Approche décentralisée des treillis de Galois pour la localisation topologique

Emmanuel Zenou[⋆†], Manuel Samuelides[⋆]

[⋆] SUPAÉRO,
10 avenue Edouard Belin, BP 4032
31055 Toulouse Cedex, France
{zenou,samuelid}@supaero.fr et
http://personnel.supaero.fr/zenou-emmanuel
[†] LAAS - CNRS,
7 avenue du colonel Roche,
31077 Toulouse cedex 4, France

Résumé

Ce papier présente une nouvelle technique pour la localisation d'un robot mobile autonome dans un environnement structuré. La localisation est topologique et se base sur les amers visuels. Ces amers sont des combinaisons de caractéristiques visuelles sélectionnées à l'aide d'un formalisme mathématique appelé treillis de Galois, ou treillis de concepts. Pour des très gros contextes, l'approche décentralisée est introduite afin de réduire le nombre de concepts et le temps de construction du treillis. Les algorithmes complets ont été validés expérimentalement et sont exposés dans ce papier.

Mots-clés *: Localisation topologique, navigation, amers visuels, treillis de Galois, treillis de concepts, robotique mobile autonome*

Abstract

This paper presents a new technique for autonomous mobile robot topological localization in a structural environment. This localization is based on visual landmarks, which are combinations of visual characteristic selected thanks to a mathematical formalism called Galois lattice or concept lattice. For large contexts, a decentralized approch is introduced in order to reduce the number of concepts and the lattice building time process. All algorithms have been validated through experimentation and are exposed in this paper.

Key-words*: Topological localization, navigation, visual landmarks, Galois lattice, concept lattice, autonomous mobile robotics*

1 INTRODUCTION

Naviguer et se localiser dans un environnement quelconque sont des aptitudes qui font appel, chez l'homme, à des processus extrêmement complexes. Transposée à une machine, à un robot mobile autonome, la technique est encore peu maîtrisée, malgré le nombre croissant de capteurs extéroceptifs (ou externes) disponibles pour apercevoir et appréhender l'environnement.

Parmi ces capteurs, les percepteurs visuels ont un statut particulier : avec un coût très faible et une mise en œuvre en général simple et rapide, ils fournissent une information extrêmement riche et très familière de l'espèce humaine. Il est en effet facile, pour un être humain, d'interpréter une scène dans une image, il est en revenche impossible de le demander à une machine aujourd'hui.

La localisation topologique consiste à repérer dans un environnement (ici structuré) des éléments (visuels) stables et distinctifs qui caractérisent les différents lieux de l'environnement. Cette approche topologique est donc plus qualitative que quantitative, contrairement aux approches métriques (Comme le *SLAM : Simultaneous Localisation and Mapping [30]*, par exemple) plus coûteuses et moins stables en général. D'un point de vue pratique, dans le flux d'images que le robot capte à l'aide de sa caméra, il s'agit de discriminer des ensembles d'images entre eux, ou en d'autres termes faire de la classification supervisée. Ces éléments caractéristiques, sous certaines conditions, s'appellent des *amers*, en l'occurrence ici des *amers visuels*. Il est alors nécessaire de les définir correctement en se posant les questions suivantes (section 2) : qu'est-ce qu'un amer ? Comment le caractériser ? Comment le sélectionner ?

Ce papier présente une nouvelle méthode pour répondre à ces questions. Toutes les images issues d'une même pièce sont regroupées dans un ensemble, et on obtient ainsi plusieurs ensembles d'images à caractériser (section 3). Dans un premier temps, pendant la phase d'apprentissage, les relations entre des ensembles d'images et des caractéristiques sont structurées et hiérarchisées à travers un formalisme appelé *treillis de Galois*, ou *treillis de concepts* (section 4). L'utilisation d'une telle structure permet à la machine de déterminer ses propres amers (section 5) attachés à chaque ensemble (chaque pièce). Dans un deuxième temps, en utilisant de nouvelles images issues du même environnement, la machine sera capable de retrouver à quel ensemble (quelle pièce) elles appartiennent. La section 6 présente le processus complet, avant d'introduire une approche décentralisée (section 7) permettant d'obtenir de meilleurs résultats avec des gros contextes. Les expérimentations sont entièrement décrites section 8 avant d'exposer une discussion et les perspectives (section 9) de notre travail.

Dans notre application, le lien entre des ensembles d'images d'un côté et des images issues de pièces d'un environnement structuré de l'autre côté est évident. Cependant la technique développée ici peut être appliquée à tous ensembles d'images pour d'autres types d'applications.

2 LES AMERS VISUELS

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, le terme de *"landmark"* revêt un sens plus général, comme un bâtiment important, un lieu à visiter [18] voire même une date importante sur le plan historique. Dans le monde de la robotique mobile, la définition la plus proche est 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" [21], ou plus simplement "un objet visuel identifiable dans un environnement" [3]. Habituellement les amers ne sont pas introduits par une définition formelle mais plutôt à travers des propriétés spécifiques comme "facilement distinguable" (saillant) [9] ou "localement unique". Dans un sens plus concret, dans le monde animal, un amer peut être le soleil (utilisation du *compas magnétique* par les abeilles [11]), ou le nord magnétique par exemple ; dans le monde de la robotique mobile autonome, un amer peut être un objet [10], une couleur [31], des points d'intérêt [28], *etc.*

Plusieurs classifications d'amers existent, notamment celle qui différencie les amers naturels des amers artificiels. 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 pré-définis : une base d'images représentant des objets est introduite dans le robot [21, 25] qu'il *"n'a plus qu'à"* reconnaître ;
- les amers partiellement pré-définis : ces amers potentiels ont une structure commune. Par exemple, dans [19], les amers sont des formes quadrangulaires planes (type affiches murales) que le robot apprend à identifier et à reconnaître ;
- les amers non pré-définis : aucune hypothèse n'est formulée *a priori* pour les amers potentiels. Les principaux travaux ayant une telle approche sont essentiellement d'inspiration biologique [20, 1, 15]

Notre démarche s'inscrit dans cette dernière catégorie : le robot doit pouvoir choisir dynamiquement et de façon autonome les amers les plus pertinents, afin de construire son propre modèle de l'environnement.

3 PRIMITIVES, CARACTÉRISTIQUES ET PROPRIÉTÉS

Différentes images sont extraites de chaque pièce d'un environnement structuré ; ainsi, à chaque pièce, est attaché un ensemble d'images qu'il est néces-

saire de caractériser. De chacune de ces images, des *primitives* sont extraites afin d'en déduire des *caractéristiques* de l'image, pour aider le robot à trouver des *propriétés* du lieu. Ainsi les caractéristiques se distinguent des propriétés par le fait qu'elles appartiennent à l'image, alors que les propriétés appartiennent au monde réel. Nous nous intéressons ici qu'aux caractéristiques.

