Du capteur à la sémantique : contribution à la modélisation d'environnement pour la robotique autonome en interaction avec l'humain

Yohan Breux

THÈSE POUR OBTENIR LE GRADE DE DOCTEUR
DE L'UNIVERSITE DE MONTPELLIER

En Génie Informatique, automatique et traitement du signal

École doctorale : Information, Structures, Systèmes

Unité de recherche LIRMM

Du capteur à la sémantique : contribution à la modélisation d'environnement pour la robotique autonome en interaction avec l'humain.

Présentée par Yohan Breux

Le 29 Novembre 2018

Sous la direction de René Zapata

Devant le jury composé de

M. Olivier Stasse, Directeur de Recherche, LAAS	Rapporteur
M. Pierre Drap, Chargé de recherche - HdR, LSIS	Rapporteur
M. Pascal Poncelet, Professeur, Université de Montpellier-LIRMM	Président du jury
M. Sébastien Lengagne, MC, Université Clermont Auvergne-Institut Pascal	Examinateur
M. René Zapata, Professeur, Université de Montpellier-LIRMM	Directeur de Thèse
M. Sébastien Druon, MC, LIRMM	Encadrant de Thèse

UNIVERSITÉ DE MONTPELLIER

Remerciements

Je tiens ici à remercier M. Sébastien Druon qui m'a encadré durant ces trois années de thèse. Il m'a laissé libre de poursuivre mes idées mais il était présent pour me remettre sur la voie lorsque j'étais dans le doute. Un grand merci également à tous les membres (présents et passés) permanents et doctorants de l'équipe EXPLORE. Je remercie tout particulièrement M. Didier Crestani pour ses nombreux conseils ainsi que M. Robin Passama pour sa précieuse aide sur des problèmes de programmation.

Résumé

La robotique autonome est employée avec succès dans des environnements industriels contrôlés, où les instructions suivent des plans d'action prédéterminés. La robotique domestique est le challenge des années à venir et comporte un certain nombre de nouvelles difficultés : il faut passer de l'hypothèse d'un monde fermé borné à un monde ouvert. Un robot ne peut plus compter seulement sur ses données capteurs brutes qui ne font qu'indiquer la présence ou l'absence d'objets. Il lui faut aussi comprendre les relations implicites entre les objets de son environnement ainsi que le sens des tâches qu'on lui assigne. Il devra également pouvoir interagir avec des humains et donc partager leur conceptualisation à travers le langage. En effet, chaque langue est une représentation abstraite et compacte du monde qui relie entre eux une multitude de concepts concrets et purement abstraits. Malheureusement, les observations réelles sont plus complexes que nos représentations sémantiques simplifiées. Elles peuvent donc rentrer en contradiction, prix à payer d'une représentation finie d'un monde "infini".

Pour répondre à ces difficultés, nous proposons dans cette thèse une architecture globale combinant différentes modalités de représentation d'environnement. Elle permet d'interpréter une représentation physique en la rattachant aux concepts abstraits exprimés en langage naturel. Le système est à double entrée : les données capteurs vont alimenter la modalité de perception tandis que les données textuelles et les interactions avec l'humain seront reliées à la modalité sémantique. La nouveauté de notre approche se situe dans l'introduction d'une modalité intermédiaire basée sur la notion d'instance (réalisation physique de concepts sémantiques). Cela permet notamment de connecter indirectement et sans contradiction les données perceptuelles aux connaissances en langage naturel.

Nous présentons dans ce cadre une méthode originale de création d'ontologie orientée vers la description d'objets physiques. Du côté de la perception, nous analysons certaines propriétés des descripteurs image génériques extraits de couches intermédiaires de réseaux de neurones convolutifs. En particulier, nous montrons leur adéquation à la représentation d'instances ainsi que leur usage dans l'estimation de transformation de similarité. Nous proposons aussi une méthode de rattachement d'instance à une ontologie, alternative aux méthodes de classification classique dans l'hypothèse d'un monde ouvert. Enfin nous illustrons le fonctionnement global de notre modèle par la description de nos processus de gestion de requête utilisateur.

Abstract

Autonomous robotics is successfully used in controlled industrial environments where instructions follow predetermined implementation plans. Domestic robotics is the challenge of years to come and involve several new problematics : we have to move from a closed bounded world to an open one. A robot can no longer only rely on its raw sensor data as they merely show the absence or presence of things. It should also understand why objects are in its environment as well as the meaning of its tasks. Besides, it has to interact with human beings and therefore has to share their conceptualization through natural language. Indeed, each language is in its own an abstract and compact representation of the world which links up variety of concrete and abstract concepts. However, real observations are more complex than our simplified semantical representation. Thus they can come into conflict : this is the price for a finite representation of an "infinite" world.

To address those challenges, we propose in this thesis a global architecture bringing together different modalities of environment representation. It allows to relate a physical representation to abstract concepts expressed in natural language. The inputs of our system are two-fold : sensor data feed the perception modality whereas textual information and human interaction are linked to the semantic modality. The novelty of our approach is in the introduction of an intermediate modality based on instances (physical realizations of semantic concepts). Among other things, it allows to connect indirectly and without contradiction perceptual data to knowledge in natural langage.

We propose in this context an original method to automatically generate an ontology for the description of physical objects. On the perception side, we investigate some properties of image descriptor extracted from intermediate layers of convolutional neural networks. In particular, we show their relevance for instance representation as well as their use for estimation of similarity transformation. We also propose a method to relate instances to our object-oriented ontology which, in the assumption of an open world, can be seen as an alternative to classical classification methods. Finally, the global flow of our system is illustrated through the description of user request management processes.

Table des matières

Table des figures **xiii**

Liste des tableaux **xvii**

Liste des algorithmes **xix**

Notations **xxi**

1 Introduction **1**

1.1 Problématiques de la représentation d'environnement pour la robotique autonome . 3

1.2 Aperçu général de notre approche 4

1.3 Contributions 8

2 Etat de l'art **11**

2.1 Représentation géométrique de l'environnement 11

2.1.1 Segmentation de scène statique en objets 11

2.1.2 Segmentation basée mouvement et modélisation cinématique 14

2.1.3 SLAM 20

2.2 Représentation basée sémantique 21

2.3 Représentations combinant sémantique et perception 24

2.3.1 Description d'objets par attributs 24

2.3.2 Description sémantique d'image 26

2.3.3 Cartographie sémantique 28

2.4 Conclusion 33

3 Modèle de base de connaissance **35**

3.1 Représentation des connaissances par ontologie 36

3.1.1 Base lexicale et ontologie WordNet . . . 36
3.1.2 Ontologie NELL . . . 38
3.1.3 Système KnowRob . . . 40
3.1.4 Discussion . . . 42
3.2 Définition du graphe de connaissance . . . 46
3.2.1 Une ontologie basée objets à partir de définitions . . . 46
3.2.2 Formalisation . . . 50
3.3 Construction du graphe de connaissance . . . 53
3.3.1 Pourquoi WordNet ? . . . 53
3.3.2 Analyse syntaxique de langage naturel . . . 54
3.3.3 Construction automatique d'ontologie à partir de définitions . . . 58
3.4 Evaluation . . . 72
3.4.1 Statistiques générales sur le graphe ontologique . . . 73
3.4.2 Analyse qualitative de l'ontologie générée . . . 74
3.4.3 Analyse quantitative de l'ontologie générée . . . 79
3.5 Notion de contexte . . . 82
3.5.1 Contexte hiérarchique basé hyperonymie . . . 82
3.5.2 Contexte basé sur la répresentation vectorielle de concepts . . . 84
3.5.3 Contexte abstrait basé sur la centralité intermédiaire . . . 84
3.5.4 Extension au contexte commun d'un ensemble de concepts . . . 89
3.5.5 Discussion . . . 92
3.6 Conclusion et perspectives . . . 93

4 Modèle de Perception **95**
4.1 Introduction . . . 95
4.2 Une approche basée sur les descripteurs profonds . . . 96
4.2.1 Réseau de neurones convolués . . . 97
4.2.2 Classification binaire et reconnaissance d'instance . . . 104
4.2.3 Application à la localisation et à la détection . . . 105
4.2.4 Descripteur image générique pour la représentation d'instance . . . 108
4.2.5 Variation des descripteurs CNN relativement à SIM(2) . . . 111
4.2.6 Applications . . . 113
4.2.7 Expérimentation . . . 123
4.2.8 Formalisation . . . 131

4.3 Conclusion et perspectives 142

5 Modèle d'instance et processus inter-modèles 145

5.1 Modèle d'instance 146
5.1.1 Définition d'une instance 146
5.1.2 Définition du modèle d'instance 148
5.1.3 Discussion 150
5.2 Sémantique vers instance 151
5.2.1 Interaction avec l'humain 151
5.2.2 Détection de concepts 153
5.3 Instance vers sémantique 154
5.3.1 Problématiques 154
5.3.2 Arbre de classification ontologique 156
5.4 Perception vers instance / instance vers perception 162
5.5 Tâches 165
5.5.1 Interprétation de requête 166
5.5.2 Résolution de la tâche 170
5.6 Conclusion et perspectives 173

6 Vers une validation expérimentale 175

6.1 Description de l'architecture logicielle 175
6.2 Dispositif expérimental 177
6.3 Cas d'une requête simple 178
6.4 Cas multi-instances 181
6.5 Usage de l'ontologie et de la notion de contexte 187
6.6 Discussion et conclusion 192

7 Conclusion 195

A Part-of-Speech tags 203

B Opérateurs matriciels 207

B.1 Définitions et propriétés 207
B.2 Opérateur de K-dérivation 211
B.3 Preuves 216
B.3.1 Preuve de la proposition 1, section 4.2.8 216

B.3.2 Preuve de la proposition 2, section 4.2.8 217
B.3.3 Preuve de la proposition 4, section 4.2.8 218
B.3.4 Preuve de la proposition 5, section 4.2.8 219

Bibliographie **221**

Table des figures

1.1 Structure générale de notre modèle global de représentation d'environnement . . . 7

2.1 Exemple de segmentation de scène avec hypothèse de surfaces lisses 12
2.2 Segmentation de scène par présegmentation en superpixels 15
2.3 Classification de méthodes de segmentation basées mouvement 16
2.4 Illustration de cartographie par SLAM 21
2.5 Exemple d'instanciation de MLN 23
2.6 Espaces de représentation des descripteurs image et sémantique (attributs) 25
2.7 Approche de Visual Question Answering par [ARDK16] 27
2.8 Approche de Visual Question Answering par [WWS+15] 28
2.9 Grille d'occupation multicouches 29
2.10 Schéma général de l'architecture proposée par [PJ12] 30
2.11 Architecture multimodale proposée par Lang [LFHP15] 31
2.12 Architecture multimodale proposée par Liu [LW14] 32

3.1 Modèle sémantique dans notre architecture 35
3.2 Illustration de la structure de WordNet 37
3.3 Extrait de l'ontologie WordNet 38
3.4 Structure de NELL 40
3.5 Aperçu global du système KnowRob 41
3.6 Exemple de sous-graphes extraits des ontologies NELL et KnowRob 43
3.7 Sous-graphe de l'ontologie générée par notre méthode à partir du concept *fork-n03383948* 46
3.8 Exemples de sous-graphes construits à partir de définitions 48
3.9 Illustration d'un graphe ontologique avec et sans utilisation de nœud facteur d'action 49
3.10 Illustration d'un graphe ontologique avec et sans utilisation de nœud facteur d'objet 50

3.11 Schéma de construction du graphe de connaissance 53
3.12 Exemple d'analyse syntaxique basée transition 56
3.13 Schéma d'un réseau de neurones basé transition 57
3.14 Résultat d'analyse syntaxique au format coNLL-X 59
3.15 Schéma de notre méthode d'extraction de relations sémantiques 61
3.16 Exemple d'analyse sémantique avec des adjectifs en conjonction 65
3.17 Exemples simples de groupes de relation 66
3.18 Exemple de construction d'un sous-graphe ontologique 71
3.19 Sous-graphe centré sur *fork-n03383948* 75
3.20 Sous-graphe centré sur *spoon-n04284002* 76
3.21 Sous-graphe centré sur *automobile-n02958343* 77
3.22 Sous-graphe centré sur *bottle-n02876657* 78
3.24 Schéma du sous-graphe de contexte d'un concept 82
3.23 Sous-graphes de contexte (profondeur $n = 2$) pour différents concepts 83
3.25 Sous-graphes de contexte (profondeur $n = 2$) avec mesure de pertinence 85
3.26 Centralité intermédiaire sur le sous-graphe de profondeur 2 centré sur *hammer-n03481172* 87
3.27 Centralité intermédiaire sur le sous-graphe de profondeur 2 centré sur *hammer-n03481172*, après distribution des valeurs des nœuds facteur 88
3.28 Centralité intermédiaire avec normalisation 89
3.29 Exemples de sous-graphes proportionnés par les valeurs de centralités intermédiaires 90
3.30 Sous-graphe centré sur les concepts *cup-n03147509* et *spoon-n04284002*, proportionné par la mesure de pertinence définie par (3.24) 91
3.31 Sous-graphe contextuel basé sur la centralité intermédiaire pour le concept *cup-n03147509* 92

4.1 Modèle de perception dans notre architecture 95
4.2 Exemple de segmentation erronée obtenue avec LCCP 97
4.3 Structure du réseau de neurones convolutifs AlexNet 97
4.4 Structure du réseau de neurones convolutifs VGG16 98
4.5 Description d'une couche de convolution 99
4.6 Fonctions d'activation d'un CNN et champ réceptif 100
4.7 Interprétation des couches de CNN par reconstruction de l'image d'entrée 102
4.8 Schéma d'un module d'Inception employé par le réseau GoogLeNet 103
4.9 Structure de GoogLeNet 104

4.10 Adaptation d'un FCN avec classifieur binaire RF et similarité cosinus 107
4.11 Moyenne des similarités intra et extra-classes pour les couches des CNNs AlexNet, VGG et GoogleNet 109
4.12 Illustration des *warped convolutions* 112
4.13 Configuration générale de la simulation 113
4.14 Projections sur les deux premières composantes principales de l'ACP pour chaque séquence d'images 114
4.15 Représentation d'une variété 116
4.16 Images de l'expérience avec occlusion 124
4.17 Erreur d'échelle et de rotation pour l'objet tête 125
4.18 Erreur d'échelle et de rotation pour l'objet livre 126
4.19 Erreur d'échelle et de rotation pour l'objet poisson 127
4.20 Erreur d'échelle et de rotation pour l'objet tasse 128
4.21 Exemple de tracking par notre approche 130
4.22 Définition des coordonnées globales et locales pour la formalisation 132
4.23 Schéma des couches à ajouter à un CNN afin de calculer le plan tangent initial . . . 139
4.24 Adaptation des couches de convolutions pour le calcul de la jacobienne du descripteur à la couche l 140
4.25 Décomposition de la variation d'un descripteur 142

5.1 Modèle d'instance et processus dans notre architecture 145
5.2 Définition d'une instance 147
5.3 Schéma du modèle d'instance 149
5.4 Processus entre le modèle de connaissance sémantique et le modèle d'instance . . . 152
5.5 Exemple de heat map obtenue par un FCN doté d'une couche Random Forest . . . 154
5.6 Schéma de notre méthode d'association d'instance à l'ontologie 155
5.7 Histogramme des distances feuille-racine de Wordnet 157
5.8 Ontologie employée pour l'évaluation de notre méthode OCT 158
5.9 Processus entre le modèle de perception et le modèle d'instance 163
5.10 Heat map pour la détection d'instance 164
5.11 Schéma d'un thread de détection concept 171
5.12 Schéma d'un thread de détection instance 172
5.13 Schéma du déroulement interne d'une tâche 173

6.1 Aperçu général des différents élements constituant l'implémentation du système . . 176

6.2 Aperçu général de l'approche multithreads de notre architecture 177
6.3 Schéma de la détection d'instance passive employée dans ce chapitre 178
6.4 Description du cas d'usage basique avant la requête utilisateur 179
6.5 Analyse syntaxique de la requête *"Bring me the cup"* obtenue par SyntaxNet 180
6.6 Etat final du système après résolution de la requête 181
6.7 Analyse syntaxique de la phrase *"My cup is to the left of the car"* 182
6.8 Etat du système après que l'utilisateur ait donné une information concernant l'instance *my_cup* 183
6.9 Système après l'introduction d'objets dans la scène 184
6.10 Analyse syntaxique de l'information *"Bring me the cup behind my cup."* 185
6.11 Fichier temporaire pour la tâche en cours 186
6.12 Etat du modèle après résolution de la requête 187
6.13 Analyse syntaxique de la requête *"Give me something for serving coffee"* 188
6.14 Requête sur l'ontologie afin de déterminer les classes correspondants à l'objet recherché 189
6.15 Résultat de la requête *Give me something for serving coffee.* 190
6.16 Sous-graphe de contexte selon les approches proposées à la section 3.5 191
6.17 Modèle d'instance résultant du cas présenté à la section 6.4 en représentant explicitement *my_cup* 192

7.1 Echelle de causalité 199
7.2 Exemple d'hypothèse de connaissance 200

Liste des tableaux

2.1 Quantificateurs et connecteurs de la logique du premier ordre 22
2.2 Classification des représentations d'environnement rencontrées dans l'état de l'art . 34

3.1 Définition de la structure de notre ontologie 53
3.2 Sous-catégories de WordNet utilisées dans nos travaux 55
3.3 Motifs utilisés pour la classification des groupes de relation 72
3.4 Description statistique globale de notre ontologie 73
3.5 Résultat de l'évaluation de l'ontologie générée 81

4.1 Topologie des réseaux AlexNet et VGG 101
4.2 Classes utilisées pour l'évaluation des descripteurs CNN 109
4.3 Erreurs et variances d'estimation du paramètre d'échelle 129
4.4 Erreurs d'estimation de l'angle de rotation 129

5.1 Relations représentées dans le graphe d'instances 150
5.2 Erreurs de classifications selon la méthode MSVM et OCT 159
5.3 Erreurs de classification par classe selon la méthode MSVM et OCT 160
5.4 Concepts employés dans l'évaluation du rattachement d'instances inconnues 161
5.5 Proportion d'instances correctement rattachées aux concepts de l'ontologie 162
5.6 Type des clauses Prolog 170

A.1 Tags **UPOS** 203
A.2 Tags **XPOS** 204
A.3 Tags **UDR** 205

Liste des Algorithmes

1 Extraction de groupes nominaux (GN) 63
2 Extraction de groupes de relation (GR) 67
3 Extraction de groupes de relation d'usage (GRU) 69
4 Extraction du type d'entrée sémantique 152
5 Algorithme d'analyse d'entrée sémantique 153
6 Processus d'association visuelle des instances détectées avec les instances connues 163
7 Algorithme d'analyse de requête 165
8 Conversion de l'analyse syntaxique en clauses Prolog 167
9 Création d'une règle Prolog définissant une tâche 168

Notations

Abréviations

ACP Analyse en Composante Principale (PCA en anglais), page 112

AE AutoEncodeur, page 25

ASM Articulated Scene Model, page 19

BLN Bayesian Logical Network, page 41

BRNN Bidirectional Recurrent Neural Network, page 27

CNN Convolutional Neural Network, page 1

coNLL Format de description sémantique (Conference on Computational Natural Language Learning), page 58

CPC Constrained Planer Cut, page 14

CPL Clause Prolog, page 167

CRF Conditional Random Field, page 30

DAP Direct Attribute Prediction, page 24

DDVG Depth Dependent Voxel Grid, page 13

EV Espace Vectoriel, page 18

FC (Couche entièrement connectée (Réseau de neurones), page 106

FNC Forme Normale Conjonctive, page 67

FPFH Fast Point Feature Histograms, page 13

GN Groupe nominal, page 62

GRU	Groupe de Relation d'Usage, page 68
GR	Groupe de Relation, page 65
HoG	Histogramme de Gradients, page 108
LCCP	Locally Convex Connected Patches, page 13
LNR	Couche de normalisation locale (Réseau de neurones), page 100
LSTM	Long Short Term Memory, page 26
MCMC	Markov Chain Monte Carlo, page 31
MLN	Markov Logical Network, page 31
MSVM	Multi-Class SVM, page 159
NN	Nearest Neighbor, page 159
OCT	Ontological Classification Tree, page 155
OWL	Web Ontology Language, page 22
POS	Part Of Speech, page 54
RCNN	Region Convolutional Neural Network, page 26
RDF	Resource Description Framework, page 22
ReLU	Unité de rectification linéaire (Réseau de neurones), page 100
RF	Random Forest, page 104
SfM	Structure from Motion, page 15
SIFT	Scale Invariant Feature Transform, page 1
SLAM	Simultaneous Localization and Mapping, page 1
SLIC	Simple Linear Iterative Clustering, page 12
SRM	Statistical Relational Model, page 22
SVM	Support Vector Machine, page 24
UDR	Universal Dependency Relation, page 55
UPOS	Universal Part-Of-Speech, page 54

VCCS Voxel Cloud Connectivity Segmentation, page 13

XPOS Language-specific Part-Of-Speech, page 55

Glossaire

Homonymie — Relation entre plusieurs formes linguistiques ayant le même signifiant graphique et/ou phonique et des signifiés totalement différents, page 47

Hypernymie — Relation sémantique hiérarchique d'une unité lexicale à une autre selon laquelle l'extension du premier terme, plus général, englobe l'extension du second, plus spécifique, page 51

Hyponymie — Relation sémantique d'un lexème à un autre selon laquelle l'extension du premier est incluse dans l'extension du second, page 53

Méronymie — Relation sémantique entre mots, lorsqu'un terme désigne une partie d'un second terme, page 37

Synonymie — Relation entre deux ou plusieurs signifiants, telle que ces signifiants sont interchangeables, sans qu'il y ait variation concomitante du signifié, page 53

Synset — Structure de données de WordNet représentant un concept, page 36

Zero-shot learning — Famille de méthodes d'apprentissage qui supposent l'absence de données d'entraînement pour un ensemble de classes. Elles utilisent une source extérieure d'informations sur ces classes. Classiquement, lorque les classes sont inférés à partir d'image, la source extérieure peut être des informations textuelles., page 23

Symboles

$\boldsymbol{\alpha}$ — Ensemble des arguments d'une clause Prolog, page 167

$\boldsymbol{\alpha}^c$ — Ensemble des arguments de type concept d'une clause Prolog, page 167

$\boldsymbol{\alpha}^f$ — Ensemble des arguments de type facteur d'une clause Prolog, page 167

$\boldsymbol{\alpha}^i$ — Ensemble des arguments de type instance d'une clause Prolog, page 167

$\boldsymbol{\alpha}_m$	Ensemble des arguments (variables et constantes) intervenant dans les clauses constituant une règle Prolog, page 167
$\boldsymbol{\varepsilon}_t$	Elément de l'algèbre $\mathfrak{sim}(2)$ représentant la pose d'un objet à un temps t, page 121
$\wedge$	et logique, page 22
$\vee$	ou logique, page 22
$\mathbf{E(n)}$	Groupe euclidien en dimension n, page 118
$\mathbf{GL_n}(\mathbb{R})$	Groupe des matrices inversibles à valeurs réelles, page 116
$\mathbf{O(n)}$	Groupe orthogonal en dimension n, page 117
$\mathbf{SE(n)}$	Groupe spécial euclidien en dimension n, page 118
$\mathbf{SO(n)}$	Groupe spécial orthogonal en dimension n, page 117
$\mathcal{M}_{m,n}(\mathbb{R})$	EV des matrices à m lignes et n colonnes à valeurs réelles, page 18
$\mathfrak{se}(n)$	Algèbre de Lie associée au groupe $SE(n)$, page 118
$\mathfrak{sim}(n)$	Algèbre de Lie associée au groupe $SIM(n)$, page 119
$\mathfrak{so}(n)$	Algèbre de Lie associée au groupe $SO(n)$, page 118
$\neg$	Opérateur logique de négation, page 22
ϕ_l	Sortie vectorisée de la $l$ème couche d'un CNN, page 115
Π_t	Elément de $\mathbf{SIM(2)}$ représentant la pose d'un objet à un temps t, page 121
$\mathbb{VERB}, \mathbb{NSUBJ}, ...$	Tags sémantiques (cf annexe A), page 54
τ	Vecteur de paramètre d'une similarité, page 122
$A \setminus B$	Différence ensembliste (A privé de B), page 169
C^k	(classe de) fonction k fois dérivable, $k^{\text{ème}}$ dérivée continue, page 117
$C^{\wedge}_{GN(S)}$	Ensemble de sous-ensembles de $GN(S)$ contraint par une conjonction, page 62
$C^{\wedge}_{GR(S)}$	Ensemble de sous-ensemble de $GR(S)$ contraint par une conjonction, page 66

$C^{\vee}_{GN(S)}$	Ensemble de sous-ensembles de $GN(S)$ contraint par une disjonction, page 62
$C^{\vee}_{GR(S)}$	Ensemble de sous-ensemble de $GR(S)$ contraint par une conjonction, page 66
d	Tag **UDR** d'un mot, page 60
E_K	Arêtes du graphe ontologique, page 50
G_K	Graphe ontologique, page 50
$GN(S)$	Ensemble des groupes nominaux d'une phrase S, page 62
$GN(S)_o$	Ensemble des groupes nominaux objet d'une phrase S, page 65
$GN(S)_s$	Ensemble des groupes nominaux sujet d'une phrase S, page 65
gn_o	Groupe nominal objet, page 65
gn_s	Groupe nominal sujet, page 65
gr	Groupe de relation, page 65
$GR(S)$	Ensemble des groupes de relation d'une phrase S, page 65
gru	Groupe de relation d'usage, page 69
$GRU(S)$	Ensemble de groupes de relation d'usage d'une phrase S, page 69
H	Ensemble d'index de mots (format coNLL), page 60
h	Index de dépendance d'un mot (pointe vers le mot en relation), page 60
IDX_{comp}	Ensemble des index des compléments d'un groupe nominal, page 62
idx_{det}	Index du déterminant d'un groupe nominal, page 62
idx_{nom}	Index du nom d'un groupe nominal, page 62
$J^{\tau}_{\phi_l}$	Matrice jacobienne de la sortie de la $l$ème couche d'un CNN fonction de $\mathbf{SIM}(\mathbf{2})$, page 122
$J^{s}_{\phi_l}$	Matrice jacobienne de la sortie de la $l$ème couche d'un CNN fonction de $\mathfrak{sim}(2)$, page 122
J_{τ}	Matrice jacobienne des paramètres $\tau(\boldsymbol{\varepsilon}) \in \mathbf{SIM}(\mathbf{2})$ fonction de son algèbre $\mathfrak{sim}(2)$, page 122

$P(E)$	Ensemble des parties d'un ensemble E, page 89
$p(x)$	Probabilité, page 24
R_m/k	Règle prolog définissant une tâche, page 167
S	Phrase (vecteur de mots), page 61
$SIM(n)$	Groupe de similarité en dimension n, page 119
T_D	Ensemble des tags **UDR**, page 60
T_U	Ensemble des tags **UPOS**, page 60
T_X	Ensemble des tags **XPOS**, page 60
T_xM	Espace tangent en x d'une variété M, page 115
u	Tag **UPOS** d'un mot, page 60
V_K	Noeuds du graphe ontologique, page 50
V_K^A	Sous-ensemble des nœuds actions, page 51
$V_K^{C,A}$	Sous-ensemble des nœuds à la fois concept et action, page 51
$V_K^{C,O}$	Sous-ensemble des nœuds à la fois concept et objet-propriété, page 51
V_K^C	Sous-ensemble des nœuds concepts, page 50
$V_K^{F,A}$	Sous-ensemble des nœuds à la fois facteur et action (nœud facteur d'action), page 51
$V_K^{F,O}$	Sous-ensemble des nœuds à la fois facteur et objet-propriété (nœud facteur d'objet), page 51
V_K^F	Sous-ensemble des nœuds facteurs, page 51
V_K^O	Sous-ensemble des nœuds objets-propriétés, page 51
W	Ensemble des mots, page 60
W_P	Ensemble des prédicats, page 66
W_{comp}	Ensemble des compléments d'un groupe nominal, page 62
w_{det}	Déterminant d'un groupe nominal, page 62
w_{nom}	Nom d'un groupe nominal, page 62
x	Tag **XPOS** d'un mot, page 60

Chapitre 1

Introduction

Les travaux de cette thèse se positionnent sur le problème de la représentation d'environnement dans le cadre de robots autonomes interagissant avec des humains. Le terme de "représentation d'environnement" n'a pas de définition unique et varie en fonction des applications. Considérons la problématique du SLAM (Simultaneous Localization and Mapping) [TBF05] [WLSM$^+$15] : cette famille de méthodes vise à déplacer de façon autonome un robot en le repérant dans une carte de l'environnement qu'il construit lui-même lors de son exploration. L'environnement sera ici classiquement représenté par une carte d'occupation ou un ensemble de primitives (point, surface, volume, mesh, ...) c'est à dire une **représentation géométrique**.

Cette représentation atteint cependant vite ses limites. Prenons l'exemple d'applications nécessitant la manipulation d'objets : il faut un modèle d'environnement segmenté représentant les différents objets à manipuler ou encore leur support (table). La représentation géométrique n'est pas adaptée car elle représente l'environnement comme un "bloc" unique non segmenté. Il faut donc rajouter des informations supplémentaires permettant de déterminer quels ensembles de primitives appartiennent à une même entité afin d'obtenir une **représentation objets** de l'environnement.

Ces informations, lorsqu'elles doivent être acquises dynamiquement, peuvent être obtenues par des méthodes non supervisées, basées par exemple sur les gradients de couleurs ou les normales aux surfaces. Récemment, les méthodes supervisées (*machine learning*) et plus particulièrement les réseaux de neurones sont l'objet de nombreuses recherches. En particulier, les réseaux de neurones convolutifs (dit CNN [1]) ont démontré leur efficacité sur les tâches de classification en vision par ordinateur. L'état de l'art précédent était basé sur des descripteurs images prédéfinis comme SIFT (Scale Invariant Feature Transform) [Low99] ou HoG (Histogram of Gradient) [DT05]. La réelle

1. *Convolutional Neural Network*

rupture apportée par les CNNs est que les descripteurs images sont appris directement à partir des données d'entraînements en une structure hiérarchique et séquentielle. Ceci permet d'adresser des problèmes complexes pour lesquels la modélisation de descripteurs est difficile. Un robot équipé d'un tel système peut rattacher ce qu'il ne voyait auparavant que comme un ensemble de points (données capteurs) à des objets qui nous sont familiers. Par extension, là où la partie cartographie du SLAM se limite à la représentation géométrique de l'environnement du robot, les réseaux de neurones convolutifs permettent de le segmenter (partiellement) en objets.

