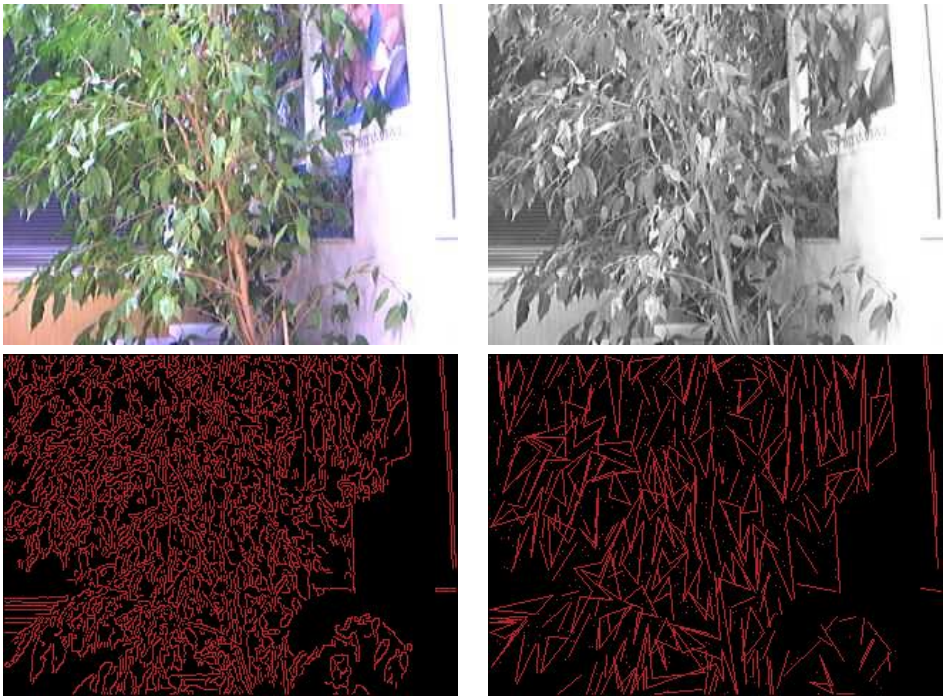


FIG. VII.21 – Extraction des segments de l'image représentant une plante verte.

VII.6.1 Phase d'apprentissage

La phase d'apprentissage consiste à piloter manuellement le robot dans un environnement donné (il n'y a pas au stade actuel d'algorithmes permettant au robot *Pekee* de se déplacer totalement seul dans un environnement structuré). Le robot se déplace comme le montre la figure VII.22.

Lors de cette phase d'apprentissage, le robot capture les images, recherche les attributs présents, construit le treillis global, et les amers sont extraits toutes les 10 images. En fin d'apprentissage, les amers sont de nouveau extraits et l'ensemble des données est enregistré.

Au total 735 images ont été utilisées pour les sept sites. Le treillis correspondant est un treillis de 108471 concepts. Le nombre d'amers maximaux correspondant à chaque site est donné tableau VII.13.

	Site n^o1	Site n^o2	Site n^o3	Site n^o4	Site n^o5	Site n^o6	Site n^o7
Amers max.	21	13	15	36	16	18	25

TAB. VII.13 – Nombre d'amers maximaux correspondant à chaque site.

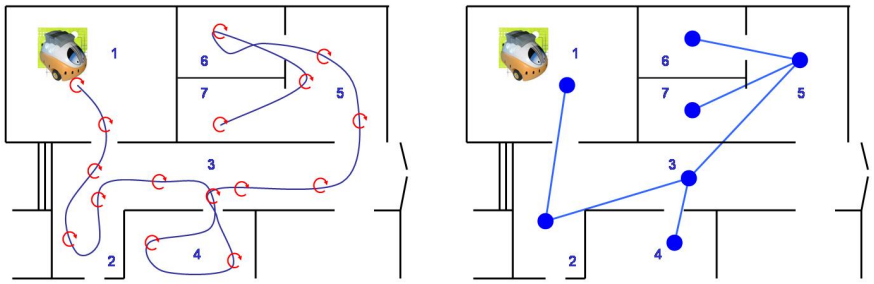


FIG. VII.22 – Topologie et changement de site.

VII.6.2 Phase de test

La phase de test consiste à rechercher des amers dans une image. Une image sera située si et seulement si tous les amers présents n'appartiennent qu'à un seul et même site.

Les résultats sont présentés tableau VII.14. Trois essais ont été menés pour chaque site de localisation. Nous nous sommes arrêtés sur 100 images à localiser pour chaque site.

Site	Images localisées	Temps moyen	Nombre d'images	Erreurs
Site n^o1	100	116s	580	8
Site n^o2	100	98,8s	494	3
Site n^o3	100	72s	360	1
Site n^o4	100	26,8s	134	9
Site n^o5	100	21,6s	108	10
Site n^o6	100	21,6s	108	8
Site n^o7	100	30s	141	13

TAB. VII.14 – Résultats de la phase de test.

On remarque ainsi qu'il existe des images mal localisées dans ce processus. Il y a deux principales raisons à notre avis :

– la première raison est que le processus d'apprentissage n'est jamais parfait, notamment lorsque la base d'apprentissage sont des images prises par le robot en flux continu ;

– la seconde raison est le choix d'attributs simples, donc peu robustes, afin de garder un processus temps réel.

Cependant, pour la localisation du robot lui-même, ces *imprécisions* n'ont pas eu de conséquences négatives.

VII.6.3 Phase de décision

La phase de décision consiste à localiser le robot. Le robot est localisé de façon certaine s'il trouve 20 images localisées dans un même site. Ensuite, les compteurs d'images situées passent à zéro, et le processus recommence. Le robot donne donc sa localisation toutes les 20 images situées dans un même site. Ce site "gagnant" est le site de localisation.

Le processus est un processus de vision active, selon une trajectoire définie section VI.4.5.

VII.6.4 Algorithme général

L'algorithme général utilisé est celui présenté figure VII.23. Il reprend les notions décrites précédemment.

VII.6.5 Résultats

Le tableau VII.15 détaille les résultats de l'un des trois essais de localisation. Chaque tableau, pour chaque processus de localisation, correspond à une "photo" prise à l'instant où le robot trouve 20 images issues d'un seul site. On a alors le nombre d'images localisées pour le site en question, et le total d'images localisées dans tous les autres sites. Par exemple, dans le premier site, pour la première localisation, dès que le robot a trouvé 20 images d'un même site, il a trouvé 20 images localisées dans le site 1 et 3 images localisées ailleurs. Sa décision consiste donc à "affirmer" qu'il est dans le site $n^o 1$.

On constate dans ce tableau, malgré des erreurs de localisation de l'image, que le robot se localise quant à lui de façon robuste. Il est donc à même de trouver ses propres amers et de se localiser correctement.

En annexe 1 est présenté quelques images d'écran prises lors de la localisation du robot dans la pièce 2. On y voit l'évolution du robot lors de sa localisation, la localisation des images (avec trois erreurs), jusqu'à ce que la décision soit prise : le robot est dans le site 2.

Une remarque importante est à noter lors de la localisation. Le robot se localise (détermine sa position topologique) à l'aide d'images qu'il "voit", et non d'images du lieu où il se situe. Or dans certains cas, même s'il se trouve dans un site donné, il lui arrive de "voir" le site d'à coté au travers par exemple d'une porte entrebâillée. Dans ce cas, une *erreur de localisation* reste à définir. De plus, lors de changements de site, il n'est pas toujours facile de savoir localiser une image.

VII.7 Approche Locale

Comme nous l'avons vu au chapitre précédent, l'approche locale permet de décentraliser les treillis et d'avoir une plus grande souplesse dans la "distribution" des amers potentiels.

Site n^o1	Site n^o1	¬Site n^o1	Décision
Localisation n^o1	20	3	Site n^o1
Localisation n^o2	20	4	Site n^o1
Localisation n^o3	20	1	Site n^o1
Localisation n^o4	20	0	Site n^o1
Localisation n^o5	20	0	Site n^o1
Erreur de localisation	-	-	0
Site n^o2	Site n^o2	¬Site n^o2	Décision
Localisation n^o1	20	3	Site n^o2
Localisation n^o2	20	0	Site n^o2
Localisation n^o3	20	0	Site n^o2
Localisation n^o4	20	0	Site n^o2
Localisation n^o5	20	0	Site n^o2
Erreur de localisation	-	-	0
Site n^o3	Site n^o3	¬Site n^o3	Décision
Localisation n^o1	20	1	Site n^o3
Localisation n^o2	20	0	Site n^o3
Localisation n^o3	20	0	Site n^o3
Localisation n^o4	20	0	Site n^o3
Localisation n^o5	20	0	Site n^o3
Erreur de localisation	-	-	0
Site n^o4	Site n^o4	¬Site n^o4	Décision
Localisation n^o1	20	0	Site n^o4
Localisation n^o2	20	1	Site n^o4
Localisation n^o3	20	8	Site n^o4
Localisation n^o4	20	0	Site n^o4
Localisation n^o5	20	0	Site n^o4
Erreur de localisation	-	-	0
Site n^o5	Site n^o5	¬Site n^o5	Décision
Localisation n^o1	20	2	Site n^o5
Localisation n^o2	20	2	Site n^o5
Localisation n^o3	20	3	Site n^o5
Localisation n^o4	20	2	Site n^o5
Localisation n^o5	20	1	Site n^o5
Erreur de localisation	-	-	0
Site n^o6	Site n^o6	¬Site n^o6	Décision
Localisation n^o1	20	3	Site n^o6
Localisation n^o2	20	0	Site n^o6
Localisation n^o3	20	0	Site n^o6
Localisation n^o4	20	0	Site n^o6
Localisation n^o5	20	6	Site n^o6
Erreur de localisation	-	-	0
Site n^o7	Site n^o7	¬Site n^o7	Décision
Localisation n^o1	20	6	Site n^o7
Localisation n^o2	20	3	Site n^o7
Localisation n^o3	20	0	Site n^o7
Localisation n^o4	20	4	Site n^o7
Localisation n^o5	20	0	Site n^o7
Erreur de localisation	-	-	0

TAB. VII.15 – Résultats de la phase de décision.

I. Apprentissage()
I.1. Pour chaque nouvelle image d'apprentissage
I.1.1. Extraire les primitives présentes dans cette image
I.1.2. Déterminer la présence ou non des attributs dans l'image
I.1.3. Mettre à jour le treillis de Galois
I.1.4. Extraire les amers visuels du treillis

II. Test()
II.1. Pour chaque nouvelle image de test
II.1.1. Extraire les primitives présentes dans cette image
II.1.2. Déterminer la présence ou non des attributs dans l'image
II.1.3. Rechercher la présence d'amers dans cette image
II.1.4. S'il n'y a pas d'amer, alors on recherche des amers probabilistes
II.1.5. S'il n'y a que des amers appartenant à un seul et même site, alors l'image est localisée dans le site en question
II.1.6. S'il y a au moins deux amers appartenant à deux sites différents, alors l'image n'est pas localisée

III. DécisionAvecVisionActive()
III.1. Tant que le robot n'est pas localisé de façon certaine
III.1.1. Bouger dans l'environnement afin de trouver une nouvelle image de test
III.1.2. Si le temps de test est supérieur à un temps donné
III.1.1.1. Le robot n'est pas localisé
III.1.1.2. Reprendre l'apprentissage
III.1.1.3. Sortie du programme
III.2. Le robot est localisé

FIG. VII.23 – Algorithme général (approche globale).

VII.7.1 Phase d'apprentissage

Dans un soucis de comparaison, la phase d'apprentissage a été la même que précédemment. Nous avons utilisé les mêmes images d'apprentissage, donc le même *"mapping"*. Pour chaque nœud un treillis local de voisinage d'ordre 1 est construit et les amers extraits, selon le processus décrit section VI.7 page 152.

VII.7.2 Phase d'apprentissage

L'algorithme général est présente figure VII.24

I. Apprentissage()
I.1. Pour chaque nouvelle image d'apprentissage
I.1.1. Extraire les primitives présentes dans cette image
I.1.2. Déterminer la présence ou non des attributs dans l'image
I.1.3. Mettre à jour les treillis locaux d'ordre 0
I.1.4. ou Mettre à jour les *"mapping"* locaux d'ordre 0
I.2. En fin d'apprentissage
I.2.1. Construire les treillis locaux
I.2.2. Extraire les amers visuels locaux de chaque treillis

II. Test()
II.1. Pour chaque nouvelle image de test
II.1.1. Extraire les primitives présentes dans cette image
II.1.2. Déterminer la présence ou non des attributs dans l'image
II.1.3. Rechercher la présence d'amers dans cette image
II.1.4. S'il n'y a pas d'amer, alors on recherche des amers probabilistes
II.1.5. S'il n'y a que des amers appartenant à un seul et même site, alors l'image est localisée dans le site en question
II.1.6. S'il y a au moins deux amers appartenant à deux sites différents, alors l'image n'est pas localisée

III. DécisionAvecVisionActive()
III.1. Tant que le robot n'est pas localisé de façon certaine
III.1.1. Bouger dans l'environnement afin de trouver une nouvelle image de test
III.1.2. Si le temps de test est supérieur à un temps donné
III.1.1.1. Le robot n'est pas localisé
III.1.1.2. Reprendre l'apprentissage
III.1.1.3. Sortie du programme
III.2. Le robot est localisé

FIG. VII.24 – Algorithme général (approche locale).