Plusieurs types de primitives peuvent être extraites dans les différentes images :

- les primitives structurelles : des segments, avec leur taille et leur orientation, des formes, *etc.* ;
- les primitives colorimétriques : on extrait les pixels et les objets rouges, verts, bleus, jaunes, *etc.* ;
- les primitives photogrammétriques, venant de chaque pixel de l'image : textures, points d'intérêt, *etc.* ;
- ...

On construit et on définit des caractéristiques à partir de ces primitives. On peut noter que la définition des caractéristiques est générale et inclut toute caractéristique potentielle, qu'elle appartienne aux images d'intérêt ou non. Leur pertinence relèvera de la connaissance a priori que nous avons sur l'environnement. Par exemple, le fait qu'*"il y a beaucoup de rouge dans cette image"* peut être une caractéristique, comme le fait qu'*"il y ait telle texture"*.

Dans la pratique, les caractéristiques seront des primitives (seules les plus stables seront sélectionnées) et des associations de plus haut niveau, les plus invariantes possibles en rotation, translation, et échelle. C'est pourquoi la présence de tel ou tel segment n'est pas toujours significative, mais une caractéristique potentielle sera plutôt le fait qu'*"il y ait un grand nombre de segments parallèles et de même longueur"* (typiquement pour une bibliothèque).

La binarisation de l'information nécessite un seuillage des attributs continus. Ce seuillage est pour le moment établi de façon experte par l'opérateur, essentiellement en fonction de la taille des images. La construction incrémentale du treillis (que nous verrons par la suite) à partir des caractéristiques des images ne permet pas d'établir une adaptativité des seuils. Si certains seuils sont mal choisis, les caractéristiques correspondantes risquent de ne plus être pertinentes et seront naturellement éliminées lors de l'apprentissage des amers. En revenche, ces *mauvaises* caractéristiques viendront alourdir inutillement le processus d'apprentissage.

La première partie de notre démarche est d'extraire les caractéristiques de chaque image et de regrouper cette information dans une table de correspondance (ou *mapping*) qui associe à chaque image les caractéristiques correspondantes. Cette structuration est présentée figure 1.

		Ppté 1	Ppté 2	Ppté 3	Ppté 4	...	Ppté N
Pièce 1	Image 1.1	X	X	X			X
	Image 1.2	X	X				
	Image 1.3	X	X				
	...	X	X				
	Image 1.n1						
Pièce 2	Image 2.1			X	X		
	Image 2.2		X	X	X	X	
	...			X	X	X	
	Image 2.n2				X	X	
...	...	X					
Pièce P	Image P.1						X
	Image P.2						X
	...			X			X
	Image P.np				X		

FIG. 1 – Structuration de l'information

4 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 [22], et certaines applications concrètes sont apparues récemment, notamment en fouille de données [29, 24], ou dans le domaine aéronautique [7]. Nous proposons ici une application au traitement d'images dans le contexte de la robotique mobile autonome.

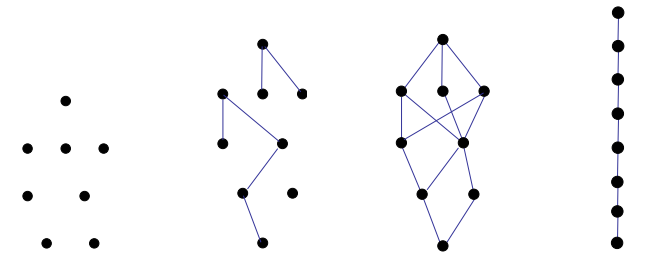
En amont de ces applications, on peut noter sur le formalisme les travaux de Birkhoff aux États-Unis (années 60) sur la correspondances de Galois et de Barbu et Monjardet en France (années 70) par la suite.

4.1 Formalisme mathématique (1)

Toute l'information visuelle peut être décrite sous forme de concepts, qui associent des images (objets) avec des caractéristiques. Ces concepts sont hiérarchisés dans un treillis, permettant la sélection des amers comme combinaisons de caractéristiques. Le formalisme complet des treillis de Galois est décrit dans [2] et [14]. Rappelons ici les principales définitions.

Définition 1
Un **treillis** *est un ensemble ordonné dans lequel deux éléments quelconques ont une borne supérieure (BS, ou sup) et une borne inférieure (BI, ou inf). Un treillis complet est un treillis pour lequel tout sous-ensemble possède une BS et une BI.*

La figure 2 montre différents types d'ensembles ordonnés : une anti-chaîne (aucun élément n'est comparable à un autre), un ensemble partiellement ordonné quelconque, un treillis et une chaîne (tous les éléments sont comparables).



(a) Anti-chaîne (b) Ens. partiellement ordonné (c) **Treillis** (d) Chaîne

FIG. 2 – Différents ensembles. Un treillis est un ensemble ordonné dans lequel deux éléments quelconques ont une borne sup. et une borne inf.

Définition 2
Un contexte $\mathcal{K}$ est un triplet $(\mathcal{O}, \mathcal{F}, \zeta)$ pour lequel $\mathcal{O}$ est un ensemble d'objets, $\mathcal{F}$ un ensemble d'attributs et ζ une application binaire de $\mathcal{O} \times \mathcal{F}$ dans $\{0, 1\}$.

Dans notre application, les objets sont les images appartenant aux différents ensembles situés, les attributs sont les caractéristiques et l'application ζ est définie par la présence d'une caractéristique $f \in \mathcal{F}$ dans une image $o \in \mathcal{O}$; on a alors $\zeta(o, f) = 1$.

Définition 3
Étant donné un contexte $\mathcal{K} = (\mathcal{O}, \mathcal{F}, \zeta)$, nous pouvons définir une application de $\mathcal{P}(\mathcal{O})$ dans $\mathcal{P}(\mathcal{F})$ et une application de $\mathcal{P}(\mathcal{F})$ dans $\mathcal{P}(\mathcal{O})$, notées identiquement par l'opérateur $'$ de la façon suivante :

$$\forall \mathcal{A} \subseteq \mathcal{O}, \mathcal{A}' = \{f \in \mathcal{F} \mid \forall o \in \mathcal{A}, \zeta(o, f) = 1\} \quad (1)$$

$$\forall \mathcal{B} \subseteq \mathcal{F}, \mathcal{B}' = \{o \in \mathcal{O} \mid \forall f \in \mathcal{B}, \zeta(o, f) = 1\} \quad (2)$$

Ces deux applications constituent la **correspondances de Galois** *du contexte ; $\mathcal{A}'$ est appelé* **dual** *de $\mathcal{A}$, et de façon similaire $\mathcal{B}'$ est appelé* **dual** *de $\mathcal{B}$.*

Pratiquement, $\mathcal{A}'$ est l'ensemble des attributs communs à tous les objets de $\mathcal{A}$, et $\mathcal{B}'$ est l'ensemble des objets qui possèdent en commun tous les attributs de $\mathcal{B}$. On peut trouver les propriétés des correspondances de Galois dans [5].