Cette représentation d'un monde vu comme un ensemble segmenté et labellisé de points n'est toujours pas entièrement satisfaisante, et ce pour plusieurs raisons :

- Elle est limitée à ce qui est visible ou plus globalement à ce qui est mesurable par des capteurs. Par comparaison, nous pouvons nous représenter et reconnaître des objets qui nous étaient inconnus (dans le sens de jamais vus) à partir de descriptions sémantiques en langage naturel.
- La majorité des travaux actuels en intelligence artificielle se focalisent sur l'amélioration des performances sur une tâche (eg. la classification) appliquée à un ensemble *borné* de classes. Comment de tels algorithmes peuvent-ils s'adapter à des classes inconnues ? Il faudrait pour cela pouvoir transférer la connaissance générale acquise lors de l'entraînement sur des cas non vues lors de la phase d'apprentissage : c'est la problématique du *Transfert learning*
- Cette représentation est limitée à la présence (ou l'absence) de primitives et/ou d'objets. Cela donne une capacité de raisonnement pauvre et en particulier ne permet pas de comprendre **pourquoi** un objet est présent. Autrement dit, cette représentation ne permet pas de rendre compte des causes et/ou des contextes sous-jacents à la présence/absence d'objets dans une scène. Prenons l'exemple de l'apprentissage de la probabilité de présence d'objets dans des scènes. Supposons que le système soit entraîné à partir d'images de cuisine et de salle de bain. Il va certainement apprendre que les cuillères et les fourchettes ont de fortes probabilités de présence dans les cuisines, et que les brosses à dent ont également une forte probabilité d'être dans une salle de bain. C'est un fait que le système connaît mais il ne sait pas **pourquoi** : il ne connaît pas la nature du lien reliant ces objets entre eux et/ou la pièce dans laquelle ils se trouvent.

On peut souligner le fait que la problématique de la modélisation d'environnement n'est pas sans rappeler la "machine-enfant" de Turing [Tur95] [Tur48]. Afin de créer une machine intelligente capable de passer le test qui porte son nom, il proposa de ne pas chercher à directement approcher les capacités d'un homme adulte mais de le faire en deux temps : tout d'abord, concevoir une structure primaire qui serait l'équivalent d'un cerveau d'un petit enfant, puis dans un second temps de "l'éduquer". Par analogie, le modèle de représentation d'environnement permettant de

comprendre le monde peut être vu comme la structure équivalente au cerveau d'un enfant, qui est ensuite enrichie à partir de données extérieures ou par l'expérience propre du robot.

Enfin, dans le cadre de la robotique d'assistance, il est primordial pour le robot d'avoir une représentation du monde qui soit proche de celle d'un humain. En particulier, il doit pouvoir raisonner et interpréter des concepts exprimés en langage naturel. Les représentations géométriques et objets sont intéressantes pour le robot lui-même pour des opérations de bas niveau, telles que la localisation ou la préhension, mais semblent insuffisantes lorsque l'on envisage l'interaction avec un utilisateur humain.

1.1 Problématiques de la représentation d'environnement pour la robotique autonome

Essayons de voir quelles difficultés doivent être surmontées par un système robotique pour pouvoir interagir dans des environnements humains :

- **Monde ouvert et capacité d'adaptation :** On a vu dans le paragraphe précédent que la plupart des avancées en intelligence artificielle se reposent sur des méthodes supervisées. Le *Transfert learning* est une tentative d'exploiter les connaissances acquises sur un domaine dans un autre domaine pour lequel les données sont limitées. Des études [BK88] ont montré qu'un enfant de trois ans est déjà doté d'une telle capacité.
- **Interaction homme/robot :** Le robot doit pouvoir comprendre notre langage afin de pouvoir interpréter des ordres, des informations disponibles sous forme textuelle ou pour interroger un utilisateur. Cette compréhension de la langue réside essentiellement dans le sens des mots. En effet, il existe un certain nombre de systèmes comme les chatbots [SSG+17] (agent conversationnel) qui se basent uniquement sur l'analyse syntaxique de grandes bases de données : on retrouve en entrée du système une séquence de mots correspondant à une question et en sortie la séquence de mots de la réponse. Ici, le sens des mots n'est pas réellement exploité. De plus, ils ne sont pas adaptés aux interactions humain/robot où les interrogations concerneront non pas de la connaissance générale (politique, film, ...) mais plutôt des objets concrets liés à l'environnement proche. Ainsi, à la requête *Donne moi mes médicaments*, le robot doit raisonner sur des instances [2] possiblement vues précédemment et non sur des lieux génériques issues de données externes (eg Internet). Il pourrait également inférer du sens de *médicament* que l'utilisateur est probablement malade.

2. définie comme une réalisation concrète d'un concept

Cette approche peut se justifier par le fait qu'il ne serait pas raisonnable d'implémenter manuellement l'ensemble des relations entre les concepts d'une langue ainsi que les règles grammaticales associées. On verra toutefois que dans le domaine de la robotique, nous nous intéressons principalement aux concepts liés à des objets physiques. Avec cette limitation, l'emploi d'un ensemble de règles simples devient envisageable.

– **Représentation du monde non unique :** Revenons sur le début de cette introduction avec la définition du mot "environnement". Imaginons un robot dans un laboratoire de recherche. Est-ce que son environnement se définit comme le plan du bâtiment (ensemble de points 2D) ? comme un ensemble de couloirs, de bureaux et de salles d'expérimentation ? De plus, selon la définition choisie, cette représentation peut varier temporellement : une salle d'expérimentation peut être ultérieurement réarrangée en bureau.

  Nous définissons ainsi trois axes caractérisant toute modélisation d'environnement

  - **Spatialité (local <–> global) :** La spatialité caractérise l'aspect local ou global d'un modèle : une image sera typiquement une représentation locale alors que l'intégration d'une carte par SLAM sera globale. Un autre exemple sémantique est de considérer le fait *Les taxis sont jaunes*. Ce fait n'est vrai que de façon locale à New-York.
  - **Temporalité (instantané <–> infini) :** La temporalité va déterminer la période de validité d'un modèle : la représentation par une image ne sera exacte qu'au moment de la prise, l'existence d'un objet (eg mes lunettes) ne sera valide que sur une période limitée de temps alors qu'un fait général (eg. *Les lunettes servent à corriger la vue*) sera intemporellement vrai.
  - **Abstraction (concret <–> abstrait) :** Enfin, on distingue les informations d'abstraction faible que sont les données sensorielles bas niveau comme par exemple une carte constituée de points 2D/3D. Au contraire, les concepts sémantiques exprimés en langage naturel sont abstraits.

1.2 Aperçu général de notre approche

Le système développé lors de cette thèse, illustré en figure 1.1, cherche à répondre de manière globale à chacune de ces problématiques. Les deux derniers points de la section précédente nous imposent d'avoir au moins deux modalités d'informations dans notre modèle : des données visuelles/géométriques (capteur RGBD) et des données sémantiques en langage naturel. Se faisant, nous remplissons également en partie le troisième point, avec une première modalité concrète et

l'autre fortement abstraite. Les données sensorielles ont une temporalité "nulle" alors que les données sémantiques considérées ici sont généralement intemporelles.

En résumé, nous avons une représentation **locale/globale/concrète** d'un côté et **globale/abstraite** de l'autre. On s'aperçoit donc qu'il nous faut un modèle **local/abstrait** afin de passer "continûment" d'une représentation à une autre. Celui-ci correspond à une description d'éléments concrets à partir de concepts abstraits. Autrement dit, ce modèle intermédiaire va décrire l'environnement courant [3] en terme d'**instances**. Celles-ci ont leur propriétés visuelles/géométriques données par des capteurs mais également une description en langage naturel réutilisant les concepts abstraits de la représentation sémantique.

Plus précisément, notre modèle vise à une représentation d'un environnement ouvert [4] et non défini en fonction de tâches précises. Ce modèle doit être une base sur laquelle on puisse ensuite greffer des modules spécifiques à l'exécution de tâches particulières. Il faut donc pouvoir être capable à la fois d'utiliser chacune des modalités de façon indépendante mais également de façon conjointe si nécessaire. L'apport d'information d'une modalité vers une autre doit également pouvoir se faire dans les deux sens. On a vu précédemment que majoritairement, l'information allait de la sémantique vers la vision. Certains travaux, au contraire, transportaient l'information de la vision vers la sémantique. Nous pensons que l'intégration de ces différentes méthodes dans un cadre commun permettra d'obtenir un modèle plus générique.

Bien sûr, une plus grande généricité implique de minimiser le nombre d'hypothèses faites. Cela limite fortement les méthodes applicables, notamment l'usage d'apprentissage automatique (profond ou non). En effet l'apprentissage automatique atteint vite ses limites pour les applications sans cadre bien défini. Ces méthodes sont le plus souvent évaluées sur des données en monde fermé (nombre limité de classes). Que se passe-t-il lorsqu'une classe inconnue est vue ? C'est le problème du zero-shot learning (cf section 2.3.1). Ces approches peuvent paraître appropriées au premier abord, mais souffrent d'un gros défaut : elles supposent que les classes vues lors de la phase de test n'ont pas été vues lors de l'entraînement. Une hypothèse difficilement atteignable en pratique ! Deux autres problèmes se posent ensuite : l'ajout continu de nouvelles classes et la mise à l'échelle. L'apprentissage profond est toutefois un outil intéressant via le *Transfert learning* : apprendre sur un grand ensemble de données afin d'obtenir une représentation générique exploitable sur des domaines différents avec peu de données disponibles.

Enfin, l'accent est mis sur la structure générale de notre modèle et non sur les différents processus secondaires nécessaires en pratique. Par exemple, considérons la segmentation de scène non

3. Plus exactement, l'environnement connu du robot
4. Dans le sens d'univers non borné, pas d'environnement extérieur

supervisée. Ce problème, difficile, est toujours un sujet actif de recherche. Nous proposons l'utilisation de certaines méthodes, mais elles peuvent être remplacées avec l'évolution de l'état de l'art sans affecter le modèle dans sa globalité.

Notre architecture est composée de trois unités : Perception, Instance et Connaissance, représentées à la figure 1.1.

En commençant par le bas de ce schéma, nous avons les entrées en langage naturel de haut niveau qui peuvent provenir d'un utilisateur ou de sources extérieures comme Internet. Elles peuvent se décomposer en trois principaux types : connaissances générales, informations sur des instances, ou requêtes (tâches). Ceci sera détaillé dans le chapitre 5. Cette entrée est naturellement liée au modèle de connaissance. A l'opposé (haut de la figure), il y a les entrées capteurs du robot avec par exemple ici un LIDAR et une caméra RGBD. Ces données sont directement exploitées par le modèle de perception. Le modèle d'instance va lui faire le pont entre ces deux modalités.

L'unité de connaissance, sujet du chapitre 3, est une représentation sémantique générale et intemporelle de l'environnement sous forme de graphe ontologique. Celle-ci est initialement construite à partir de sources extérieures provenant d'Internet. Elle est basée sur l'ontologie WordNet [Mil95] et enrichie de relations extraites de ses définitions. Elle peut être mise à jour automatiquement par l'analyse de données textuelles tirées d'Internet ou par des informations données par l'utilisateur. Cette unité permet au robot de raisonner sur les concepts à travers la notion de **contexte**. Cela peut être la simple extension aux concepts parents/fils lorsque l'utilisateur emploie un terme mais en sous-entendant une spécialisation de celui-ci : *Give me a knife* signifie généralement *Give me a table knife* lors d'un repas. Le contexte sert également à désambiguïser des termes ayant plusieurs sens [5] eg. *fork* (fourchette) et *fork* (fourche). Enfin, il intervient dans l'inférence d'objets dans une scène donnée. En effet, ce type d'inférence est typiquement basée sur la cooccurrence d'objets. Hors, celle-ci n'est qu'une conséquence de l'existence d'un contexte commun plus ou moins explicite entre les entités composant une scène.

L'unité de perception est abordée au chapitre 4. C'est une représentation basée capteurs [6] de l'environnement à l'instant courant. Elle traite les données capteurs bas niveau afin de structurer l'environnement en instances, que ce soit de manière passive (segmentation de scène non supervisée, suivi d'une instance connue) ou active (détection d'un objet précis).

L'unité d'instance représente l'ensemble des instances de l'environnement dont on connaît l'existence sous forme de graphe. C'est donc une représentation spécifique à l'environnement connu du robot sur une période de temps limitée. Une instance est définie selon chacune des modalités :

5. homographe
6. Dans notre cas, cela se limite à l'aspect visuelle et géométrique.

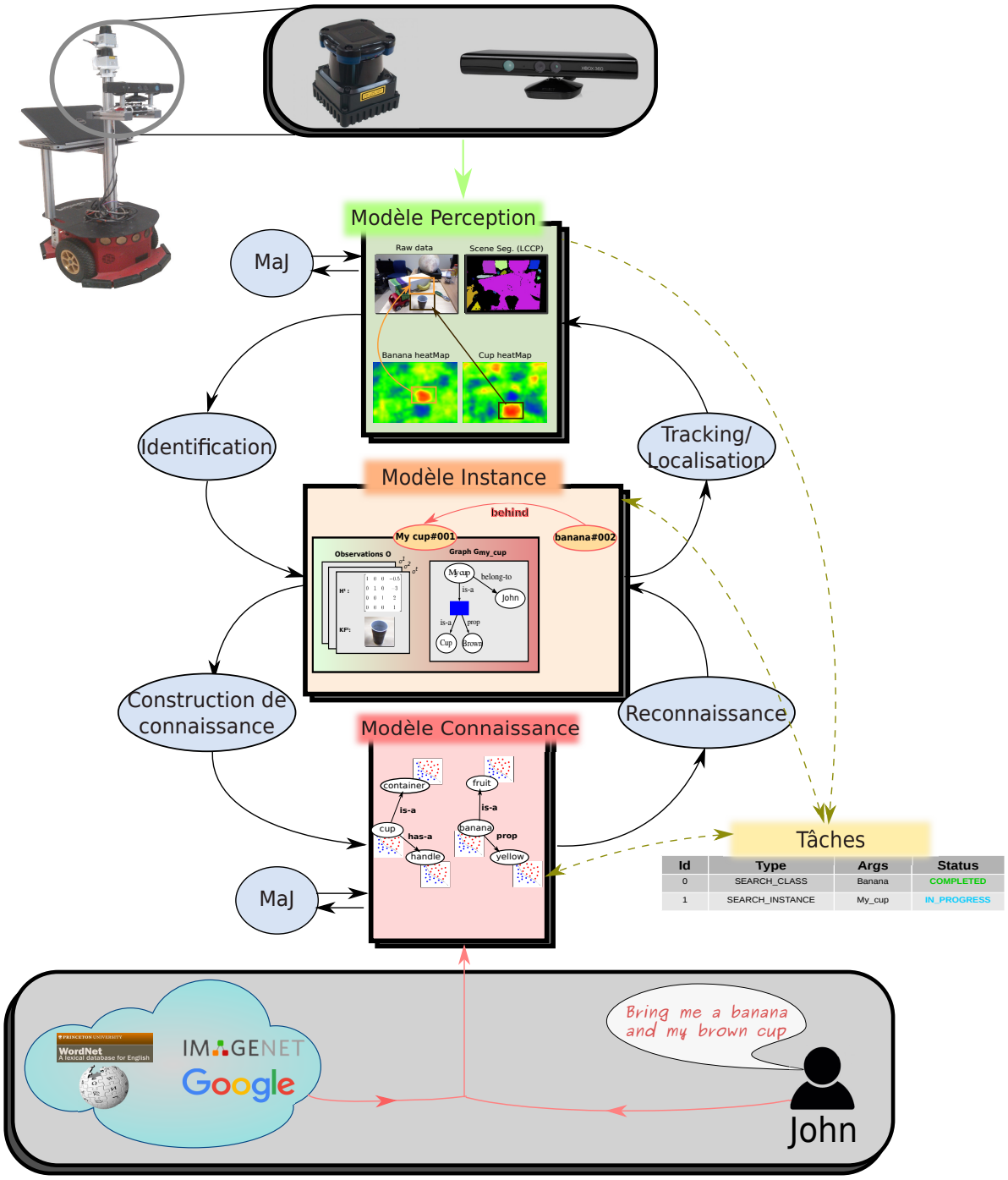


FIGURE 1.1 – Structure générale de notre modèle global de représentation d'environnement

une représentation géométrique (nuage de points 3D) / visuelle (vecteur de description image) et une représentation sémantique décrivant l'objet à partir des concepts du modèle de connaissance. Les instances peuvent être initialement créées à partir d'une seule modalité : une détection du mo-

dèle de perception générera une instance avec une représentation sémantique vide. *A contrario*, les informations données en langage naturel comme *Ma tasse est rouge* généreront une instance définie sémantiquement mais sans représentation visuelle. Elle gère l'ajout/retrait/fusion/mise à jour d'instance que ce soit par le biais du modèle de perception ou par des informations externes en langage naturel.

Enfin, on peut voir dans le schéma la présence d'un gestionnaire de tâches. La création de tâche est faite à partir de requête utilisateur eg *Give me my cup* et le gestionnaire vérifie de manière asynchrone la complétion des tâches à mesure que le système évolue. Ceci sera abordé plus en détail dans la section 5.5. Les principaux processus reliant les unités et entrées entre elles ont également été notés sur la figure 1.1. Les passages du modèle d'instance au modèle de connaissance seront abordés dans le chapitre 5.

La suite de ce manuscrit est structurée de la façon suivante :

- **Le chapitre 2** traite des différentes représentations d'environnements proposées dans la littérature robotique.
- **Le chapitre 3** est consacrée à la définition et à la méthode de construction de notre modèle de connaissance sémantique.
- **Le chapitre 4** décrit la modalité visuelle de notre système. En particulier, elle se concentre sur l'analyse de propriétés des descripteurs image issus de couches intermédiaires de réseaux de neurones convolutifs.
- **Le chapitre 5** est consacré aux processus entre les différentes unités du modèle : passage des données perceptuelles à la notion d'instance (section 5.4), des données sémantiques (modèle de connaissance et entrées utilisateurs) aux instances (section 5.2) et enfin le rattachement des instances au modèle de connaissance (section 5.3). La dernière section 5.5 détaille la représentation des tâches induites par les requêtes utilisateurs.
- **Le chapitre 6** présente une validation expérimentale de l'implémentation de notre modèle.
- **Le chapitre 7** est la conclusion de cette thèse.

1.3 Contributions

Ce manuscrit présente un système de représentation d'environnement multimodale focalisé principalement sur la notion d'instance. C'est une architecture globale à 3 couches reliant les données perceptuelles aux connaissances sémantiques au travers d'une représentation intermédiaire basé instances. Nous insistons sur le caractère générique de l'approche en posant l'hypothèse d'un

monde ouvert, c'est à dire avec un nombre de classe non défini, et indépendant des applications ou tâches qui seront effectuées par le robot.

En particulier, l'attention est focalisée sur la notion d'instance et la description sémantique d'objets en créant notre propre ontologie. Elle est générée automatiquement par une méthode originale d'analyse syntaxique des définitions d'un dictionnaire. Cette méthode d'analyse permet également l'analyse d'autres sources textuelles et notamment les informations ou requêtes utilisateurs. Ces travaux ont fait l'objet d'une publication [BDZ18].

Il a été montré dans la littérature récente la bonne généralisation des descripteurs images issus de couches intermédiaires de CNNs. Ce résultat est confirmé et affiné dans le cadre de la représentation d'instance. Nous remarquons également que ces descripteurs varient de façon remarquablement régulières lorsque l'image d'entrée est soumise à une transformation de similarité (translation, rotation et mise à l'échelle). Cette propriété sera exploitée afin d'inférer les paramètres de la transformation.

Dans l'objectif d'obtenir une justification théorique de la propriété précédente, une formalisation en écriture matricielle sera proposée. Quelques nouvelles propriétés sur l'opérateur de dérivation Kronecker, servant à exprimer sous forme matricielle les gradients de fonctions multivariées et multidimensionnelles, seront démontrées.

Nous proposons ensuite des mécanismes permettant de relier les différentes modalités du modèle, avec notamment une méthode originale permettant de rattacher des instances de classes inconnues à notre modèle de connaissance en se basant sur la structure de notre ontologie. On discute enfin de la gestion des requêtes utilisateur en langage naturel : transcription en termes de "tâches" internes et leur résolution. On se limite ici à des requêtes basées sur la recherche d'objet, le but étant d'illustrer l'utilisation de notre modèle.

Chapitre 2

Etat de l'art

Dans ce chapitre, nous allons nous concentrer sur les différentes modalités de représentation d'un environnement par un robot en nous appuyant sur la grille d'analyse proposée en introduction. Les différentes méthodes sont séparées selon le type de représentation : **géométrique**, **objets** et **sémantique** tout en gardant à l'esprit les trois principaux axes caractéristiques de toute représentation (Spatialité, Temporalité, Abstraction).

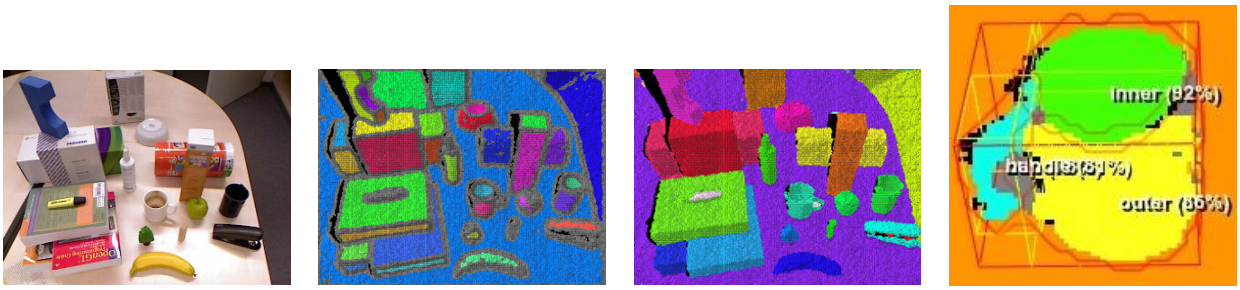
Tout d'abord les représentations géométriques et objets principalement utilisées en robotique à partir de traitement des données capteurs (principalement LIDAR, caméra monoculaire/Stéréo et RGBD) sont passées en revue. Ensuite, nous étudierons des représentations plus abstraites basées sur la sémantique. Enfin, nous regarderons les approches mêlant ces différentes modalités. Notons que nous nous intéressons principalement à des méthodes non supervisées avec aucune ou peu d'information *a priori* sur l'environnement du robot. En conclusion, une comparaison de l'état de l'art abordé selon les critères de spatialité, temporalité et d'abstraction est proposée.

2.1 Représentation géométrique de l'environnement

2.1.1 Segmentation de scène statique en objets

La segmentation de scène constitue une première famille de méthodes locales, à temporalité nulle et à faible degré d'abstraction, c'est à dire proche des données capteurs brutes. Un exemple de méthode est celle d'Uckermann *et al.* [ÜHR13] qui propose de segmenter une scène en se basant sur un capteur RGBD (Kinect). Elle repose sur les deux étapes suivantes : une présegmentation de la scène en surfaces lisses puis un regroupement des surfaces correspondant à un même objet. La présegmentation est effectuée par un calcul de normale en chaque point puis par une détection des

bordures à partir du produit scalaire de ces normales. Une analyse en composants connectés est ensuite appliquée sur l'image binaire précédemment obtenue pour avoir l'ensemble des surfaces lisses de l'image. Pour le partitionnement des surfaces en objets, un graphe pondéré modélisant la probabilité qu'ont deux surfaces d'appartenir à un même objet est construit. Les poids associés à chaque lien de ce graphe sont calculés selon différentes heuristiques telles que la coplanarité des surfaces. La segmentation optimale est finalement obtenue par une coupe du graphe selon les liens dont les poids sont inférieurs à un certain seuil. Dans un second papier du même auteur [ÜEHR14], cette méthode est étendue en générant une hiérarchie de segmentation. Pour cela, ils séparent itérativement le graphe non plus en coupant les liens selon un seuil fixé, mais avec une méthode de flot maximum/coupe minimum. Ils obtiennent alors un arbre binaire, avec chaque nœud correspondant à une segmentation à une certaine échelle. Par exemple, la figure 2.1d montre la segmentation d'une tasse en trois parties (extérieur, intérieur et anse).



(a) Image RGB original (b) Présegmentation (c) Segmentation finale (d) Segmentation en parties

FIGURE 2.1 – (a)(b)(c) Segmentation de scène basée sur l'hypothèse de surfaces lisses [ÜHR13] et (d) segmentation en parties [ÜEHR14]

D'autres approches de segmentation de scène non supervisées se basent sur une représentation intermédiaire d'une image se situant entre le pixel et la segmentation en objets : les superpixels [ASS+12] [WGB12] [FVS09] [VBM10] (supervoxels [PASW13] [SSPW14] [ABS14] [SPW15] dans le cas 3D). L'objectif est de remplacer la segmentation par pixel, qui ne tient pas compte du contexte global et qui est coûteuse à inférer, en une segmentation d'ensemble de pixels regroupés selon certains critères de similarité : couleur, position, normale à la surface tangente. Une des méthodes classiques pour effectuer la présegmentation d'une image en superpixels est nommée SLIC [ASS+12] (Simple Linear Iterative Clustering). Chaque pixel est représenté par un vecteur de dimension 5 (2 coordonnées spatiales et 3 coordonnées colorimétriques dans l'espace couleur Lab). La segmentation est initialisée en créant des centres de régions réparties uniformément sur l'image puis en appliquant un algorithme de K-means local, c'est à dire qu'un pixel ne peut appartenir à une région que s'il se trouve dans un rayon fixe autour de son centre. Cette région de

recherche est représentée à la figure 2.2a pour un pixel (point rouge) : seuls les centres (point vert) inclus dans cette région sont considérés.

Une adaptation de cette méthode aux capteurs RGB-D a été proposée par Weikersdorfer *et al.* [WGB12]. Ils exploitent l'information de profondeur afin d'avoir des superpixels de taille proportionnelle à la surface réelle correspondante, ceci afin d'obtenir une segmentation uniforme de la scène 3D observée. La segmentation de la scène est enfin obtenue après application d'une méthode de partitionnement spectral (Spectral Clustering) [SM00] sur le graphe dont les nœuds sont les superpixels.

Les supervoxels sont l'équivalent 3D des superpixels. Papon *et al.* [PASW13] ont proposé une méthode de sursegmentation d'un nuage de points 3D en supervoxels appelée VCCS (*Voxel Cloud Connectivity Segmentation*). La méthode est similaire au cas 2D : l'espace 3D est discrétisé puis un graphe de voisinage aux 26-ppv [1] est construit (deux voxels sont voisins s'ils partagent une face, une arête ou un sommet). Chaque point p est représenté par un vecteur $\mathbf{F}$ comprenant sa position, sa couleur dans l'espace Lab, mais également un vecteur $\mathbf{f}$ de description 3D FPFH (*Fast Point Feature Histograms*) [RBB09] décrivant les propriétés (invariantes selon la pose) du modèle de la surface locale au point :

$$\mathbf{F} = \begin{bmatrix} x & y & z & L & a & b & \mathbf{f}^T \end{bmatrix}^T \tag{2.1}$$

Asif et al [ABS14] proposent une approche similaire mais avec une définition différente du vecteur F de description des points 3D :

$$\mathbf{F} = \begin{bmatrix} x & y & z & \nabla & \alpha \end{bmatrix} \tag{2.2}$$

avec ∇ le gradient local moyen (moyenne des projections entre la normale au point et les normales alentours) et α l'orientation de la surface par rapport au point de vue courant. De la même façon que [WGB12] améliora SLIC en tenant compte de la densité décroissante des superpixels en fonction de la distance, Stein *et al.* [SSPW14] proposent une extension de VCCS dite DDVG (*Depth Dependent Voxel Grid*) en utilisant un octree dont la taille des nœuds augmente en fonction de la distance.

Une fois cette présegmentation en supervoxels effectuée, il reste à les regrouper en entités de plus haut niveau que sont les "objets". Les expériences effectuées par Worgotter [WST15] et Tamosiunaite [TSSW15] montrent que les humains définissent principalement le concept d'objet à partir de la convexité. Ainsi, les segmentations basées uniquement sur la couleur ne sont généralement pas capables de décomposer une scène selon des objets réels ou ayant un sens physique.

Les approches de Asif [ABS14] et Stein [SSPW14] utilisent l'information sur la convexité lo-

1. plus proches voisins

cale des surfaces. Le premier regroupe simplement les régions adjacentes convexes avec un traitement spécial pour gérer les ouvertures du type intérieur/extérieur d'une tasse. Le second développe une méthode dite LCCP (*Locally Convex Connected Patches*) qui se base sur un processus de région croissante : un supervoxel est choisi aléatoirement puis labellisé. Ce label se propage dans le graphe de voisinage des supervoxels selon les arêtes convexes [2]. Lorsque la propagation se termine, l'opération est répétée en sélectionnant aléatoirement un autre supervoxel ne possédant aucun label. L'avantage du processus de région croissante proposé par [SSPW14] par rapport à [ABS14] est d'éviter les segmentations erronées le long de concavités isolées.

Une extension de LCCP a été proposée par les mêmes auteurs [SPW15] dans le but de segmenter non pas une scène en objets mais un objet en parties. Cette méthode est dénommée CPC-LCCP (CPC pour *Constrained Planer Cut*). En effet, la méthode LCCP permet de regrouper des régions malgré des concavités isolées. Cela est une bonne propriété pour la séparation d'objets distincts mais désavantageux pour une segmentation en parties d'un objet comme illustré dans la figure 2.2c où l'on voit que l'épaule ne permet pas de séparer totalement le bras du torse. En effet, le dessus de l'épaule n'est pas concave. CPC-LCCP suit exactement la méthode LCCP à la différence que le graphe de voisinage peut être dorénavant coupé selon des arêtes convexes. Cette méthode est par nature sensible aux concavités des objets déformables, comme par exemple les plis d'un vêtement, qui causent de la sursegmentation.

Cette méthode peut être une bonne initialisation dans le cadre de modélisation cinématique d'objets articulés. C'est ce que nous allons voir dans la section suivante.

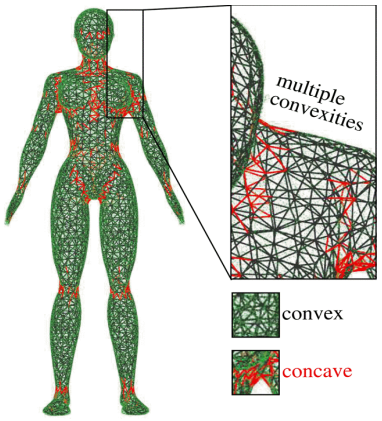
2.1.2 Segmentation basée mouvement et modélisation cinématique

Dans la section précédente, nous avons présenté quelques méthodes récentes de segmentation de scène non supervisées en régions correspondant à d'hypothétiques objets. En se référant à l'analyse faite dans le chapitre d'introduction, ces méthodes ont un niveau d'abstraction très faible. Elles peuvent seulement indiquer la présence d'objets ainsi que certaines de leur propriétés physiques (apparence visuelle, géométrie), ne donnant ainsi qu'une représentation superficielle de l'environnement proche du robot. Jusqu'à présent, les méthodes étudiées se basaient sur des scènes fixes. Nous allons voir maintenant des méthodes qui reposent sur l'analyse des mouvements (trajectoires) dans une scène afin de la segmenter en objets indépendants mais également de décomposer les objets articulés selon leur parties. Ces méthodes permettent de monter d'un cran le niveau d'abs-

2. Chaque arête du graphe a été préalablement classifié comme convexe ou concave selon les angles entre les normales aux surfaces des sommets et le segment joignant les sommets.