VII.7.3 Résultats

Le tableau VII.16 montre le nombre de concepts et le nombres d'amers locaux attachés à chaque nœud, illustré figure VII.25.

Le premier constat est que le nombre d'amers pour chaque nœud est plus important que précédemment. Ensuite, le nombre total de concepts est de 150244, ce qui est supérieur à l'approche globale (108471). Ceci s'explique par le fait que les attributs d'une image sont "utilisés" plusieurs fois, à savoir pour le nœud dans lequel l'image est attachée mais également les nœuds

	Site n^o1	Site n^o2	Site n^o3	Site n^o4	Site n^o5	Site n^o6	Site n^o7
Concepts	22777	33323	32964	11314	31682	8877	9301
Amers max.	46	78	81	65	84	31	53

TAB. VII.16 – Nombre d'amers maximaux correspondant à chaque site.

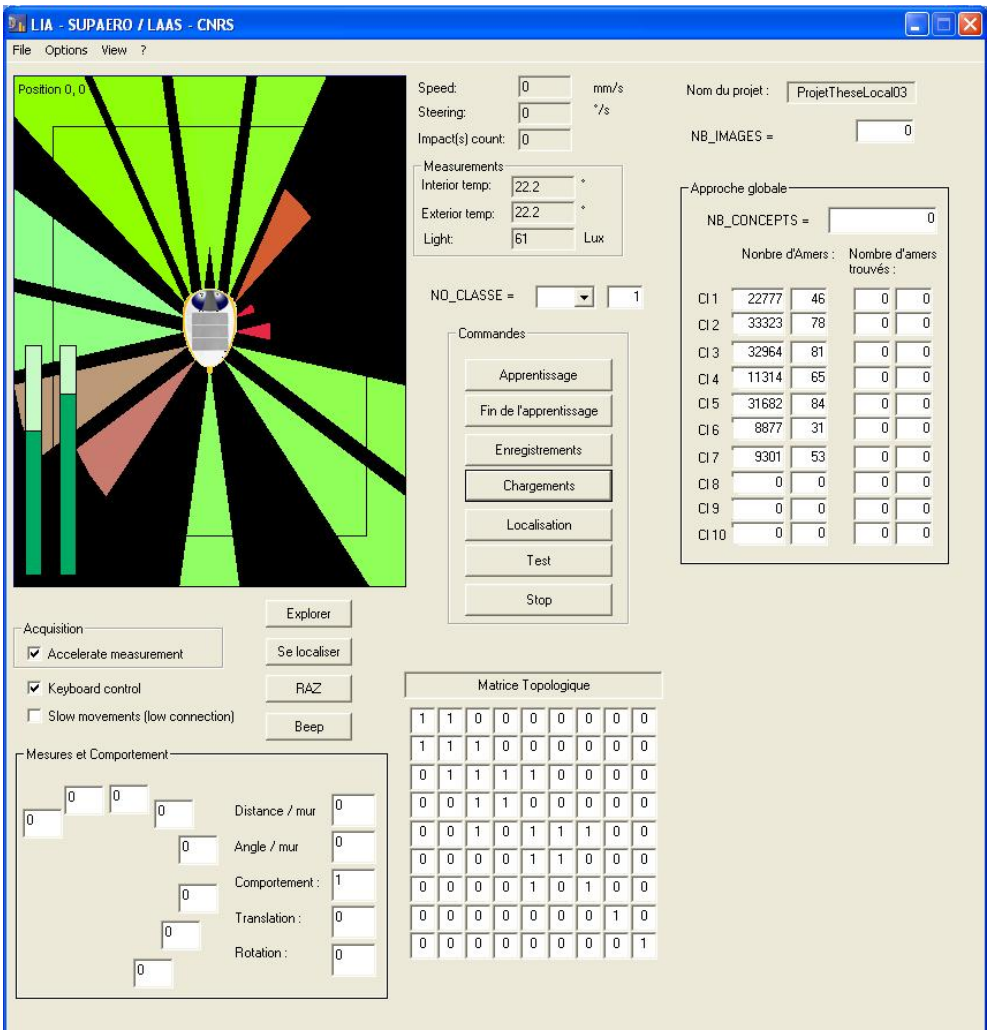


FIG. VII.25 – Début de la phase de localisation en approche locale.

voisins. Notre expérience n'est pas assez grande (en nombre de nœuds) pour constater cet avantage. Cependant, chaque treillis est bien plus petit que le "grand" treillis global, ce qui permet une manipulation plus rapide des concepts pour la recherche d'amers.

VII.7.4 Phases de Test et de décision

La phase de test est identique à précédemment. La phase de décision a été "raccourcie" et la décision se prend sur 10 images, et non plus 20, afin d'être plus réactif. L'inconvénient est une très grande sensibilité aux erreurs de test.

Le robot localise ainsi les images et se localise lui-même tout au long de son déplacement, en flux continu. Dans le tableau VII.17, sont donnés pour chaque sité visité par le robot (parcours), le nombre d'images concernées, le nombre d'image localisées, le nombre de décisions prises (ensemble de 10 images localisées à un même endroit), et le nombre d'images localisées *ailleurs*.

Parcours	Temps dans le site	Temps total	Nombre de décisions	Nombre d'images mal localisées
Site n^o1	49s	49s	4	3
Site n^o2	43s	1 :32s	3	2
Site n^o3	44s	2 :16s	1	0
Site n^o4	47s	3 :03s	11	3
Site n^o3	46s	3 :49s	3	1
Site n^o5	33s	4 :21s	3	4
Site n^o7	32s	4 :53s	3	2
Site n^o5	24s	5 :17s	2	1
Site n^o6	32s	5 :49s	3	2

TAB. VII.17 – Nombre d'amers maximaux correspondant à chaque site.

On remarque comme précédemment que le site 3 est pauvre en amers. Ceci s'explique par deux raisons : la première raison est que ce site est un couloir pauvre en informations visuelles. La seconde raison est que le nœud correspondant est relié à trois autres nœuds, et donc le nombre d'amers à "partager" est plus faible.

En revanche, le site 4 est très riche en information et donc le robot localise un très grand nombre d'images. De plus, contrairement à précédemment, ce nœud n'est relié qu'au nœud n^o2, pauvre en amers, donc le site 4 est très riche en amers visuels.

Chapitre VIII

Conclusion et Perspectives

Caractériser un environnement structuré au sens topologique et se localiser dans cet environnement est une problématique qui soulève de nombreuses questions : comment soi-même se repère-t-on dans un environnement ? De quels éléments visuels se sert-on pour appréhender son environnement ? Que retient-on d'une scène ?

Même si notre approche n'est pas directement inspirée des sciences cognitives, ces questions se sont posées tout au long de cette recherche. Ces questions sont essentielles dans le monde de la robotique mobile, puisque aucune machine n'est capable à ce jour de concurrencer les capacités humaines de localisation dans un environnement de grande échelle.

Reste cependant l'idée de combiner des attributs visuels pour caractériser l'environnement. Caractériser certaines images à l'aide de combinaisons d'attributs est directement lié à la notion de concepts, et l'organisation structurelle et hiérarchique de ces concepts nous a mené vers le formalisme que nous avons introduit dans le monde de l'image et de la robotique mobile : les treillis de concepts, ou treillis de Galois.

Après une introduction à la vision humaine et à la vision par ordinateur, suivie d'un tour d'horizon de la navigation et de la localisation à l'aide d'amers en robotique mobile et d'une introduction générale à l'apprentissage, nous avons présenté les treillis de Galois et les algorithmes de construction associés.

Nous avons dans un premier temps modifié ces algorithmes afin de pouvoir classer des objets (images) de façon incrémentale. Ensuite, nous avons introduit un aspect probabiliste dans ces treillis puis défini la notion d'amers visuels à partir des concepts du treillis. Nous avons introduit la notion d'amer probabiliste puis développé les treillis locaux dans une topologie liée à l'environnement.

Enfin nous avons validé, au sein du Laboratoire d'Informatique et d'Automatique de SUPAERO, ces notions et les algorithmes associés dans une application réelle de robotique, qui permet à un robot mobile de construire une représentation topologique de l'environnement et de se localiser par la suite.

Les perspectives futures liées à ce travail sont nombreuses, nous en exposons ici quelque unes.

FIG. VIII.1 – Mars Exploration Rover

La première perspective serait d'améliorer les attributs visuels utilisés et d'en définir de plus haut niveau ; notre système est évolutif et permet d'introduire d'autres attributs, et l'utilisation d'attributs de plus haut niveau et plus stables permettrait plus de robustesse dans la localisation des images.

Une deuxième perspective serait de monter ces algorithmes sur des robots plus évolués et d'utiliser une caméra montée sur un "pan-tilt"[1]. Ceci permettrait de sélectionner dans le champ visuel des zones attentionnelles (points d'intérêt [Harris 88, Schmid 96], maxima locaux de densité de contours par exemple) et limiter ainsi l'espace de recherche d'images caractéristiques.

Un autre avantage de l'utilisation d'une caméra montée sur un "pan-tilt" est l'introduction d'attributs particuliers dans les concepts comme l'orientation de la caméra. De cette manière, il serait intéressant de localiser les amers dans l'espace. Par exemple, telle pièce pourrait être caractérisée par son plafond, si le concept-amer correspondant indique que le "pan-tilt" est orienté vers le haut. Autre exemple, si le robot est muni d'une boussole, on peut associer un amer un point cardinal ou un azimut. On peut imaginer ainsi associer comme attribut des données intrinsèques et extrinsèques au robot. C'est l'avantage d'un système "ouvert" comme le nôtre : il permet de prédéfinir tout type d'attributs potentiels sans changer les algorithmes développés dans ce mémoire.

Dans le cas de données intrinsèques ou extrinsèques relatives à une position ou une orientation du robot, la recherche d'amers peut alors se faire de façon plus "active" que dans notre application. Si les amers de telle pièce sont *a priori* au plafond, il suffit au robot de chercher au plafond si des amers sont présent. La notion de vision active prendrait ainsi un sens plus général, et les processus de recherche d'amers seraient optimisés.

Enfin, une dernière perspective intéressante est d'introduire une notion d'apprentissage

[1] banc qui permet d'orienter la caméra par rapport au robot

"semi-supervisé", dans le sens où, lors de l'apprentissage, on fournirait à la machine les *changements de classes*, et non plus les classes elles-mêmes. Dans la pratique, ceci reviendrait à détecter les changements de sites par le robot. Si les sites sont des pièces d'un environnement structuré, ceci reviendrait à repérer -à l'aide d'une ceinture d'ultrasons par exemple- les passages de porte. Toute une théorie sur l'apprentissage semi-supervisé serait à explorer, celle-ci n'existant pas à notre connaissance. Elle permettrait une cartographie topologique complète et autonome du robot d'un environnement structuré.

Annexe A

Images de la localisation du robot

Ici sont présentés des écrans relatifs à la localisation du robot dans la pièce 2.

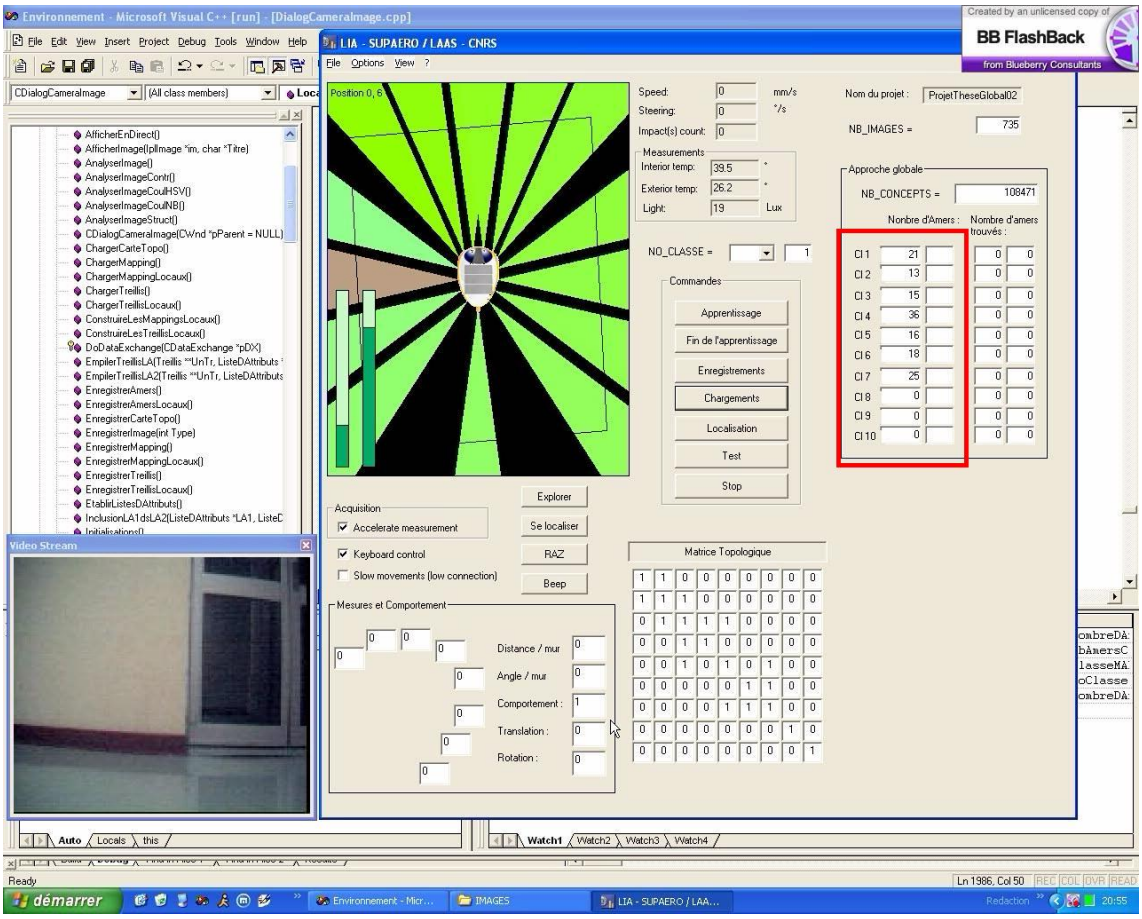


FIG. A.1 – Image 1 : début de la localisation.