Nous pouvons maintenant énoncer la définition d'un concept :

Définition 4
Etant donné un contexte $\mathcal{K} = (\mathcal{O}, \mathcal{F}, \zeta)$, le couple $\mathcal{C} = (\mathcal{A}, \mathcal{B})$ est appelé un **concept** *de $\mathcal{K}$ si et seulement si $\mathcal{A}' = \mathcal{B}$ et $\mathcal{B}' = \mathcal{A}$.*

Définition 5
*$\mathcal{A}$ est appelé l'***extension** *du concept $\mathcal{C}$ et $\mathcal{B}$ est appelé son* **intension**. *On note $\mathcal{A} = \texttt{extent}(\mathcal{C})$ et $\mathcal{B} = \texttt{intent}(\mathcal{C})$.*

L'ensemble de tous les concepts d'un contexte $\mathcal{K}$ s'appelle $\mathcal{L}(\mathcal{K})$ ou simplement $\mathcal{L}$ pour abréger les notations s'il n'y a pas d'ambigüité. On peut prouver le théorème suivant [14].

Théorême 1
Soient $\mathcal{C}_1 = (\mathcal{A}_1, \mathcal{B}_1)$ et $\mathcal{C}_2 = (\mathcal{A}_2, \mathcal{B}_2)$ un couple de concepts alors $\mathcal{C}_1 \vee \mathcal{C}_2 = ((\mathcal{A}_1 \cup \mathcal{A}_2)'', \mathcal{B}_1 \cap \mathcal{B}_2)$ et $\mathcal{C}_1 \wedge \mathcal{C}_2 = (\mathcal{A}_1 \cap \mathcal{A}_2, (\mathcal{B}_1 \cup \mathcal{B}_2)'')$ sont des concepts.

Ce résultat peut être étendu à tout ensemble $\mathcal{I}$ de concepts. On notera $\mathcal{C}_\mathcal{I} = (\mathcal{A}_\mathcal{I}, \mathcal{B}_\mathcal{I}) = \bigvee_{i \in \mathcal{I}} \mathcal{C}_i$ et de même $\mathcal{C}^\mathcal{I} = (\mathcal{A}^\mathcal{I}, \mathcal{B}^\mathcal{I}) = \bigwedge_{i \in \mathcal{I}} \mathcal{C}_i$
Alors l'ensemble des concepts $\mathcal{L}$ muni de la relation d'ordre $\subseteq$ (inclusion) sur ses extensions est un treillis complet.

Définition 6
Le treillis complet $\mathcal{L}(\mathcal{K})$ des concepts d'un contexte $\mathcal{K}$ est appelé le **treillis de Galois** *ou le* **treillis de concepts**.

4.2 Les algorithmes de construction de treillis

Il existe deux familles d'algorithmes de construction de treillis : les algorithmes incrémentaux et les algorithmes non incrémentaux. Les algorithmes incrémentaux (*Godin* [17], *Carpineto & Romano* [6], *Norris* [27], ...) construisent le treillis au fur et à mesure que les objets arrivent, alors que les algorithmes non incrémentaux (*Cheïn* [8], *Ganter* [13], *Bordat* [4],...) construisent le treillis une fois le contexte entièrement connu.

Plusieurs travaux se sont attachés à comparer les algorithmes de construction de treillis entre eux, essentiellement en termes de complexité algorithmique et temps de calcul. Notons ceux effectués par Kuznetsov *et al.* [23] et par Nguifo *et al.* [26].

5 LES AMERS

Une fois l'information synthétisée dans un treillis de concepts, il est nécessaire d'extraire les amers du treillis sous forme de combinaisons de caractéristiques.

5.1 Utilisation des concepts pour la sélection d'amers

Le lien entre concept et amer se définit de la façon suivante :

Définition 7
Considérons un contexte $(\mathcal{O}, \mathcal{F}, \zeta)$ et un sous-ensemble d'objets $\mathcal{A} \subseteq \mathcal{O}$. On dit qu'un sous-ensemble $\mathcal{B}$ de $\mathcal{F}$ est un **amer** *de $\mathcal{A}$ si et seulement si*
- $\mathcal{B}'' = \mathcal{B}$,
- *et* $\mathcal{B}' \subseteq \mathcal{A}$.

La première condition permet de limiter l'espace des combinaisons de caractériques possibles ($2^{card(\mathcal{F})}$). La seconde condition permet la caractérisation de l'ensemble souhaité.

Un amer est donc l'intension d'un concept du treillis construit.

5.2 Définitions formelles des amers dans un contexte partitionné

5.2.1 Définition générale

On se donne dans ce qui suit un contexte propre $(\mathcal{O}, \mathcal{F}, \zeta)$, et une partition $(\mathcal{O}_\theta)_{\theta \in \Theta}$ de son ensemble d'objets. L'ensemble Θ est appelé **ensemble des sites**, ou plus généralement **ensemble des classes** dans le contexte d'une classification générale.

Définition 8
On dit qu'un sous-ensemble $\mathcal{B}$ de $\mathcal{F}$ est un **amer** *de $\theta \in \Theta$ si et seulement si*
- $\mathcal{B}'' = \mathcal{B}$,
- *et* $\mathcal{B}' \subseteq \mathcal{O}_\theta$.

Ainsi un amer est un ensemble de caractéristiques dont l'occurrence simultanée n'est effective que dans *certaines* images du site à caractériser.

5.2.2 Amer plein

En particulier, si un amer $\mathcal{B}$ est un ensemble de caractéristiques simultanément présentes dans *toutes* les images du site à caractériser, et seulement dans celles-là, on parle alors d'**amer plein** :

Définition 9
On dit qu'un sous-ensemble $\mathcal{B}$ de $\mathcal{F}$ est un **amer plein** *de $\theta \in \Theta$ si et seulement si*
- $\mathcal{B}'' = \mathcal{B}$,
- *et* $\mathcal{B}' = \mathcal{O}_\theta$.

L'existence d'amers pleins rend la classification simple à mettre en œuvre, mais en général la propriété de présence dans toutes les images n'est pas vérifiée.

5.2.3 Amer maximal

Dans le cas où il n'existe pas d'amer plein, afin de limiter le nombre d'amers à considérer (et à retenir), il est intéressant d'introduire les amers maximaux :

Définition 10
Un **amer maximal** *est un amer minimal pour l'inclusion dans l'ensemble des amers d'un site.*

En effet, si deux amers sont comparables, un amer sera supérieur à l'autre s'il décrit un ensemble d'images plus important.