(a) Segmentation en Superpixel (SLIC). Modifié à partir de [ASS+12]

(b) Segmentation d'une scène par LCCP. Extrait de [SSPW14]



(c) Limite de LCCP pour la segmentation en partie avec concavité isolée. Extrait de [SPW15]

FIGURE 2.2 – Segmentation de scène basée sur une présegmentation en superpixel/supervoxel

traction sur la connaissance d'un objet : elles permettent en effet d'inférer des liaisons mécaniques (e.g. liaison rotule, pivot) et des chaînes cinématiques. Ce type de connaissance est primordiale pour un robot évoluant dans un milieu "humain" : pour ouvrir une porte, il faut prendre en compte qu'elle est en liaison pivot avec un mur.

Zappella *et al* [ZLS09] proposent une revue complète des méthodes de segmentation mouvement antérieures à 2009, en les classifiant par type et propriété (figure 2.3). Nous allons dans la suite introduire les principales approches à ce problème.

Méthode par factorisation

Une des méthodes pionnières dans la segmentation basée sur l'analyse de trajectoires est la méthode par factorisation [TK92] [KVD91] [CK98] [BHB00] [XCK06] [MC09]. Elle trouve son origine dans le travail de Tomasi *et al.*. [TK92] et est plus souvent connue sous le nom de SfM (Structure from Motion). En supposant un modèle de caméra orthographique et un objet statique [3], ils montrent que l'on peut factoriser la matrice $W \in \mathcal{M}_{2F,N}(\mathbb{R})$, composée des trajectoires de N points 2D $\mathbf{x}_{n,t} = \begin{bmatrix} u_{n,t} & v_{n,t} \end{bmatrix}^T$ appartenant à l'objet suivi sur F images, en un produit de deux

3. De manière équivalente, on peut considérer un objet en mouvement et une caméra statique

Image Diff.	(Cavallaro et al, 2005)	F/D	✓	✓	✓		✓	✓	
	(Cheng and Chen, 2006)	D	✓	✓		✓	✓	✓	X
	(Li et al, Aug. 2007)	D		✓	✓		✓	✓	
	(Colombari et al, 2007)	D	✓	✓	✓	✓	✓	✓	
Statistical MAP	(Rasmussen and Hager, 2001)	D	✓	✓	✓		✓	✓	X
	(Cremers and Soatto, 2005)	D	✓	✓	✓		✓	RA	X
	(Shen et al, 2007)	D	✓	✓	✓	✓	✓	✓	CX
PF	(Vaswani et al, 2007)	D	✓	✓	✓		✓	RN	X
EM	(Stolkin et al, 2008)	D	✓	✓	✓	✓	✓	R	CX
Wavelets	(Wiskott, 1997)	F		✓			✓	R	
	(Kong et al, 1998)	F	✓	✓		✓	✓	R	X
O.F.	(Zhang et al, 2007)	F		✓			I	RA	C
	(Xu et al, 2008)	D	✓	✓	✓		I	✓	
	(Klappstein et al, 2009)	F	✓	✓			I	RA	X
	(Bugeau and Pérez, 2009)	F/D	✓	✓		✓	I	R	
	(Ommer et al, 2009)	F	✓	✓			I	R	
Layers	(Kumar et al, 2008)	F	✓	✓	✓	✓	✓	RA	X
	(Min and Medioni, 2008)	D	✓	✓	✓	✓	✓	✓	X
Manifold Clustering Iter	(Fischler and Bolles, 1981)	F			✓	✓	I	RA	C
	(Ho et al, 2003)	F			✓		✓	✓	CD
	(da Silva and Costeira, 2008)	F			✓		✓	✓	X
Stat	(Kanatani and Matsunaga, 2002)	F			✓		I	R	
	(Sugaya and Kanatani, 2004)	F			✓		I	R	C
	(Gruber and Weiss, 2004b)	F			✓	✓	I	R	X
	(Gruber and Weiss, 2006)	F	✓	✓	✓	✓	I	R	X
ALC	(Rao et al, 2008)	F	✓		✓	✓	I	R	
Fact	(Costeira and Kanade, 1998)	F			✓		I	R	
	(Ichimura and Tomita, 2000)	F			✓		I	R	
	(Zelnik-Manor and Irani, 2003)	F			✓		✓	RA	CD
	(Zhou and Huang, 2003)	F	✓		✓	✓	✓	RA	CD
Subspaces	(Vidal and Hartley, 2004)	F	✓		✓		✓	R	C
	(Yan and Pollefeys, 2006, 2008)	F			✓		✓	✓	CDX
	(Julia et al, 2008)	F	✓		✓		✓	R	C
	(Goh and Vidal, 2007)	F			✓		✓	R	CD
	(Vidal et al, 2008)	F	✓		✓		✓	R	C
	(Goh and Vidal, 2008)	F			✓		✓	R	CD
	(Chen and Lerman, 2009)	F			✓		✓	✓	CD
	(Zappella et al, 2009)	F			✓	✓	✓	✓	C
Features (F) / Dense (D)									
Occlusion or Missing Data									
Spatial Continuity									
Temporary Stopping									
Robustness (Noise, Outliers, Initialization)									
Dependency (I independent, D dependent, ✓all)									
Kind (R rigid, N non-rigid, A articulated, ✓all)									
Prior knowledge (C Clusters number, D Subspace dimension, X Other, T Training)									

FIGURE 2.3 – Classification de méthodes de segmentation basées mouvement. Extrait de [ZLS09]

matrices de la façon suivante :

$$W = \begin{bmatrix} u_{11} & \dots & u_{1N} \\ \vdots & & \vdots \\ u_{F1} & \dots & u_{FN} \\ v_{11} & \dots & v_{1N} \\ \vdots & & \vdots \\ v_{F1} & \dots & v_{FN} \end{bmatrix} = \begin{bmatrix} \mathbf{i}_1^T & t_{x_1} \\ \vdots & \\ \mathbf{i}_F^T & t_{x_F} \\ \mathbf{j}_1^T & t_{y_1} \\ \vdots & \\ \mathbf{j}_F^T & t_{y_F} \end{bmatrix} \begin{bmatrix} \mathbf{s}_1 & \dots & \mathbf{s}_N \end{bmatrix} = MS \tag{2.3}$$

$i_f,\ j_f$ représentent les axes du repère associé à la caméra à la f-ème image et t_x, t_y la position du centre du repère. La matrice de mouvement $M \in \mathcal{M}_{2F,4}(\mathbb{R})$, ayant chacune de ces lignes i et $F+i$ correspondantes à la position de la caméra à la i-ème image, représente le mouvement de la caméra. Les colonnes de la matrice de forme $S \in \mathcal{M}_{4,N}(\mathbb{R})$ correspondent aux positions 3D homogènes des points suivis.

La propriété remarquable est que, quelque soient l'objet et le mouvement de la caméra, on a toujours $r = Rang(W) \leq 4$ d'après l'équation (2.3). Cette propriété sur le rang sera exploitée par la suite dans l'inférence des liaisons mécaniques. Costeira *et al.* [CK98] étendent la méthode de factorisation dans le cas de plusieurs objets ayant des mouvements indépendants. Bregler *et al.* [BHB00] et Xia *et al.* [XCK06] proposent une méthode de factorisation pour les objets non rigides en utilisant non pas une matrice de forme mais une base de matrices de forme composée de K formes clés. La matrice de forme S_f devient donc ici une variable du temps et est définie par une combinaison linéaire de formes clés B_i :

$$S_f = \sum_{i=1}^{K} c_{fi} B_i,\ B_i \in \mathcal{M}_{3,N}(\mathbb{R}) \tag{2.4}$$

On a alors la décomposition de la matrices des trajectoires W, en posant $R_f^T = [\mathbf{i}_f, \mathbf{j}_f]$

$$W = \begin{bmatrix} c_{11}R_1 & \dots & c_{1K}R_1 & \mathbf{t}_1 \\ \vdots & \vdots & \vdots & \vdots \\ c_{F1}R_F & \dots & c_{FK}R_F & \mathbf{t}_F \end{bmatrix} \begin{bmatrix} B_1 \\ \vdots \\ B_K \\ \mathbf{1}_{N,1}^T \end{bmatrix} \tag{2.5}$$

On déduit de l'équation (2.5) que pour les objets non-rigides on a $r \leq 3K + 1$.

Ces approches permettent donc, à partir des trajectoires 2D, d'estimer la forme d'objets. Le problème est qu'elles requièrent des matrices pleines c'est à dire qu'il faut connaître la position des

points à chaque instant. Or, en pratique, il est difficile de suivre un point précis sur un intervalle de temps relativement long sans erreur en raison des problèmes d'occlusions ou des erreurs du flot optique (changement d'illumination, etc ...). Marques *et al.* [MC09] proposent une variation permettant de gérer les cas où les trajectoires ne sont pas complètes dans le cas d'un seul objet. L'extension de Costeira [CK98] permet de segmenter une scène en différents objets, mais elle ne prend pas en compte les objets articulés et non-rigides (mouvement dépendant). Au contraire, les méthodes applicables aux objets non-rigides se limitent à un seul objet.

Estimation de sous-espaces vectoriels

Une extension naturelle de la méthode par factorisation est l'estimation de sous-espaces vectoriels. Yan et Pollefeys [YP08] ont montré que les trajectoires d'objets appartiennent à un ensemble d'intersections de sous-espaces vectoriels dont les dimensions dépendent de leur rigidité et de leur liaisons mécaniques.

Les contraintes cinématiques peuvent être estimées à partir du rang r de la matrice des trajectoires W [YP05] définie à l'équation 2.3. Dans le cas d'un seul objet rigide non dégénéré, l'équation 2.3 nous a donné $r = 4$. On aura $r = 3$ (resp. $r = 2$) dans les cas dégénérés d'objets planaires (resp. linéaires). De même, on a obtenu $r = 3K + 1$ dans le cas général d'un objet non rigide représenté par K formes de base.

Les auteurs montrent que les sous-espaces de mouvement de deux objets liés par une liaison rotule (resp. liaison pivot) ont une intersection de dimension 1 (resp. 2). On a donc pour la matrice des trajectoires des deux objets $W = [W1|W2]$ avec $rang(W) = 8 - 1 = 7$ dans le premier cas et $rang(W) = 8 - 2 = 6$ dans le second. On peut en déduire une formule générale dans le cas d'un objet articulé composé de N_r (resp. N_{nr}) parties rigides (resp. non rigides), de N_{rot} liaisons rotule et de N_{piv} liaisons pivot :

$$rang(W) = 4N_r + \sum_{i=1}^{N_{nr}} 3k_i + 1 - 2N_{piv} - N_{rot} \tag{2.6}$$

Une autre approche pour l'estimation de liaisons cinématiques est proposée par Jacquet *et al.* [JAP13]. Ils définissent un couple d'entier (signature) qui capture la structure de la liaison entre deux objets. Ici, ce sont les mouvements relatifs entre objets et non une matrice de trajectoire qui sont exploités. Les données d'entrées, à chaque instant t, sont donc des éléments du groupe **SE(3)** représentés par

$$H_t = \begin{bmatrix} R_t & \boldsymbol{t}_t \\ \mathbf{0}^T & 1 \end{bmatrix} \tag{2.7}$$

Afin de simplifier leur analyse, ils projettent ces mouvements dans un espace vectoriel de dimension supérieure en considérant la matrice M suivante :

$$M = \begin{bmatrix} vec(R_1 - I_3)^T & \boldsymbol{t}_1^T \\ \vdots & \vdots \\ vec(R_\tau - I_3)^T & \boldsymbol{t}_\tau^T \end{bmatrix} = [M_{:,1:9}|M_{:,10:12}] \in R^{\tau \times 12} \tag{2.8}$$

où $vec : \mathcal{M}_{m,n}(\mathbb{R}) \mapsto \mathbb{R}^{mn}$ empile les colonnes d'une matrice les une sur les autres afin de convertir une matrice en un vecteur colonne. La signature d'un mouvement relatif entre deux objets est définie par :

$$sig(M) = \{r = rang(M_{:,1:9}),\ d = rang(M) - rang(M_{:,1:9})\} \tag{2.9}$$

$r \in \{0, 2, 8, 9\}$ détermine le nombre d'axes de rotation fixe (0, 1, 2 ou 3) et $d \in \{0, 1, 2, 3\}$ est la dimension du sous-espace engendré par la composante de translation. On a donc par exemple une signature (2,2) pour une liaison plan-plan et (0,1) pour une liaison glissière.

Les méthodes d'estimation de liaisons mécaniques ci-dessus supposent une présegmentation des données par objet. Une autre famille de méthode, dite *Articuled Structure From Motion* (ASfM) [RTZ10] [FRA11] [JCD15], vise à estimer simultanément les types de liaisons ainsi que la chaîne cinématique.

Ces méthodes sont intéressantes dans le cadre de la robotique de manipulation. Il est en effet nécessaire pour un robot de connaître le type d'articulation d'un objet tel qu'un ciseau afin de pouvoir l'utiliser. Cependant, elles ne s'appliquent que sur un seul objet et nécessitent donc une présegmentation de la scène. De plus, elle partagent le même défaut que les méthodes SfM, à savoir le besoin de trajectoires complètes. Un autre problème pratique est qu'à cause du bruit dans les données et des erreurs numériques, le rang r de la matrice des trajectoires W sera supérieur à sa valeur théorique. Il faut donc employer des méthodes d'estimation de rang.

Modèle de scène articulée

Swadzba [SBWK10] et Ziegler [Zie15] proposent une approche différente appelée ASM (Articulated Scene Model). Le premier analyse une scène avec un capteur 3D ToF (Time of Flight) statique en la séparant en trois couches qui évoluent dynamiquement (d'où l'indice temporel t) :

- une couche S_t correspondant aux objets statiques (arrière plan statique)
- une couche D_t correspondant aux objets en mouvement durant la période d'observation

– une dernière couche O_t composée des objets déplaçables c'est-à-dire des objets dont on n'a pas observé directement le déplacement mais qui ont changé de position entre deux observations

L'union $D_t \cup O_t$ correspond donc à l'avant-plan de la scène. La séparation en trois couches du nuage de point 3D donné par le capteur est effectuée de la manière suivante : tout d'abord, le vecteur vitesse 3D est calculé en chaque point en se basant sur le flot optique 2D à partir de l'image RGB et de l'image de profondeur. Leur objectif est donc de regrouper les potentiels points dynamiques courants (ie n'appartenant pas à S_{t-1}) en objets qui constitueront la couche D_t. Une méthode de regroupement hiérarchique (*complete-linkage clustering*) est employée en se basant sur une fonction de distance dépendant de la vitesse et de la position spatiale des regroupements. Une sursegmentation de la scène est ainsi obtenue et les ensembles de vitesse similaire sont regroupés selon des modèles cylindriques. Chaque objet est ensuite suivi par un filtre particulaire à noyau gaussien. Ziegler *et al.* [Zie15] améliorent l'ASM en considérant des acquisitions selon différents points de vue, ce qui permet son intégration avec un système de localisation type SLAM.

Contrairement aux méthodes précédentes, ASM ne nécessite pas des trajectoires complètes de même durée pour chaque point de la scène : elle se base sur le vecteur vitesse 3D des points calculé d'image en image. Par contre, la méthode n'est pas adaptée à des objets articulés ou non rigides, dont une partie peut rester immobile tandis que les autres sont en mouvement.

2.1.3 SLAM

Les approches vues précédemment sont essentiellement locales. Les techniques de SLAM (Simultaneous Localization and Mapping) [TBF05] [FM12] permettent à un robot de générer une carte globale de son environnement tout en s'y localisant. La carte de l'environnement peut être de différentes natures en fonction des capteurs disponibles comme par exemple des cartes d'occupation 2D ou des nuages de points 3D (figure 2.4). Plus d'information et de détails se trouvent dans les ouvrages [TBF05] [FM12]. Cependant, nous retrouverons le SLAM dans la section 2.3 où l'on verra des approches cherchant à exploiter et à inférer des informations sémantiques de plus haut niveau sur la carte de l'environnement.

Cette section a présenté des approches non supervisées de représentation de l'environnement physique du robot, proche des données brutes capteur. L'inférence de contraintes mécaniques apporte une compréhension plus fine de l'environnement, ce qui peut permettre au robot de s'adapter

(a) Exemple de carte d'occupation 2D et de trajectoire obtenue par une approche SLAM. Extrait de [BSB+15]

(b) Exemple de carte par nuage de points 3D. Extrait de [MAT17]

FIGURE 2.4 – Illustration de cartographie par SLAM

à des situations préalablement inconnues. Ceci est primordial dans le cadre de la robotique domestique, où un robot doit pouvoir ouvrir des portes ou des tiroirs. Cependant, dans le cadre d'un robot évoluant en interaction avec l'homme, ces informations sont nécessaires mais non suffisantes : il faut pouvoir faire le lien entre l'environnement observé et sa description en langage naturel par l'homme.

2.2 Représentation basée sémantique

Jusqu'à présent, nous nous sommes intéressés aux représentations physiques de l'environnement à partir de données capteurs communes dans le domaine de la robotique et de la vision par ordinateur. Une autre façon plus abstraite de se représenter l'environnement, que nous utilisons tous plus ou moins consciemment, vient de notre langage et de sa structure : c'est une représentation sémantique. Cette représentation est principalement exploitée par le web sémantique [MMM+04] et en traitement du langage naturel. Depuis quelques années, elle est également utilisée de multiples façons en robotique, comme nous le verrons dans la section 2.3. Nous nous concentrerons principalement sur les concepts et aspects pertinents dans le cadre de cette thèse. Le chapitre 3 décrira en détails le modèle de connaissance sémantique de notre architecture.

La représentation formelle de relations sémantiques peut se faire à l'aide de la *logique du premier ordre*. Elle permet de définir des *formules* à partir des éléments suivants :

– *Constantes* représentant des symboles dans un domaine d'intérêt $\mathcal{D}$. Exemple : $Tasse \in \mathcal{D}$

Symbole	Sens
$\neg$	Négation
$\forall$	Universelle ("Pour tout")
$\exists$	Existence
$\vee$	"ou" logique
$\wedge$	"et" logique
$\Rightarrow, \Leftarrow$	Implication
$\Leftrightarrow$	Équivalence

TABLE 2.1 – Quantificateurs et connecteurs de la logique du premier ordre

- *Variables* à valeurs dans le domaine considéré $\mathcal{D}$. Il faut noter que les variables et les constantes peuvent être typées.
- *Termes* qui regroupe les variables et les constantes.
- *Fonctions* qui envoient des symboles sur des symboles. L'image d'une fonction est également un terme.
- *Prédicats* représentant une relation entre des symboles du domaine, qui sont ses arguments. Un *prédicat de base* (*ground predicate*) est un prédicat dont tous les arguments sont des constantes.
- *Connecteurs/Quantificateur* qui sont définis dans le tableau 2.1.

Ainsi, la formule suivante :

$$\forall x, y, z \in \mathcal{D},\ isSubClassOf(x, y) \wedge hasA(y, z) => hasA(x, z) \tag{2.10}$$

représente la règle d'héritage de la méronymie [4] : si une classe possède un méronyme, ses sous-classes le possèdent également. $x,\ y,\ z$ sont des variables et $isSubClassOf,\ hasA$ sont des prédicats. Si l'on considère le domaine d'intérêt $\mathcal{D} = \{Tasse, Anse, Conteneur\}$ alors

$$isSubClassOf(Tasse, Conteneur),\ hasA(Tasse, Anse)$$

sont des prédicats de base.

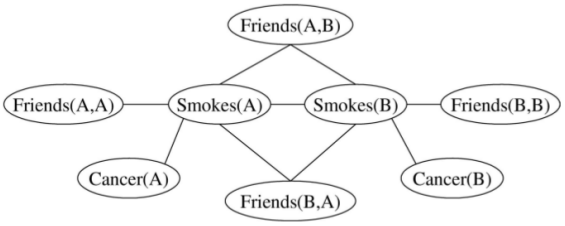
Une *base de connaissance du premier ordre* est simplement définie comme une collection de formules et de prédicats exprimés selon la syntaxe de la logique du premier ordre.

Il y a de nombreux langages permettant l'expression de relations logiques du premier ordre : on peut citer par exemple RDF (Resource Description Framework) [Bec04] qui est un formalisme

4. Relation sémantique désignant la partie d'un tout : par exemple, *anse* est un méronyme de *tasse*

English	First-order logic	Clausal form	Weight
Smoking causes cancer	$\forall x\ \mathtt{Sm}(x) \Rightarrow \mathtt{Ca}(x)$	$\neg\mathtt{Sm}(x) \vee \mathtt{Ca}(x)$	1.5
If two people are friends, either both smoke or neither does	$\forall x \forall y\ \mathtt{Fr}(x, y) \Rightarrow (\mathtt{Sm}(x) \Leftrightarrow \mathtt{Sm}(y))$	$\neg\mathtt{Fr}(x, y) \vee \mathtt{Sm}(x) \vee \neg\mathtt{Sm}(y)$, $\neg\mathtt{Fr}(x, y) \vee \neg\mathtt{Sm}(x) \vee \mathtt{Sm}(y)$	1.1 1.1

(a) Formules du premier ordre



(b) MLN instancié à partir des deux constantes A et B

FIGURE 2.5 – Exemple d'instanciation de MLN à partir de formules du premier ordre et de deux constantes A,B. Extrait et modifié à partir de [RD06]

simple représentant les relations sémantiques par un triplet $< Sujet,\ Predicat,\ Objet >$. OWL (Web Ontology Language) [MPSP+09] se base sur RDF pour représenter un plus grand nombre de relations. Enfin, nous pouvons citer le langage de programmation et moteur d'inférence Prolog [WSTL12] que nous utilisons dans l'implémentation de notre système.

Un défaut non négligeable de la logique du premier ordre est qu'elle ne peut rendre compte de situations réelles dans le sens où il n'y a peu de règles sans exception. Cela implique qu'un monde défini par un ensemble de règles aura une probabilité d'existence nulle si une seule de ces règles n'est pas vérifiée.

Afin de palier cela fut développée une approche mêlant modèle graphique probabiliste et logique du premier ordre, permettant d'exploiter l'expressivité de la logique et la robustesse des modèles probabilistes : ce sont des *modèles de Relations Statistiques* (*Statistical Relational Models* (SRM) dans la littérature). Markov Logic Network (MLN) [RD06] en est un exemple représentatif.

MLN est un template de champ de Markov générique défini à partir de formules en logique du premier ordre et qui s'instancie à partir d'un ensemble de constantes. Les prédicat de base (dont les arguments sont des constantes) constituent les nœuds du champ et les arêtes relient les prédicats de base apparaissant ensemble dans une même formule. On associe à chaque formule un poids relatif qui caractérise sa pertinence. La figure 2.5b est le champ de Markov instancié à partir de deux constantes A et B selon les formules de la figure 2.5a.

Ces approches permettent d'inférer de nouvelles relations dans une base de connaissance. Cependant, l'inférence est difficile d'autant plus que le graphe instancié croît exponentiellement avec le nombre de constantes.

2.3 Représentations combinant sémantique et perception

Récemment, de nombreuses recherches ont été faites sur l'utilisation combinée de représentations physiques (principalement visuelles) et sémantiques dans différents contextes. La suite de cette section va tenter de regrouper ces approches bimodales en fonction de leur objectifs.

2.3.1 Description d'objets par attributs

Les travaux de Lambert [LNH09] et Farhadi [FEHF09] ont posé les bases de la reconnaissance d'objets par attributs. L'idée est que l'on peut définir un grand nombre de concepts (classes) avec un ensemble borné d'attributs communs. Par exemple, on peut affecter à la classe *zèbre* les attributs *rayé*, *noir* et *blanc*. Ces attributs constituent donc une représentation intermédiaire des classes, faisant le pont entre la représentation visuelle et purement sémantique. Cette méthode a deux avantages principaux : elle permet à la fois de représenter des classes non présentes dans les données d'entraînement en les définissant par leur vecteur d'attributs, mais également d'inférer de nouveaux attributs à des classes connues. Dans la littérature, l'apprentissage de nouvelles classes sans données d'entraînement associées est appelé *Zero-shot learning*. Dans [LNH09] est proposée une méthode dite DAP (*Direct Attribute Prediction*), où les probabilités de présence $p(a|x)$ d'un attribut a dans une image x sont obtenues par l'apprentissage de classifieurs binaires SVM (*Support Vector Machine*) non-linéaires. L'inférence de la classe z d'une nouvelle image x est alors simplement

$$p(z|x) = \sum_m p(z|a_m)p(a_m|x). \tag{2.11}$$

Il est à noter qu'à chaque classe z est associée un vecteur attribut a^z, de telle sorte que

$$p(a|z) = 1 - \delta_{a-a^z}. \tag{2.12}$$

où δ [5] est le symbole de Kronecker. Ceci est une contrainte forte et qui, en pratique, n'est pas très robuste. Elle souffre du même problème évoqué précédemment concernant la logique du premier ordre : il suffit qu'une instance ne possède pas un attribut pour avoir une probabilité nulle d'appartenir à une certaine classe.

Romera-Paredes [RPT15] et Akata [APHS16] prolongent l'approche par attribut en cherchant un espace de représentation sémantique compatible avec les vecteurs de descriptions image associés (figure 2.6). La différence avec [LNH09] est qu'aucun classifieur par attribut n'est directement

5. δ_a vaut 1 si $a = 0$ et 0 sinon

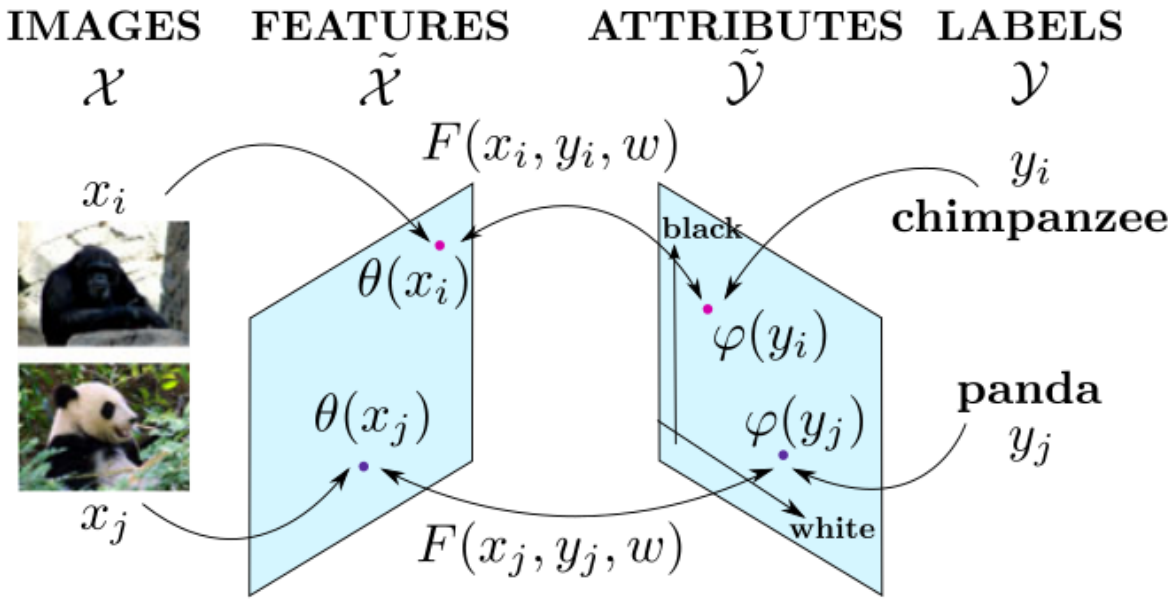


FIGURE 2.6 – Espaces de représentation des descripteurs image et sémantique (ici attributs). F est une fonction mesurant la compatibilité entre les deux représentations (visuelles et sémantique) d'un même concept. Extrait de [APHS16]

appris. L'accent est mis sur l'apprentissage de classes non connues et non sur la reconnaissance d'attributs. De plus, cette méthode plus générale permet également d'intégrer, en plus des attributs, d'autres sources d'informations telles que des hiérarchies de classes (ontologies).

Fu [FXKG15] considère le zero-shot learning en utilisant la distance sémantique, basée sur une représentation vectorielle de mots [MCCD13] ou par attributs comme précédemment, pour classifier des classes non vues lors de l'apprentissage. Pour cela, ils apprennent une projection des vecteurs de description image dans cette espace sémantique. L'ensemble des classes vues et non vues vont constituer un graphe sémantique, avec des liens entre les k plus proches voisins. Afin d'inférer la classe d'une nouvelle image test x^*, ils définissent une chaîne de Markov absorbante [6] sur leur graphe sémantique.

Zhang [ZXG16] développe une approche similaire qui n'apprend pas qu'une projection du visuelle vers le sémantique, mais également l'espace de représentation visuelle. Cela est fait via un réseau de neurones profond à deux branches : une classique servant à construire l'espace de représentation visuelle et une autre construisant la projection des données sémantiques dans l'espace de description visuelle. La fonction coût de ce réseau vise ensuite à minimiser la distance entre les vecteurs de description visuelles et la projection des représentations sémantiques.

6. Seuls les nœuds correspondants aux classes non vues sont absorbants ie ayant une probabilité nulle de transition avec toutes les autres classes

Les travaux de Luo [LLH+18] s'attachent à corriger un défaut inhérent à l'apprentissage de la projection des représentations visuelles dans l'espace sémantique. En effet, il y a un biais d'apprentissage lié aux domaines des classes considérées dans les données d'entraînement, donnant des résultats peu satisfaisants ensuite sur des données test issues de classes non vues provenant de domaines différents. Pour résoudre cela, au lieu de considérer une représentation sémantique fixe par classe, les auteurs proposent de la mettre à jour à partir des projections apprises sur les données visuelles. Pour les classes non vues, la mise à jour se fait à partir des relations avec la représentation sémantique des classes vues voisines.