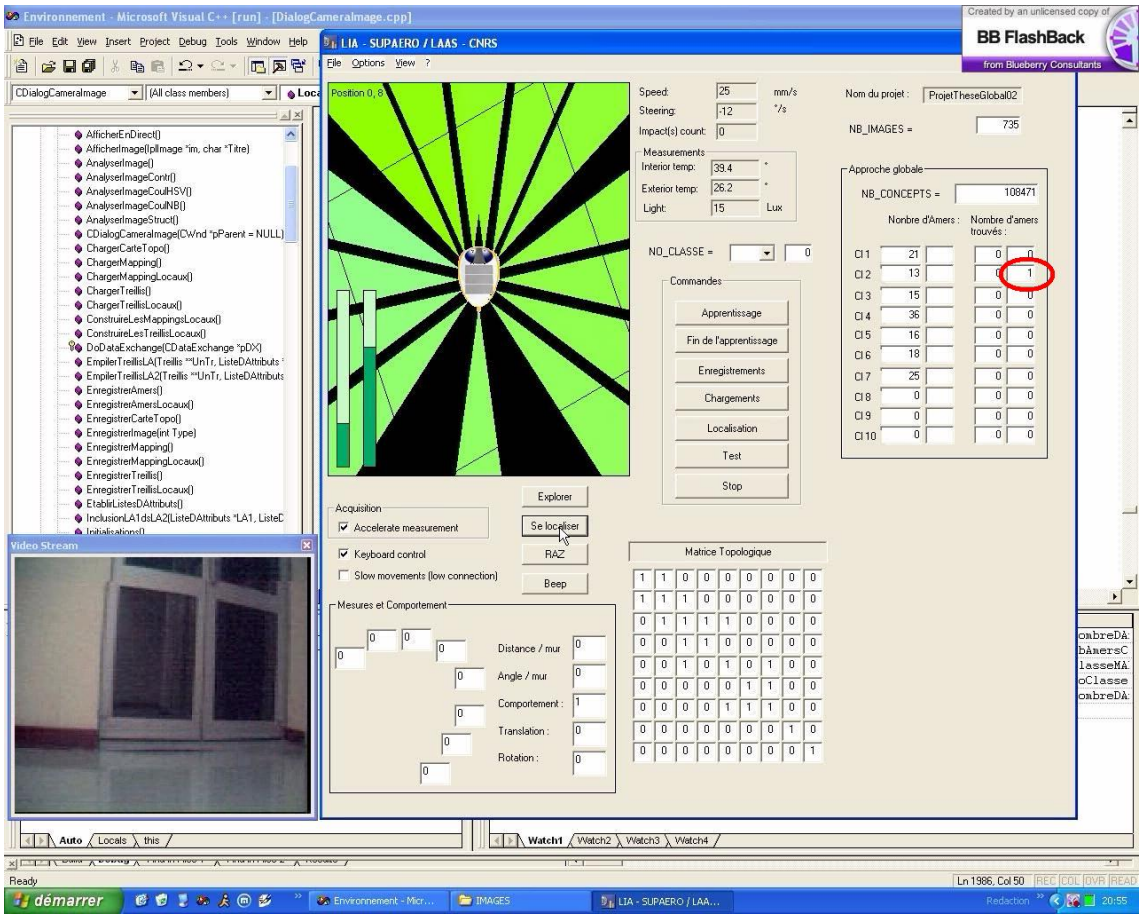


FIG. A.2 – Image 2 : le robot à repéré un amer de la pièce 2.

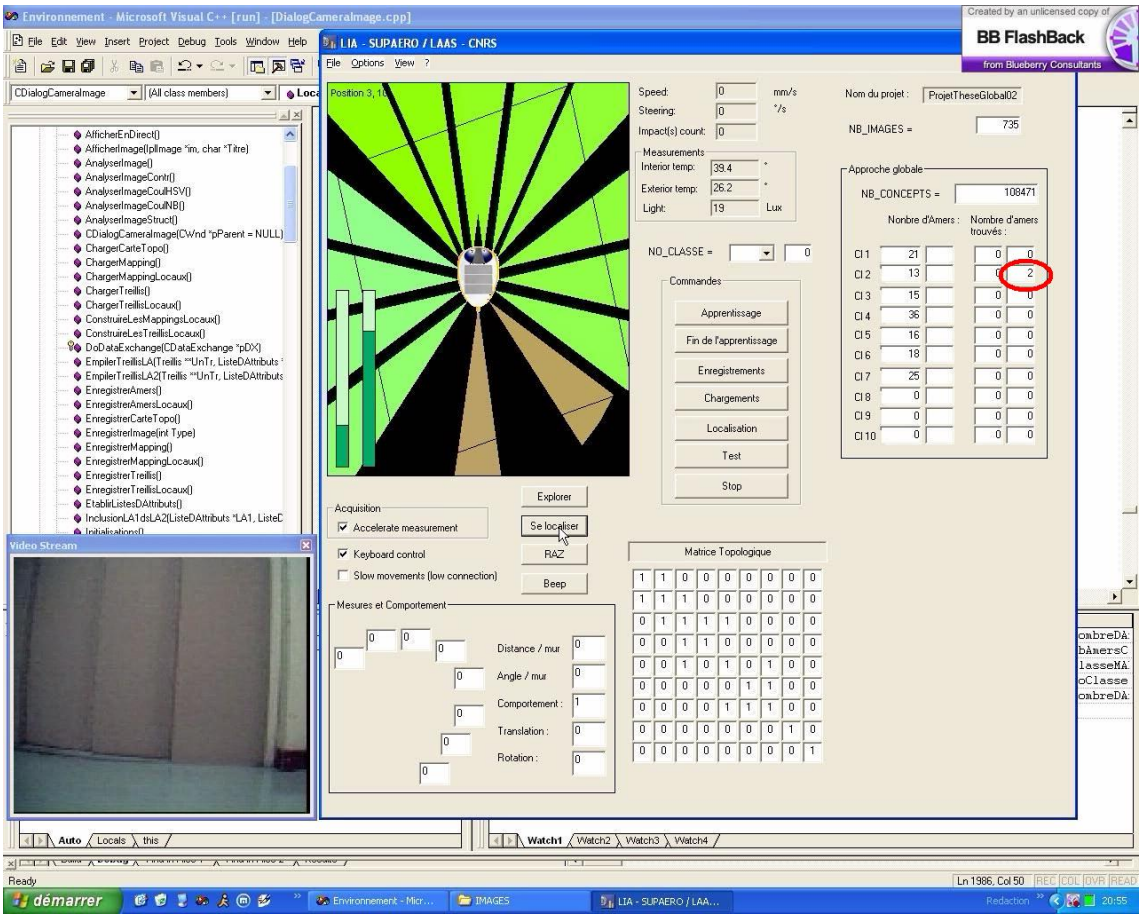


FIG. A.3 – Image 3 : Un deuxième amer est détecté.

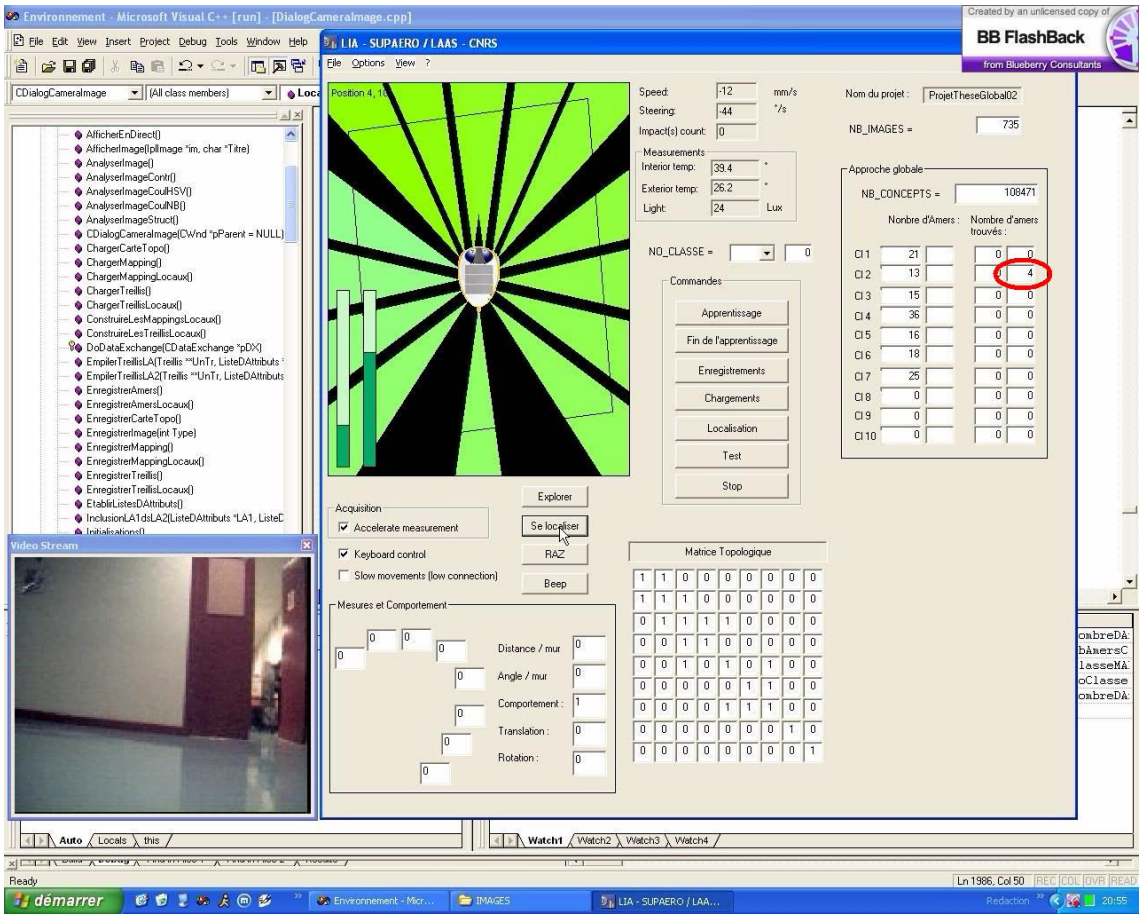


FIG. A.4 – Image 4 : à ce stade, quatre amers sont détectés.

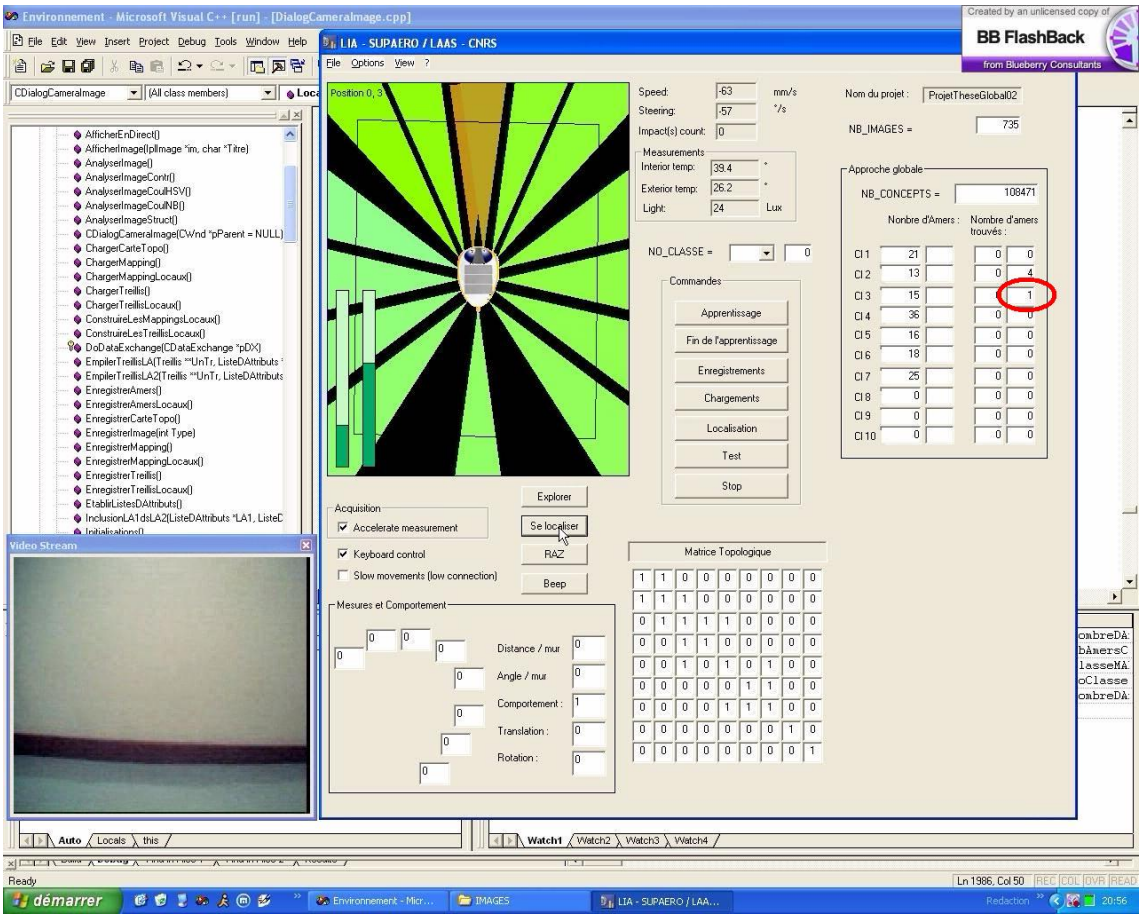


FIG. A.5 – Image 5 : une image est mal localisée (dans la pièce 3).