5.2.4 Couverture

La couverture d'un site se définit de la façon suivante :

Définition 11
Considérons les images $\mathcal{O}_\theta$ d'un site $\theta \in \Theta$ et l'ensemble N_θ des amers $\{\mathcal{B}_i\}_{i=1\ldots N_\theta}$ de ce site. On dit que le site θ est **couvert***, ou que les images du site sont* **entièrement couvertes***, si et seulement si $\bigcup_{i=1\ldots N_\theta}\{\mathcal{B}_i^{'}\} = \mathcal{O}_\theta$*

Dans le cas où il existe un amer plein d'un site, la couverture est évidente. Dans l'autre cas, certaines images peuvent ne pas être couvertes par des amers. Il est clair que s'il existe une couverture du site, elle se fait par les amers (non pleins) maximaux.

6 CONSTRUIRE UN CLASSIFIEUR À L'AIDE D'AMERS

Dans cette partie, nous exposons le raisonnement complet relatif à l'extraction d'amers de chaque ensemble d'images dans un premier temps, puis l'utilisation des ces amers dans un deuxième temps.

Notre approche ici diffère essentiellement de par l'aspect topologique de l'environnement, qui amène vers l'approche décentralisée développée plus loin.

Détaillons quelque peu notre application. Nous disposons d'un ensemble d'images issues d'un environnement structuré. Chaque image appartient à une pièce particulière de cet environnement, lieu où elle a été prise par un robot équipé d'une caméra CCD. L'ensemble des images est donc partitionné en fonction du nombre de pièces. Notre objectif est de donner au robot les moyens de se repérer sur une carte topologique[3] (et, dans un futur proche,

[3] Une carte topologique d'un environnement structuré est un graphe pour lequel (en général, mais pas nécessairement) chaque nœud représente une pièce de l'environnement et chaque arc un passage entre deux pièces.

de construire la carte) de cet environnement. Il s'agit, de façon basique, d'un problème de classification supervisée. La règle de décision est extraite à partir d'un ensemble d'images appartenant à une base d'apprentissage. La règle est formalisée pour chaque sous-ensemble (un sous-ensemble correspondant à une pièce) en y associant un ensemble d'amers. Certaines images peuvent échapper à la règle de décision, et des erreurs d'apprentissage peuvent apparaître.

Un amer est une entité propre à chaque pièce ; même si son occurrence est très faible, cette entité est caractéristique de la pièce. Ce qui diffère des approches plus traditionnelles de classification supervisée (comme les treillis de Galois alpha [32], ou tout autre système de classification supervisée décrits dans [12]).

Il existe classiquement deux phases dans notre processus : la phase d'apprentissage, qui étiquette à chaque pièce un ensemble d'amers, et la phase de généralisation (ou de classemement), qui utilise ces amers pour reconnaître telle ou telle pièce à partir de nouvelles images.

6.1 Phase d'apprentissage : extraction des amers

Dans un premier temps, il est nécessaire d'extraire les primitives de chaque image. Les algorithmes utilisés sont relativement classiques.

Pour extraire les segments, l'image est dérivée à l'aide du filtre de *Canny*. Ensuite, les contours sont extraits puis une approximation polygonale de type *Douglas-Peucker* est appliquée. Enfin, les segments sont issus des polygones présents dans l'image.

Pour extraire les couleurs, on utilise la représentation ITS (Intensité, Teinte, Saturation) relativement stable car la teinte (longueur d'onde principale) et la saturation ("pureté" de la teinte) sont isolées de l'intensité. Ensuite, pour extraire des objets dans l'image, c'est-à-dire des ensembles de pixels connexes de même intensité, des filtres et opérateurs morphologiques permettent rapidement de les détecter. Un petit élément structurant (carré 15×15) permet de détecter un petit objet si l'érodé est non vide, et de la même façon un grand élément structurant permet de détecter un grand objet.

Ensuite, dans un deuxième temps, les éventuelles caractéristiques sont déduites de ces primitives, et l'on remplit ainsi la table de correspondance entre les images d'un côté, et les caractéristiques de l'autre.

Dans un troisième temps, le treillis de Galois de la table est construit, hiérarchisant ainsi les ensembles de caractéristiques les unes par rapport aux autres. Dans la pratique, un autre treillis est construit, appelé *treillis d'héritage* [16]. Un treillis d'héritage est un treillis de *h-concepts*. Un h-concept $\tilde{\mathcal{C}}$, correspondant à un concept $\mathcal{C}$, ne comporte dans son extension que les nouveaux objets relatifs à ses descendants, et dans son intension que des nouvelles caractéristiques relatifs à ses ascendants.

Il est nécessaire enfin, dans un quatrième temps, de sélectionner les amers. Pour cela, il est possible d'utiliser soit le treillis "classique" construit, soit le

treillis d'héritage.

Considérant tous les concepts $\mathcal{C}_\theta$ du treillis classique relatifs à une même pièce θ, c'est-à-dire les concepts qui ont pour extension que des images de cette pièce, les amers sont les intensions de ces concepts.

Considérant tous les h-concepts $\tilde{\mathcal{C}}_\theta$ du treillis d'héritage relatifs à une même pièce θ, il est nécessaire de sélectionner...

1. les h-concepts relatifs à la pièce θ ;
2. les h-concepts ascendants et descendants des h-concepts relatifs à cette pièce ;
3. et les h-concepts qui n'ont pas comme descendants un h-concept dont l'extension renferme une image issue d'une autre pièce $\theta^* \neq \theta$.

Les amers sont les intensions des h-concepts ainsi sélectionnés. Dans certains cas où les h-concepts du treillis d'héritage ne donnent pas de résultat, il est alors nécessaire de retrouver l'intension du concept correspondant, en regroupant les attributs des h-concepts amonts.

La notion de h-concept maximal est identique à précédemment, mais la définition est différente en ce sens où il n'y a plus d'inclusion possible entre les sous-ensembles d'attributs. Aussi un h-concept maximal se définit-il structurellement dans le treillis, à savoir qu'un h-concept est maximal s'il n'existe pas de h-concept en amont dans le treillis.

Une fois les images analysées et le contexte général extrait, l'algorithmique de sélection d'amers est présentée figure 3.

1. Construire le treillis de Galois ou le treillis d'héritage correspondant au contexte
2. Si (treillis de Galois), pour chaque pièce :
   21. Sélectionner les concepts ayant pour extension uniquement des images de la pièce considérée
   22. Sélectionner les amers comme intensions des concepts sélectionnés
3. Si (treillis d'héritage), pour chaque pièce :
   31. Sélectionner les h-concepts ayant pour extension uniquement des images de la pièce considérée
   32. Sélectionner les ascendants et les descendants de ces h-concepts
   33. Éliminer les h-concepts qui ont pour descendants des h-concepts qui renferment des images d'autres pièces dans leur extension
   34. Sélectionner les h-concepts maximaux
   35. Enfin, si nécessaire, lire les intensions des concepts correspondants aux h-concepts

FIG. 3 – Algorithme général de sélection d'amers.