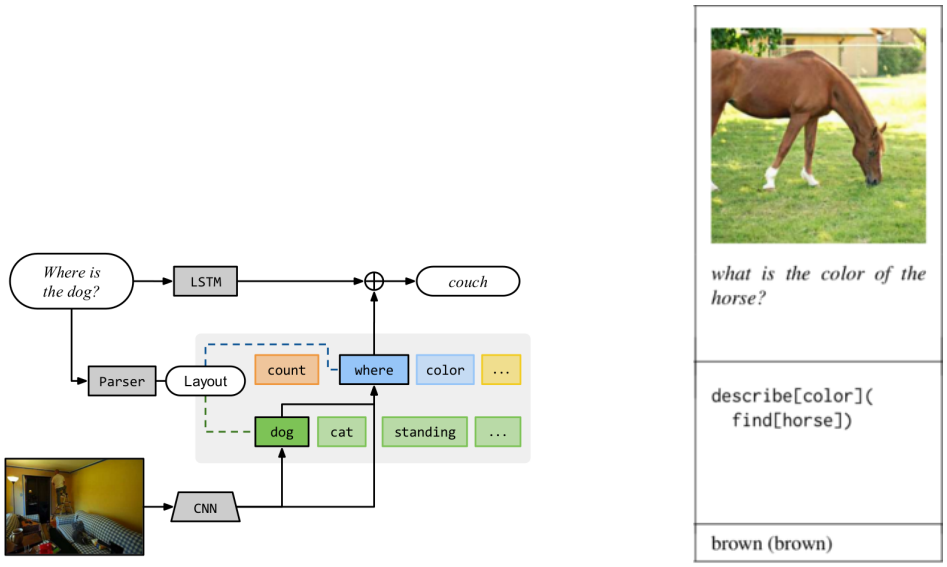
Jusqu'à présent, toutes les approches vues ont pour but d'exploiter les données textuelles afin d'améliorer la description visuelle. Silberer [SFL17] propose l'opposé : exploiter les données visuelles associées à chaque mot afin d'obtenir une représentation sémantique rendant mieux compte des similarités entre mots. Pour cela, trois autoencodeurs (AE) [GBC16] sont entraînés : un pour la représentation visuelle par vecteur d'attributs, un pour la représentation textuelle et enfin un AE final qui fusionne les deux modalités.

Les méthodes ci-dessus développent des idées intéressantes, mais un de leur défaut est qu'elles ne considèrent que des classes non vues dans leur ensemble de test. C'est à dire qu'une nouvelle image x^* est considérée par avance comme étant une instance d'une classe non vue. Or, dans des applications concrètes robotiques, cela est rarement le cas. Pour essayer de répondre à ce problème, Chao *et al.* [CCGS16] introduisent le *Generalized Zero-shot learning*. Ils montrent qu'en appliquant naïvement ces méthodes dans le cas général, la majorité des instances de classes non vues sont incorrectement classifiées en classes vues. Ceci se comprend aisément du fait que les données d'entraînement ne concernent que les classes vues, l'information sur les classes non vues étant réduite à la seule représentation sémantique.

2.3.2 Description sémantique d'image

Nous allons traiter dans cette section des méthodes cherchant à décrire en langage naturel des images. Plus particulièrement, nous considérons les problématiques de *légende* (*captioning* en anglais) et de réponse visuelle à des questions (*Visual Question Answering*) principalement issues de la recherche en vision.

Andreas [ARDK16] et Malinowski [MRF15] proposent de répondre à des questions telles que *Où est le chien ?*, *Qu'est ce qui se trouve derrière la table ?* à partir d'une image. Pour cela, ils utilisent un réseau de neurones convolués (CNN) [KSH12] [SZ14] [GBC16] pour représenter l'image. La sortie de celui-ci est ensuite passée à chaque couche cachée d'un réseau LSTM (*Long Short Term*



(a) Schéma du modèle (b) Exemple de résultat

FIGURE 2.7 – Approche proposée par [ARDK16]. Images extraites de [ARDK16]

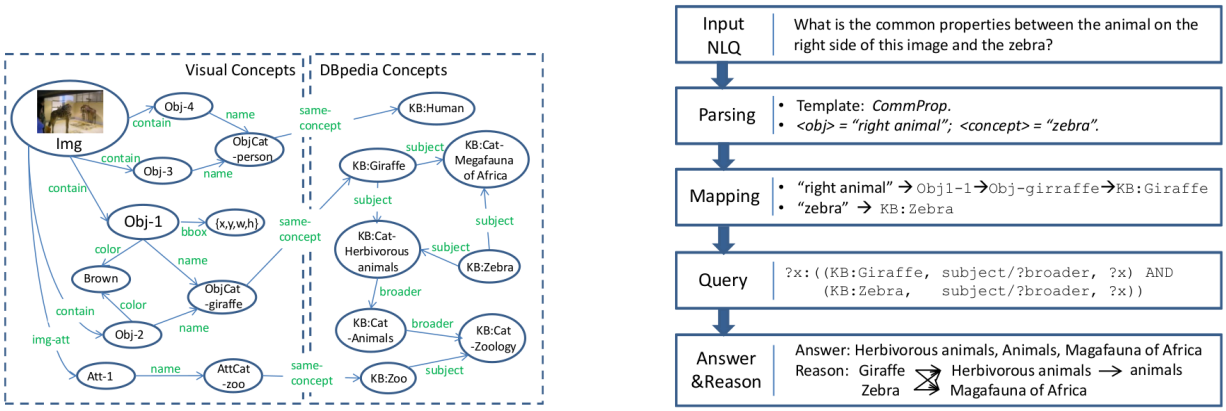
Memory) [HS97] [GBC16] avec en entrée la question posée. Plus particulièrement, [ARDK16] convertit la question en langage naturel en une série d'instructions du type :

Quelle est la couleur du chat ? => describe[color](find[chat])

Chacune de ces instructions constitue un module à part. L'intérêt est de pouvoir adapter ce système à d'autres domaines en ajoutant/remplaçant certains modules puis en les entraînant conjointement.

Wang [WWS+15] développe un système qui utilise l'ontologie DBpedia afin de pouvoir raisonner sur des informations extra-visuelles. L'approche est ensuite similaire à [ARDK16] : la question est convertie en un triplet RDF puis chaque sujet/objet est détecté dans l'image par un réseau de neurones convolués Fast-RCNN (*Region Convolutional Neural Network*) [Gir15]. La différence ici est que l'action requise pour répondre aux questions est gérée par une requête sur la base de connaissance alors que [ARDK16] entraîne un réseau de neurones peu profond spécialisé pour chaque type de requête.

Bien que les résultats présentés soient prometteurs, ils restent fortement dépendant de l'ensemble d'entraînement : les systèmes sont limités aux classes et modèles de questions présents lors de la phase d'apprentissage. Ceci est d'autant plus problématique dans le contexte de la robotique d'exploration où notre objectif est de permettre à notre système d'apprendre continuellement sur le monde l'environnant. Ramakrishnan [RPSM17] soulève ce point en montrant la perte importante de



(a) Liens entre concepts sémantiques et visuels

(b) Processus d'analyse d'une question en langage naturel

FIGURE 2.8 – Approche proposée par [WWS+15]. Images extraites de [WWS+15]

performance lorsque des classes non vues durant l'entraînement sont présentes dans l'ensemble de test. Pour limiter ce problème, leur méthode pré-entraîne une architecture composée d'un LSTM et d'un CNN en utilisant des données auxiliaires (vocabulaire élargie, images extérieures issues d'ImageNet [DDS+09]).

Du côté du *captioning*, Karpathy [KFF15] propose un système permettant de légender automatiquement une image. La méthode s'inspire à la fois du *zero-shot learning* en projetant le texte descriptif de l'image et l'image elle-même dans un même espace. Pour cela, il utilise un BRNN (*Bidirectional Recurrent Neural Network*) [SP97] qui permet d'estimer un espace de projection pour une phrase complète et non des mots individuellement comme cela était le cas précédemment. La détection des objets dans l'image se fait ici aussi via RCNN.

2.3.3 Cartographie sémantique

Les travaux qui vont être présentés dans cette section se positionnent dans un contexte robotique plus classique, en particulier l'apport de connaissance sémantique dans la cartographie d'environnement.

Dans la littérature, le mot *sémantique* est souvent employé mais il n'a pas toujours la même portée. En effet, une tâche de classification d'objets ou de segmentation dite sémantique n'en n'est qu'une utilisation superficielle où le sémantique n'est qu'un label. On pourrait remplacer ces labels par des indexes sans ne rien changer aux algorithmes d'apprentissage. C'est d'autant plus le cas en cartographie sémantique. Des applications telles que [WS08] ou [DFRDW11] qui visent à segmen-

ter l'environnement en régions labellisées en sont des exemples. Nous allons plutôt nous attarder sur des approches exploitant plus finement les données en langage naturels.

Sunderhauf [SDM+16] [SPL+16] introduit un système de cartographie par grille d'occupation [TBF05] avec la particularité ici non pas d'associer la probabilité de présence [7] à chaque élément de la grille mais un vecteur de probabilité sur le type de pièce via un classifieur basé sur un réseau de neurones convolués. Cette carte est donc organisée sous forme de couches comme illustré à la figure 2.9, chacune correspondant à une grille d'occupation classique liée à une classe de pièce.

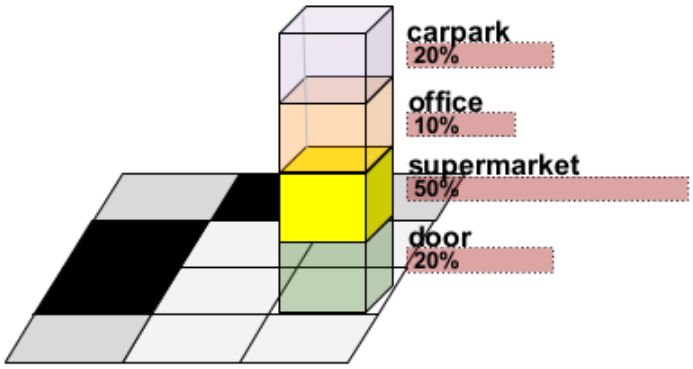


FIGURE 2.9 – Grille d'occupation de [SDM+16] où chaque couche correspond à une classe distincte. Extrait de [SDM+16]

Ils étudient également l'utilisation de cette donnée afin d'améliorer la détection d'objet, en définissant une probabilité de présence à priori d'objets en fonction du type de pièce. L'idée est qu'une fourchette a de fortes chances de se trouver dans une cuisine mais peu d'être dans une salle de bain.

Pronobis [PJ11] [PJ12] développe une architecture pour la cartographie sémantique d'intérieur en raisonnant sur des modalités hétérogènes telles que l'observation d'objets, la taille et la forme des pièces. Leur représentation spatiale est composée de quatre couches allant des données capteurs bas niveau aux relations conceptuelles abstraites (figure 2.10) :

- Une couche contenant la carte métrique, en pratique une grille d'occupation classique en SLAM.
- Une couche qui contient une discrétisation de l'espace en *lieux*, qui sont définis par un ensemble de propriétés : occurrence de classe d'objet, forme géométrique du lieu, surface.
- Une couche qui contient des modèles statiques d'objets, les modèles géométriques et d'apparences des pièces.

7. Plus précisément, le logit (*log odd*) de cette probabilité

– Enfin, la dernière couche contient les concepts et instances présents dans l'environnement en utilisant une ontologie. En particulier, ils décrivent des relations :

  - concept-concept e.g. *a Living-room* ***has-a*** *TV*
  - instance-concept e.g. *object1* ***is-a*** *TV*
  - instance-instance e.g. *place1* ***has-a*** *object1*

  En pratique, l'ontologie est représentée par un graphe probabiliste où les nœuds correspondent aux concepts et aux instances. Les arêtes sont de différents types (e.g. *is*, *has-a*) et représentent les liens entre les concepts et les instance. Les fonctions potentielles associées à ces arêtes sont apprises de manière supervisée.

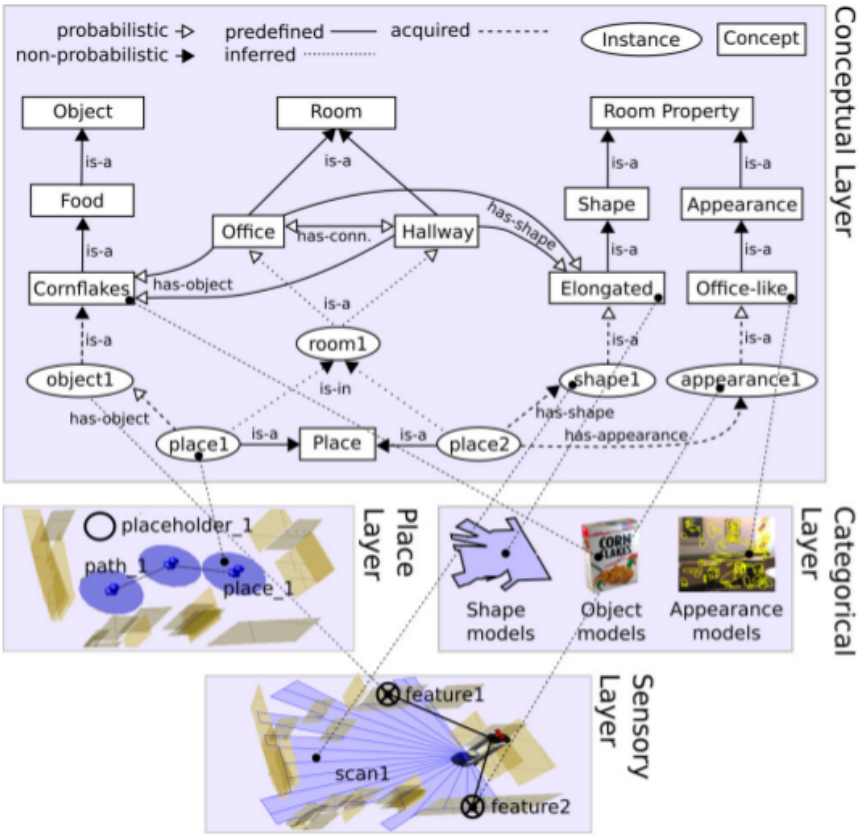
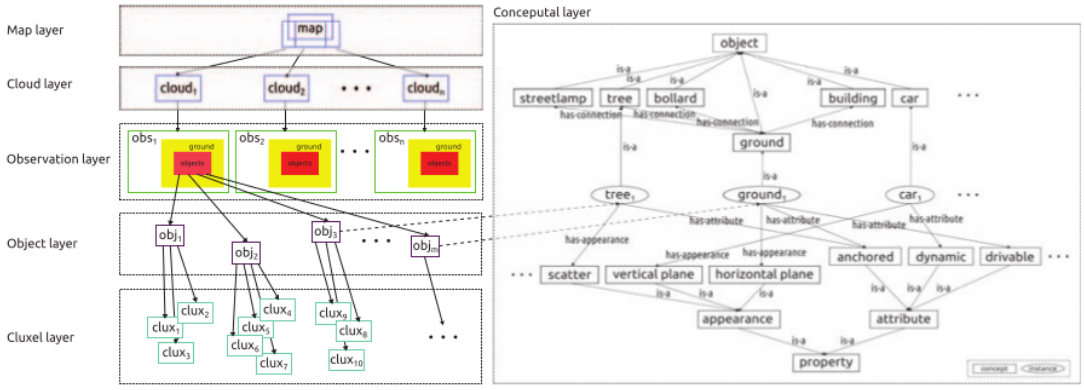
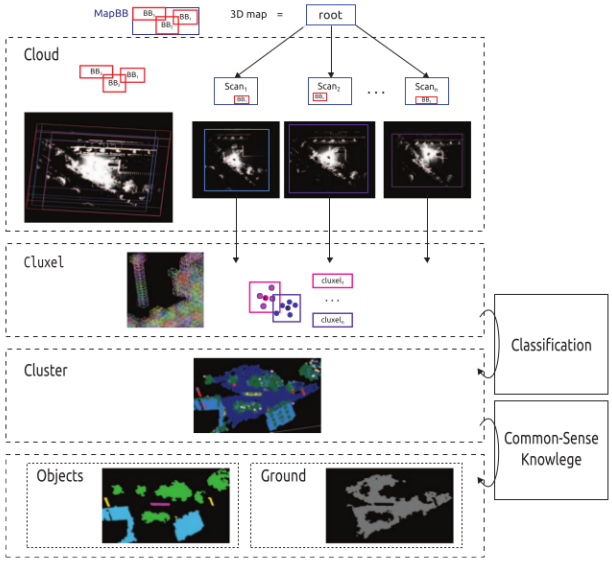


FIGURE 2.10 – Schéma général de l'architecture proposée par Pronobis [PJ12]. Extrait de [PJ12]

Lang [LFHP14] [LP14] [LFHP15] introduit une architecture cette fois-ci destinée à la cartographie sémantique d'environnements extérieurs (figure 2.11). Elle se décompose en six couches :



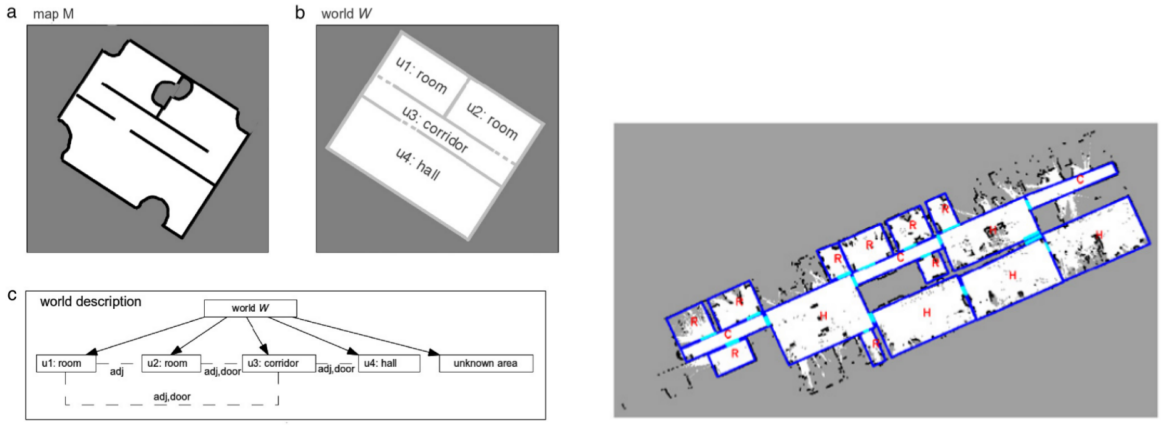
(a) Architecture globale proposée par [LFHP15]



(b) Construction de la carte sémantique extérieure

FIGURE 2.11 – Architecture multimodale proposée par Lang [LFHP15]. Extrait de [LFHP15]

- La première couche est le nœud racine de la carte, qui intègre les nuages de points des couches supérieures dans un système de coordonnées globales.
- La couche suivante contient l'ensemble des points 3D issus des données capteurs. Ces nuages sont ensuite sursegmentés en *cluxels* (combinaison de *cluster* et de *voxel*), eux-même regroupés ensuite en Observations.
- Une couche stockant les cluxels issus de la couche précédente. Un cluxel est défini par un volume autour d'un point. Il est à noté que les cluxels peuvent se superposer.
- Une couche classifiant les cluxels en Observations de différentes classes : plan horizontal,



(a) Aperçu globale de la méthode. Extrait de [LW14]

(b) Exemple de carte sémantique obtenue. Extrait de [LW14]

FIGURE 2.12 – Architecture multimodale proposée par Liu [LW14]

plan vertical, colonne, petite structure et éparpillé. Cette classification est faite à partir d'un champ aléatoire conditionné (CRF) construit en reliant les cluxels qui s'intersectent.

– Une fois les cluxels classifiés, cette couche regroupe les cluxels en objets en se basant sur une ontologie incluant par exemple des informations telles que *object is-a tree* ou *tree has-a scatter appearance*.

Liu [LW14] présente une architecture générique pour la cartographie sémantique d'intérieurs basée sur les réseaux de Markov logiques (MLN) [RD06] et les chaînes de Markov Monte Carlo basées données (data-driven MCMC). Il considère trois types d'unité (pièce, couloir et halle) et a pour objectif d'obtenir le modèle $W = \{U, T, R, \Theta\}$ maximisant la vraisemblance de la grille d'occupation M observée :

$$W = arg\max_{W'} p(W'|M) = arg\max_{W'} p(M|W)p(W) \tag{2.13}$$

Ce modèle est défini par quatre éléments :

– U est un ensemble d'unité rectangulaire, composée de quatre sommets. Cela correspond à la segmentation de la grille.

– T est le type (ou classe) de chacune des unités dans U. Le type est déterminé à partir de simples classifieurs sur la taille et le ratio longueur/largeur.

– R est une matrice dont l'élément en position i, j représente la relation d'adjacence entre les unités $u_i,\ u_j \in U$, qui s'obtient en dilatant chaque unité et en regardant leur intersection.

– Θ est une matrice binaire indiquant si deux unités $u_i,\ u_j$ doivent avoir un mur commun avec la même longueur.

Une fois T et R définis, ils sont utilisés afin de construire les prédicats de base qui vont instancier leur MLN avec des nœuds tels que *Room(u_i)* et *Adjacent(u_i, u_j)*. Des formules spécifiques à la tâche considérée sont également utilisées, comme par exemple le fait que deux unités adjacentes qui sont des pièces doivent avoir un mur en commun de même longueur. Ce MLN est ensuite exploité pour inférer la probabilité que deux unités aient un mur commun de même longueur, ce qui permet par la suite de calculer la probabilité à priori du modèle $p(W)$. Le terme postérieur $P(M|W)$ est lui calculé par MCMC.

Ces approches par couches qui intègrent à la fois les données capteurs avec la sémantique se rapproche de l'objectif de nos travaux. Cependant, elles sont fortement orientées vers l'application finale pour laquelle elles ont été conçues. Lang *et al.* limitent leur ontologie et ses relations à un ensemble d'objets prédéterminés (arbre, sol, ...). De même, les systèmes proposés par Sunderhauf et Pronobis ont pour objectif la classification de scène. Notre approche vise plutôt à une compréhension générale, indépendante de l'application finale, de l'environnement au travers des instances le constituant.

2.4 Conclusion

Dans ce chapitre, nous avons répertorié des travaux représentant les principales façon d'aborder la représentation d'environnement, que ce soit de manière directe ou indirecte au travers d'une application particulière. On remarque donc que naturellement, la représentation d'environnement considérée est fortement fonction de l'application. La table 2.2 récapitule l'ensemble des familles de méthodes vues et les classe selon nos trois axes définis en introduction : spatialité, temporalité et abstraction. Seules les approches de cartographie sémantique multicouches proposent des architectures combinant différents degrés de spatialité et d'abstractions. Cela est également le cas d'une autre approche proposée par le système KnowRob [Ten11] [TB17], qui sera abordée au chapitre 3 concernant notre modèle sémantique.

Notre architecture globale, qui sera le sujet des chapitres suivants, s'inspire de toutes ces méthodes et principalement des approches multi-couches. La différence majeure est que dans notre cas la représentation d'environnement n'est pas un intermédiaire utilisé pour accomplir un objectif donné. Nos travaux se focalisent sur cette notion de modélisation d'environnement en tant que telle. Le but est d'obtenir une base générique qui pourra alors être agrémentée de modules spécifiques en fonction des applications envisagées.

Approche	Spatialité	Temporalité	Abstraction
Superpixels/Supervoxels [ASS+12] [WGB12] [FVS09] [VBM10] [PASW13]	+	+	+
Segmentation objets basée surfaces lisses [ÜHR13] [ÜEHR14]	+	+	++
Segmentation objets basée convexité [ABS14] [SSPW14] [SPW15]	+	+	++
Segmentation objets basée mouvement [TK92] [KVD91] [CK98] [BHB00] [XCK06] [MC09] [SBWK10] [Zie15]	+	++	++
Estimations de liaisons mécaniques [YP05] [YP08] [JAP13] [RTZ10] [FRA11] [JCD15]	+	++	+++
SLAM [TBF05] [FM12]	++	+	+
Cartographie sémantique : classification de scène [SDM+16] [SPL+16]	++	+	++
Cartographie sémantique multi-couches [PJ11] [PJ12] [LFHP14] [LP14] [LFHP15] [LW14]	+/+++	++	+/+++
Base de connaissance [MMM+04] [WSTL12] [MPSP+09] [MCH+15]	++++	++++	++++
Description objet par attributs sémantiques [LNH09] [FEHF09] [RPT15] [APHS16] [FXKG15] [ZXG16] [LLH+18]	++++	++++	+++
Visual Question Answering [ARDK16] [MRF15] [WWS+15]	+	+	+++
Captioning [KFF15]	+	+	+++
KnowRob [Ten11] [TB17]	+++	+ / ++++	+ / ++++

TABLE 2.2 – Classifications des différentes représentations rencontrées dans l'état de l'art selon les trois axes de spatialité, de temporalité et d'abstraction (noté de + à ++++)

Chapitre 3

Modèle de base de connaissance

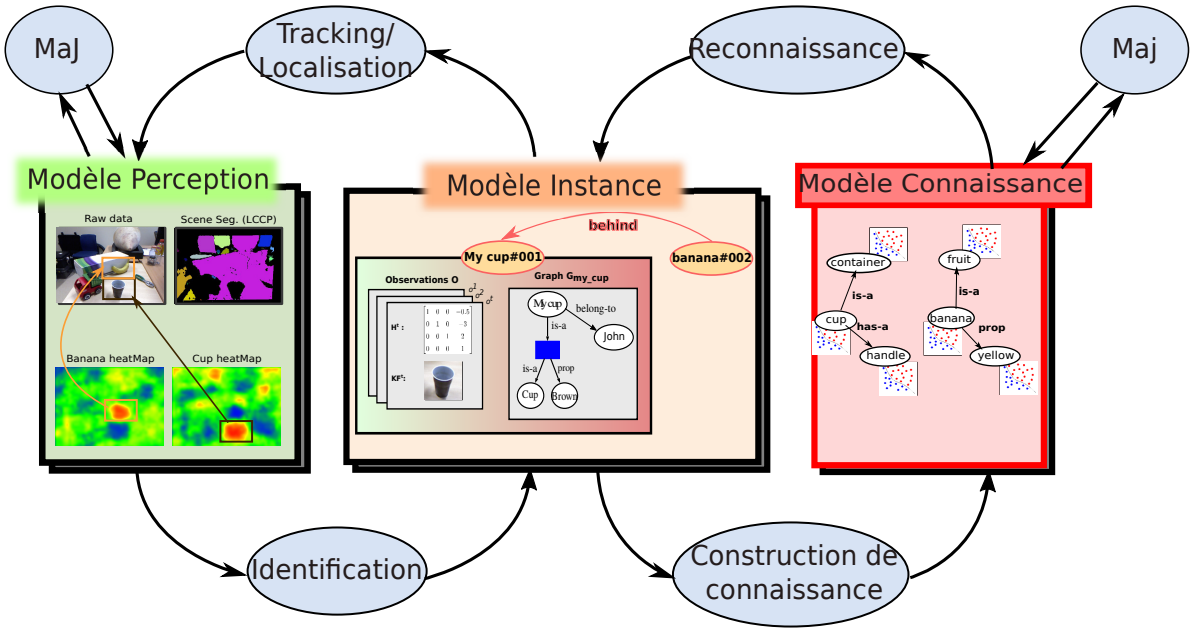


FIGURE 3.1 – Modèle sémantique dans notre architecture

Ce chapitre est dédié au modèle de connaissance introduit lors de la description générale de l'architecture (cf figure 3.1) [BDZ18]. Celui-ci va constituer une représentation compacte de l'environnement telle que l'homme l'a définie au travers d'un langage naturel. En effet, quelque soit la langue, les mots la constituant sont hiérarchisés et segmentés selon les concepts qu'ils représentent. De plus, de nouveaux concepts peuvent être décrits à partir de concepts et de relations déjà existants. On peut prendre l'exemple d'un dictionnaire où des termes complexes sont définis par une mise en relation de termes plus simples et génériques. Cet ensemble de relations, que l'on nomme

ontologie (section 3.1), est un condensé générique de connaissances. Afin d'exploiter cela, il nous faut transcrire ces relations en langage naturel dans une représentation facilement exploitable par un système robotique.

Cette section va s'articuler de la façon suivante. Tout d'abord, nous allons revenir sur l'état de l'art concernant la représentation de connaissance. Des exemples d'ontologies seront données en insistant particulièrement sur WordNet [Mil95] et NELL [MCH+15]. Dans un deuxième temps, nous décrirons le système KnowRob [TB17] qui se rapproche des travaux de cette thèse. Nous verrons ensuite en quoi ces ontologies ne nous semblent pas parfaitement adaptées dans le cadre de la robotique autonome. Notre ontologie sera définie formellement puis nous détaillerons notre méthode de construction automatique à partir de définitions extraites de dictionnaire. Enfin, la pertinence de l'ontologie ainsi construite est évaluée à partir de l'analyse quantitative des relations extraites et qualitativement par l'observation des sous-graphes de connaissances obtenus.

3.1 Représentation des connaissances par ontologie

Une ontologie est un ensemble structuré de concepts, basé sur un ensemble de relations et pouvant se représenter sous forme de graphe. Elle peut être globale ou contenir des relations spécifiques à certains domaines d'application. OpenCyc [Len95] [MCWD06] et DBpedia [ABK+07] (créée à partir de la base de données de Wikipedia) sont des exemples d'ontologies globales construites manuellement par des experts. Elles représentent des concepts pouvant être très abstraits et concernant des domaines telles que l'Histoire ou la politique. Dans le cadre de la robotique, les concepts liés à ces domaines peuvent être vus comme du "bruit". De plus, les concepts plus concrets concernant des objets physiques ne sont pas suffisamment détaillés et adaptés à une application robotique.

Notre ontologie se fonde sur celle de WordNet [Mil95] présentée ci-dessous.

3.1.1 Base lexicale et ontologie WordNet

WordNet [Mil95] est une large base lexicale en anglais construite manuellement. Les mots correspondant à un même concept sont regroupés en une structure dite *synset* à laquelle est associée un identifiant unique, une définition et parfois des phrases d'exemple. Cet identifiant est nommé *WordnetId*. Il est composé d'une lettre indiquant la nature grammaticale du concept (*n* pour nom, *a* pour adjectif et *v* pour verbe) suivie par une séquence de 8 chiffres. Dans la suite de ce document, nous noterons les concepts sous la forme *concept-wordnetId* afin de distinguer les homonymes comme par exemple *fork-n03383948* (fourchette) et *fork-n03384167* (fourche). L'ontologie WordNet est

FIGURE 3.2 – Illustration de la structure de WordNet à partir d'une requête sur le mot *fork* via l'interface web. La totalité des résultats n'est pas représentée

donc un graphe de connaissance dont les nœuds représentent des synsets. Ceux-ci sont reliés entre eux par des liens d'hyper/hyponymies (*isA*), de méronymies (*hasA*) et de synonymies.

Par exemple, le synset ayant pour WordnetId $n02958343$ correspond au concept car et contient :

- des synonymes $\{auto,\ automobile,\ machine,\ motorcar\}$.
- une définition : *a motor vehicle with four wheels ; usually propelled by an internal combustion engine*.
- des hyponymes comme par exemple *ambulance-n02701002*
- un hyperonyme : *motor vehicle-n03791235*
- des méronymes comme par exemple *car seat-n02970685*

Elle est largement utilisée comme ontologie de référence dans des applications liées à la vision et plus particulièrement à la tâche de classification, notamment par la base de données image ImageNet [DDS+09].