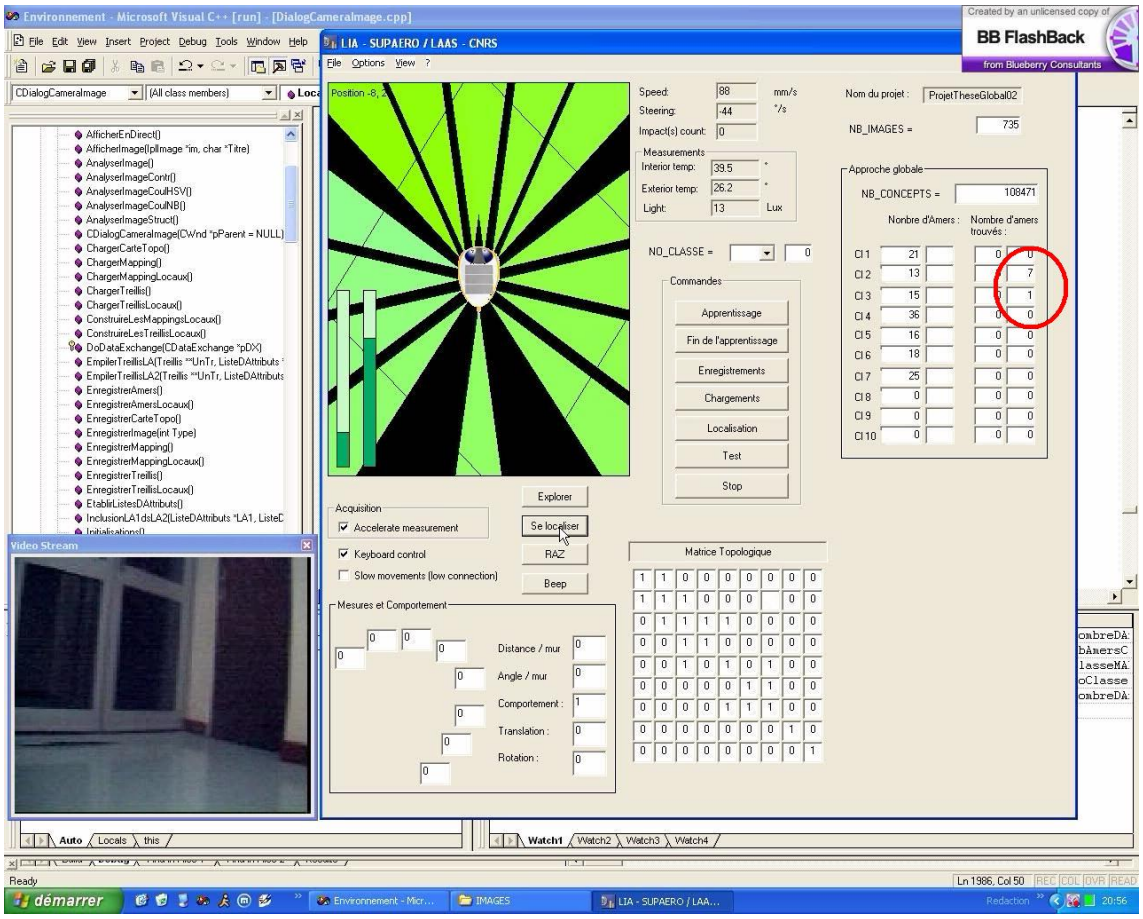


FIG. A.6 – Image 6 : le robot poursuit sa localisation.

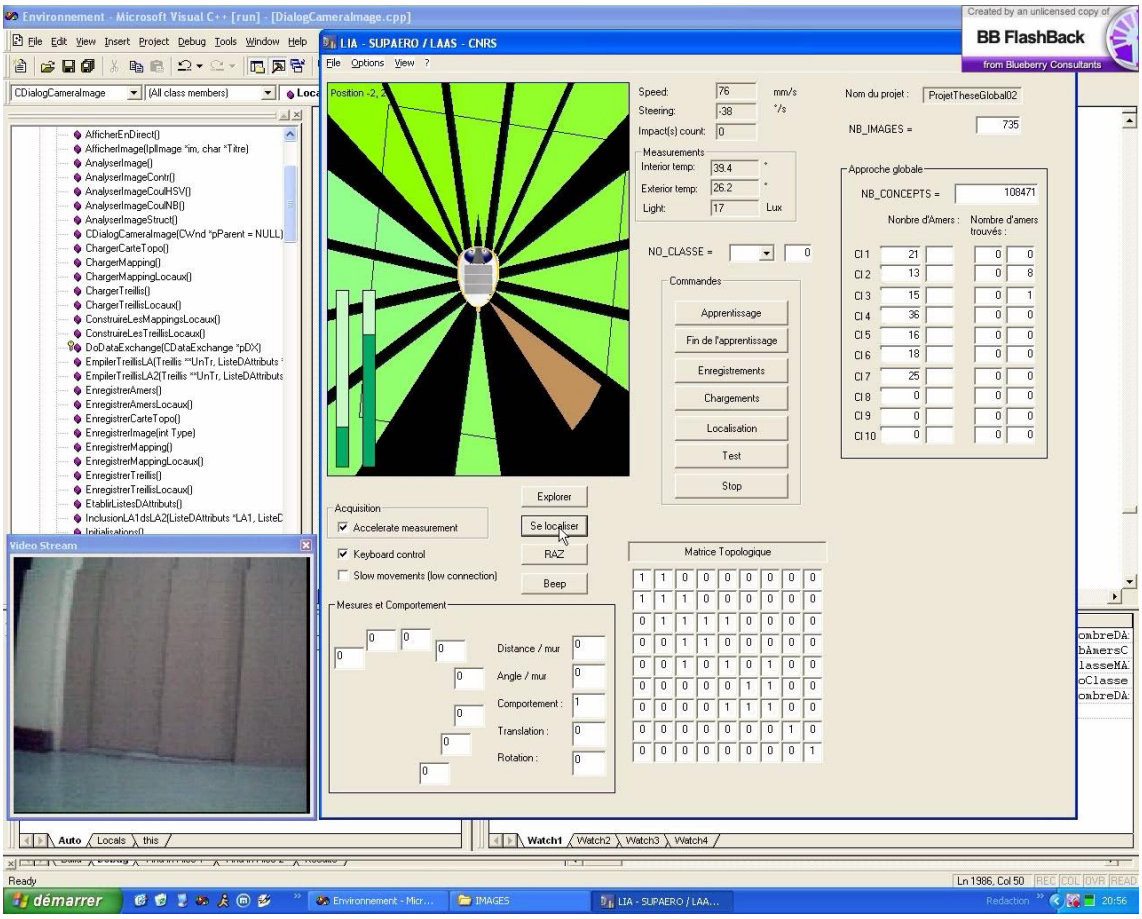


FIG. A.7 – Image 7 : le robot poursuit sa localisation.

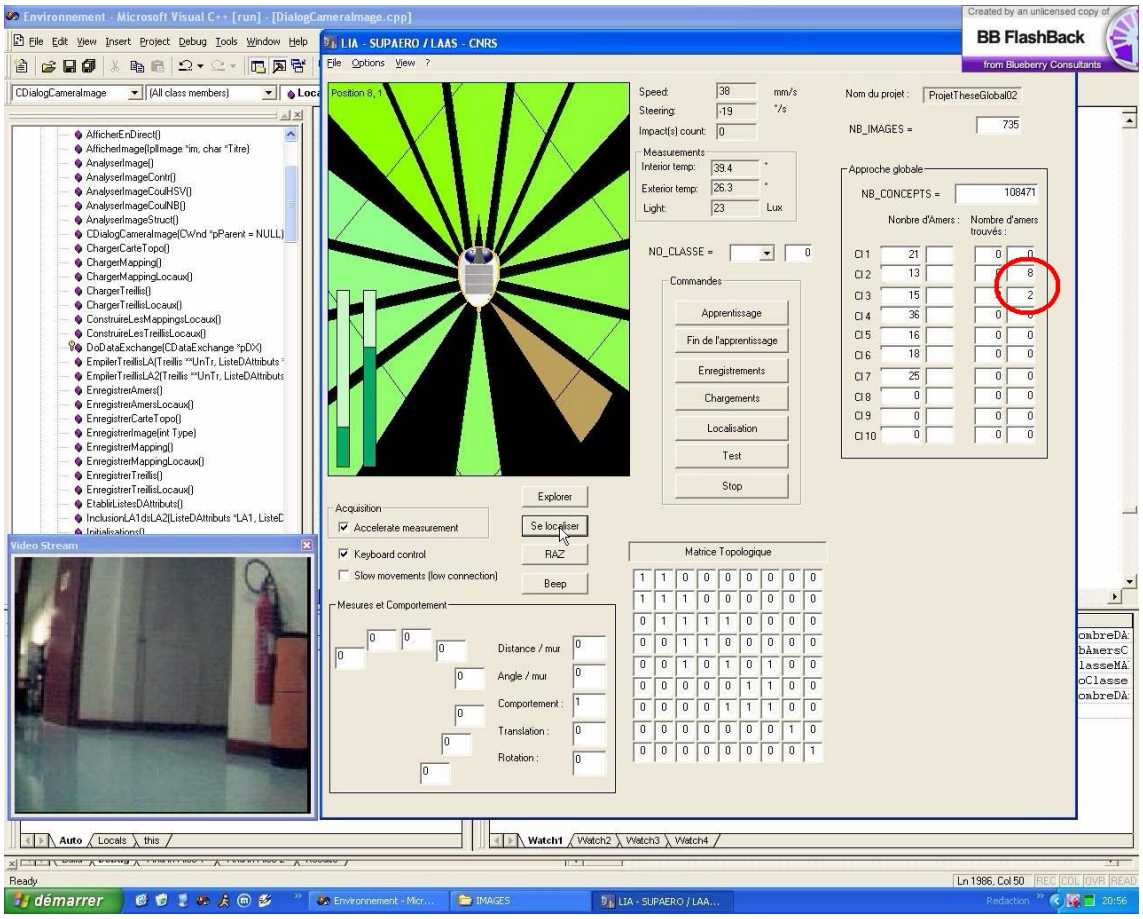


FIG. A.8 – Image 8 : une deuxième erreur se produit pour la localisation de l'image.

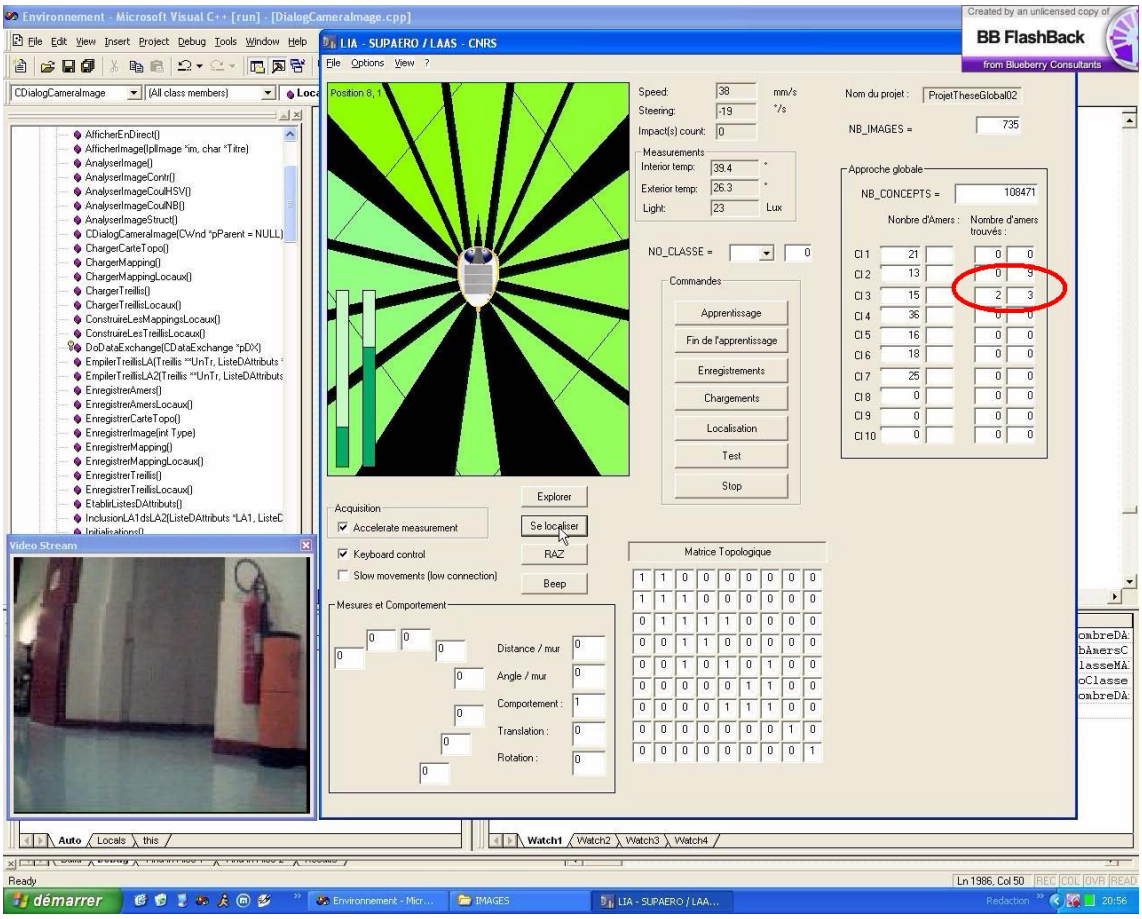


FIG. A.9 – Image 9 : une deuxième erreur se produit pour la localisation de l'image.