6.2 Phase de généralisation : Reconnaissance de la pièce

Une fois les amers sélectionnés, nous considérons une nouvelle image issue des mêmes pièces mais différente de celles utilisées lors de l'apprentissage. Les primitives sont extraites et les caractéristiques déterminées. Plusieurs cas sont possibles :

- si l'ensemble des caractéristiques inclut un ou plusieurs amers d'une seule et unique pièce, alors il est possible de conclure que l'image appartient à cette pièce ;
- si l'ensemble des caractéristiques n'inclut aucun amer, on ne peut pas conclure et le robot a pour tâche de se déplacer et de prendre une autre image (on parle alors de *"vision active"*) ;
- si l'ensemble des caractéristiques inclut un ou plusieurs amers de plusieurs pièces différentes, alors non seulement on ne peut conclure, mais également le processus d'apprentissage est remis en cause.

Une des propriétés particulières de classification à l'aide de treillis est l'absence d'erreur d'apprentissage. En revenche, il admet des "non-décisions" sur certaines images. La remise en cause du processus d'apprentissage lors de la phase de généralisation consiste donc à deux choses : la première est de signifier que l'échantillon d'apprentissage n'est pas "suffisamment" représentatif ; la seconde -lorsque cela est possible- est de mettre à jour le treillis si cette image est correctement située *a posteriori*.

7 L'APPROCHE DÉCENTRALISÉE

La précédente approche donne de très bons résultats pour des contextes de petite taille (moins de 100 objets / caractéristiques). Cependant, la complexité exponentielle des algorithmes (voir [22]) ne permet pas de travailler dans de très gros contextes. De plus, l'exploitation du treillis risque d'être délicate de par sa complexité hiérarchique entre les concepts : il peut y avoir tant de concepts et tant de correspondances entre eux qu'aucun amer ne pourrait être sélectionné.

C'est pourquoi nous introduisons ici une approche *décentralisée* pour gérer les données. L'idée principale est de construire non plus un seul treillis pour l'ensembles des images mais autant de treillis qu'il existe de pièces. Ces treillis seront de taille plus réduite, mais il sera alors nécessaire de considérer les interactions entre treillis. Cette approche présente beaucoup d'avantages (voir section 8.2) en termes de temps de calcul et de lisibilité.

7.1 Formalisme mathématique (2)

Nous définissons ici les contextes, concepts et treillis locaux puis locaux modifiés dont nous avons besoin plus loin dans notre application.

Définition 12
Le **contexte général** *ou* **contexte global** $\mathcal{K}$ *est le contexte 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* $\mathcal{K} = (\mathcal{O}, \mathcal{F}, \zeta)$.

Le contexte général est alors divisé en sous-contextes, relatifs à chaque ensemble.

Définition 13
Soit $\mathcal{O} = \{\mathcal{O}_\theta\}_{\theta \in \Theta}$ *les sous-ensembles d'objets du contexte général* $\mathcal{K}$*, on a alors* $\mathcal{O} = \bigoplus_j \theta_j$*.* θ_j *est appelé* **classe d'objets** *relatif au sous-ensemble d'objets concerné.*

Définition 14
On appelle **application locale** ζ_θ *la restriction de* ζ *à la classe d'objets* θ.

On a alors : $\zeta = \bigoplus_{\theta \in \Theta} \zeta_\theta$

Définition 15
Soit $\mathcal{K}$ *un contexte multi-classes, on appelle* **contexte local** $\mathcal{K}_\theta$ *relative à la classe* θ *le contexte* $\mathcal{K}_\theta = (\theta, \mathcal{F}, \zeta_\theta)$

Une fois les contextes locaux définis, il est simple de définir les concepts locaux et treillis locaux :

Définition 16
On appelle **concept local** $\mathcal{C}_\theta$ *tout concept issu du contexte local* $\mathcal{K}_\theta$.

Définition 17
On appelle **treillis de Galois local** $\mathcal{L}_\theta$ *le treillis issu du contexte local* $\mathcal{K}_\theta$.

Cependant, comme nous allons le voir, les contextes locaux vont être modifiés par la sélection préalable d'attributs. Nous définissons donc :

Définition 18
Soit $\mathcal{K}_\theta$ *un contexte local et* ζ'_θ *une modification de l'application locale* ζ_θ*. On appelle* **contexte local modifié** *le contexte* $\mathcal{K}'_\theta = (\theta, \mathcal{F}, \zeta'_\theta)$.

De la même façon que précédemment, il est possible de définir les concepts locaux modifiés et treillis locaux modifiés :

Définition 19
On appelle **concept local modifié** $\mathcal{C}'_\theta$ *tout concept issu du contexte local modifié* $\mathcal{K}'_\theta$.

Définition 20
On appelle **treillis de Galois local modifié** $\mathcal{L}'_\theta$ *le treillis local issu du contexte local modifié* $\mathcal{K}'_\theta$.

Nous pouvons également définir des amers pleins à partir de la structure des treillis locaux modifiés.

7.2 Sélection de caractéristiques

À chaque sous-ensemble est ainsi associé un treillis, qui sera simplifié en enlevant des caractéristiques *non essentielles* dans une classe. En effet, il est nécessaire de retirer des caractéristiques soit qui alourdissent inutilement un contexte local, soit que ne permettent pas de distinguer les classes entre elles.

Une caractéristique i sera considérée comme **essentielle** dans la classe θ si sa fréquence d'apparition $r_{i,\theta}$ dans la classe θ est bien supérieure à sa fréquence d'apparition $r_{i,\theta^* \neq \theta}$ dans toute autre classe θ^* du contexte général.

Notre heuristique se base donc sur un taux relatif $\tau_{i,\theta,\theta^*} = r_{i,\theta}/r_{i,\theta^*}$. Si $\forall \theta^* \neq \theta$, $\tau_{i,\theta,\theta^*} \gg 1$, la caractéristique i est gardée pour la classe θ ; sinon, elle en est retirée.

Ainsi, l'ensemble des caractéristiques relatives à chaque classe, hiérarchisés dans le treillis, forment les amers de chaque sous-ensemble.