WordNet organise ses synsets selon leur nature : nom, adjectif, adverbe ou verbe. Chaque type est ensuite sous-divisé en fonction de catégories syntaxiques, comme le montre la table 3.2 de la section 3.3. Un exemple de recherche sur le mot *fork* via l'interface web [1] est donnée en figure 3.2. On peut y voir la représentation des synsets comprenant le mot *fork*. Par exemple, le synset correspondant au deuxième sens regroupe les synonymes *{branching, ramification, fork, forking}*. Les résultats sont classifiés selon leur nature (nom, verbe) avec l'indication de leur sous-catégorie (soulignée en vert). On a également un aperçu des relations concernant le premier sens.

La figure 3.3 représente une sous-partie de l'ontologie WordNet en partant du concept *fork*.

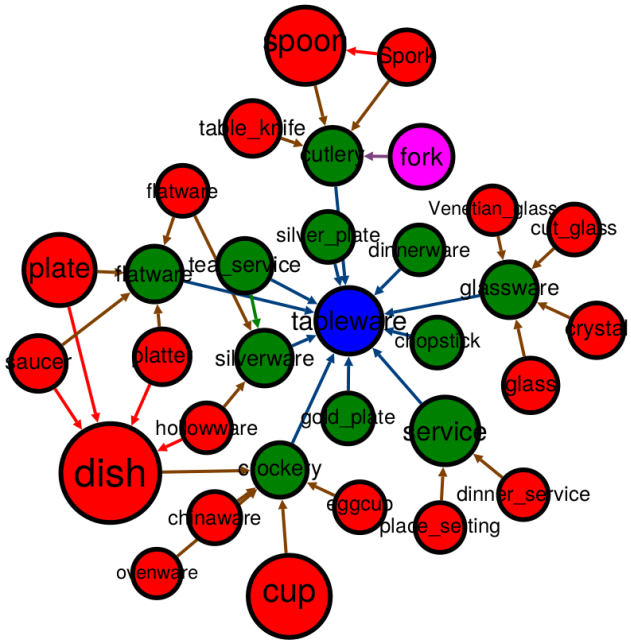


FIGURE 3.3 – Extrait de l'ontologie WordNet en partant du concept *fork*. Chaque arête (orientée) du graphe correspond à une relation d'hyperonymie. La couleur des nœuds indique la distance en terme de niveau hiérarchique : rouge pour 0, vert pour 1 et bleu pour 2

3.1.2 Ontologie NELL

La quasi-totalité des ontologies sont conçues manuellement. Le projet NELL (Never Ending Language Learning) [CBK+10] [MCH+15] propose un apprentissage continue de relations sémantiques à partir d'Internet. Le système se base sur une ontologie de départ composée d'une centaine de catégories (concepts) (eg. Sport, Athlète) et de relations binaires (eg AthletePlaysSport(x,y)).

1. http ://wordnetweb.princeton.edu/

Il faut noter ici que, contrairement à notre approche, les instances (qu'ils dénomment par *entités* ou *membres*) sont inclues dans l'ontologie : *Usain Bolt* sera représenté comme étant un *Athlète* à travers la relation *Generalization* (équivalent de *isA*). A l'initialisation, une douzaine d'exemples sont donnés pour chaque catégorie et chaque relation. Un grand ensemble de pages web est ensuite analysé afin d'extraire de nouvelles connaissances en se basant sur les connaissances précédemment acquises. Pour cela, différentes méthodes sont employées conjointement comme on peut le voir sur la figure 3.4. Il est intéressant de noter que le système intègre une modalité basée sur l'analyse d'image en incorporant également le projet NEIL (Never Ending Image Learning) [CXSG13], qui extrait des connaissances sémantiques à partir de cooccurrences visuelles. Le module *Learning embeddings* apprend une représentation vectorielle de chaque catégorie X_i (concept) et entité Y_i (instance) de l'ontologie ainsi qu'une matrice M représentant la relation *Generalization*. Formellement, cela permet de calculer un score de confiance sur une assertion $Generalization(X_i, Y_i)$:

$$S(X, Y) = \mathbf{v_{X_i}}^T M \mathbf{v_{Y_i}} \tag{3.1}$$

où $\mathbf{v_{X_i}}, \mathbf{v_{Y_i}} \in \mathbb{R}^{\mathbf{d}}$ sont des vecteurs de dimension d correspondant à la catégorie X_i et l'entité Y_i.

Nous renvoyons le lecteur sur les publications de référence [CBK+10] [MCH+15] pour plus de détails sur les autres méthodes employées. Des extraits de cette ontologie sont présentés aux figures 3.6a et 3.6b.

Cette génération semi-automatique d'ontologie est intéressante dans le cadre de nos travaux. En effet, le but est que le robot se fasse une représentation de son environnement de façon autonome. Cependant, dans le cas d'une application robotique, NELL souffre des mêmes défauts que les autres ontologies : manque de détails ou de relations pertinentes au domaine, avec ici également un sur-apprentissage de concepts (e.g. *plastic fork* ou *crest promise salad fork sterling 1948*). Nous verrons par la suite qu'une de nos contributions vise justement à séparer les concepts abstraits généraux de concepts correspondants à des instances, évitant ainsi cette sur-création de concepts.

La construction manuelle d'ontologie par des experts permet d'avoir des relations fiables mais non exhaustives : les domaines couverts par ces approches sont très larges, allant de la philosophie jusqu'aux atomes en passant par l'Histoire et la Géographie. Une grande partie de cette connaissance, quoi que exacte, constitue du bruit dans le cadre d'une application robotique. En effet, la robotique s'intéresse aux concepts concrets décrivant des objets réels et des actions. De plus, de part leur globalité, les types de relations proposés par ces ontologies ne sont pas adaptés à la description d'objets du quotidien.

Nous allons maintenant aborder un système robotique qui se base principalement sur la repré-

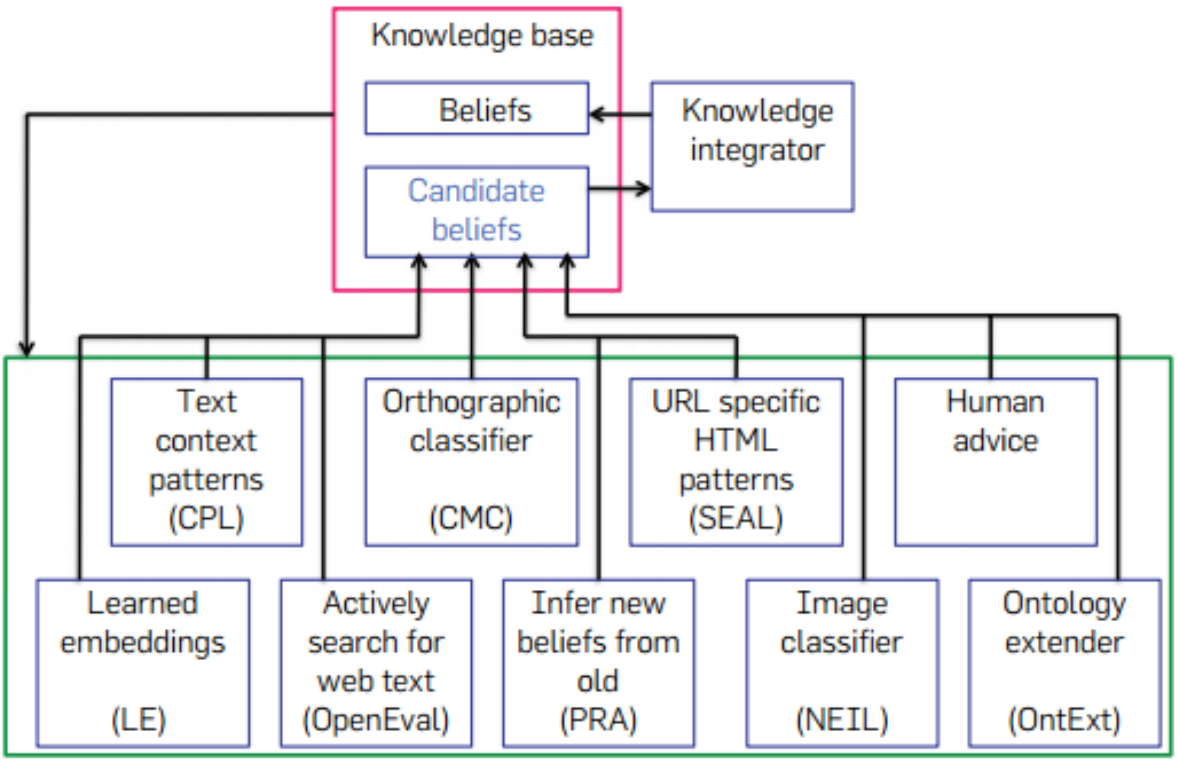


FIGURE 3.4 – Structure logicielle de NELL présentant l'ensemble des méthodes utilisées pour l'inférence de nouvelles connaissances. Extrait de [MCH+15]

sentation de connaissance sémantique : KnownRob.

3.1.3 Système KnowRob

KnowRob [Ten11] [TB17] est une architecture reposant sur la représentation de connaissance dans le cadre de la robotique d'assistance. Le but est de pouvoir faire effectuer des tâches de haut niveau à des robots dans des environnements humains. En particulier, il a été développé dans le but de réaliser des plats en suivant des instructions (recettes de cuisine) exprimées en langage naturel. La difficulté vient du fait qu'un certain nombre d'informations et d'étapes sont implicites dans une recette de cuisine : le robot va devoir les inférer à partir de ces propres connaissances et de ses capteurs. Il est intéressant de noter que l'approche proposée est basée sur le sémantique et utilise la vision afin de corriger les erreurs d'inférences. Les travaux robotiques précédents utilisant conjointement la sémantique et les données sensorielles fonctionnaient dans le sens inverse : ils se basaient sur la vision, la sémantique représentant des contraintes ou estimant des probabilités a-priori.

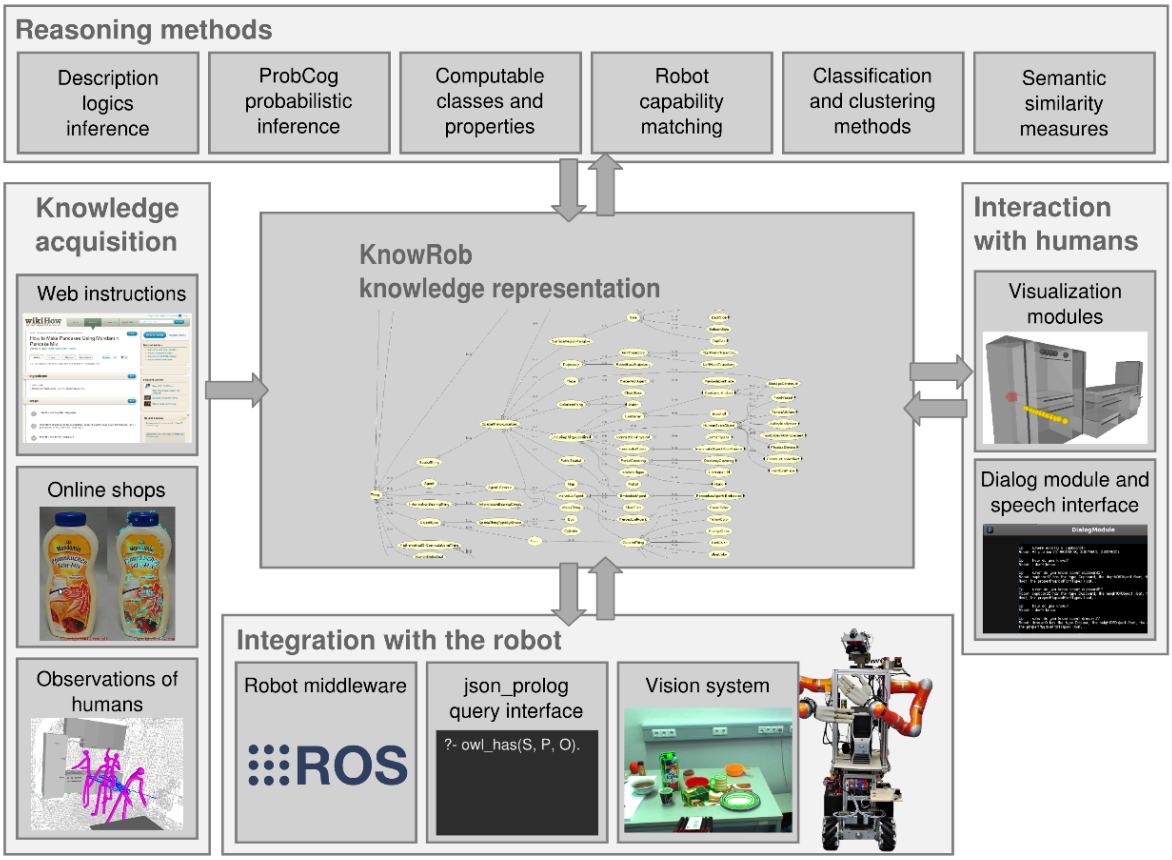


FIGURE 3.5 – Aperçu global du système KnowRob. Extrait de [Ten11]

KnowRob utilise une base de connaissance symbolique qui donne, en plus d'une ontologie classique, une représentation sémantique des différentes structures de données de plus bas niveau. Par exemple, leur concept *SpatialThing* va regrouper des sous-concepts tels que *Point* ou *Trajectory*. *MathematicalThings* regroupera lui *Vector* et *CoordinateSystem*. Ce sont des concepts correspondant à des données exploitées par des algorithmes d'inférence au niveau capteur (eg. RGBD). Ceci permet donc de "projeter" les informations bas niveau dans une ontologie sémantique commune. Se faisant, les requêtes peuvent combiner des inférences sur la base de connaissance ainsi que des inférences à partir de données capteurs à la volée. Cette unification permet de compenser les défauts des représentations purement sémantiques, à savoir la trop grande rigidité et généralité des contraintes entre concepts. Lorsque le système doit traiter des informations incertaines et non représentables de façon déterministe, il utilise un réseau de logique Bayésien (BLN) [JWB09]. Développés par les mêmes auteurs, les projets Open-EASE [BTW15] et RoboEarth [RTDM+15] exploitent une version cloud et participative de KnowRob. L'ontologie utilisée par KnowRob a été construite ma-

nuellement sur la base d'OpenCyc [Len95] [MCWD06]. En plus de représenter des structures de données, ils ont également introduit un certain nombre d'objets liés à leur application. On peut voir un extrait de leur ontologie aux figures 3.6c et 3.6d.

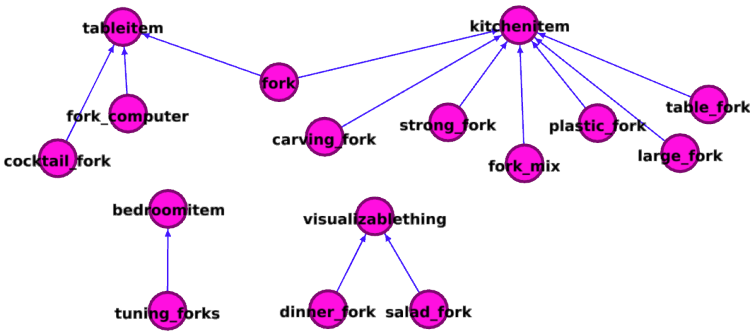
KnowRob est aussi intégré dans le système RoboSHERLOCK [NBBB14] [BBBB$^+$15]. Ce système permet de fusionner le résultat de différents algorithmes pour la reconnaissance d'objets. En effet, des règles sont attachées à chaque algorithme de perception afin de savoir dans quels cas ils doivent être utilisés eg. objet texturé, détection de texte. En supposant une scène segmentée en un ensemble d'objets, ceux-ci sont annotés à partir des résultats de chacun des algorithmes spécialisés. Par exemple, un algorithme de détection de couleur peut donner *color(object1,red)*. Ces annotations sont utilisées pour la construction de règles associées à chaque algorithme. Chaque objet est donc annoté avec un ensemble de prédicats qui vont former un MLN, à partir duquel la catégorie de l'objet sera inférée. L'utilisation d'un MLN permet ainsi de combiner des annotations contradictoires. Cependant, cela nécessite un apprentissage supervisé des poids associés aux différents prédicats et donc d'avoir accès à des données d'entraînement appropriées.

3.1.4 Discussion

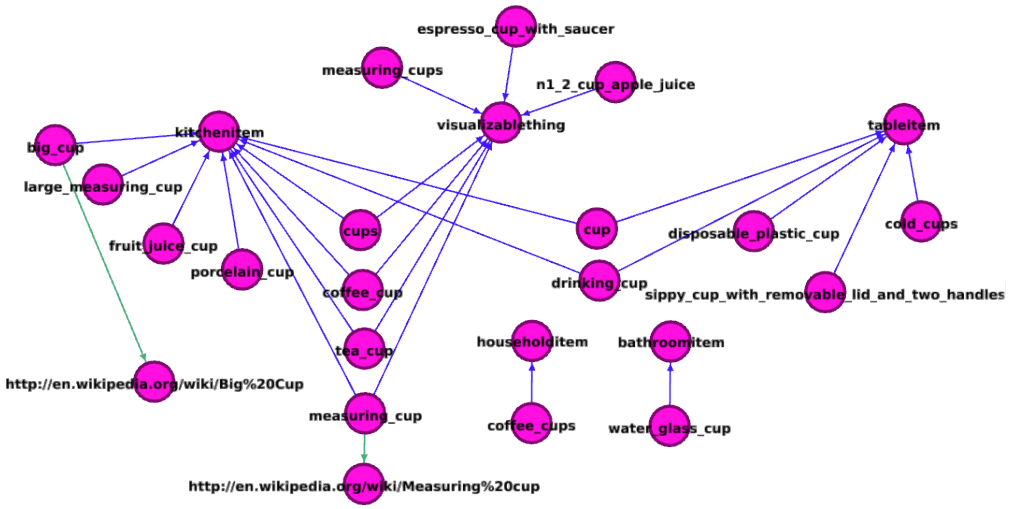
Un certain nombre d'approches sur la représentation/construction de connaissances sémantiques ont été présentées. Abordons maintenant l'adéquation de ces ontologies dans le cadre de nos travaux. Notre objectif est d'avoir une ontologie centrée sur la description sémantique de concepts représentant des objets physiques. Nous voulons une séparation claire entre un concept abstrait et ces différentes réalisations physiques (instances) qui seront elles représentées dans notre modèle d'instance (section 5.1). Il faut se rappeler que notre modèle ne cherche pas à résoudre une tâche précise : il est conçu comme une base générale à laquelle pourraient se greffer des modules spécialisés. L'ontologie doit donc rester générale.

KnowRob a été développé en vue d'être intégré sur un robot cuisinier. Il doit pouvoir interpréter et exécuter une tâche décrite par une recette. Dans ce système, l'ontologie liée aux différentes tâches a été définie manuellement (cf figure 3.6). On voit que l'information concernant les objets est relativement sommaire et centrée sur l'application visée. Certaines propriétés des objets (comme *small* ou *big*) ne sont pas représentées explicitement. En pratique, ces propriétés sont intégrées dans les modèles géométriques/visuels des objets mais ne sont pas représentés de façon abstraite dans l'ontologie. Il faut souligner que leur base de connaissance a surtout pour but de créer une représentation commune pour les données (*Angle*, *Distance*,...), les algorithmes et les concepts (objet/tâche).

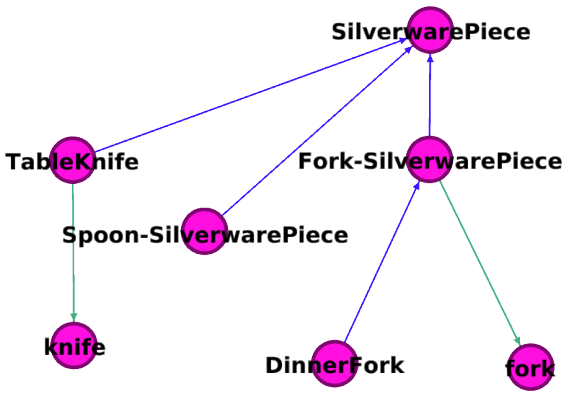
Le principal défaut, du point de vue de notre modèle, est donc de devoir créer manuellement l'information en fonction de la tâche envisagée. Comme nous nous plaçons dans un monde ouvert



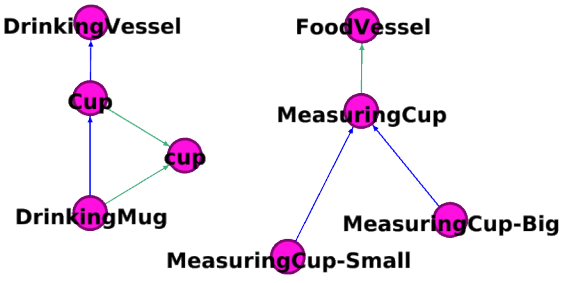
(a) *fork-NELL*



(b) *cup-NELL*



(c) *fork-KnowRob*



(d) *cup-KnowRob*

FIGURE 3.6 – Exemple de sous-graphes extraits de l'ontologie NELL [MCH+15] et KnowRob [TB17]. Pour des raisons de visibilité seule une partie significative de ces sous-graphes est représentée.
NELL : Les liens correspondent aux propriétés *generalizations* (hypernymie, en bleu) et *haswikipediaurl* (en vert)
KnowRob : Les liens correspondent aux propriétés *subClassOf* (bleu) et *hasValue* (vert). Les relations *disjointWith* ne sont pas représentées

sans tâche spécifique, nous devons considérer des méthodes de génération automatique.

Avec internet, nous avons accès à une source énorme d'informations en langage naturel qui permettrait potentiellement de construire une ontologie complète. Cependant, il y a un certain nombre de difficultés à surmonter avant d'atteindre un tel objectif. NELL propose une création automatique d'ontologie en parcourant continuellement l'Internet. Cependant, la grande généricité de ce système entraîne également un sur-apprentissage de concepts qui ne sont qu'une combinaison de concepts plus simples. Pour tenter d'illustrer en quoi ce type d'ontologie ne peut convenir tel quel en robotique, nous avons extrait les sous-graphes concernant les concepts *fork* et *cup* en se limitant à leur sens principal. La figure 3.6 en propose une représentation graphique.

On observe, notamment pour *cup*, des concepts très (trop ?) spécialisés comme

sippy_cup_with_removable_lid_and_two_handles ou *expresso_cup_with_saucer*

En effet, ces deux concepts sont simplement une combinaison de concepts plus génériques. Dans notre modèle de représentation, ils seraient représentés par des nœuds facteurs (cf section 3.2). Les pluriels

cup/*coffee_cup*, *cups*/*coffee_cups*

sont également ici dissociés, bien que les concepts soient identiques. On peut également apercevoir des relations qui peuvent être vraies mais qui ne sont pas pertinentes de manière générale [2] comme

<*water_glass_cup, generalizations, bathroomitem*>

<*tuning_forks, generalizations, bedroomitem*>

Enfin, on observe un certain nombre de relations manquantes entre les concepts présents, comme par exemple

<*salad_fork, generalizations,fork*>

<*table_fork, generalizations, fork*>

Bien que chaque méthode contienne son lot d'erreurs, le problème est ici qu'il ne reste que peu d'informations exploitables par un robot et elle est principalement en lien avec le contexte spatial (*kitchenitem*, *tableitem*, *householditem*, etc..).

2. un diapason (*tuning fork*) peut effectivement se trouver dans une chambre mais de manière plus générale dans un studio de musique ou une salle de répétition. De même, il est plus probable de voir un verre en plastique (pour les brosses à dent) qu'un verre d'eau dans une salle de bain.

A notre connaissance, cela est un point fondamental dont souffrent la majorité des ontologies, même celles adaptées à des applications robotiques [PJ12] [TB17]. Le fait qu'une fourchette soit généralement trouvée dans une cuisine n'est qu'une conséquence d'une cause plus profonde. En effet, la cuisine est le lieu où l'on cuisine (!) et où l'on mange. Hors, ces deux actions sont fortement liées à la notion de nourriture (*food* en anglais). C'est donc le **contexte commun** qui relie les deux concepts de cuisine et de fourchette. En raisonnant de la sorte, on peut donc correctement inférer la possible présence de fourchettes dans une salle de restaurant, chose que n'implique pas le concept de *kitchenitem*. On peut observer cela à la figure 3.7 qui présente le sous-graphe associé au concept *fork* obtenu par notre méthode détaillée plus loin dans ce chapitre. Sans rentrer dans les détails du graphe, on voit clairement apparaître ici le contexte de nourriture.

Notre modèle de base de connaissance ainsi que sa génération sont motivés par les points soulevés précédemment. Nous n'avons pas trouvé d'ontologie réellement adaptée à notre application, qu'elle soit créée manuellement ou de manière automatique. En effet, on s'intéresse principalement à une description physique et fonctionnelle d'objets. De plus, les ontologies existantes se concentrent sur une hiérarchisation "verticale" des concepts (*fork is a kitchenitem*) alors que l'information importante est contenue dans les contextes sous-jacents : ce sont les nœuds intermédiaires reliant ces deux concepts ie *fork relateTo food, kitchen relateTo food*.

Nous allons maintenant définir la structure de notre ontologie et décrire l'algorithme de construction de celle-ci.

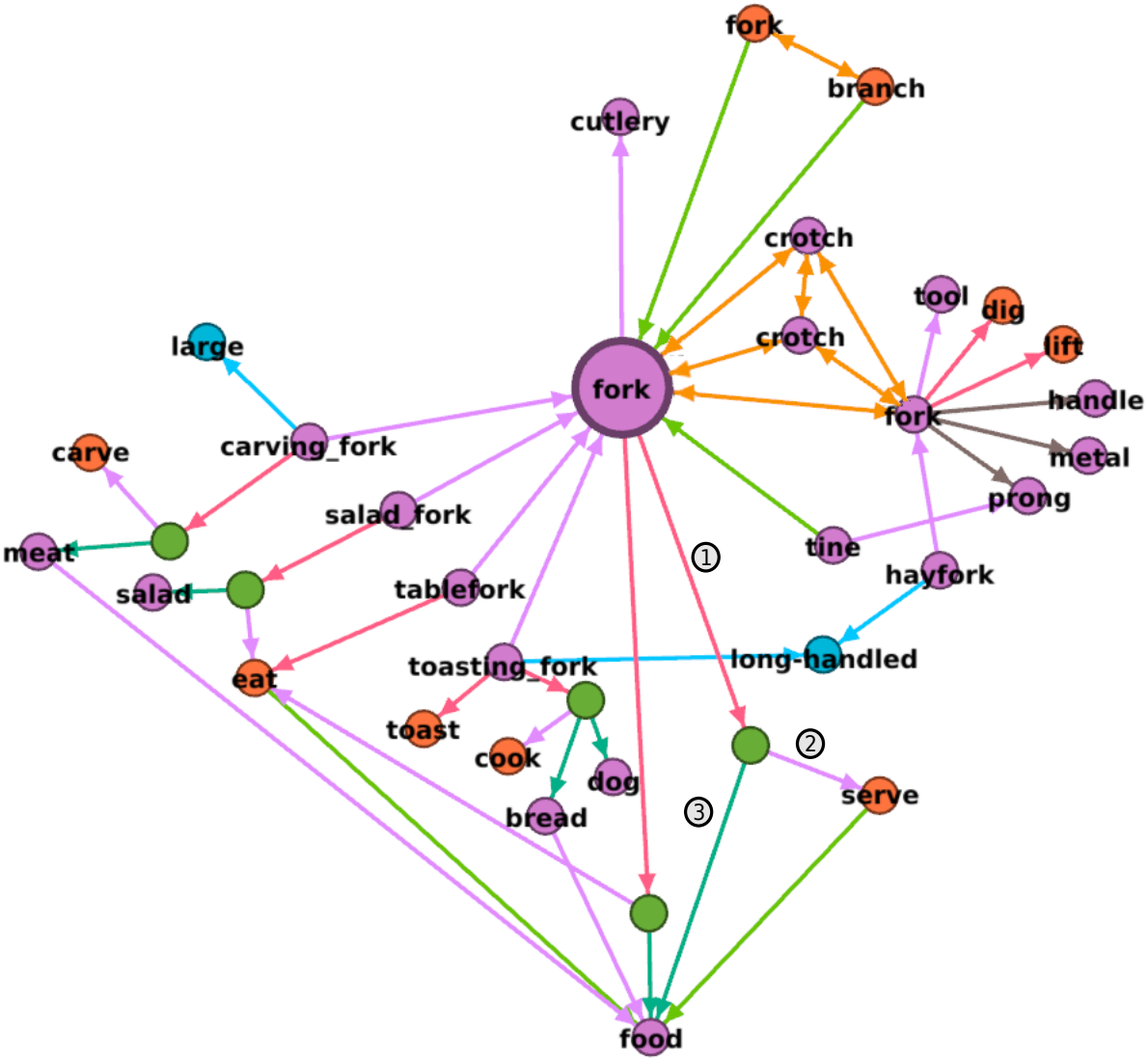


FIGURE 3.7 – Sous-graphe de l'ontologie générée par notre méthode à partir du concept *fork*. Plus précisément, c'est l'ensemble des nœuds à distance 1 (en ne comptant pas les nœuds facteurs) du concept *fork* et de ces hyponymes (*carving_fork*, *salad_fork*, *table_fork*, *toasting_fork*). Les mots associés à chaque nœud correspondent au premier mot de l'ensemble des synonymes donnés par WordNet.

3.2 Définition du graphe de connaissance

3.2.1 Une ontologie basée objets à partir de définitions

Avant de rentrer dans la formalisation du graphe de connaissance, commençons par expliquer les raisons qui nous ont menées à celle-ci. Il a été mis en évidence le fait, à travers l'ontologie

NELL, qu'un apprentissage automatique à partir des différentes sources d'Internet (Wikipedia,...) est une tâche difficile. Il est donc naturel de se demander s'il n'y aurait pas des sources d'information suffisamment simples et riches permettant une représentation ni trop détaillée ni trop grossière de l'environnement. Observons maintenant plus précisément les définitions liées à des objets physiques, tels que ceux qu'un robot peut rencontrer dans un environnement humain. On remarque qu'un objet est principalement défini selon deux axes : son **apparence physique** (visuelle et structurelle) et/ou sa **fonction**. Par exemple, le *Cambridge Dictionary* [3] nous donne pour le mot *fork* :

- Premier sens (fourchette) : *a small object with three or four points and a handle, that you use to pick up food and eat with*
- Second sens (fourche) : *a tool with a long handle and three or four points, used for digging and breaking soil into pieces*

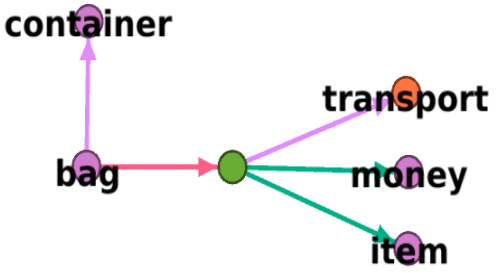
Le concept de fourchette est ici défini par son apparence (*small, with three or four points and a handle*) et par sa fonction (*used to pick up food and eat*). Il en est de même pour la fourche. Ceci est le point central qui guide notre choix de formalisation et la méthode de génération d'ontologie.