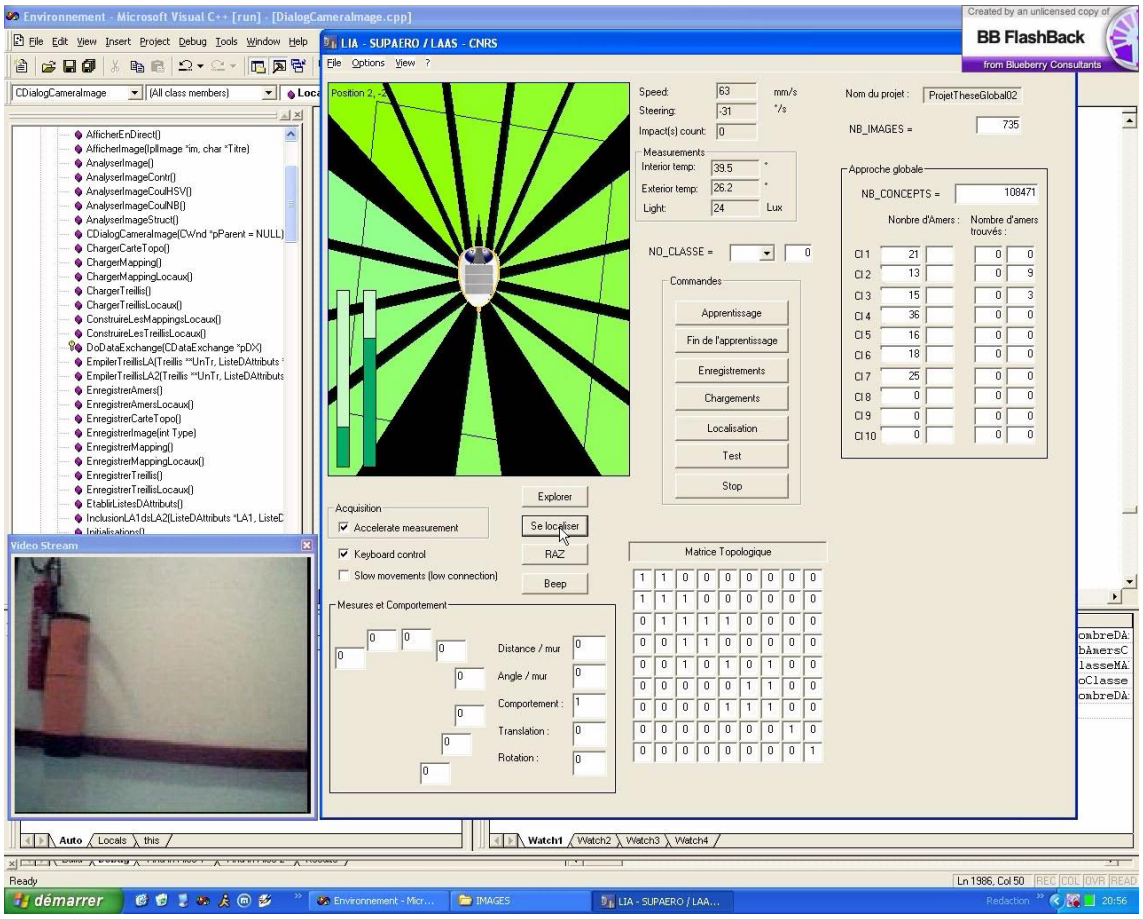


FIG. A.10 – Image 10 : le robot poursuit sa localisation.

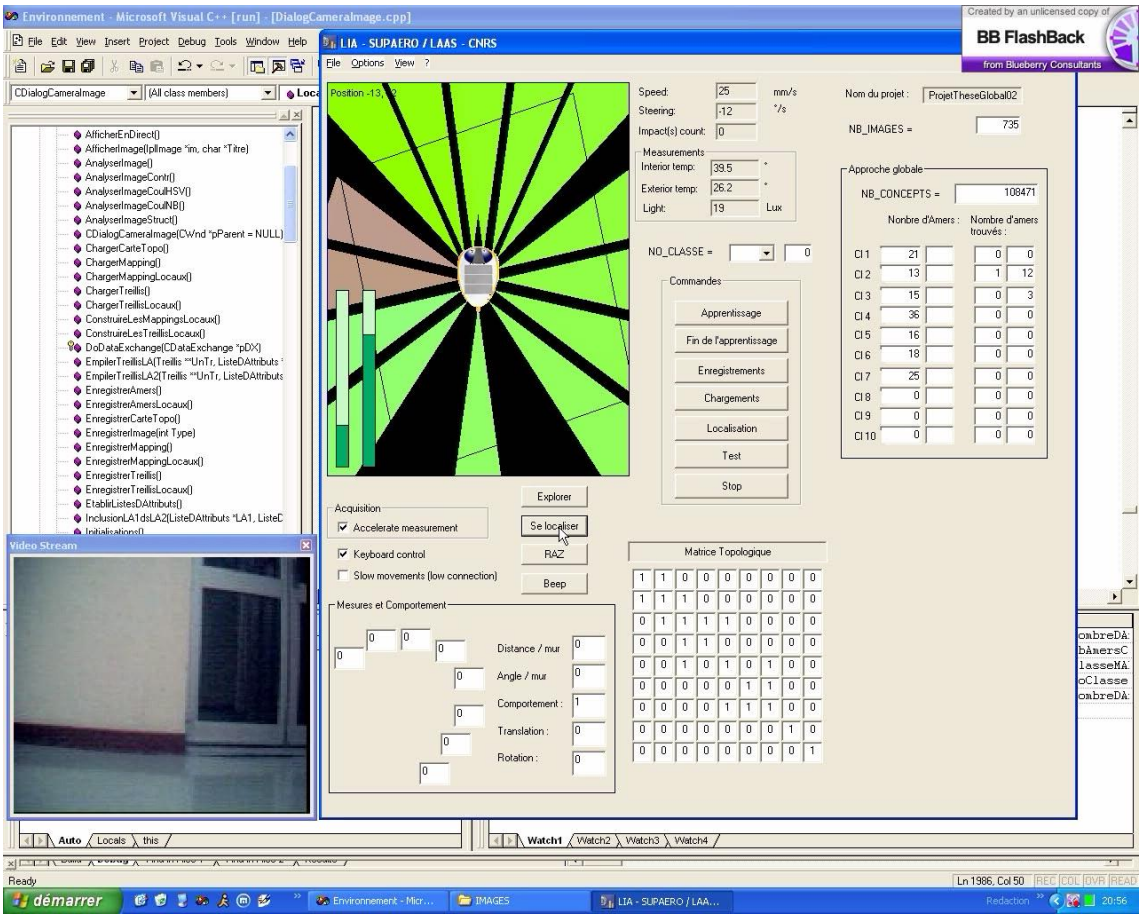


FIG. A.11 – Image 11 : le robot poursuit sa localisation.

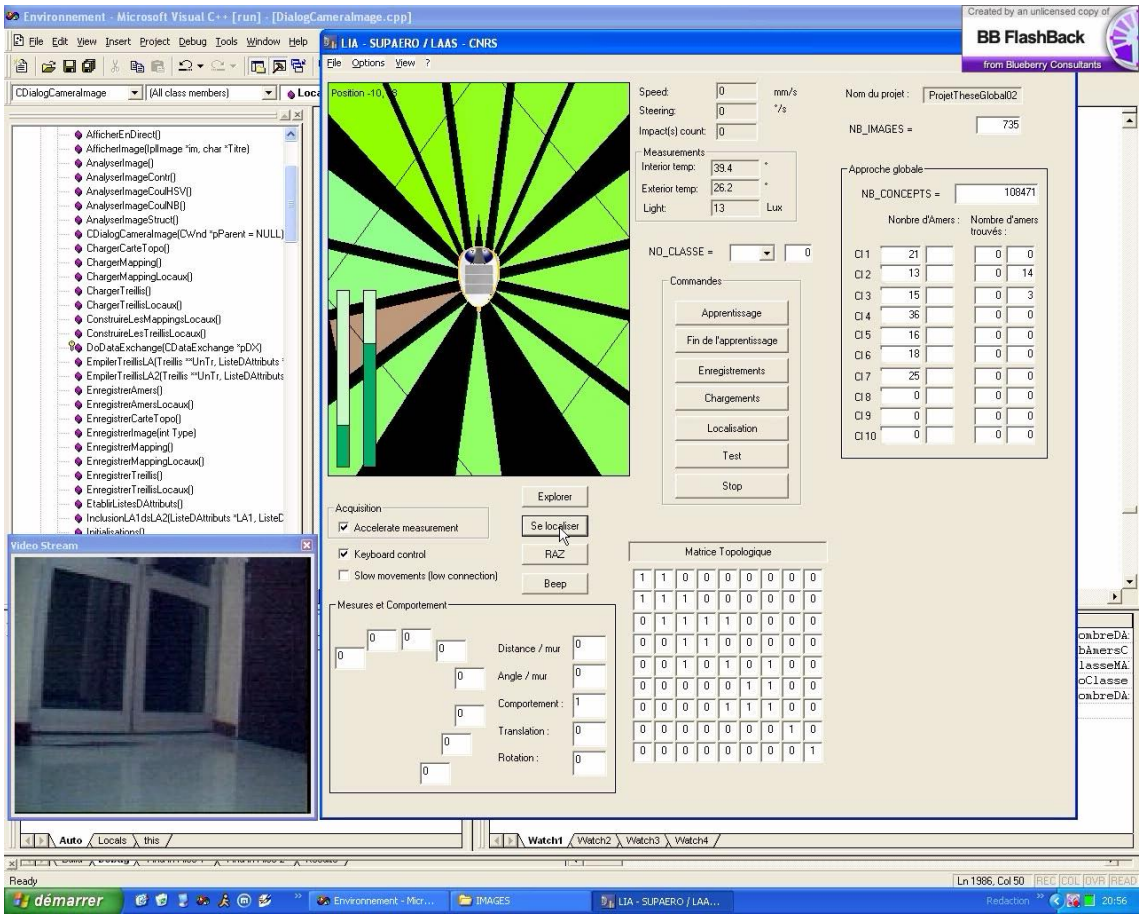


FIG. A.12 – Image 12 : le robot poursuit sa localisation.