7.3 Algorithme général

L'algorithme général de sélection d'amers avec l'approche décentralisée est exposé figure 4. Notons que le processus de décision est identique que précédemment (section 6.2).

```
1.  Pour chaque site θ,
2.    Pour chaque caractéristique i,
3.      Calculer le taux de présence r_{i,θ}
4.    Fin Pour
5.  Fin Pour
6.  Pour chaque site θ,
7.    Pour chaque caractéristique i,
8.      Pour chaque site θ* ≠ θ,
9.        Si τ_{i,θ,θ*} ≃ 1 or τ_{i,θ,θ*} ≪ 1
10.         Modifier le contexte local
            (càd retirer i du site θ)
11.       Fin Si
12.     Fin Pour
13.   Fin Pour
14. Construire les treillis locaux modifiés
15. Extraire les amers
16. Fin Pour.
```

FIG. 4 – Algorithme général de la sélection d'amers par approche décentralisée.

8 EXPÉRIMENTATIONS

Plusieurs expérimentations ont été menées afin de valider ces nouvelles techniques de sélection d'amers visuels. Nous en exposons trois différentes :

- la première expérimentation (approche centralisée) expose les résultats détaillés avec un petit contexte (15 images d'apprentissage, 22 images de test, et 50 caractéristiques), le but étant de concrétiser et d'expliciter les notions théoriques développées dans ce papier ;
- la seconde expérimentation (approche décentralisée) utilise le même contexte que ci-dessus, puis compare les temps de traitement pour des contextes stochastiques de taille variable ;
- la troisième expérimentation expose un cas réel de robotique mobile, avec un contexte beaucoup plus important (177 images d'apprentissage, 151 images de test, et 66 caractéristiques) et compare les résultats obtenus avec un réseau de neurones optimisé.

Dans tous les cas, le processus de prise d'images se déroule en deux temps : un premier temps pour la phase d'apprentissage, suivi d'un deuxième temps pour la phase de test.

8.1 Approche centralisée

Nous avons mis en œuvre cette approche à partir d'images issues de pièces différentes d'un environnement structuré, en l'occurrence une partie du LAAS.

Un ensemble de 15 images ont servi de base d'apprentissage, réparties en 4 pièces différentes, et nous considérons ici 50 caractéristiques potentielles, numérotées de #1 à #50 :

- peu de pixels (> 50), des pixels (> 300), beaucoup (> 3000) de pixels de rouge, vert, bleu, cyan, magenta, jaune ($3 \times 6 = 18$ caractéristiques), petit ($> 15 \times 15$), moyen ($> 50 \times 50$) ou gros ($> 75 \times 75$) objet R-V-B-C-M-J (18) ;
- un grand nombre (> 50), un très grand nombre (> 100) de segments horizontaux, verticaux, ou de direction quelconque (3×2), 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).

Ensuite, la table de correspondance a été remplie et le treillis correspondant construit (algorithme de Carpineto & Romano[6]), figure 5.

Pour chaque ensemble d'images, les concepts correspondants ont été trouvés et l'on obtient (table 1) :

- pour le premier ensemble (images 1,2), {#42 et #45} est un amer plein et {#22.#28.#39} un amer non plein ; {#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 non pleins ; {#5} (*objet rouge moyen*)

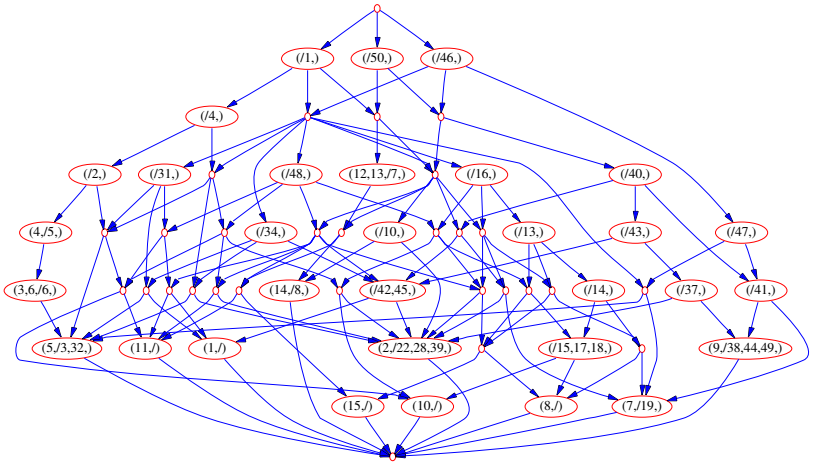


FIG. 5 – *Treillis global correspondant à notre application.*

est l'amer maximum qui couvre ce site ;

- pour le troisième ensemble (images 7,8,9,10), pas d'amer plein mais quatre amers non pleins : {#19}, {#41}, {#38.#44.#49}, {#15.#17.#18} et {#14}. Parmis ces amers non pleins, 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 gand 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 non plein. {#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 non pleins. Dans le cas contraire, parmi les amers non pleins, on ne retiendra que les amers maximaux.

Enfin, pour la phase de test, 20 nouvelles images ont été prises dans les quatre pièces différentes. 18 ont été correctement situées (c'est-à-dire que le système a su reconnaître le lieu où elles ont été prises), les deux dernières ne permettant aucune conclusion.

Dans l'application qui nous concerne, si une image n'est pas située, cela n'a aucune importance dans le sens où le robot est capable de "regarder ailleurs" et d'extraire d'autres images de la pièce où il se situe pour reconnaître cette pièce.

Lieu	**Amer plein**	**Amers non pleins**	**Amers maximaux**
Pièce n^o1	{#42.#45}	{#22.#28#39}	{#42.#45}
Pièce n^o2	{#5}	{#6}, {#3.#32}	{#5}
Pièce n^o3	$\emptyset$	{#19}, {#41}, {#38. #44.#49}, {#15.#17.#18}, {#14}	{#41}, {#14}
Pièce n^o4	{#7}	{#8}	{#7}

TAB. 1 – Les différents amers

8.2 Approche décentralisée

Nous avons fait deux expériences : la première montre l'intérêt d'une approche décentralisée en terme de temps de calcul et nombre de concepts global, et la seconde reprend le contexte précédent en comparaison avec l'approche centralisée.

8.2.1 Temps de calcul et nombre de concepts

L'algorithme ci dessus a été testé sur des contextes carrés aléatoires de taille 5×5 à 35×35, avec trois classes. La densité est de 20%, similaire à celle obtenue en expérimentation réelle. L'algorithme λ est l'algorithme de Carpineto & Romano. La figure 6 montre l'avantage, au delà de 25 objets, de l'approche décentralisée en terme de temps de calcul et de nombre de concepts global.