Nous allons maintenant décrire notre ontologie en se basant sur la figure 3.7 et la table 3.1. Comme n'importe quel graphe, notre ontologie est composée d'un ensemble de nœuds reliés par des arêtes représentant différents type de relations : hyperonymie (*isA*, violet), méronymie (*hasA*, marron), propriété (*prop*, bleu), usage (*usedFor*, rouge), objet sur lequel s'exerce l'usage (*on*, vert foncé), relation de type inconnu (*linkedTo*, vert clair) et homonymie (*homonym*, orange).

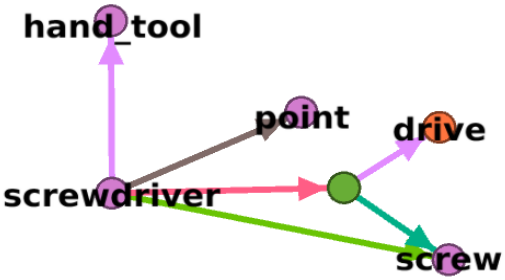
Deux principaux types de nœuds sont à distinguer : les nœuds facteurs (en vert) et les nœuds concepts (le reste). Ces derniers, comme leur nom l'indique, représentent différentes catégories de concepts :

- liés à des objets eg. *fork*, *food* (violet)
- liés à des propriétés eg. *large* (bleu)
- liés à des actions eg. *serve*, *eat* (orange)

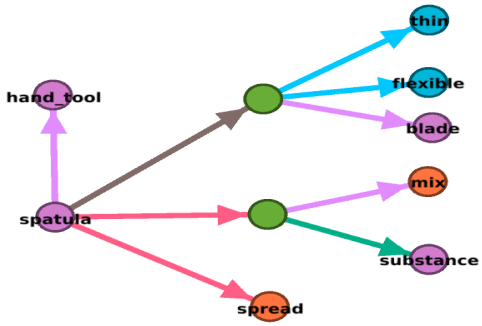
3. https ://dictionary.cambridge.org/dictionary/english/fork



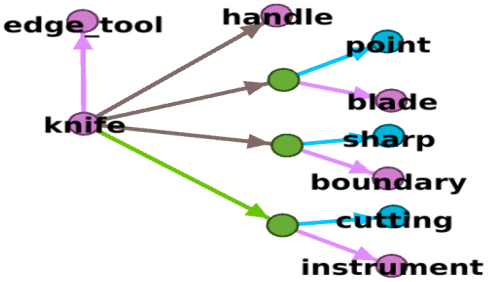
(a) handbag-n02774152



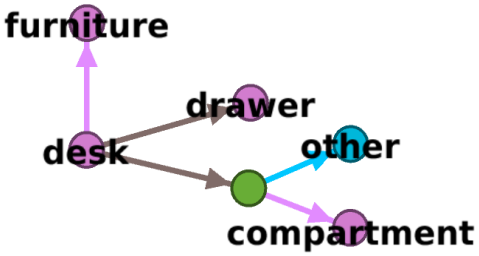
(b) screwdriver-n04154565



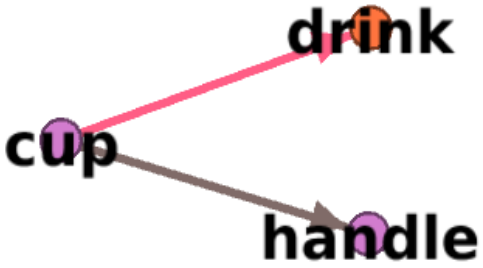
(c) spatula-n04269944



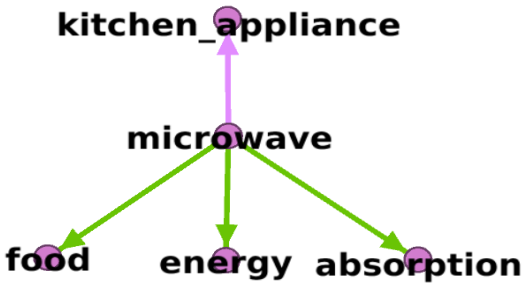
(d) knife-n03623556



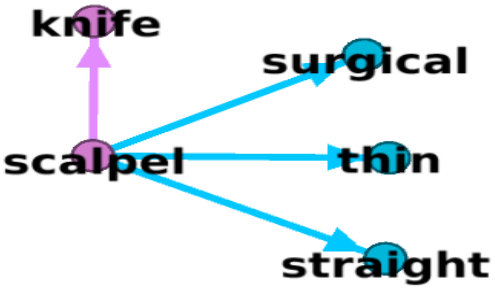
(e) desk-n03179701



(f) cup-n03147509

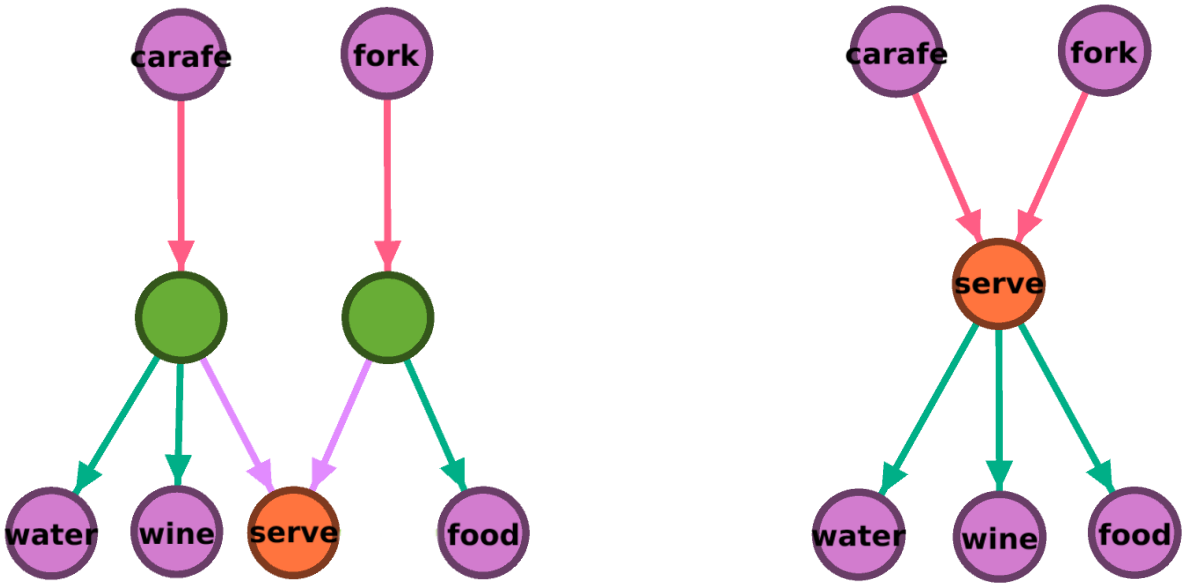


(g) microwave-n03761084



(h) scalpel-n02774152

FIGURE 3.8 – Exemples de sous-graphes construits à partir de définitions selon notre méthode. Les liens *isA* directement issus de WordNet ne sont pas affichés.



(a) Avec noeuds facteur (b) Sans noeuds facteur

FIGURE 3.9 – Illustration d'un graphe ontologique avec et sans utilisation de nœud facteur d'action $V_K^{F,A}$. Dans le cas (b), on voit clairement qu'il n'est pas possible de savoir à quel sous-ensemble d'objets s'applique (ou non) l'action *serve* dans le cas d'une fourchette (*fork*) : cette ontologie considère donc qu'une fourchette peut potentiellement servir du vin.

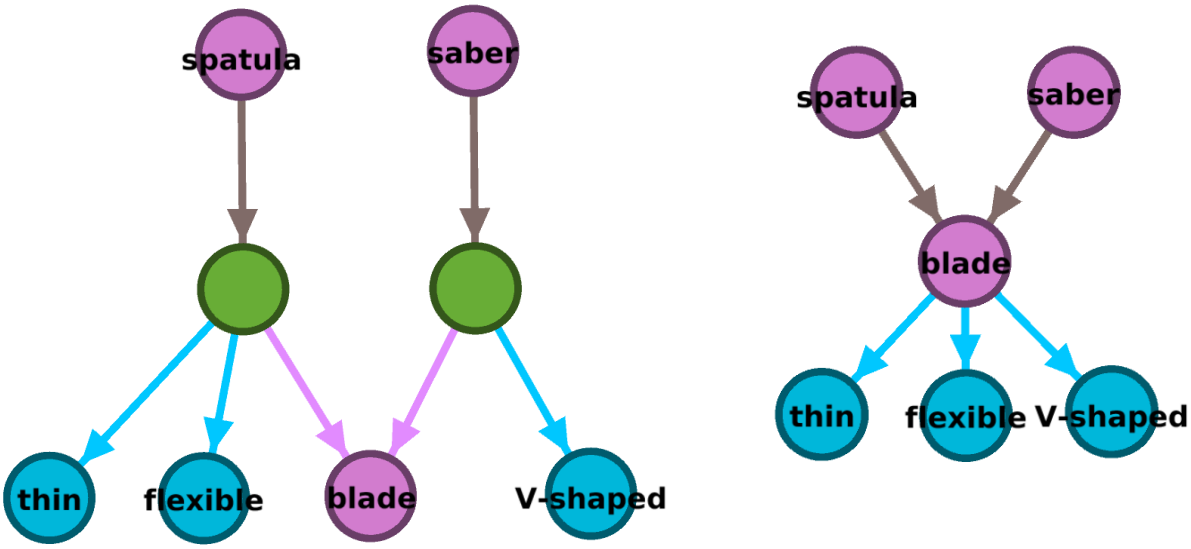
Les nœuds facteurs, au contraire, ne sont qu'un intermédiaire permettant de combiner un ensemble de concepts. On peut voir ainsi que le fait *fork is used to serve food* se traduit dans notre ontologie par trois relations (notées respectivement 1, 2 et 3 sur la figure 3.7) :

$$usedFor(fork, noeudFacteur),\ isA(noeudFacteur, serve),\ on(noeudFacteur, food)$$

Un tel nœud facteur, qui représente une **spécialisation** d'une action, est donc un nœud facteur d'action. Sans cet intermédiaire, on ne pourrait pas représenter cette relation. En effet, considérons la figure 3.9a. Supposons que l'on supprime le nœud facteur reliant *fork* et *serve* : on devra donc relier directement *serve* à *food* et *fork* à *serve* de tel sorte qu'on aura les relations (figure 3.9b)

$$on(serve, food),\ usedFor(fork, serve)$$

Le problème est que si *serve* est par exemple également relié à d'autres usages comme *on(serve, wine)*, on ne peut plus déterminer exactement sur quels objets peut agir *fork*.



(a) Avec noeuds facteur (b) Sans noeuds facteur

FIGURE 3.10 – Illustration d'un graphe ontologique avec et sans utilisation de nœud facteur d'objet $V_K^{F,O}$

Un autre exemple de l'utilité des nœuds facteurs peut se voir à la figure 3.10. La relation *spatula has a thin flexible blade* est représentée par quatre relations :

$$hasA(spatula, noeudFacteur),\ isA(noeudFacteur, blade),$$
$$prop(noeudFacteur, thin),\ prop(noeudFacteur, flexible)$$

Ce nœud facteur représente une **spécialisation** d'un concept que l'on nomme donc nœud facteur d'objet. Là encore, ce type de relation ne pourrait être représenté sans l'utilisation de nœud facteur : les propriétés *thin* et *flexible* ne sont pas intrinsèques à *blade*. Autrement dit, on ne peut pas relier directement ces concepts par des relations comme *prop(blade, thin)* (figure 3.10b).

3.2.2 Formalisation

Notre ontologie est représentée par un graphe de facteur mixte $G_K = \{V_K, E_K\}$ avec V_K l'ensemble des nœuds correspondant aux concepts (eg. *fork*) et E_K l'ensemble des arêtes correspondant aux relations sémantiques (eg. *isA(A,B)*).

On décompose l'ensemble des nœuds V_K de notre ontologie selon deux catégories :

– Les nœuds concepts V_K^C eg. *fork*, *large*, *eat*.

– Les nœuds facteurs V_K^F correspondant à une factorisation de concepts. Par exemple, la repré-

sentation de *spatula has a thin blade* fait intervenir un nœud facteur équivalent à un concept factorisé *thin blade*.

On a ainsi

$$\begin{cases} V_K = V_K^C \cup V_K^F \\ V_K^C \cap V_K^F = \emptyset \end{cases} \tag{3.2}$$

Les nœuds se caractérisent également par le type de concept qu'ils représentent :

- Les nœuds liés à un objet (*fork*) ou une propriété (*large*) V_K^O.
- Les nœuds liés à une action (*eat*) V_K^A.

On peut donc définir par intersection une décomposition finale :

$$V_K = V_K^{C,O} \cup V_K^{C,A} \cup V_K^{F,O} \cup V_K^{F,A} \tag{3.3}$$

avec pour $t \in \{C, F\}$:

- Les nœuds concepts (resp. facteurs) liés à un objet ou une propriété $V_K^{t,O} = V_K^t \cap V_K^O$
- Les nœuds concepts (resp. facteurs) liés à une action $V_K^{t,A} = V_K^t \cap V_K^A$

A chacun de ces nœuds est associé un ensemble de mots (synonymes) et un identifiant unique, qui est le WordnetId [Mil95] du mot correspondant. Nous reviendrons sur ce point dans la section 3.3 concernant la construction du graphe.

Nous séparons les concepts V_K^O liés à une description physique et les concepts V_K^A liés à des fonctions car, comme évoqué en introduction, un objet est défini par son apparence physique et sa fonction. Voyons maintenant l'utilisation des nœuds facteurs.

A notre connaissance, aucune autre ontologie n'en fait usage : les nœuds y sont tous des concepts bien définis. De plus, la plupart des ontologies souffrent d'un surapprentissage (ou surconceptualisation). Cela se traduit par la présence de concepts trop détaillés qui ne sont en fait qu'une **combinaison** d'autres concepts. On le voit clairement sur la figure 3.6 avec des concepts tels que *MeasuringCup-Small* pour KnowRob et *sippy_cup_with_removable_lid_and_two_handles* pour NELL. Notre ontologie remplace ces combinaisons de concepts par des nœuds facteur, ce qui réduit de manière importante le nombre de concepts afin de n'en garder que les plus élémentaires. La table 3.4a montre qu'environs 23% des nœuds de notre graphe de connaissance sont des nœuds facteurs. Cela indique une forte réduction du nombre de concepts par rapport à une structure ontologique classique.

Enfin, une dernière distinction doit être faite selon la nature du mot associé à un concept comme indiqué dans la table 3.1a. L'ensemble des arêtes E_K est également segmenté en fonction du type

de relation représenté, actuellement au nombre de sept. Chacune de ces relations est définie par une fonction R :

$$R : \begin{cases} V_{depart} \times V_{arrive} \times \mathbb{N}^* \to 0, 1 \\ u, v, n \mapsto R(u, v, n) \end{cases} \tag{3.4}$$

avec V_{depart}, $V_{arrive} \subset V_K$. n représente l'arité de la relation lorsque cela a du sens et vaut 1 par défaut. Les types de relation sont :

- **isA** : Relation d'hyperonymie (sous-concept). Exemple : *isA(fork, cutlery)*.
  Elle est présente dans toutes les ontologies vues précédemment (*generalizations* dans NELL ou *subClassOf* pour OpenCyc/KnownRob). Elle permet de hiérarchiser l'ontologie en partant de concepts spécialisés pour remonter à des concepts de plus en plus abstraits.
- **hasA** : Relation de méronymie (partie). Exemple : *hasA(fork, handle)*.
  Elle permet de définir un concept par ses parties, qu'elles soient visibles (*hasA(car, wheel)*) ou non (*hasA(car, motor)*).
- **prop** : Relation de propriété, généralement exprimée par un adjectif.
  Exemple : *prop(banana, yellow)*.
- **usedFor** : Relation d'usage exprimée par un verbe d'action. Exemple : *usedFor(fork,eat)*.
- **on** : Relation spécifiant l'objet sur lequel s'effectue une action. Exemple : *on(eat, food)*.
- **linkedTo** : Relation entre concepts dont le type est inconnu. Elle peut correspondre soit à une relation non (encore) représentée dans notre ontologie, soit à une incapacité de notre algorithme de génération d'inférer le bon type. Exemple : *linkedTo(chemistry_lab, chemistry)*.
- **homonym** : Relation liant les homonymes, c'est-à-dire les nœuds qui sont définis par un ensemble de synonymes d'intersection non nulle. Exemple : *homonym(alchemy, chemistry)*.
  C'est la seule relation non dirigée du graphe.

Le dernier type, **homonym**, est très important pour faire face au problème de *désambiguïsation lexicale*. Il est en effet difficile de savoir dans une phrase à quel sens possible fait référence un mot. Comme on le verra dans la section 3.3, notre méthode se base sur l'analyse de définition. Nos relations obtenues après analyse sont donc à homonymie près.

Nature	Type	Légende
Nom	$V_K^{C,O}$	
Adjectif	$V_K^{C,O}$	
Verbe	$V_K^{C,A}$	
Facteur	V_K^{F}	

(a) Ensemble de définition des nœuds en fonction de la nature des mots associés

Relation	V_{depart}	V_{arrive}	Légende
isA	V_K	V_K	
hasA	V_K^O	V_K^O	
prop	V_K^O	$V_K^{C,O}$	
usedFor	$V_K^{C,O}$	V_K^A	
on	$V_K^{F,A}$	V_K^O	
linkedTo	$V_K^{C,A}$	$V_K^{C,O}$	
homonym	V_K^C	V_K^C	

(b) Ensemble de définition de chacune des relations de notre ontologie

TABLE 3.1 – Définition des ensembles correspondant aux différents éléments de notre ontologie. Sauf mention contraire, la légende s'applique à toutes les représentations graphiques de l'ontologie présentée dans ce manuscrit

3.3 Construction du graphe de connaissance

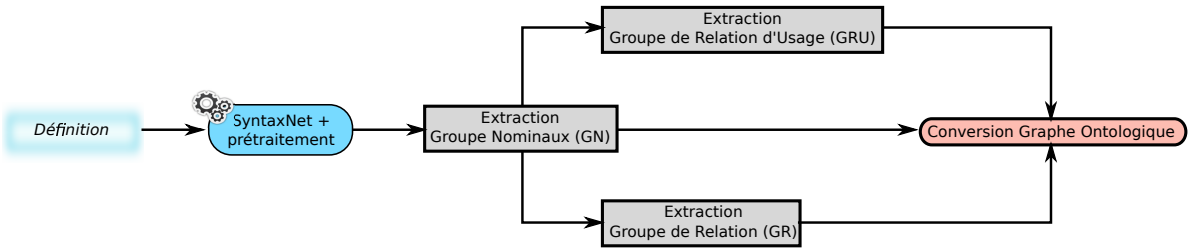


FIGURE 3.11 – Schéma de construction du graphe de connaissance

Maintenant que le graphe de connaissance a été défini, voyons en détail sa construction. Elle suit le schéma présenté à la figure 3.11. L'ontologie est initialisée par celle de WordNet et la méthode d'extraction de relations sémantiques se base sur ses définitions. Cette partie va donc tout d'abord présenter les raisons du choix de WordNet comme base du modèle de connaissance. On rentrera ensuite dans le détail de l'analyse syntaxique des définitions et comment elles sont transformées en un ensemble de relations comme défini en section 3.2.2. On s'attardera enfin sur une évaluation quantitative et qualitative de l'ontologie ainsi créée.

3.3.1 Pourquoi WordNet ?

Nous allons ici expliquer pourquoi notre choix d'ontologie initiale s'est porté sur WordNet. Ces principaux avantages sont :

- de contenir des relations simples (synonymie, hypero/hyponymie, méronymie), facilement exploitables dans le contexte d'application robotique. Les concepts sont hiérarchisés des plus abstraits au plus concrets.
- de gérer nativement les problèmes d'homonymie en ne considérant que des synsets uniques, regroupant les mots ayant un sens commun.
- de classifier les synsets par nature et par sous-catégorie de sens (principalement pour les noms et les verbes).
- de fournir une définition associée à chaque synset.

Cependant, l'ontologie WordNet doit être complétée pour construire cette ontologie. En effet, elle ne donne aucune relation de type *usedFor*, *on* nécessaire à notre graphe de connaissance. De plus, beaucoup de liens entre concepts sont manquants : *handle* n'est pas indiqué comme méronyme de *cup*, *saucer* (sous-tasse) n'est pas considéré comme hyponyme de *dish* (vaisselle, plat). Cependant, les définitions associées aux synsets correspondant contiennent ces informations :

"(a cup is) a small open container usually used for drinking ; usually has a handle"

"(a saucer is) a small shallow dish for holding a cup at the table"

Notre ontologie est initialisée à partir d'un sous-ensemble de WordNet approprié à notre application, centré sur les objets et les actions. Les catégories retenues apparaissent soulignées dans la table 3.2. Chaque nœud de notre graphe de connaissance est donc associé à un synset de WordNet et nous convertissons les liens d'hypero/hyponymies en relation *isA*. Dans la suite, nous détaillons l'extraction des relations constituant notre ontologie à partir des définitions contenues dans chaque synset.

3.3.2 Analyse syntaxique de langage naturel

Rappelons le vocabulaire associé à l'analyse syntaxique :

- **Lemme :** En morphologie (linguistique), le lemme est la représentation canonique d'un ensemble de mots ayant le même sens (appelé *lexème*). Par exemple, les mots *{mangeons, mangeais, mange, manger}* forme un lexème dont *mange* est le lemme. De même, *fleur* est le lemme du lexème *{fleur, fleurs}*.
- **POS (part-of-speech) :** La nature d'un mot : nom, verbe, adjectif, déterminant, etc.

4. qui peut être défini comme "de ou se rapportant" à un autre mot

Nature	Catégories
Nom	act, animal, artifact, attribute, body, cognition, communication, event, feeling, food, group, location, motive, object, person, phenomenon, plant, possession, process, quantity, relation, shape, state, substance, time
Adjectif	all,pert(pertainym [4]), ppl (participial)
Adverbe	all
Verbe	body, change, cognition, communication, competition, consumption, contact, creation, emotion, motion, perception, possession, social, stative, weather

TABLE 3.2 – Ensemble des sous catégories de WordNet par nature. Les catégories utilisées dans nos travaux ont été soulignées

- **UPOS (Universal part-of-speech) [PDM11] :** Cela correspond aux principales natures universelles (c'est-à-dire commune à tous les langages) telles que $\mathbb{VERB}$ (verbe) ou $\mathbb{NOUN}$ (nom).
- **XPOS (Language-Specific part-of-speech) [San90] :** Natures spécifiques à un langage en particulier (l'anglais dans nos travaux) comme par exemple les différentes conjugaisons d'un verbe ($\mathbb{VBZ}$ pour la conjugaison à la 3$^{\text{ème}}$ personne, $\mathbb{VBG}$ pour les verbes au gérondif).
- **UDR (Universal Dependency Relation) [DMM08] [DMDS$^+$14] [NDMG$^+$16] :** Représente les liens syntaxiques entre les mots d'une phrase. Par exemple, dans la phrase *Le chat mange la souris*, *chat* est lié à *mange* par une relation $\mathbb{NSUBJ}$ (sujet nominal), et ce verbe est lui-même lié à *souris* par une relation $\mathbb{DOBJ}$ (cod).

Les différents tags existants (**UPOS**, **XPOS** et **UDR**) et leur significations sont référencées en Annexe A.

La construction de notre ontologie se fait à partir de l'analyse syntaxique de chaque définition associée aux synsets de WordNet. Par analyse syntaxique, nous entendons l'inférence de la nature et des relations grammaticales liant les mots dans une phrase. Nous nous basons ensuite sur cette analyse "bas niveau" en suivant une approche dite "bottom-up", c'est-à-dire en regroupant les mots en ensembles qui sont ensuite eux-même regroupés en relations. L'analyse est effectuée par l'analyseur en langage naturel *Parsey McParseface*, qui est un réseau SyntaxNet [AAW$^+$16] pré-entraîné sur un large corpus en langue anglaise.

SyntaxNet est un réseau de neurones basé transition qui permet l'analyse syntaxique d'une phrase. Cette analyse se passe en deux temps : on cherche tout d'abord à estimer la nature des mots

Transition	Stack	Buffer	A
	[ROOT]	[He has good control .]	$\emptyset$
SHIFT	[ROOT He]	[has good control .]	
SHIFT	[ROOT He has]	[good control .]	
LEFT-ARC(nsubj)	[ROOT has]	[good control .]	$A \cup$ nsubj(has,He)
SHIFT	[ROOT has good]	[control .]	
SHIFT	[ROOT has good control]	[.]	
LEFT-ARC(amod)	[ROOT has control]	[.]	$A \cup$amod(control,good)
RIGHT-ARC(dobj)	[ROOT has]	[.]	$A \cup$ dobj(has,control)
...	...	...	...
RIGHT-ARC(root)	[ROOT]	[]	$A \cup$ root(ROOT,has)

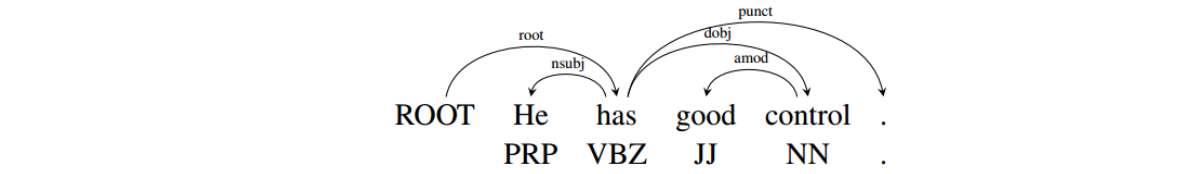


FIGURE 3.12 – Exemple d'analyse de dépendance par un système basé transition. La pile est indiquée en bleu et le tampon en gris. Modifié à partir de [CM14]

en suivant la nomenclature des tags **UPOS** ou **XPOS**. Dans la suite, ce sont ces derniers qui sont utilisés. Une deuxième étape vise ensuite, en utilisant les natures estimées, à inférer les relations de dépendances entre les mots (tags **UDR**).

Les systèmes basés transition [CM14] cherchent séquentiellement la meilleure "transition" en tenant compte de la configuration courante de l'analyse. De manière générale, la configuration d'une analyse $c = (s, b, A)$ d'une phrase consiste en une pile (stack) s, un tampon (buffer) b et un ensemble d'arcs de dépendance (correspond au **UDR**) A . Pour plus de clarté, on se réfère à la figure 3.12. Dans cet exemple, on cherche à analyser la séquence de mots *"He has good control."*, que l'on note formellement $w_1, \ldots, w_n$ (ici $n = 4$). La première colonne correspond aux transitions inférées par le système et les trois autres forment la configuration c. Il faut noter que cet exemple se place directement dans la seconde étape, à savoir que les tags **XPOS** de chaque mot de la séquence sont connus. L'état initial correspond à la première ligne du tableau. On peut y voir que le tampon est rempli par la séquence de mots et que la pile est vide. Bien sûr, l'ensemble des dépendances l'est également. A partir de là, le système va estimer la meilleure transition (et le label associé selon son type) à partir de la configuration actuelle. On note par s_i le $i^{\text{ème}}$ élément à partir du haut de la pile et b_i le $i^{\text{ème}}$ élément du tampon. Il y a trois types de transitions :

– **LEFT-ARC**(l) : Ajout d'une dépendance de type $l \in$ **UDR** de s_1 vers s_2 et on retire s_2

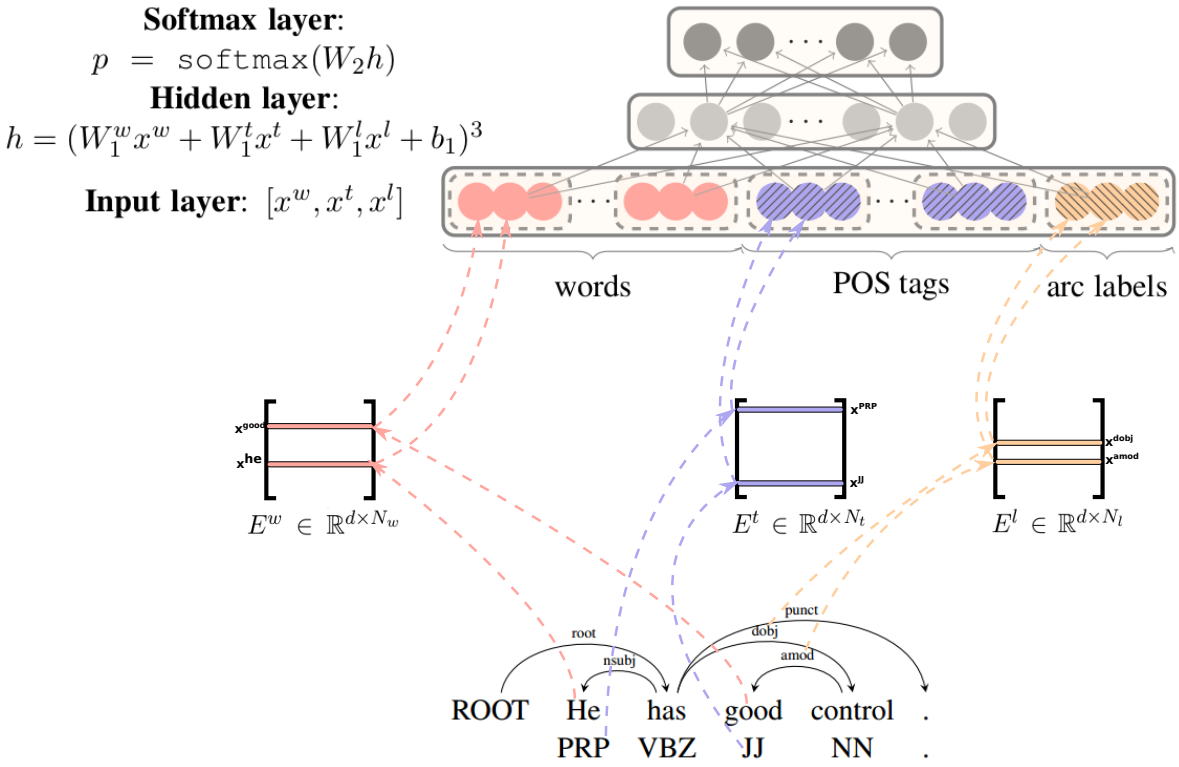


FIGURE 3.13 – Schéma d'un réseau de neurones basé transition. Chaque descripteur (mot w, tags **POS** t et dépendance l) de chaque élément de la phrase d'entrée est représenté sous forme vectorielle via respectivement les matrices E^w, E^t et E^l. Modifié à partir de [CM14]

de la pile. On peut en voir un exemple à la transition entre la 3ème et 4ème ligne du tableau. On a, avant la transition, $s_1 = has$ et $s_2 = He$. Le système estime la transition LEFT-ARC($\mathbb{NSUBJ}$), ce qui enlève He de la pile et ajoute la relation de dépendance à l'ensemble A.