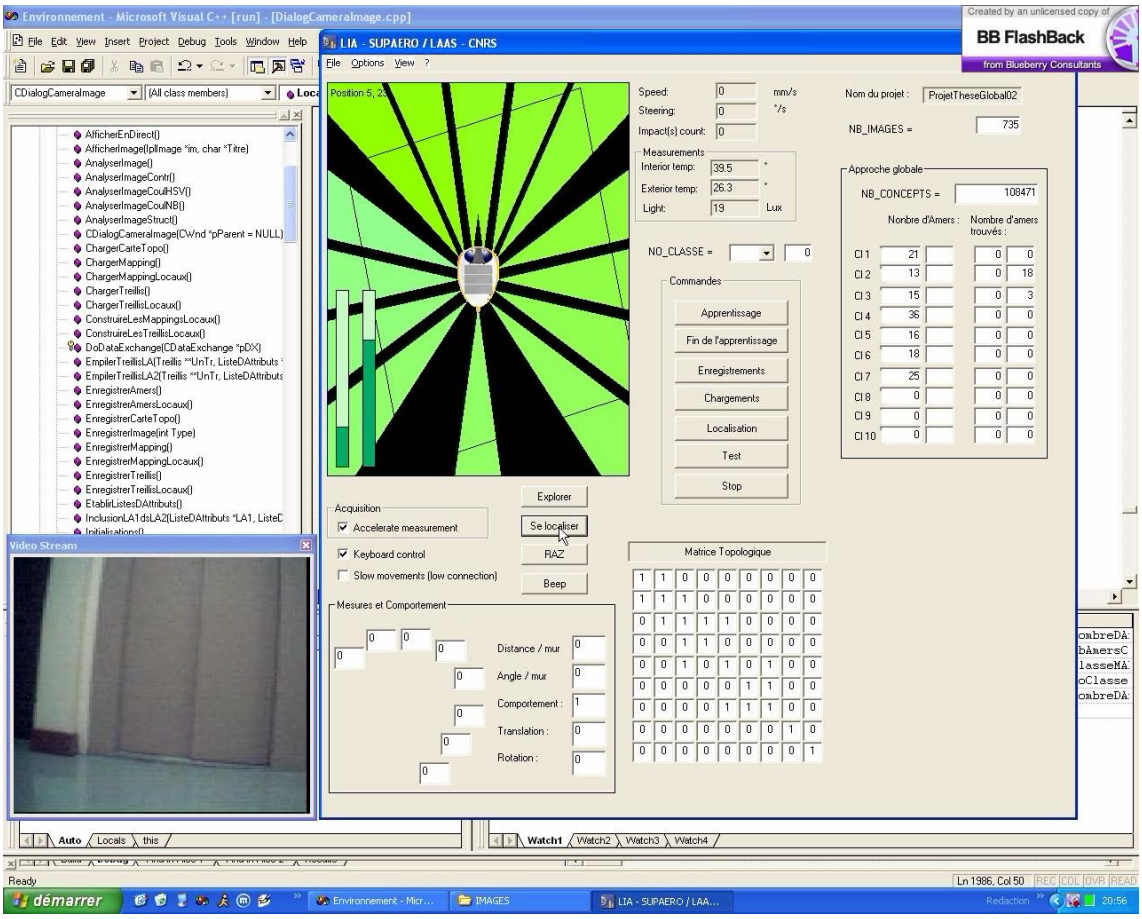


FIG. A.13 – Image 13 : le robot poursuit sa localisation.

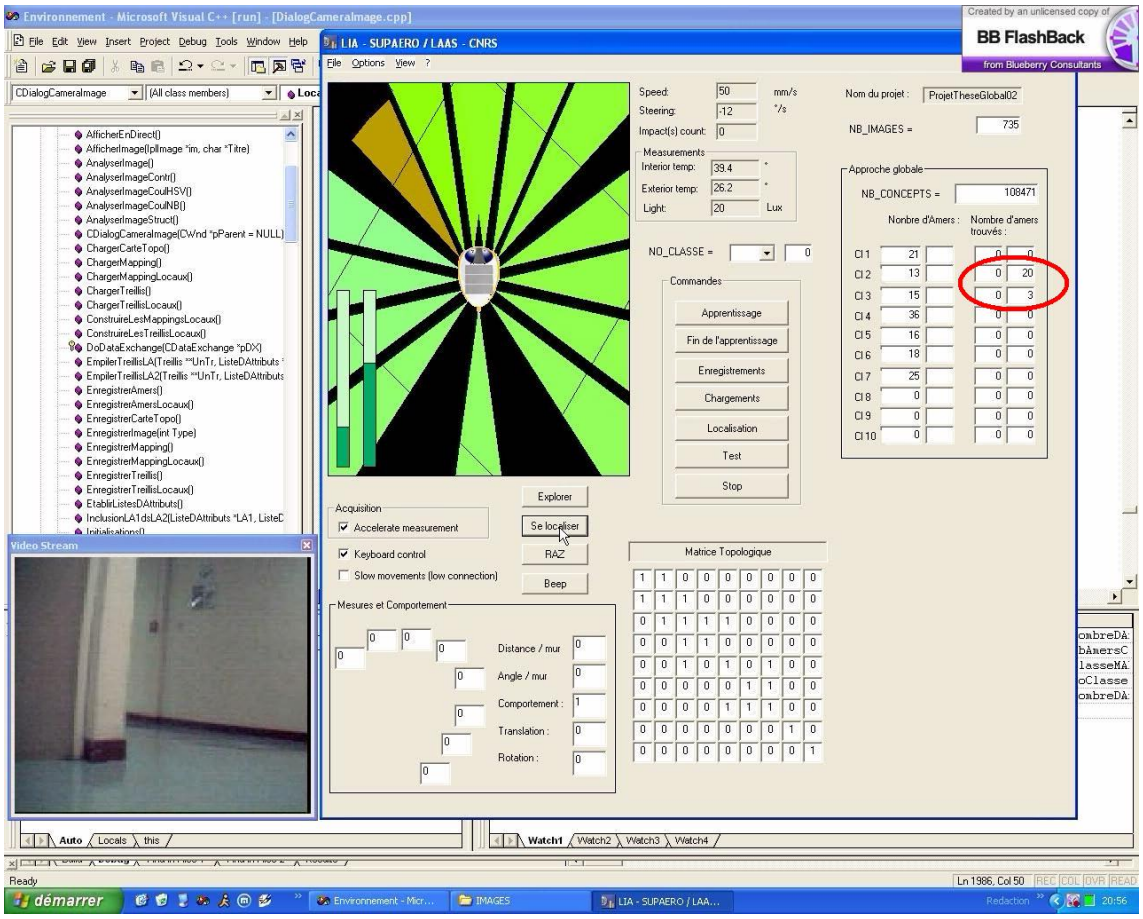


FIG. A.14 – Image 14 : le robot a détecté 20 images.

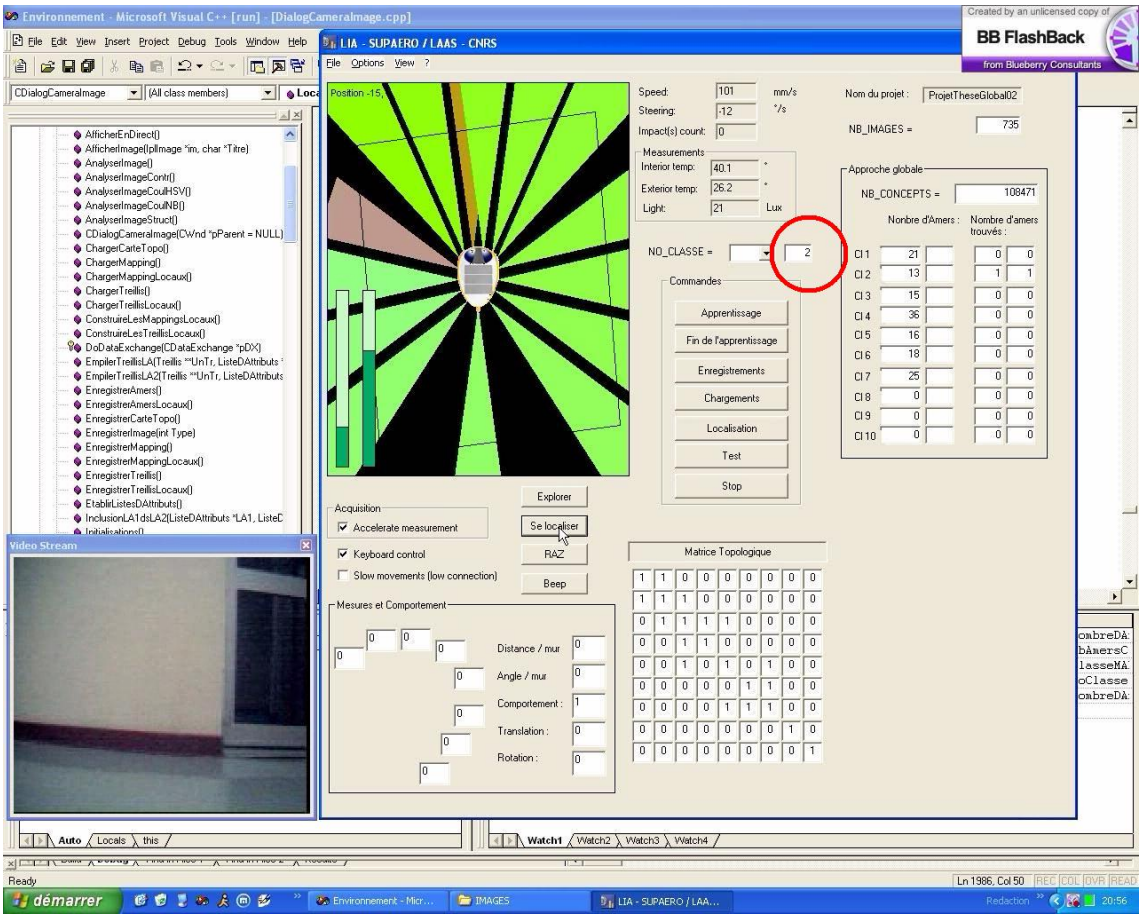


FIG. A.15 – Image 15 : la décision est prise par le robot : il se site dans la pièce 2.

Annexe B

Publications

JOURNAUX

Journal :	Journal of Universal Computer Sciences (J.UCS)
Titre :	*"Galois Lattice Theory for Probabilistic Visual Landmarks"*
Auteurs :	Emmanuel Zenou, Manuel Samuelides
Volume :	10
Numéro :	8
Année :	2004
Résumé :	This paper presents an original application of the Galois lattice theory, the visual landmark selection for topological localization of an autonomous mobile robot, equipped with a color camera. First, visual landmarks have to be selected in order to characterize a structural environment. Second, such landmarks have to be detected and updated for localization. These landmarks are combinations of attributes, and the selection process is done through a Galois lattice. This paper exposes the landmark selection process and focuses on probabilistic landmarks, which give the robot thorough information on how to locate itself. As a result, landmarks are no longer binary, but probabilistic. The full process of using such landmarks is described in this paper and validated through a robotics experiment.

Journal : EURASIP Journal of Applied Signal Processing (J.ASP)
Titre : *"Characterizing Image Sets using Formal Concept Analysis"*
Auteurs : Emmanuel Zenou, Manuel Samuelides
Volume :
Numéro :
Année :
Résumé : This article presents a new method for supervised image classification. Given a finite number of image sets, each set corresponding to a place of an environment, we propose a localization strategy, which relies upon supervised classification. For each place the corresponding landmark is actually a combination of features that have to be detected in the image set. Moreover, these features are extracted using a symbolic knowledge extraction theory, "formal concept analysis". This paper details the full landmark extraction process and its hierarchical organization. A real localization problem in a structured environment is processed as an illustration. This approach is compared with an optimized neural network based classification, and validated with experimental results. Further research to build up hybrid classifier is outlined on discussion.

CONFÉRENCES INTERNATIONALES

Conférence : European Conference on Artificial Intelligence (ECAI)
Lieu : Valence, Espagne
Titre : *"Topological Visual Localization Using Decentralized Galois Lattices"* (Poster)
Auteurs : Emmanuel Zenou, Manuel Samuelides
Année : 2004
Résumé : This paper presents a new decentralized method for selecting visual landmarks in a structured environment. Different images, issued from the different places, are analyzed, and primitives are extracted to determine whether or not features are present in the images. Subsequently, landmarks are selected as a combination of these features with a mathematical formalism called Galois -or concept- lattices. The main drawback of the general approach is the exponential complexity of lattice building algorithms. A decentralized approach is therefore defined and detailed here : it leads to smaller lattices, and thus to better performance as well as an improved legibility.

Conférence :	Int. Conf. on Advanced Concepts for Intelligent Vision Systems (ACIVS)
Lieu :	Ghent, Belgique
Titre :	*"Characterization of Image Sets : Learning with the Galois Lattice Approach"*
Auteurs :	Emmanuel Zenou, Manuel Samuelides
Année :	2003
Résumé :	This paper presents a new method for supervised image classification. One or several landmarks are attached to each class, with the intention of characterizing it and discriminating it from the other classes. The different features, deduced from image primitives, and their relationships with the sets of images are structured and organized into a hierarchy thanks to an original method relying on a mathematical formalism called Galois (or Concept) Lattices. Such lattices allow us to select features as landmarks of specific classes. This paper details the feature selection process and illustrates this through a robotic example in a structured environment. The class of any image is the room from which the image is shot by the robot camera. In the discussion, we compare this approach with decision trees and we give some issues for future research.

WORKSHOP INTERNATIONAUX

Conférence :	IEEE Workshop on Machine Learning for Signal Processing (MLSP)
Lieu :	Sao-Luis, Brésil
Titre :	*"Learning-Based Visual Localization Using Formal Concept Lattices"*
Auteurs :	Manuel Samuelides, Emmanuel Zenou
Année :	2004
Résumé :	We present here a new methodology to perform active visual localization in the context of autnomous mobile robotics. The robot is endowed with a topological map of its environment. During the learning phase, the robot takes a lot of pictures from the environment ; each picture is labelled by its origin place in the topological map. After the learning phase, the robot is supposed to locate itself in the learnt environment using the visual sensor. Since the discriminating information is sparse, the usual supervised classification techniques as neural networks are not sufficient to perform efficiently this task. Therefore, we propose to use a symbolic learning approach, the "formal concept analysis". The relevant information is gathered into one concept lattice. A formal classification rule is proposed to achieve localization on the topological map. In order to improve the response rate of the decision process, the original formal landmark set is extended to plausible landmarks for a given confidence level. Experimental results in a structured environment support this approach. Perspectives for implementing active strategy to look for visual information and to improve on-line learning and localization process are presented in the final discussion.