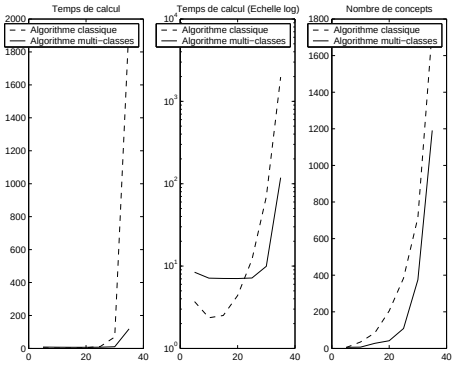
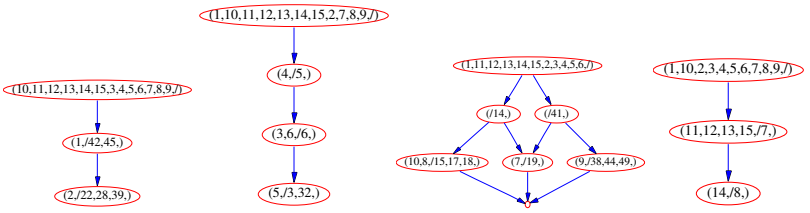


FIG. 6 – Nombre d'objets/propriétés, temps et nombre de concepts.

8.2.2 Résultats

L'approche décentralisée a permis de construire les treillis locaux présentés figure 7. Il suffit alors de lire les amers (table 1) à partir de ces treillis (figure 7), ce qui se fait de façon bien plus simple que précédemment. Il est à noter que les résultats (amers) sont identiques, c'est pourquoi nous nous référons au même tableau.



(a) Pièce 1 (b) Pièce 2 (c) Pièce 3 (d) Pièce 4

FIG. 7 – Les treillis locaux correspondants aux quatre pièces de l'environnement.

8.3 Application réelle à la robotique mobile autonome

Une expérimentation réelle de plus grande échelle a été menée afin de comparer notre approche avec une autre approche plus classique, les réseaux de neurones.

Le robot se déplace dans quatre pièces du laboratoire et acquiert les images dans un flux continu (2 Hz). Le nombre de caractéristiques 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 langage C) 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 (non pleins) maximaux ont été retenus (tableau 2).

Pendant la phase de test, 32 images différentes sont issues de la première pièce. Le robot y cherche dans chacune d'elles des amers enregistrés lors de l'apprentissage : une image contient deux amers ambigus (un de la place 1, un de la place 3), une image contient un amer d'une autre pièce, 14 ne contiennent pas d'amers, et 17 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% (1/32) et relatif de 5,88% (1/17). Tous les résultats sur la base de test sont regroupé tableau

Place	**Amers pleins**	**Amers non pleins**	**Amers maxi-maux**
Place #1	0	194	9
Place #2	0	316	8
Place #3	0	291	17
Place #4	0	82	8
Total :	0	883	42

TAB. 2 – Extraction d'amers.

3 (NI : nombre d'images ; NR : nombre de réponses ; NR+ : nombre de réponses justes ; NR- : nombre de réponses fausses.)

Place	**NI**	**NR**	**NR+**	**NR-**
Place #1	32	17 (53.1%)	16 (94.1%)	1 (5.88%)
Place #2	50	13 (26%)	12 (92.3%)	1 (15%)
Place #3	31	10 (32.3%)	10 (100%)	0 (0%)
Place #4	38	20 (52.6%)	19 (95%)	1 (5%)
Total :	151	60 (39.8%)	57 (95%)	**3 (5%)**

TAB. 3 – Résultats obtenus avec un 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 d'apprentissage, 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.

Afin de comparer ces résultats, un réseau de neurones à été mis en œuvre sous MATLAB composé de 66 neurones dans la couche d'entrée (correspondants aux 66 caractéristiques), 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, eu égard aux très nombreuses configurations différentes testées pour 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.

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 sur la base d'apprentissage est de 5% et sur la base de test est de 30%... De plus, une très grande variabilité caractérise ces résultats : ces chiffres optimum ne sont donnés par le réseau qu'un 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 adaptatif afin d'avoir le même taux de "non-réponse". Ainsi, le réseau ne répond que s'il est relativement sur de sa réponse. Les résultats présentés tableau 4, bien qu'optimaux pour un réseau de neurones, montrent une efficacité moins importante que l'approche que nous proposons ici.

Place	**NI**	**NR**	**NR+**	**NR-**
Place #1	32	17 (53.1%)	17 (100%)	0 (0%)
Place #2	50	17 (26%)	17 (100%)	0 (0%)
Place #3	31	14 (45.2%)	10 (71.4%)	4 (28.6%)
Place #4	38	12 (31.6%)	10 (83.3%)	2 (16.6%)
Total :	151	60	54	**6 (10%)**

TAB. 4 – Résultats avec un réseau de neurones optimisé.

9 DISCUSSION ET PERSPECTIVES

Nous avons décrit dans ce papier une nouvelle application de la théorie des treillis de Galois, afin de caractériser par apprentissage des ensembles d'images. La motivation principale de ce travail vient du monde de la robotique mobile autonome : notre but est de permettre à un robot de caractériser les ensembles d'images issues des différentes pièces d'un environnement structuré, afin de reconnaître chaque ensemble et donc de savoir se situer dans l'environnement.

Nous avons validé notre architecture générale à travers des expérimentations sur des ensembles d'images réelles du laboratoire. Nous avons développé une approche dite *décentralisée*, dans le sens où l'on ne construit plus un treillis général de l'environnement mais n treillis pour les n pièces : on les appelle alors *treillis locaux* attachés aux différentes pièces. Les premières expérimentations dans ce sens donnent de bons résultats.

Les primitives que nous utilisons ici sont relativement simples et de bas niveau. Il est souhaitable pour perfectionner les résultats d'avoir des caractéristiques plus évoluées, éventuellement avec une sémantique quelconque. En

revanche, notre système est *ouvert*, dans le sens où toute information (et pas seulement visuelle) peut être intégré dans notre processus.

D'autres méthodes de classification supervisée pourrait utiliser les mêmes données initiales pour résoudre des problèmes de classification. Les réseaux de neurones ou les arbres de décisions semblent les plus indiqués pour cela, et seront testés dans un futur proche. Entre autres, les arbres de décision ont été utilisés pour caractériser les caractères manuscrits utilisant des *tags* [33] qui peuvent être comparés à nos caractéristiques. Ces techniques ont leurs propres limitations. Les réseaux de neurones convergent très difficilement dès lors que le nombre de neurones dans la couche d'entrée est important. De plus, leur initialisation aléatoire rend l'apprentissage non déterministe. Enfin, la structure du réseau reste très problématique à définir. Les arbres de décision, quant à eux, sont peu flexibles et il est nécessaire d'utiliser une heuristique extrinsèque (comme utiliser la propriété qui sépare l'ensemble des données en deux parties identiques) pour chaque nœud de l'arbre.