- **RIGHT-ARC**(l) : La même chose en inversant le sens de la relation (ie de s_2 vers s_1) et en retirant s_1 à la place de s_2. Exemple visible à la 8ème ligne du tableau.
- **SHIFT** : Quand aucune dépendance n'est trouvée, on ajoute le premier élément du tampon b_1 à la pile (2ème lignes du tableau).

L'analyse est terminée lorsque la pile et le tampon sont vides.

Voyons maintenant comment les transitions et les labels sont inférés en pratique. Pour cela, on commence par représenter les mots mais également les natures de mots **XPOS** ainsi que les dépendances **UDR** sous forme vectorielle (bas de la figure 3.13). Ces représentations sont apprises en même temps que les poids du réseau de neurones (haut de la figure 3.13) dont elles sont l'entrée. Chaque entrée (mot, nature **XPOS** et dépendance **UDR**) est un vecteur avec une coordonnée à 1 et

le reste à zéro. Dans le cas de mots, chaque coordonnée représente un mot du dictionnaire. Dans le cas de **POS**, chaque coordonnée correspond à une valeur de tag **XPOS** (cf Table A.2 en annexe) et de même pour les relations de dépendances (tag **UDR**). Dans le schéma, d est la dimension des vecteurs associés à chaque descripteur (mot, **POS** ou relation de dépendance). N_w est la taille du dictionnaire (nombre de mots connus), N_t le nombre de tags **XPOS** et N_l le nombre de tags **UDR**.

L'ensemble des types de transitions et de labels (tags **UDR**) étant discret, on se retrouve avec un problème de classification : la transition (et le label) inférée est donc celle avec la probabilité la plus élevée en sortie du réseau de neurone. Le vecteur d'entrée est une concaténation des représentations vectorielles des mots, des tags **POS** et des tags **UDR** pour des ensembles de groupes (allant de 1 à 3) de mots.

Jusque là, il a été supposé que les **POS** étaient connus. En pratique, ils ne le sont pas. SyntaxNet estime le **POS** d'un mot w_i de la phrase à partir d'un autre réseau de neurones en utilisant les représentations vectorielles de ce mot et de ces voisins. Les représentations vectorielles des tags **POS** précédemment inférées pour $j < i$ sont également utilisées.

La sortie de SyntaxNet suit le format **coNLL-X** [BM06], qui va constituer l'entrée de notre algorithme. ce format contient différentes entrées dont certaines sont dépendantes des applications. Lorsque qu'une entrée est inutilisée ou non déterminée, elle est représentée par un signe _. La figure 3.14 est un exemple de phrase décrite en format coNLL-X. Dans la suite, nous ignorons les colonnes liées aux descriptions spécifiques (colonne 6), les relations de dépendances projectives (colonne 10) et les index correspondants (colonne 9). L'index de dépendance (*head* en anglais) est l'index du mot auquel est dépendant le mot courant par la relation spécifiée par son tag **UDR**. On lit ainsi que *food* est lié à *eating* par une relation de complément d'objet direct. On rappelle que les domaines de définition des différents tags **UPOS**, **XPOS** et **UDR** sont donnés en annexe A. Notons que SyntaxNet ne donne pas les lemmes de chaque mot. Nous les ajoutons par la suite en utilisant le processeur morphologique Morphy [5] de WordNet.

3.3.3 Construction automatique d'ontologie à partir de définitions

Cette section va détailler la méthodologie employée pour l'extraction de relations. Elle se base sur la structure générale des définitions WordNet par reconnaissance de motifs (pattern matching). L'idée d'extraire des relations sémantiques à partir des définitions WordNet n'est pas nouvelle. Novischi [Nov02] cherche à désambiguïser le sens des mots dans les définitions de WordNet, en

5. https ://wordnet.princeton.edu/documentation/morph3wn

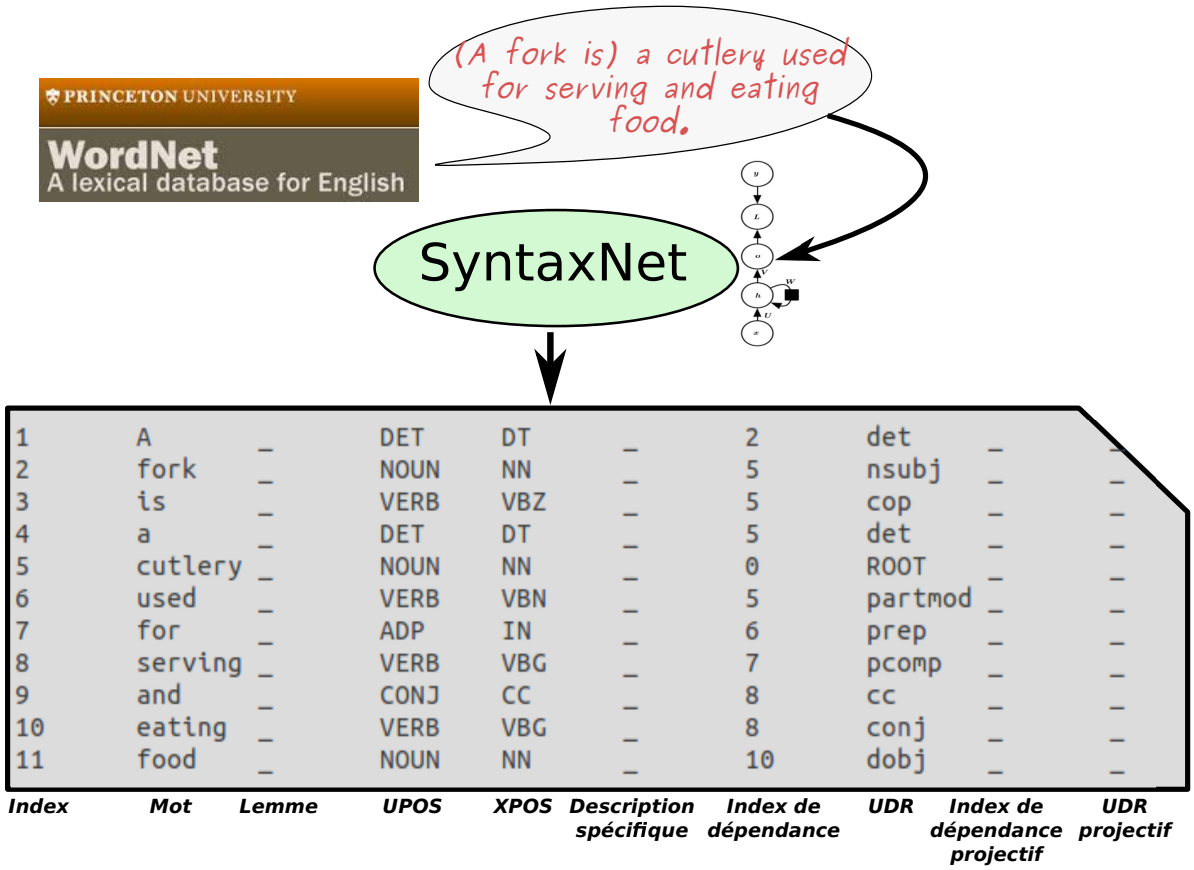


FIGURE 3.14 – Exemple de résultat d'analyse syntaxique obtenue au format coNLL-X à partir de la définition WordNet de *fork*

privilégiant une trés haute précision [6] au détriment de la couverture [7]. Il construit des motifs en considérant des mots successifs dans l'ensemble des définitions, qui seront ensuite fusionnés ou filtrés selon leur occurrence et certaines heuristiques. La désambiguïsation se fait ensuite comme suit :

- on recherche le meilleur motif correspondant à chaque mot relatif à la définition analysée.
- on assigne alors au mot le même sens que celui qu'il porte lorsqu'il apparaît dans le motif sélectionné.

La méthode présentée ne cherche donc pas à extraire directement des relations sémantiques mais montre la difficulté liée aux homonymes : la méthode obtient une précision de 98% pour une couverture de seulement 6.6%.

6. Définie comme le pourcentage de résultats corrects
7. Pourcentage de mots sur lesquelles la méthode a pu s'appliquer

DeBoni [DBM02] propose d'extraire automatiquement des relations téliques [8] du type *usedFor*. Ils se basent sur un ensemble de motifs correspondant aux relations recherchées. Un mot O issu d'une relation extraite à partir d'une définition D d'un mot O' est ensuite désambiguïsé en utilisant une mesure de similarité. Elle est basée sur la cooccurrence des termes dans la définition D et dans l'ensemble constitué des définitions de O et de ses hypero/hyponymes. Leur résultat sur un petit échantillon (10%) de WordNet donne une précision de 60%. Bracewell [BRK06] décrit également une approche sur la reconnaissance de motifs dans le but d'extraire la fonction d'agent (eg. *un médecin soigne*). Ils ne considèrent cependant pas le problème de la désambiguïsation.

La plupart des approches plus récentes essaient d'exploiter des sources d'information plus riches mais moins structurées, comme par exemple Wikipedia. Cependant, comme l'exemple de NELL le montre, un bon compromis entre la richesse des données à analyser et la pertinence des relations extraites est difficile à obtenir. Tout comme [Nov02], nous privilégions la précision et la pertinence des relations sur la couverture. Les avantages de la reconnaissance de motifs sur les méthodes de machine learning sont dans notre cas :

- La non-nécessité d'avoir des données d'entrainement.
- Une meilleur interprétabilité des résultats.
- La possibilité d'amélioration incrémentale de l'algorithme.

Notre approche se différencie de [DBM02] [Nov02] principalement par le fait que nous ne cherchons pas directement à désambiguïser les termes trouvés dans les relations. En effet, ce problème est complexe et les erreurs dues à l'assignement d'un mauvais sens sont critiques : elles vont fortement corrompre le contexte des concepts de notre ontologie. C'est pourquoi nous avons choisi de conserver les liens d'homonymies afin de désambiguïser "à la volée", en fonction du contexte courant du robot. De plus, dans notre cas les relations de dépendance (**UDR**) entre les mots sont également exploitées ce qui permet le développement de nouvelles heuristiques.

Définitions générales

Nous définissons l'ensemble des tags **UPOS**, **XPOS** et **UDR** par respectivement T_U, T_X et T_D. L'ensemble des mots W est défini par

$$W = T_U \times T_X \times T_D \times H,\ H \subset \mathbb{N} \tag{3.5}$$

8. verbe ou syntagme verbal présentant une action (ou un événement) comme mené à son terme.

Chaque mot est un 4-uplet. H est un ensemble d'indices tel que, pour $\mathbf{w} = [u\ x\ d\ h]^T$, h correspond à l'indice de dépendance (*head*) du mot $\mathbf{w}$. De là, nous pouvons définir des sous-ensembles de mots :

$$\forall\, V \subset T_U \times T_X \times T_D,\ W_V = \left\{ \mathbf{w} = [u\ x\ d\ h]^T \in W \;\middle|\; u \in U \cap V,\ x \in X \cap V,\ d \in D \cap V \right\} \tag{3.6}$$

Par exemple, $W_{\{\text{NOUN,NSUBJ,IOBJ}\}}$ représente le sous-ensemble de mots correspondant à des noms sujets ou compléments d'objet indirect.

Une phrase (définition dans notre cas) S de taille n se définit par la matrice suivante :

$$S = [\mathbf{w}_1 \dots \mathbf{w}_n] \in W^n \tag{3.7}$$

Nous passons d'une phrase vue comme chaîne de caractères à la représentation ci-dessus par l'analyse syntaxique de SyntaxNet. L'index i des mots dans la formule 3.7 correspond à l'index du format coNLL-X (colonne 1 de la figure 3.14). L'approche est *bottom-up* : les mots sont agrégés en groupes nominaux (*GN*), qui sont eux-même regroupés en groupes de relations (*GR*) ou en groupes de relations d'usage (GRU). La figure 3.15 donne un aperçu global de notre méthode.

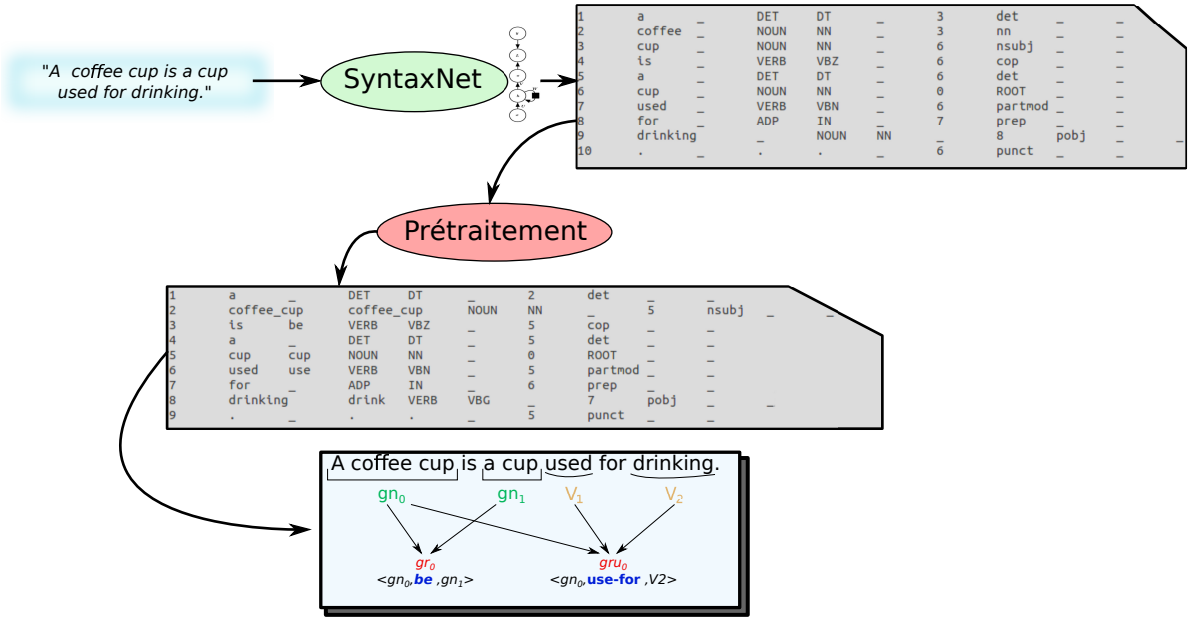


FIGURE 3.15 – Schéma de fonctionnement de notre méthode d'extraction de relations sémantiques

Prétraitements

L'API de WordNet permet d'obtenir les descriptions attachées à chacun de ces synsets sous la forme *définition ;"phrase_exemple"*. La première étape de prétraitement consiste donc à ne récupérer que la définition. Nous retirons également les parties de définitions entre parenthèses comme par exemple :

stinger : a sharp organ of offense or defense (as of a wasp or stingray or scorpion) often connected with a poison gland.

Afin de faciliter l'analyse par SyntaxNet, nous remplaçons un ensemble de motifs par des expressions similaires plus simples (alias) ou nous les retirons si possible. Actuellement, ces motifs sont

$$\{kind\ /piece\ /type\ /variant\ of\}$$

Une fois l'analyse syntaxique de SyntaxNet obtenue, nous opérons une deuxième phase de traitement :

- Les virgules liées à une conjonction de coordination sont remplacées par une conjonction identique.
- Certains verbes en *-ing*, considérés à tort comme étant des noms, sont alors modifiés en verbe au gérondif.

Une fois ces prétraitements effectués, nous pouvons passer à l'analyse des définitions en commençant par la décomposition en groupes nominaux.

Décomposition en groupes nominaux (*gn*)

Nous définissons un groupe nominal *gn* comme étant un nom accompagné éventuellement d'un déterminant et d'un ensemble le complémentant. Nous considérons comme complément du nom les adjectifs, les modificateurs numériques ainsi que les verbes au participe passé employés comme adjectif. Formellement,

$$gn = \left\{ \mathbf{w}_{det} \in W_{\mathtt{DET}},\ \mathbf{w}_{nom} \in W_{\mathtt{NOUN}},\ W_{comp} \subset W_{\{\mathtt{ADJ},\mathtt{NUM}\}} \cup W_{\{\mathtt{VBN},\mathtt{AMOD}\}} \right\} \tag{3.8}$$

Dans la suite, nous utiliserons les *gn* en se référant indifféremment aux mots ou aux index (eg. idx_{det} pour w_{det}) correspondant dans la phrase. Ainsi, sur la figure 3.14, on peut définir le *GN* associé à

Algorithme 1 Extraction de groupes nominaux (GN)

Entrée: Phrase $S = [\mathbf{w}_1 \dots \mathbf{w}_n]^T$, $\mathbf{w}_i = [u_i\ x_i\ d_i\ h_i]^T \in W$
Sortie: Ensemble de GN représenté par les index des mots correspondant dans la phrase S, Ensembles $C^{\wedge}_{GN}$ et $C^{\vee}_{GN}$

function EXTRACTIONGROUPENOMINAUX(S)
  $GN \leftarrow \emptyset,\ C^{\wedge}_{GN} \leftarrow \emptyset, C^{\vee}_{GN} \leftarrow \emptyset$
  for $i \in [\![1, n]\!]$ t.q. $u_i = \mathbb{NOUN}$ **do**
    $i_{nom} \leftarrow i,\ i_{det} \leftarrow \emptyset,\ I_{comp} \leftarrow \emptyset$
    if $\exists j$ t.q. $u_j = \mathbb{DET} \wedge h_j = i$ **then**
      $i_{det} \leftarrow j$
    end if
    for $k \in [\![1, i]\!]$ t.q. $u_k \in \{\mathbb{ADJ}, \mathbb{NUM}\} \vee (x_k = \mathbb{VBN} \wedge d_k = \mathbb{AMOD})$ **do**
      if $h_k = i$ **then**
        $I_{comp} \leftarrow k$
      else if $d_k = \mathbb{CONJ} \wedge \exists j$ t.q. $w_j \in \{"and", "or"\} \wedge h_j \in I_{comp}$ **then**
        $gn_{constr} \leftarrow \{i_{nom}, i_{det}, \{k\}\}$
        if $w_j =$ "and" **then**
          $C_{GN} \leftarrow C^{\wedge}_{GN}$
        else
          $C_{GN} \leftarrow C^{\vee}_{GN}$
        end if
        if $\exists c \in C_{GN},\ \exists gn_c \in c$ t.q. $idx_{nom}(gn_c) = i_{nom}$ **then**
          $c \leftarrow gn_{constr}$
        else
          $gn \leftarrow gn' \in GN$ t.q. $idx_{nom}(gn') = i_{nom}$
          $C_{GN} \leftarrow \{gn, gn_{constr}\}$
        end if
        $GN \leftarrow gn_{constr}$
      end if
    end for
    if $d_i = \mathbb{CONJ} \wedge u_{h_i} = \mathbb{NOUN}$ **then**
      $gn_{conj} \leftarrow gn' = (i'_{nom},\ i'_{det},\ I'_{comp} = \{i'_{c,1},\ \dots, i'_{c,|I'_{comp}|}\}) \in GN$ t.q. $i'_{nom} = h_i$
      if $|I'_{comp}| > |I_{comp}|$ **then**
        $I_{comp} \leftarrow \{\{i'_c\}_1, \dots, \{i'_c\}_{|I'_{comp}|-|I_{comp}|}\}$
      end if
    end if
    $gn \leftarrow \{i_{nom}, i_{det}, I_{comp}\}$
    $GN \leftarrow gn$
  end for
  for $i \in [\![1, n]\!]$ t.q. $u_i = \mathbb{ADJ} \wedge d_i = \mathbb{ROOT}$ **do**
    $GN \leftarrow \{\emptyset, \emptyset, \{i\}\}$ ▷ cas d'un adjectif attribut du sujet
  end for
end function

fork par $gn = \{1,\ 2,\ \emptyset\}$ ou $gn = \{a, fork, \emptyset\}$.

On notera $GN(S)$ l'ensemble des groupes nominaux d'une phrase S. $C^{\wedge}_{GN(S)}$(resp. $C^{\vee}_{GN(S)}$) est un ensemble de sous-ensembles de $GN(S)$ contraints par une relation de conjonction et (resp. ou). Si l'on considère la figure 3.16, on aura $GN(S) = \{gn_1, \ldots, gn_4\}$ avec :

$$\begin{aligned} gn_1 &= \{a, banana, \emptyset\} \\ gn_2 &= \{a, fruit, \{red\}\} \\ gn_3 &= \{a, fruit, \{yellow\}\} \\ gn_4 &= \{a, fruit, \{green\}\} \end{aligned} \tag{3.9}$$

On en déduit

$$\begin{aligned} C^{\vee}_{GN(S)} &= \{\{gn_2,\ gn_3,\ gn_4\}\} \\ C^{\wedge}_{GN(S)} &= \emptyset \end{aligned} \tag{3.10}$$

où $\emptyset$ désigne l'ensemble vide.

L'algorithme 1 décrit l'extraction des groupes nominaux. On crée un *gn* par nom dans la phrase. Les déterminants et compléments sont ensuite ajoutés à ce *gn* en cherchant les mots pointant vers ce nom avec les tags **UPOS/XPOS** adéquats : les lignes 5-7 cherche le déterminant associé (dont le tag **UPOS** vaut $\mathbb{DET}$) et les lignes 8-10 s'occupent des compléments qui peuvent être

- des adjectifs (tag **UPOS** valant $\mathbb{ADJ}$)
- des nombres (tag **UPOS** valant $\mathbb{NUM}$)
- des verbes au participe passé (tag **XPOS** valant $\mathbb{VBN}$) qui modifient le sens du nom (relation de dépendance **UDR** valant $\mathbb{AMOD}$)

On construit ensuite (ligne 11-25 et 28-35) les ensembles $C^{\wedge}_{GN(\mathbf{s})}$, $C^{\vee}_{GN(\mathbf{s})}$ si des adjectifs sont en relation de conjonction (tag **UDR** valant $\mathbb{CONJ}$) entre eux via une conjonction de coordination et ou ou. En particulier, les lignes 11-25 concernent la conjonction entre les adjectifs comme par exemple *La tasse est rouge et petite* alors que les lignes 28-35 gèrent la distribution d'adjectif entre noms reliés par une conjonction. Ainsi, *Light red cups and green glasses* est équivalent à *Light red cups and light green glasses*. L'adjectif *light* se distribue donc aux deux *gn* $\{\emptyset, cups, \{red\}\}$ et $\{\emptyset, glasses, \{green\}\}$. Il faut noter que les alternatives du type *un objet est un *** ou un **** ne sont actuellement pas considérées dans notre ontologie, bien que pris en compte dans l'extraction des *gn* puis des groupes de relations *gr*. Enfin, un dernier cas particulier à traiter est celui du verbe *être* (lignes 36-38). En effet, on distingue *a red cup* de *a cup is red*. Dans le premier cas, on obtient

un *gn* classique $\{a, cup, \{red\}\}$. Dans le second cas, on va construire un *gn* à partir de l'adjectif seul : $\{a, cup, \emptyset\}$ et $\{\emptyset, \emptyset, \{red\}\}$. Cela permet de garder une structure du type gn-Prédicat-gn qui est employée plus tard dans notre regroupement en groupe de relation.

Les virgules correspondant à une conjonction de plus de deux éléments sont prétraitées avant l'analyse syntaxique. Notons h_c l'index de dépendance de la virgule, qui correspond à l'index auquel elle est reliée par une relation de dépendance.

Soit w le mot d'index idx_w tel que $h_c = idx_w$. S'il existe une conjonction de coordination qui "pointe" également vers w alors nous remplaçons la virgule par une copie de la conjonction, comme dans la figure 3.16.

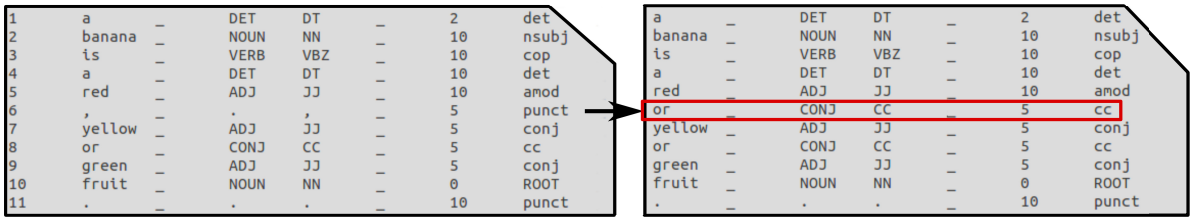


FIGURE 3.16 – Exemple d'analyse sémantique avec des adjectifs en conjonction

Il est également nécessaire de gérer les mots composés. Cela comprend également les lemmes de WordNet qui ne sont pas grammaticalement de véritables mots composés mais qui, en pratique, sont constitués de plusieurs mots comme par exemple *wheeled_vehicle-n04576211*. En théorie, nous devrions pouvoir les détecter à partir de la relation de dépendance de tag **UDR** COMPOUND (littéralement "composé" en français) entre deux noms. Cependant, SyntaxNet ne prend pas en compte cette relation (L'annexe A précise les relations prises en compte ou non par SyntaxNet). Plusieurs motifs peuvent correspondre à un mot composé :

- NN-NN. Par exemple, *water_bottle-n03543735*.
- VBN-NN. Par exemple, *wheeled_vehicle-n04576211*.
- VBG-NN. Par exemple, *washing_machine-n04554684*.

Dans tous les cas, nous vérifions l'existence du mot composé dans WordNet. S'il existe, nous fusionnons les deux mots en gardant les différents tags associés au second terme. Il faut noter que SyntaxNet considère très souvent les verbes en $-ing$ en tant que nom (NN) et non en tant que verbe au gérondif (VBG).

On notera dans la suite $GN(S)_s$ (resp. $GN(S)_o$) l'ensemble des groupes nominaux sujets (resp. objets). Cette distinction se fait directement en fonction du type de dépendance **UDR** du nom (NSUBJ ou NSUBJPASS).

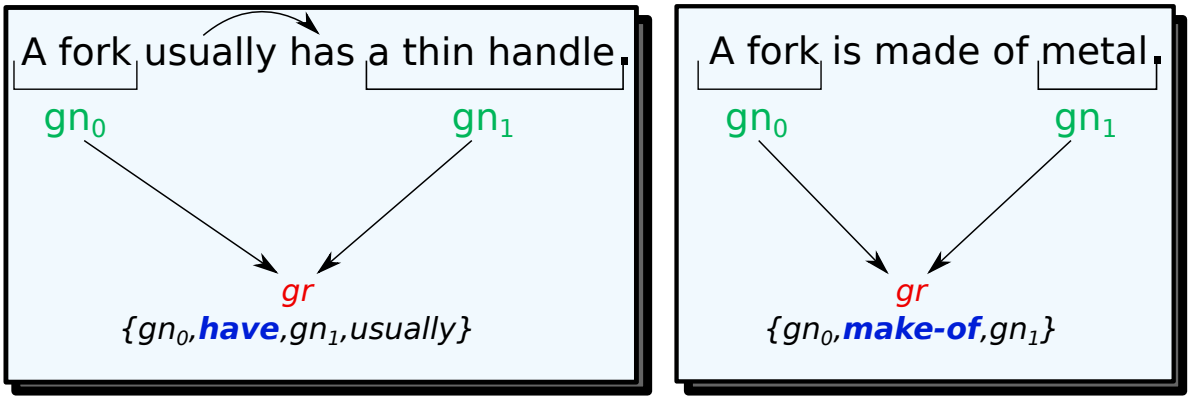


FIGURE 3.17 – Exemples simples de groupes de relation (GR)

Groupe de relation (GR)

Une fois l'ensemble $GN(S)$ d'une phrase S déterminé, nous les regroupons sous forme de groupe de relation (GR). Ils sont définis à partir d'un prédicat $\mathbf{w}_{prédicat}$ qui relie un groupe nominal sujet $gn_s \in GN(S)_s$ à un groupe nominal objet $gn_o \in GN(S)_o$. A cela peut s'ajouter un ensemble d'adverbes W_{adv} modifiant le prédicat. Formellement,

$$\forall gr \in GR(S),\ gr = \{gn_s \in GN(S)_s,\ \mathbf{w}_{predicat} \in W_P,\ gn_o \in GN_o,\ W_{adv} \subset W_{\mathbb{ADV}}^{n_{adv}}\} \quad (3.11)$$

n_{adv} correspond au nombre d'adverbe relatif au GR. W_P désigne l'ensemble des mots définissant un prédicat :

$$\begin{aligned} W_P = \{\mathbf{w} = [u\ x\ d\ h]^T \in W \mid \\ (u = \mathbb{VERB} \wedge d \in \{\mathbb{ROOT}, \mathbb{CONJ}, \mathbb{PARTMOD}, \mathbb{RCMOD}, \mathbb{PCOMP}\}) \\ \vee (u = \mathbb{NOUN} \wedge d = \mathbb{ROOT}) \vee PrepositionPredicat(w)\} \end{aligned} \quad (3.12)$$

La fonction $PrepositionPredicat$ vérifie si une préposition telle que {*with, of, for, on*} agit comme un prédicat et qu'elle n'est pas liée à un verbe précédent comme dans le cas de *made of* ou *designed with*. Dans ce dernier cas, le prédicat considéré sera alors le verbe avec sa préposition. Des exemples de *GRs* sont visibles à la figure 3.17. L'exemple de droite montre un groupe de relation défini par

$$gr = \{gn_s = gn_0 = \{a, fork, \emptyset\},\ \mathbf{w}_{predicat} = have, gn_o = gn_1 = \{a, blade, \{thin\}\}, W_{adv} = \emptyset\}$$

L'algorithme 2 présente la méthode d'extraction des *GRs*. Tout d'abord, des prédicats sont recherchés selon les contraintes définies par (3.12) (lignes 2-3). Les *GNs* sujets et objets qui leur sont

Algorithme 2 Extraction de groupes de relation (GR)

Entrée: Phrase $S = [\mathbf{w}_1 \dots \mathbf{w}_n]^T$, $\mathbf{w}_i = [u_i\ d_i\ x_i\ h_i]^T \in W$, $GN(S)$, $C^{\wedge/\vee}_{GN(S)}$
Sortie: Ensemble des groupes de relation GR, Ensembles $C^{\wedge/\vee}_{GR}$

1: **function** EXTRACTIONGROUPERELATION(S,$GN(S)$,$C^{\wedge/\vee}_{GN(S)}$)
2: **for** $i \in [\![1, n]\!]$ t.q. $\mathbf{w}_i \in W_P$ **do**
3: $\mathbf{w}_{predicat} \leftarrow \mathbf{w}_i$, $W_adv \leftarrow \emptyset$
4: **for** $j \in [\![1, n]\!]$ t.q. $u_j = \mathbb{ADV} \wedge h_j = i$ **do**
5: $W_{adv} \leftarrow \mathbf{w}_j$
6: **end for**
7: **for** $(gn_s \in GN_s(S),\ gn_o \in GN_o(S))$ t.q. $h_{idx_{nom}} = h_{idx'_{nom}} = i$ **do**
8: $GR \leftarrow \{gn_s,\ \mathbf{w}_{predicat},\ gn_o,\ W_{adj}\}$
9: **end for**
10: **for** $conjType \in \{\wedge,\ \vee\}$ **do**
11: **for** $c_{gn} \in C^{conjType}_{GN}$ **do**
12: $c_{gr} \leftarrow \emptyset$
13: **for** $gr \in GR$ t.q. $gn_o \in c_{gn}$ **do**
14: $c_{gr} \leftarrow gr$
15: **end for**
16: $C^{conjType}_{GR} \leftarrow c_{gr}$
17: **end for**
18: **end for**
19: **end for**
20: **end function**

rattachés sont ensuite construits (lignes 7-8). Dans le même temps les adverbes modifiant le prédicat courant sont également ajoutés au groupe de relation (lignes 4-5). Les contraintes liées aux conjonctions $C^{\wedge/\vee}_{GN}$ sont propagées aux *GRs* définis par les GN_o correspondant (lignes 10-18). On note de même ces ensembles $C^{\wedge/\vee}_{GR}$. Ainsi, en reprenant l'exemple de la figure 3.16, on obtient

$$\begin{aligned} GR_{i-1} &= \{GN_1,\ be,\ GN_i\},\ i \in \{2, 3, 4\} \\ C^{\vee}_{GR} &= \{GR_1,\ GR_2,\ GR_3\} \end{aligned} \tag{3.13}$$

Dans certains cas, dont celui des définitions de WordNet, le sujet est éludé eg *(a fork is) a cutlery used for ....* Ceci implique qu'il n'y a donc pas de *GN* sujet. Dans ce cas, on assigne à tous ces prédicats le sujet global de la phrase. Actuellement ce sujet est considéré comme étant le mot dont nous analysons la définition.