CONFÉRENCES FRANCOPHONES

Conférence :	Reconnaissance de Formes et Intelligence Artificielle (RFIA)
Lieu :	Toulouse, France
Titre :	*"Utilisation des treillis de Galois pour la caractérisation d'ensembles d'images"*
Auteurs :	Emmanuel Zenou, Manuel Samuelides
Année :	2004
Résumé :	Ce papier présente une nouvelle méthode pour la classification supervisée d'ensembles d'images. À chacun de ces ensembles est attaché, pour le caractériser et le distinguer des autres, une entité appelée amer. Différentes caractéristiques, déduites de primitives visuelles, sont extraites de chaque image, et la relation qui lie certaines images à certaines caractéristiques sont structurées et hiérarchisées grâce à un formalisme mathématique appelé treillis de Galois (ou treillis de concepts). De tels treillis permettent de sélectionner des combinaisons de caractéristiques qui serviront d'amers visuels pour chaque ensemble d'images. Cette approche s'inscrit dans le cadre de la robotique mobile autonome et de la localisation topologique dans un environnement structuré, pour laquelle chaque ensemble d'images est issu de d'une même pièce qu'il est nécessaire de caractériser. Ce papier expose cette démarche originale et les résultats obtenus après expérimentation.

Conférence :	Conférence des Jeunes Chercheurs en Sciences Cognitives (CJC)
Lieu :	Ministère de la Recherche, Paris, France
Titre :	*"Apprentissage d'Amers Visuels à l'Aide des Treillis de Galois"*
Auteurs :	Emmanuel Zenou, Manuel Samuelides
Année :	2002
Résumé :	Cette communication expose un moyen de localisation dans l'espace d'un robot mobile autonome à l'aide d'amers visuels : le robot sélectionne et apprend lui-même ses propres points de repère, qui lui permettront ensuite de reconnaître un lieu particulier dans un espace donné. Ces amers sont locaux, et le contexte est modélisé par l'ensemble de ces amers à l'aide d'un formalisme mathématique appelé *treillis de Galois* et d'une heuristique particulière.

TRAVAUX ANTÉRIEURS

Type :	Mémoire de DEA
Lieu :	LIRMM, Montpellier
Titre :	*"Simulation de systèmes récurrents et non linéaires à l'aide de réseaux de neurones. Application au* TAÏPAN*"*
Auteur :	Emmanuel Zenou
Année :	1998

Résumé :

Type : Mémoire de Maîtrise
Lieu : Université LAVAL, Quebec, Canada & ENS Cachan
Titre : *"Projet de programmation appliquée l'intelligence artificielle distribuée"*
Auteur : Emmanuel Zenou
Année : 1997
Résumé :

AUTRES

Type : Article très hautement scientifique
Lieu : NASA Aimes Research Center, Mountain View, CA, USA
Titre : *"De l'art de découvrir les mystères du vélocipède jaune et de son interaction avec le sol états-unien - Application à l'aéro-pneulogie et illustrations"*
Auteur : Emmanuel Zenou
Année : 2003
Résumé : Cet article s'inscrit dans la droite ligne des Grands Travaux de l'Aéro-Pneulogie. L'étude considérée ici présente une analyse issue d'une problématique concrète, à savoir la crevaison répétitive et inexpliquée des pneus d'un vélocipède jaune. Une fois les faits exposés, des hypothèses sont élaborées, puis un travail minutieux de dissection de pneu a permis de résoudre le problème. Enfin, une étude sur le temps de réparation du Vélocipède Jaune a été étudiée de façon studieuse.

Références bibliographiques

[Angluin 88] D. Angluin. *Queries and Concept Learning*. Machine Learning, vol. 2, pages 319–342, 1988.

[Angluin 97] D. Angluin, M. Krikis, R. Sloan & G. Turan. *Malicious omission and errors in answers to membership queries*. Machine Learning, vol. 28, pages 211–255, 1997.

[Arleo 98] A. Arleo, D. Floreano & W. Gerstner. *Modélisation de l'Hippocampe : Représentation Spatiale et Navigation des Systèmes Autonomes*. Journées Jeunes Chercheurs en Robotique, 1998.

[Arleo 01] A. Arleo, F. Smeraldi, S. Hug & W. Gerstner. *Place Cells and Spatial Navigation based on 2d Visual Feature Extraction, Path Integration, and Reinforcement Learning*. Advances in Neural Information Processing Systems 13, MIT-Press, 2001.

[Arleo 02] A. Arleo & W. Gerstner. *A model of the rat spatial learning system : Studying the interrelation between idiothetic and allothetic cues.* Third Forum of European Neuroscience, Paris, France, 2002.

[Ayala 00] V. Ayala, J.-B. Hayet, F. Lerasle & M. Devy. *Visual Localization of a Mobile Robot in Indoor Environment Using Planar Landmarks*. IROS, 2000.

[Balakrishnan 97] Karthik Balakrishnan & Vasant Honavar. *Spatial Learning for Robot Localization*. In Genetic Programming 1997 : Proceedings of the Second Annual Conference, pages 389–397, 1997.

[Barbut 70] M. Barbut & B. Monjardet. Ordre et classification. Hachette Université, 1970.

[Bianco 99] G. Bianco & A. Zelinsky. *Biologically-Inspired Visual Landmarks Learning and Navigation for Mobile Robots*. IEEE/RSJ, IROS, 1999.

[Bianco 00] G. Bianco, A. Zelinsky & M. Lehrer. *Visual Landmark Learning*. IEEE/RSJ, IROS, 2000.

[Bressloff 01] P. Bressloff, J. Cowan, M. Golubitsky, P. Thomas & M. Wiener. *Geometric Visual Hallucinations, Euclidian Symetry and the Functionnal Architecture of Striate Cortex*. Philosophical Transactions of The Royal Society, pages 299–330, 2001.

[Bruce 03] V. Bruce, P. Green & M. Georgeson. Visual perception, physiology, psychology and ecology. Psychology Press, 2003.

[Carpineto 93] C. Carpineto & G. Romano. *Galois : An Order-Theoric Approach to Conceptual Clustering*. In Proceedings of ICML'93, pages 33–40, 1993.

[Carpineto 96] C. Carpineto & G. Romano. *A Lattice Conceptual Clustering System and Its Applications to Browsing Retrieval*. In Machine Learning, volume 24, pages 95–122, 1996.

[Chaudron 03] L. Chaudron, N. Maille & M. Boyer. *The (CUBE) Lattice Model and its Applications*. Applied Artificial Intelligence, vol 19, no 3, 2003.

[Cocquerez 95] J.P. Cocquerez & S. Philipp. Analyse d'images : filtrage et segmentation. Masson, 1995.

[Collet 96] T. Collet. *Insect Navigation En Route to the Goal : Multiple Strategies for the Use of Landmarks*. The Journal of Experimental Biology, 1996.

[Cornuéjols 02] A. Cornuéjols & L. Miclet. Apprentissage artificiel, concepts et algorithmes. Eyrolles, 2002.

[Cristianini 00] N. Cristianini & J. Shawe-Taylor. An introduction to support vector machine. Cambridge University Press, 2000.

[Culloch 43] W. S. Mc Culloch & W. Pitts. *A logical calculus of ideas immanent in nervous activity*. In Bull. Mathematical Biophysics, 1943.

[Damasio 03] A. Damasio. Spinoza avait raison. Odile Jacob, 2003.

[Deco 96] G. Deco & D. Obradovic. An information-theoric approach to neural computing. Springer, 1996.

[Dodds 97] Z. Dodds & G. Hager. *A Color Interest Operator for Landmark-Based Navigation*. In AAAI/IAAI, pages 655–660, 1997.

[Dojat 99] M. Dojat. Systèmes cognitifs pour le traitement d'informations cliniques ou biologiques. Mémoire de synthèse, Habilitation à Diriger les Recherches, Univerté Joseph Fourier, 1999.

[Douglas 73] D.H. Douglas & T.K. Peucker. *Algorithms for the reduction of the number of points required to represent a digitalized line or its caricature*. In The Canadian Cartographer, volume 10 (2), pages 112–122, 1973.

[Dubreil 64] P. Dubreil & M.L. Dubreil-Jacotin. Leçons d'algèbre moderne. Dunod, 1964.

[Dudek 00] G. Dudek & D. Jugessur. *Robust Place Recognition using Local Appearance Based Methods*. IEEE Int. Conf. on Robotics and Automation, 2000.

[Duquenne 03] V. Duquenne, C. Chabert, A. Cherfouh, J.-M. Delabar, A.-L. Doyen & D. Pickering. *Structuration of Phenotypes / Genotypes through Galois Lattices and Implications*. In American Association of Immunologists, 2003.

[Dutech 98] A. Dutech. *Apprentissage d'Environnements : Approches Cognitives et Comportementales*. PhD thesis, École Nationale Supérieure de l'Aéronautique et de l'Espace, Toulouse, 1998.

[Dyer 96] Fred C. Dyer. *Spatial Memory and Navigation by Honeybees on the Scale of the Foraging Range*. The Journal of Experimental Biology, vol. 199, pages 147–154, 1996.

[Filliat 01] D. Filliat. *Cartographie et Estimation Globale de la Position pour un Robot Mobile Autonome*. PhD thesis, Université Paris XI, Paris, 2001.

[Foggia 99] P. Foggia, C. Sansone, F. Tortorella & M. Vento. *Definition and Validation of a Distance Measure Between Structural Primitives*. Pattern Analysis and Applications, 1999.

[Fox 98] D. Fox, W. Burgard, S. Thrun & A. Cremers. *Position Estimation for Mobile Robots in Dynqmic Environnement*, 1998.

[Fox 99] D. Fox, W. Burgard & S. Thrun. *Probabilistic methods for Mobile Robot Mapping*. novembre 1999.

[Frigerio 01] J.-M. Frigerio. Perception humaine de la couleur (vue par un physicien). Laboratoire d'Optique des Solides UMR 7601 - CNRS, 2001.

[Ganter 99] B. Ganter & R. Wille. Formal concept analysis. Springer-Verlag, 1999.

[Gaussier 00] P. Gaussier, C. Joulin, J.P. Banquet, S. Leprêtre & A. Revel. *The Visual Homing Problem : An Example of Robotics/Biology Cross Fertilization*. Robotics and Autonomous Systems, 2000.

[Gazzaniga 00] M. Gazzaniga, R. Ivry & G. Mangun. Neurosciences cognitives, la biologie de l'esprit. DeBoeck Unversity, 2000.

[Gerstner 02] W. Gerstner & W. Kistler. Spiking neuron models. Cambridge University Press, 2002.

[Godin 91] R. Godin, R. Missaoui & H. Alaoui. *Learning Algorithms using a Galois Lattice Structure*. In IEEE Int. Conf. on Tools for Artificial Intelligence, 1991.

[Godin 95] R. Godin, G. Mineau & R. Missaoui. *Incremental Concept Formation Algorithms based on Galois (Concept) Lattices*. Computational Intelligence, vol 11, 1995.

[Govaert 03] G. Govaert. Analyse de données. Lavoisier - Hermes Science, 2003.

[Gregory 00] R. Gregory. L'œil et le cerveau, la psychologie de la vision. DeBoeck Unversity, 2000.

[Hall 00] D. Hall, V. Colin de Verdière & James L. Crowley. *Object recognition using coloured receptive fields*. ECCV, 2000.

[Harris 88] C. Harris & M. Stephens. *A Combined Corner and Edge Detector*. In 4th Alvey Vision Conference, pages 147–151, 1988.

[Hayet 00] J.-B. Hayet, F. Lerasle & M. Devy. *Planar Landmarks to Localise a Mobile Robot*. Int. Symp. on Intelligent Robotic Systems, 2000.

[Hayet 02a] J.-B. Hayet. *Contribution à la navigation d'un robot mobile sur amers visuels texturés dans un environnement structuré*. PhD thesis, Université Paul Sabatier - LAAS/CNRS, Toulouse, 2002.

[Hayet 02b] J.B. Hayet, F. Lerasle & M. Devy. *A Visual Landmark Framework for Indoor Mobile Robot Navigation*. Int. Conf. on Robotics and Automation, 2002.

[Hebb 49] D.O. Hebb. The organization of behavior. a neuropsychological theory. J. Wiley & Sons, 1949.

[Hérault 94] J. Hérault & C. Jutten. Réseaux neuronaux et traitement du signal. Hermes, 1994.

[Hernandez 84] A. Marin Hernandez. *Vision dynamique pour la navigation d'un robot mobile*. PhD thesis, Université Paul Sabatier - LAAS/CNRS, Toulouse, 1984.

[Hershberger 92] J. Hershberger & J. Snoeyink. *Speeding Up the Douglas-Peucker Line-Simplification Algorithm*. In 5th Int. Symp. on Spatial Data Handling, volume 1, pages 134–143, 1992.

[Hopfield 82] J.J. Hopfield. *Neural Networks and Physical Systems with Emergent Collective Computational Abilities*. In National Academy of Sciences, volume 79, 1982.

[Hyvarinen 01] Hyvarinen, Karhunen & Oja. Principal component analysis. John Wiley and Sons, 2001.

[Itti 98] L. Itti, C. Koch & 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, pages 1254–1259, 1998.

[Jain 95] R. Jain, R. Kastury & B. Schunck. Machine vision. McGraw-Hill, 1995.

[Jedynak 96] B. Jedynak & F. Fleuret. *Reconnaissance d'objets 3D à l'aide d'arbres de classification*. image'com, 1996.