Habituellement il est préférable d'avoir un processus de navigation et d'orientation en ligne, afin de limiter les erreurs propres à tout approche statistique : si une erreur survient ou une indétermination apparaît, le robot doit pouvoir réagir en temps réel pour par exemple prendre de nouvelles images afin d'affiner sa décision, et éventuellement modifier sa base d'apprentissage, le (ou les) treillis correspondants et donc ses amers. Notre approche permet une certaine souplesse par la redondance d'informations, à travers les nombreux concepts. Et ce contrairement aux arbres de décision classiques. Dans le processus de construction des arbres de décision de [33], les nœuds pertinents des arbres sont sélectionnés en maximisant une fonction d'entropie. Il sera ainsi difficile pour le système de gérer des informations contradictoires. De fait il y a un compromis à trouver entre redondance et robustesse dans l'apprentissage symbolique des treillis de Galois d'un côté, et leur complexité d'autre part.

La principale limitation de notre approche est le faible taux de réponse donné sur des images d'entrée. Dans le contexte de la robotique mobile, le robot est actif, dans le sens où il lui est possible, s'il ne trouve pas d'amers, de se mouvoir et de chercher dans le lieu des amers. On parle alors de *vision active*. Nous nous attachons à développer un processus complet incluant cette notion.

Enfin, les amers considérés dans ce papier sont des amers visuels, dans le sens où ils ne sont que des caractéristiques issues de la perception visuelle, et ne sont pas reliés à des objets physiques. Le passage de la vision (caractéristiques) au monde réel (propriétés) n'est pas évident à instaurer, mais pourra être considéré à plus ou moins long terme afin d'améliorer notre apprentissage et d'avoir une approche plus "humaine" des amers de l'environnement.

RÉFÉRENCES

[1] A. Arleo, F. Smeraldi, S. Hug et 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.

[2] M. Barbut et B. Monjardet. *Ordre et Classification*. Hachette Université, 1970.

[3] G. Bianco et A. Zelinsky. Biologically-inspired visual landmarks learning and navigation for mobile robots. *IEEE/RSJ, IROS*, 1999.

[4] J. Bordat. Calcul pratique du treillis de galois d'une correspondance. *Mathématique, Informatique et Sciences Humaines*, 1986.

[5] M. Boyer. *Induction de régularités dans une base de connaissance, application au phénomène bruit/Gêne et ses extensions*. PhD thesis, École Nationale Supérieure de l'Aéronautique et de l'Espace, 2001.

[6] C. Carpineto et G. Romano. A lattice conceptual clustering system and its applications to browsing retrieval. In *Machine Learning*, volume 24, pages 95–122, 1996.

[7] L. Chaudron, N. Maille et M. Boyer. The (cube) lattice model and its applications. *Applied Artificial Intelligence, vol 19, no 3*, 2003.

[8] M. Chein. Algorithme de recherche des sous-matrices premières d'une matrice. *Bulletin Mathématique de la Sociologie Scientifique de la R.S. de Roumanie*, 1969.

[9] Z. Dodds et G. Hager. A color interest operator for landmark-based navigation. In *AAAI/IAAI*, pages 655–660, 1997.

[10] G. Dudek et D. Jugessur. Robust place recognition using local appearance based methods. *IEEE Int. Conf. on Robotics and Automation*, 2000.

[11] Fred C. Dyer. Spatial memory and navigation by honeybees on the scale of the foraging range. *The Journal of Experimental Biology*, 199 :147–154, 1996.

[12] H. Fu et E. Mephu Nguifo. How well go lattice algorithms on currently used machine learning testbeds ? In *Quatrièmes journées d'Extraction et de Gestion des Connaissances*, 2004.

[13] B. Ganter. Two basics algorithms in concept analysis. Technical report, Technische Hochschule Darmstadt, 1984.

[14] B. Ganter et R. Wille. *Formal Concept Analysis*. Springer-Verlag, 1999.

[15] P. Gaussier, C. Joulin, J.P. Banquet, S. Leprêtre et A. Revel. The visual homing problem : An example of robotics/biology cross fertilization. *Robotics and Autonomous Systems*, 2000.

[16] R. Godin, G. Mineau et R. Missaoui. Incremental concept formation algorithms based on galois (concept) lattices. *Computational Intelligence, vol 11*, 1995.

[17] R. Godin, R. Missaoui et H. Alaoui. Learning algorithms using a galois lattice structure. In *IEEE Int. Conf. on Tools for Artificial Intelligence*, 1991.

[18] 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.

[19] J.B. Hayet, F. Lerasle et M. Devy. A visual landmark framework for indoor mobile robot navigation. *Int. Conf. on Robotics and Automation*, 2002.

[20] L. Itti, C. Koch et E. Niebur. A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(11) :1254–1259, 1998.

[21] M. Knapek, R.-O. Oropeza et D. Kriegman. Selecting promising landmarks. *IEEE Int. Conf. on Robotics and Automation*, 2000.

[22] S.O. Kuznetsov et 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.

[23] S.O. Kuznetsov et S. Obiedkov. Comparing performance of algorithms for generating concept lattices. *Journal of Experimental and Theorical Artificial Intelligence*, 14(2-3), 2002.

[24] M. Liquière et J. Sallantin. Structural machine learning with galois lattice and graphs. *International Conference on Machine Learning*, 1998.

[25] M. Mata, J. M. Armingol, A. de la Escalera et M.A. Salichs. A visual landmark recognition system for topological navigation of mobile robots. *IEEE International Conference on Robotics and Automation (ICRA01)*, 2001.

[26] E. Mephu Nguifo et P. Njiwoua. Supervised classification and formal concept analysis. *International Conference on Formal Concept Analysis, Technische Universität Darmstadt*, 2003.

[27] E.M. Norris. An algorithm for computing the maximal rectangles in a binary relation. *Revue Roumaine de Mathématiques Pures et Appliquées 23 (2)*, 1978.

[28] S. Se, D. Lowe et J. Little. Local and global localization for mobile robots using visual landmarks. In *IEEE International Conference on Intelligent Robots and Systems, IROS 2001*, 2001.

[29] G. Stumme, R. Taouil, Y. Bastide, N. Pasquier et L. Lakhal. Computing iceberg concept lattices with titanic. *Data Knowledge Engineering*, 42(2) :189–222, 2002.

[30] S. Thrun, D. Fox, W. Burgard et F. Dellaert. Robust monte carlo localization for mobile robots. *Artificiel Intelligence*, 2000.

[31] I. Ulrich et I. Nourbakhsh. Appearance-based place recognition for topological localisation. *IEEE Int. Conf. on Robotics and Automation*, 2000.

[32] V. Ventos, H. Soldano et T. Lamadon. Treillis de galois alpha. In *Conférence d'apprentissage*, pages 175–190, 2004.

[33] Y.Amit et D.Geman. Shape quantization and recognition with randomized trees. *Neural computation*, 1997.