[Jung 03] I.-K. Jung & S. Lacroix. *High Resolution Terrain Mapping using Low Altitude Aerial Stereo Imagery*. In Int. Conf. on Computer Vision, 2003.

[Kanizsa 97] G. Kanizsa. La grammaire du voir (la grammatica del vedere -1980). Diderot Editeur, 1997.

[Kansdel 00] E. Kansdel, J. Schwartz & T. Jessell. Principle of neural science (fourth edition). McGraw-Hill, 2000.

[Knapek 00] M. Knapek, R.-O. Oropeza & D. Kriegman. *Selecting Promising Landmarks*. IEEE Int. Conf. on Robotics and Automation, 2000.

[Kodratoff 88] Y. Kodratoff & G. Tecuci. *Learning Based on Conceptual Distance*. IEEE Transactions on Patern Analysis and Machine Intelligence, 1988.

[Kohonen 82] T. Kohonen. *Self-Organized Formation of Topologically Correct Features Maps*. Biological Cybernetics, 1982.

[Kunt 00] M. Kunt. Reconnaissance des formes et analyse des scenes. Presses Polytechniques et Universitaires Romandes, 2000.

[Kuznetsov 01] S.O. Kuznetsov & S. Obiedkov. *Algorithms for the Construction of Concept Lattices and Their Diagram Graphs*. Principles of Data Mining and Knowledge Discovery (PKDD 2001), Freiburg, Germany, 2001.

[Kuznetsov 02] S.O. Kuznetsov & S. Obiedkov. *Comparing Performance of Algorithms for Generating Concept Lattices*. Journal of Experimental and Theorical Artificial Intelligence, vol. 14, no. 2-3, 2002.

[Leeser 98] M. Leeser, N. Kitaryeva & J. Crisman. *Spatial and Color Clustering on an FPGA-based Computer System*. In SPIE, pages 25–33, 1998.

[Levitt 90] T. Levitt & D. Lawton. *Qualitative Navigation for Mobile Robots*. Artificial Intelligence, vol. 44, pages 305–360, 1990.

[Lin 98] Dekang Lin. *An Information-Theoric Definition of Similarity*. International Conference on Machine Learning, 1998.

[Liquière 90] M. Liquière & E. Mephu Nguifo. *LEGAL : LEarning with GAlois Lattice*. In 5ièmes Journées Françaises sur l'Apprentissage, pages 93–113, 1990.

[Liquière 98] M. Liquière & J. Sallantin. *Structural machine learning with Galois lattice and Graphs*. International Conference on Machine Learning, 1998.

[Lowe 99] D. Lowe. *Object recognition from local scale-invariant features*. Int. Conf. on Computer Vision, 1999.

[Madsen 98] C. Madsen & C. Andersen. *Optimal Landmark Selection for Triangulation of Robot Position*. Journal of Robotics and Autonomous Systems, 1998.

[Majumder 00] S. Majumder, J. Rosenblatt, S. Scheding & H. Durrant-Whyte. *Map Building and Localization for Underwater Navigation*. In Int. Symp. on Experimental Robotics, 2000.

[Mitchell 97] T. Mitchell. Machine learning. McGraw-Hill, 1997.

[Moller 97] R. Moller & D. Lambrinos. *Modelling the landmark navigation behavior of the desert ant Cataglyphis*, 1997.

[Moller 98a] R. Moller, D. Lambrinos, R. Pfeifer, T. Labhart & R. Wehner. *Modeling ant navigation with autonomous agents*, 1998.

[Moller 98b] R. Moller, D. Lambrinos, R. Pfeifer & R. Wehner. *Insect Strategies of Visual Homing in Mobile Robots*, 1998.

[Nehmzow 95] U. Nehmzow. *Animal and robot navigation*, 1995.

[Nguifo 93] E. Mephu Nguifo. *Une nouvelle approche basée sur les treillis de Galois pour l'apprentissage de concepts*. Mathématiques, Informatique et Sciences Humaines, 1993.

[Nguifo 94] E. Mephu Nguifo. *Galois Lattice : A Framework for Concept Learning. Design, Evaluation and Refinement*. In Proc. 6th Int. Conf. on Tools with Artificial Intelligence, pages 461–467, 1994.

[Nguifo 98] Engelbert Mephu Nguifo & Patrick Njiwoua. *Using Lattice-Based Framework as a Tool for Feature Extraction*. In European Conference on Machine Learning, pages 304–309, 1998.

[Njamen 00] P. Njiwoua Njamen. *Contribution à l'apprentissage symbolique automatique par l'usage du treillis de Galois*. PhD thesis, Université d'Artois, 2000.

[Oosthuizen 88] G. Oosthuizen. *The Use of a Lattice in Knowledge Processing*. PhD thesis, University of Strathclyde, Glasgow, 1988.

[Oosthuizen 91] D. Oosthuizen. *Lattice-Based Knoledge Discovery*. 1991.

[Pastor 95] P. Pastor. *Étude et application des méthodes d'apprentissage pour la navigation en environnement inconnu*. PhD thesis, École Nationale Supérieure de l'Aéronautique et de l'Espace, Toulouse, 1995.

[Pratt 01] W. Pratt. Digital image processing (third edition). J. Wiley & Sons, 2001.

[Provost 01] F. Provost & T. Fawcett. *Robust Classification for Imprecise Environments*. Machine Learning, pages 203–231, 2001.

[Ranganathan 01] P. Ranganathan, J.B. Hayet, M. Devy, S. Hutchinson & F. Larasle. *Topological Navigation and Qualitative Localization for Indoor Environment Using Multisensory Perception*. Int. Symp. on Intelligent Robotic Systems, Toulouse, 2001.

[Ricordeau 04] M. Ricordeau. *Q-concept-learning : généralisation à l'aide de treillis de Galois dans l'apprentissage par renforcement*. In RFIA, 2004.

[Rosenblatt 62] F. Rosenblatt. Principles of neurodynamics. Spartan Press, East Lansin, 1962.

[Rukoz 02] M. Rukoz, M. Manouvrier & G. Jomier. *Distances de Similarité d'images basées sur les arbres quaternaires*. $18^{\text{èmes}}$ Journées Bases de Donnés Avancées, Evry, France, 2002.

[Sahami 95] M. Sahami. *Learning Classification Rules Using Lattices*. In Proceedings of ECML'95, pages 343–346, 1995.

[Sallantin] J. Sallantin. *Rationalité et Calcul : vers une cognition rationelle*.

[Schmid 96] C. Schmid. *Appariement d'Images par Invariants Locaux de Niveaux de Gris*. PhD thesis, Institut National Polytechnique de Grenoble, 1996.

[Schmid 97] C. Schmid & R. Mohr. *Local Greyvalue Invariants for Image Retrieval*. IEEE TPAM, 1997.

[Shatkay 97] Hagit Shatkay & Leslie Pack Kaelbling. *Learning Topological Maps with Weak Local Odometric Information*. In IJCAI (2), pages 920–929, 1997.

[Shavlik 90] J. Shavlik & T. Dietterich. *Readings in Machine Learning*, 1990.

[Smith 87] R.C. Smith & P. Cheeseman. *On the Representation and Estimation of Spatial Uncertainty*. International Journal of Robotics Research, vol. 5, pages 56–68, 1987.

[Stumme 02] G. Stumme, R. Taouil, Y. Bastide, N. Pasquier & L. Lakhal. *Computing iceberg concept lattices with Titanic*. Data Knowledge Engineering, vol. 42, no. 2, pages 189–222, 2002.

[Sutton 84] R. S. Sutton. *Temporal Credit Assignment in Reinforcement Learning*. PhD thesis, University of Massachusetts, 1984.

[Theiler 97] J. Theiler & G. Gisler. *A Contiguity-Enhanced k-means Clustering Algorithm for Unsupervised Multispectral Image Segmentation*. In SPIE, pages 108–118, 1997.

[Theocharous 01] G. Theocharous, K.r Rohanimanesh & S. Mahadevan. *Learning Hierarchical Partially Observable Markov Decision Process for Robot Navigation*. In Int. Conf. on Robotics and Automation, 2001.

[Thorpe 96] S. Thorpe, D. Fize & C. Marlot. *Speed of processing in the human visual system*. Nature, 1996.

[Thrun 96] Sebastian Thrun & Arno Bucken. *Integrating Grid-Based and Topological Maps for Mobile Robot Navigation*. In AAAI/IAAI, Vol. 2, pages 944–950, 1996.

[Thrun 98a] S. Thrun. *Learning Maps for Indoor Mobile Robot Navigation*. Artificial Intelligence, vol. 99, no. 1, pages 21–71, 1998.

[Thrun 98b] S. Thrun. *Learning Metric-Topological Maps for Indoor Mobile Robot Navigation*. Artificial Intelligence, vol. 99, no. 1, pages 21–71, 1998.

[Thrun 98c] S. Thrun, S. Gutmann, D. Fox, W. Burgard & B. Kuipers. *Integrating Topological and Metric Maps for Mobile Robot Navigation : a Statistical Approch*. AAAI / IAAI, 1998.

[Thrun 99] S. Thrun, M. Bennewitz, W. Burgard, A. Cremers, F. Dellaert, D. Fox, D. Haehnel, C. Rosenberg, N. Roy, J. Schulte & D. Schulz. *Minerva : a second generation mobile tour-guide robot*. Int. Cong. on Robotics and Automation, 1999.

[Trullier 97] O. Trullier, S. Wiener, A. Berthoz & J. Meyer. *Biologically-based Artificial Navigation Systems : Review and Prospects*, 1997.

[Tversky 77] Amos Tversky. *Features of Similarity*. Psychological Review, 84(4), 1977.

[Ulrich 00] I. Ulrich & I. Nourbakhsh. *Appearance-Based Place Recognition for Topological Localisation*. IEEE Int. Conf. on Robotics and Automation, 2000.

[Valtchev 02] P. Valtchev, R. Missaoui & P. Lebrun. *A partition-based approach towards constructing Galois (concept) lattices*. Discrete Mathematics, vol. 256, pages 801–829, 2002.

[Vandapel 00] N. Vandapel. *Perception et Sélection d'Amers en Environnement Polaire pour la Navigation d'un Robot Mobile*. PhD thesis, Université Paul Sabatier - LAAS/CNRS, Toulouse, 2000.

[Vapnik 96] V. Vapnik. The nature of statistical learning theory. Springer-Verlag, New-York, 1996.

[Veelaert 96] P. Veelaert & H. Peremans. *Adaptative Animat Navigation based on a Flexibility Model for the Environnement*. From Animals to Animats 4, 1996.

[Wall 84] K. Wall & P. Danielsson. *A Fast Sequential Method for Polygonal Approximation of Digitized Curves*. Computer Vision, Graphics and Image Processing, vol. 28, pages 220–227, 1984.

[Widrow 60] H. Widrow. *Adaptative Switching Circuit*. In IRE Western Electronic Show and Correction, volume 96, pages 96–104, 1960.

[Zwynsvoorde 00] D. Van Zwynsvoorde. *Construction Incrementale de Modeles Topologiques pour la Navigation d'un Robot Mobile*. PhD thesis, Université Paul Sabatier - LAAS/CNRS, Toulouse, 2000.

LOCALISATION TOPOLOGIQUE, AMERS VISUELS ET TREILLIS DE GALOIS

Cette thèse s'inscrit dans le cadre de la classification d'images supervisée appliquée à la robotique mobile autonome en milieu structuré. Pour naviguer, et en particulier pour se repérer, un robot utilise des amers attachés à des sites de l'environnement, modélisé par un graphe topologique. À chaque nœud du graphe est associé un amer. Les amers sont des combinaisons d'attributs visuels optimales, obtenues au travers d'un formalisme particulier appelé "treillis de Galois" ou "treillis de concepts". Des algorithmes de construction de treillis ont été modifiés afin de permettre, de façon incrémentale, l'établissement d'une classification supervisée des images et l'extraction des amers visuels. Une extension probabiliste et une approche locale ont été implémentées pour améliorer les performances de l'apprentissage. Enfin, une application robotique complète a été implémentée dans les laboratoires du LAAS-CNRS et de SUPAERO.

Mots clés : Analyse d'images, Vision par ordinateur, Apprentissage, Classification supervisée, Treillis de Galois, Treillis de Concepts, Robotique mobile autonome, Localisation topologique.

TOPOLOGICAL LOCALIZATION, VISUAL LANDMARKS AND GALOIS LATTICES

The subject if this Thesis is situated within the field of image supervized classification applied to autonomous mobile robotics in a structural environment. To navigate, and more especially to locate itself, a robot needs landmarks, attached to each site of the environment, which is represented by a topological graph. The robot attaches a landmark to each node of the topological graph. Landmarks are combinations of features obtained through a specific formalism called "Galois lattices" or "concept lattices". Some lattice building algorithms are modified to allow incremental supervized image classification to extract visual landmarks. A probabilistic expansion and a local approach have been developed to perform our system. Finally, a robotics application is described and has been implemented in the LAAS-CNRS and SUPAERO laboratories.

Keywords : Image analysis, Computer vision, Learning, Supervized classification, Galois lattices, concept lattices, Autonomous mobile robotics, Topological localization.

10, avenue Édouard-Belin - BP 54032 - 31055 Toulouse cedex 4 - France
Tél : 33 (0)5 62 17 80 80 - Fax : 33 0(5) 62 17 83 30 - http://www.supaero.fr
Établissement public à caractère administratif sous tutelle du Ministère de la Défense