Nous ne propageons pas les éventuelles contraintes entre GN_s car non représentables dans notre

modèle de connaissance. En fait, notre modèle de connaissance ne peut être construit qu'à partir de formules logiques dites *formes normales conjonctives* (FNC). Ce sont des conjonctions ($\wedge$) de clauses, une clause étant une disjonction ($\vee$) de littéraux (comme nos *GRs*). Cela veut dire que les données textuelles utilisées doivent être de la forme :

$$\bigwedge_{i=1}^{n} \left(\bigvee_{j=1}^{n_i} GR_{ij} \right) \tag{3.14}$$

Il est possible de convertir des disjonctions en conjonctions en utilisant des négations mais dans notre cas, les relations négatives (absence de relation) ne sont pas explicitement représentées. Pour illustrer ceci, considérons la phrase *Les bouteilles et/ou les béchers contiennent de l'eau et/ou de l'huile.* avec quatre variations possibles. Nous avons là quatre *GNs* :

$$GN_1 = \{les, bouteilles, \emptyset\},\ GN_2 = \{les, \textbf{béchers}, \emptyset\}$$
$$GN_3 = \{le, eau, \emptyset\},\ GN_4 = \{le, huile, \emptyset\}$$

qui forment également quatre *GRs* :

$$GR_1 = \{GN_1, contenir, GN_3\},\ GR_2 = \{GN_1, contenir, GN_4\}$$
$$GR_3 = \{GN_2, contenir, GN_3\},\ GR_4 = \{GN_2, contenir, GN_4\}$$

Nous pouvons donc représenter chacune des variations de phrase possible par les FNCs :

Les bouteilles et les béchers contiennent de l'eau et de l'huile	$\rightarrow$	$GR_1 \wedge GR_2 \wedge GR_3 \wedge GR_4$
Les bouteilles et les béchers contiennent de l'eau ou de l'huile	$\rightarrow$	$(GR_1 \vee GR_2) \wedge (GR_3 \vee GR_4)$
Les bouteilles ou les béchers contiennent de l'eau et de l'huile	$\rightarrow$	$(GR_1 \wedge GR_2) \vee (GR_3 \wedge GR_4)$
Les bouteilles ou les béchers contiennent de l'eau ou de l'huile	$\rightarrow$	$GR_1 \vee GR_2 \vee GR_3 \vee GR_4$

Les deux dernières variations ne peuvent être représentées dans notre modèle, car celui-ci ne peut représenter qu'un seul monde possible. Cependant, il est rare de voir ce genre de structure dans des définitions ou des descriptions de concepts. Nous allons maintenant voir une variante particulière des groupes de relation dans le cas où le prédicat indique un usage tel que *used for* ou *designed for*.

Algorithme 3 Extraction de groupes de relation d'usage (GRU)

Entrée: Phrase $S = [\mathbf{w}_1 \dots \mathbf{w}_n]^T$, $\mathbf{w}_i = [u_i\ x_i\ d_i\ h_i]^T \in W$, GN sujet gn_s
Sortie: Ensembles des groupes de relation d'usage GRU

function EXTRACTIONRELATIONUSAGE(S,gn_s)
  $GRU \leftarrow \emptyset$
  for $i \in [\![1, n]\!]$ t.q. $w_i = "for" \vee w_i = "to" \vee (w_i = "in" \wedge w_{i-1} = "used")$ **do**
    for $j \in [\![i+1, n]\!]$ t.q. $h_j = i \wedge$
      $[\ [(w_i = "for" \vee w_i = "in") \wedge x_j = \mathbb{VBG}] \vee (w_i = "to" \wedge x_j = \mathbb{VB})]$ **do**
      $\mathbf{w}_{action} \leftarrow w_j, W_{cible} \leftarrow \emptyset$
      for $k \in [\![j+1, n]\!]$ t.q. $h_k = j \wedge d_k = \mathbb{DOBJ}$ **do**
        $W_{cible} \leftarrow w_k$
      end for
    end for
    $GRU \leftarrow \{gn_s, \mathbf{w}_{action}, W_{cible}\}$
  end for ▷ Distribution W_{cible} comme dans "a fork is used for serving or eating food"
  for $gru \in GRU$ t.q. $d(\mathbf{w}_{action}(gru)) = \mathbb{CONJ}$ **do**
    $gru_{conj} \leftarrow gru' \in GRU$ t.q. $h(\mathbf{w}_{action}(gru')) = \mathbf{w}_{action}(gru)$
    $W_{cible}(gru_{conj}) \leftarrow W_{cible}(gru)$
  end for
end function

Groupe de relations d'usage (GRU)

L'extraction des *GRs* liés aux relations *usedFor* et *on* se fait de façon différente. En effet, l'objet de ces relations n'est pas un *GN* mais un verbe avec son propre COD. Ces groupes de relation d'usage vont relier un groupe nominal sujet gn_s à un verbe d'action $\mathbf{w}_{action}$ avec optionnellement l'ensemble des mots W_{cible} objets de l'action. Formellement, en notant $GRU(S)$ l'ensemble des groupes de relation d'une phrase S

$$\forall gru \in GRU(S),\ gru = \{gn_s \in GN_s,\ \mathbf{w}_{action} \in W_{\mathbb{VERB}},\ W_{cible} \subset W^{n_c}_{\mathbb{NOUN}}\} \tag{3.15}$$

avec $n_c = |W_{cible}|$. Ainsi, dans la phrase *a cup is used for drinking tea or coffee*, nous pouvons extraire le gru :

$$\{\{a, cup, \emptyset\},\ drink,\ \{tea, coffee\}\}$$

Actuellement, le groupe nominal sujet gn_s est considéré comme étant le sujet global (objet de la définition).

L'algorithme 3 présente la méthode d'extraction de ces groupes de relation d'usage. On com-

mence tout d'abord par rechercher certaines prépositions indiquant la présence de prédicat d'usage, à savoir *for*, *to* et *in* (limité à l'expression *used in*) (ligne 2). On récupère ensuite les verbes d'actions pointant vers les prépositions trouvées (on rappelle que h_j est l'index du mot vers lequel pointe le mot w_j, avec une relation d_j) (ligne 3) :

- Dans le cas de *for* et *in*, ce doit être un verbe en -ing (**XPOS** x valant $\mathbb{VBG}$).
- Pour *to*, ce doit être un verbe à l'infinitif (**XPOS** x valant $\mathbb{VB}$).

Une fois le prédicat d'usage et le verbe d'action trouvés, d'éventuels objets sur lesquels s'exerce le verbe d'action sont recherchés (lignes 6-7). Ceux-ci correspondent à des mots pointant vers le verbe d'action avec une relation de complément d'objet direct (**UDR** d valant $\mathbb{DOBJ}$). La fin de l'algorithme (lignes 12-15) gère la distribution des mots cibles W_{cible} en présence de conjonction. Par exemple, *a fork is used for serving and eating food* sera équivalent à *a fork is used for serving food and eating food*.

Revenons sur une remarque précédente concernant SyntaxNet, qui peut parfois considérer certains verbes en $-ing$ en tant que nom [9]. Pour gérer ce cas, on considère en tant que $\mathbf{w}_{action}$ possible également les noms finissant en $-ing$. On recherche ensuite dans WordNet si ce mot peut correspondre à un verbe. Si c'est le cas, son **UPOS** est modifié en $\mathbb{VERB}$ et son **XPOS** en $\mathbb{VBG}$. Cette opération se fait dans l'étape de prétraitement, avant l'extraction des groupes nominaux. Ce cas est observable à la figure 3.15 sur le verbe $drinking$. Précisons qu'actuellement ne sont considérés que des noms en tant que cibles et non des *GNs* complets afin de rester plus général. Cependant, il serait immédiat d'étendre les cibles à des *GNs* complets.

Conversion en graphe ontologique

A ce stade une définition en langage naturel est décomposée en un ensemble de *GNs*, regroupés en *GRs* et en *GRUs*. Il ne nous reste plus qu'à convertir ces relations en liens ontologiques comme défini à la section 3.2. La figure 3.18 est un exemple de construction à partir de la définition du concept *hammer-n03481172*.

Tout d'abord, un nœud est créé pour chaque lemme de synset (entrée de WordNet) dont on a analysé la définition. Ensuite les nœuds correspondant aux noms et adjectifs des différents *GNs* ainsi que les verbes d'action des *GRUs* sont également créés. Si un *GN* est complet (avec au moins un adjectif), alors on crée un nœud facteur le représentant. C'est le cas du GN "a heavy rigid head" de la figure 3.18. Le nom y est relié par une arête *isA* et les adjectifs le sont par une arête *prop*.

9. Empiriquement, cela arrive lorsque le verbe en question n'a aucun complément.

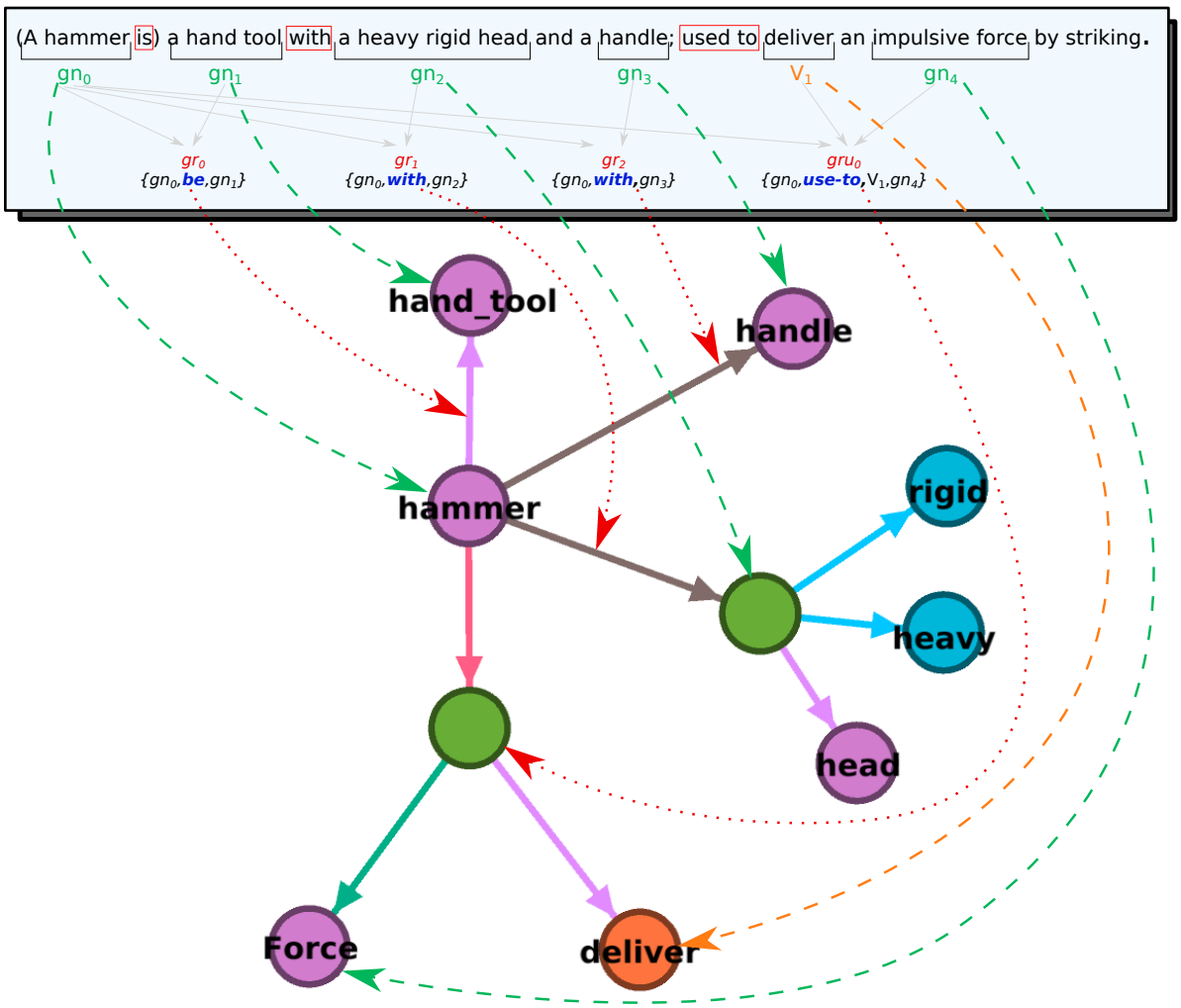


FIGURE 3.18 – Exemple de construction du sous-graphe ontologique du concept *hammer-n03481172*. Les flèches indiquent les correspondances entre les éléments issus de la décomposition syntaxique avec les nœuds et arêtes du graphe

On crée un nœud facteur pour chaque *GRU* complet (c'est à dire avec au moins un concept cible) : gru_0 de la figure 3.18 en est un exemple. On associe ensuite à chacun de ces nœuds facteur le verbe correspondant par une arête *isA* et les cibles par une arête *on*.

Les arêtes bidirectionnelles d'homonymies sont créées entre nœuds ayant un ensemble de synonymes d'intersection non vide et les liens d'hypero/hyponymies de WordNet sont convertis en arêtes de type *isA*.

Maintenant, il faut analyser les prédicats des *GRs* afin de savoir à quel type d'arête ils corres-

Relation	Prédicat	XPOS	Prépositions	COD UPOS
prop	make	VBN, VBZ, VBP	of	NOUN
hasA	have	VBP, VBZ, VBG	of	NOUN
	consist	VBP, VBZ, VBG	of	NOUN
	design	VBN	with	NOUN
	equip	VBN	with	NOUN
	support	VBN	on,by	NOUN
	with [10]	IN		NOUN
useFor	design	VBN	for	VERB
	use	VBN	for,to	VERB

TABLE 3.3 – Motifs utilisés pour classifier les *GRs* en fonction de leur prédicat

pondent. Pour cela, nous définissons un ensemble de motifs sous la forme :

$$\{predicat, \{xpos_i\}, \{prepositions_i\}, \{upos_i\}\} \tag{3.16}$$

définis par un prédicat contraint selon les prépositions associées, ces types d'**XPOS** et les types d'**UPOS** de ces COD. La table 3.3 montre les différents motifs utilisés pour chaque type de relation. Les *GRs* ayant un prédicat ne correspondant à aucun motif sont convertis en relation *linkedTo*. A noter qu'à la figure 3.18 l'adjectif *Impulsive* n'est pas considéré dans la relation *on* mais seulement le nom *force* : les GN complets ne sont pas actuellement pris en compte pour la relation *on*.

3.4 Evaluation

Nous allons maintenant inspecter l'ontologie obtenue par notre méthode. Tout d'abord, quelques statistiques générales seront données sur les caractéristiques du graphe de connaissance. Une analyse qualitative de celui-ci sera ensuite réalisée à partir d'exemples de sous-graphes. L'évaluation quantitative d'une ontologie est difficile car il n'y a pas, à notre connaissance, de métrique permettant de mesurer la pertinence des relations obtenues. Nous sommes donc limités à une inspection manuelle d'un sous-ensemble de notre ontologie.

	Nature		Quantité	Ratio(%)
Noeud	*Nom*		23945	58.38
	Facteur		9338	22.77
	Verbe		5517	13.45
	Adjectif		2219	5.41
Arête	*isA*	WordNet	21191	20.06
		Auto	15079	14.27
	linkedTo		30379	28.76
	prop		17000	16.09
	homonym		13092	12.39
	hasA		3679	3.48
	usedFor		3020	2.86
	on		2198	2.08

(a) Proportions des différents types de nœuds/arêtes

Caractéristique	*Valeur*
Degré moyen	2.575
Diamètre [11]	16
Distance moyenne	5.227

(b) Caractéristiques globales

small,large,form,material,substance,device,part,body,food, surface,structure,fabric,metal,instrument,component,drug, long,building,liquid,side,compound,acid,area,treat,wood, river,object,end,place,various,line,white,make,thin,vein, flesh,use,cell,artifact,ground,head,other,plant,stone,bone, point,source,system,person,several,mixture,arm,mineral,meat, skin,container,light,hold,color

(c) Noeuds classés selon *eigenvector centrality*

TABLE 3.4 – Description statistique globale de notre ontologie

3.4.1 Statistiques générales sur le graphe ontologique

La table 3.4a présente les informations concernant les types de nœuds et arêtes de notre graphe. On remarque que les relations d'homonymie constituent une part importante des relations, confirmant l'importance du problème de désambiguïsation. Notre méthode a également été capable de générer un nombre non négligeable (14.27%) de relations d'hypero/hyponymie.

La table 3.4b présente des caractéristiques génériques de notre graphe. Le degré d'un nœud est le nombre d'arêtes qui lui sont incidentes. Le diamètre d_G d'un graphe $G = \{V, E\}$ est défini comme la plus grande distance entre deux nœuds, la distance étant ici la longueur du plus court chemin γ_{min}. Formellement, en notant $\Gamma(v_i, v_j)$ l'ensemble des chemins possibles entre deux nœuds v_i, v_j, on a :

$$d_G = \max_{v_i, v_j \in V} |\gamma_{min}(v_i, v_j)|,\ \gamma_{min}(v_i, v_j) = arg \min_{\gamma \in \Gamma(v_i, v_j)} |\gamma(v_i, v_j)|$$

10. Ici l'équivalent du COD est un nom qui pointe vers *with* avec l'**UDR** $\mathbb{POBJ}$
11. En considérant le graphe non dirigé

La table 3.4c montre les nœuds classés selon leur *eigenvector centrality* : c'est une mesure de l'influence d'un nœud dans un graphe. Le principe est d'assigner un score à chaque nœud fonction du score des nœuds avec lesquels il est connecté. Autrement dit, un nœud avec un haut score est un nœud connecté à un nombre important de nœuds ayant également un haut score. Comme on peut s'y attendre, les premiers nœuds correspondent à des termes communs à beaucoup de concepts : *small*, *large*, *form*, *etc*.

3.4.2 Analyse qualitative de l'ontologie générée

Notre analyse qualitative va se baser sur des sous-graphes centrés sur les concepts suivants :

- *fork-n03383948* (figure 3.19)
- *spoon-n04284002* (figure 3.20)
- *automobile-n02958343* (figure 3.21)
- *bottle-n02876657* (figure 3.22)

Ces graphes sont construits comme l'ensemble des nœuds à distance 1 (en ne comptant pas les nœuds facteurs) du concept central et de ces hyponymes. Cela permet de garder des graphes relativement visibles. Il faut garder à l'esprit qu'il peut exister d'autres relations reliant les nœuds apparaissant dans ces graphes avec d'autres n'y apparaissant pas. Nous rappelons que la légende est décrite à la table 3.1.

fork-n03383948 : (figure 3.19) Ce sous-graphe est composé de deux composantes principales. La première est issue du concept central *fork-n03383948* (fourchette) et la seconde de son hyponyme *fork-n03384167* (fourche) à sa droite. Les hyponymes de *fork-n03383948* sont reliés à des actions (*eat*, *carve*, *toast*, *cook*) fortement connectés à la notion de nourriture et de cuisine. Cela apparaît également à travers les relations *on* de ces actions : *food*, *meat*, *salad*, *bread*. Le concept *dog* rattaché à *toasting fork* peut sembler étrange mais il n'a pas ici le sens de *chien* : c'est un synonyme de *hot dog*. Plus précisément, la définition de *toasting fork* contient le mot *frankfurter* :

"long-handled fork for cooking or toasting frankfurters or bread etc."

Ces trois mots (*hot dog*, *dog*, *frankfurter*) correspondent à un seul et même synset d'id $n07676602$. On peut voir également que l'adjectif composé *long-handled* a été reconnu directement comme un adjectif par SyntaxNet (section 3.3.2).

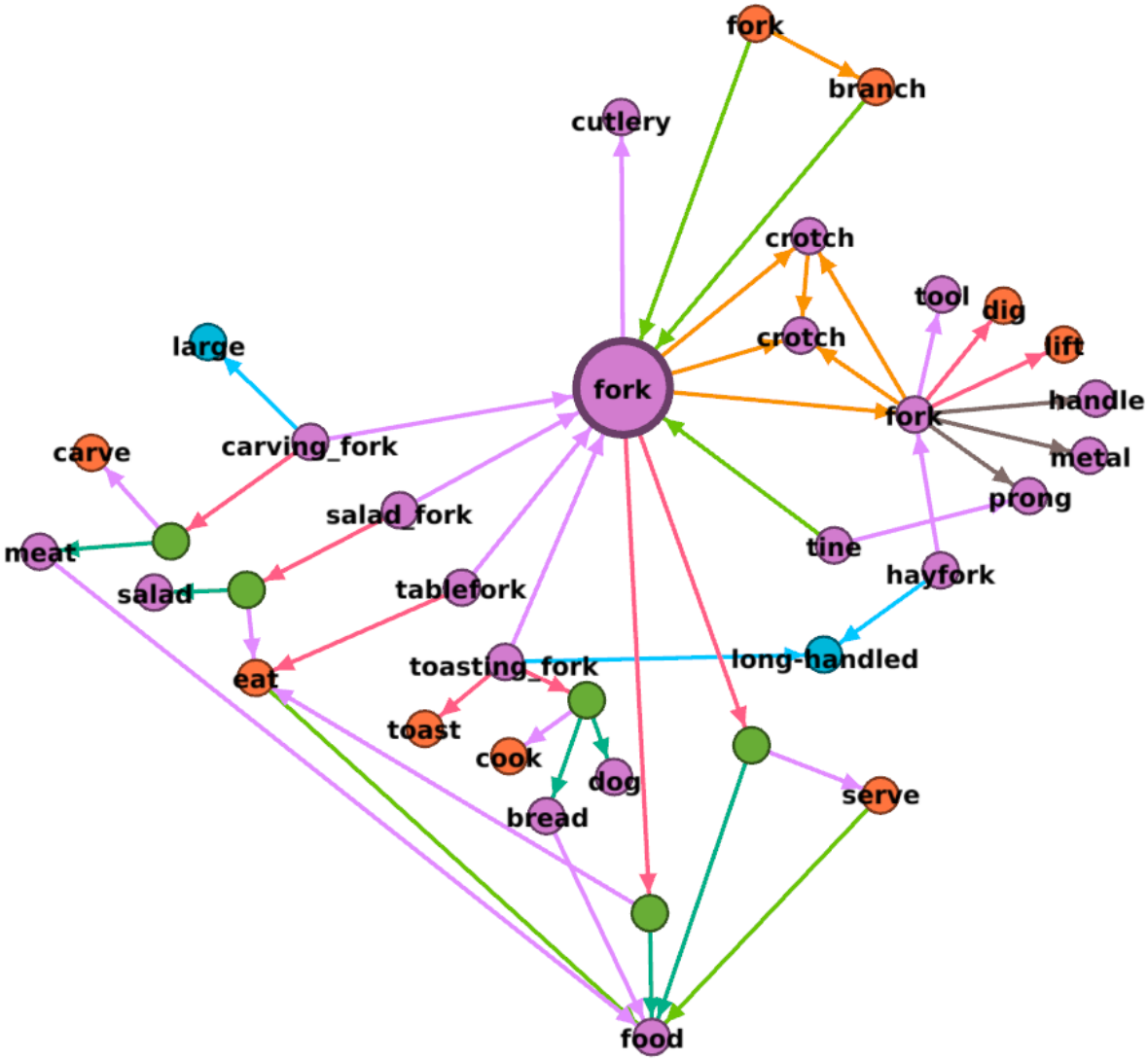


FIGURE 3.19 – Sous-graphe centré sur *fork-n03383948*

spoon-n04284002 : (figure 3.20) Une forte connexion au contexte de la nourriture via les concepts *food*, *eat*, *serve* est aussi observée ici. Les relations *prop* autour du nœud *spoon bread* comme par exemple *prop(spoon bread, rice)* peuvent sembler erronées au premier abord. Ce n'est en fait pas le cas : *prop* peut représenter la composition ou le matériau d'un objet physique. On peut s'en assurer en vérifiant la définition de *spoon bread* :

soft bread made of cornmeal and sometimes rice or hominy ; must be served with a spoon

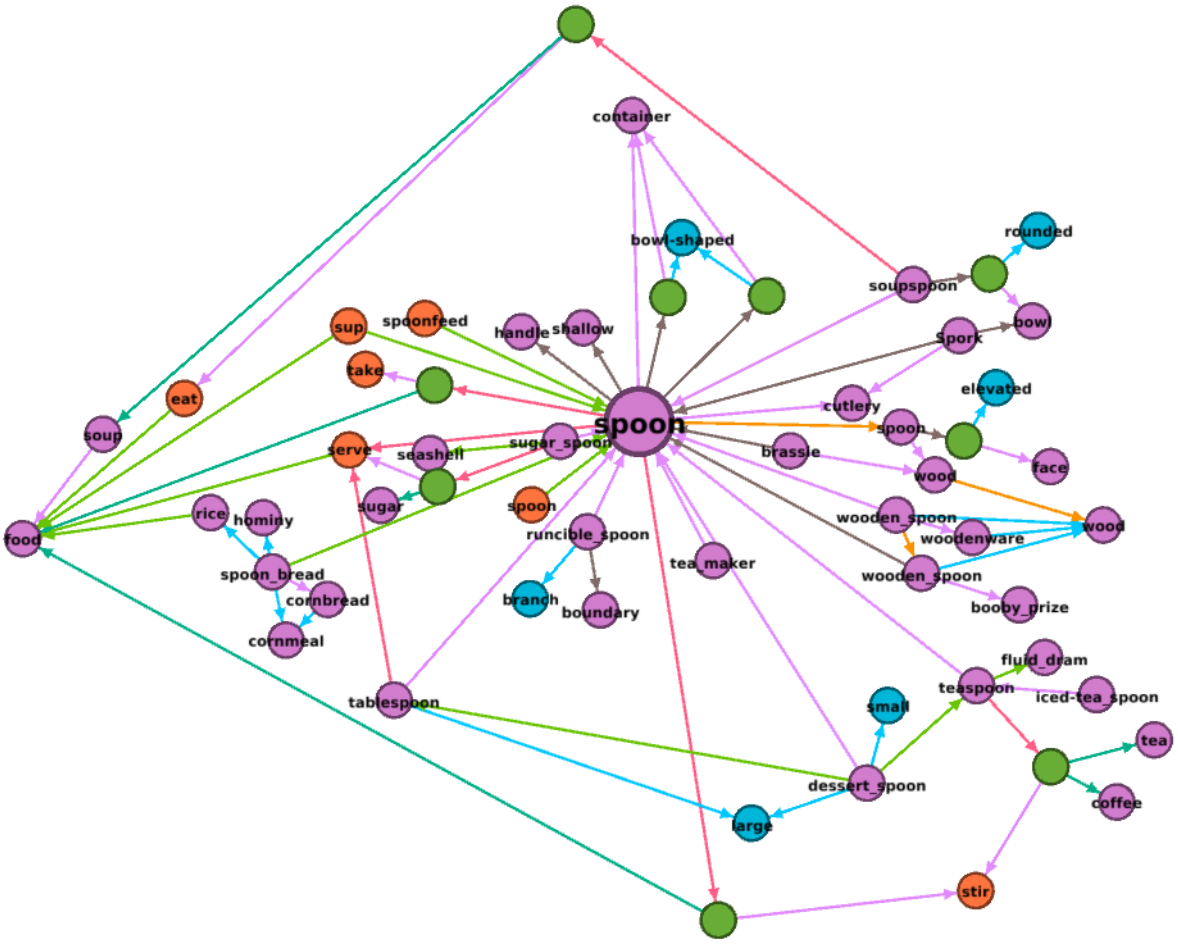


FIGURE 3.20 – Sous-graphe centré sur *spoon-n04284002*

Intéressons nous maintenant au concept *dessert spoon*. Il est lié à deux propriétés contradictoires : *small* et *large*. Cela se comprend en retournant à sa définition :

a spoon larger than a teaspoon and smaller than a tablespoon

Notre ontologie ne considère actuellement pas les comparatifs, qui ont donc été interprétés comme des adjectifs.

Enfin, une véritable erreur peut se voir au-dessus du nœud central. On peut y voir une relation *hasA(spoon, shallow)* qui n'a pas de sens ainsi qu'un doublon avec deux nœuds facteurs regroupant les mêmes concepts *bowl-shaped* et *container*. L'erreur n'est pas due à notre méthode d'analyse mais à une erreur de SyntaxNet qui considère *shallow* comme un nom (tag **UPOS** NOUN) et non comme un adjectif (tag **UPOS** ADJ). Cette erreur entraîne par la suite la création du doublon.

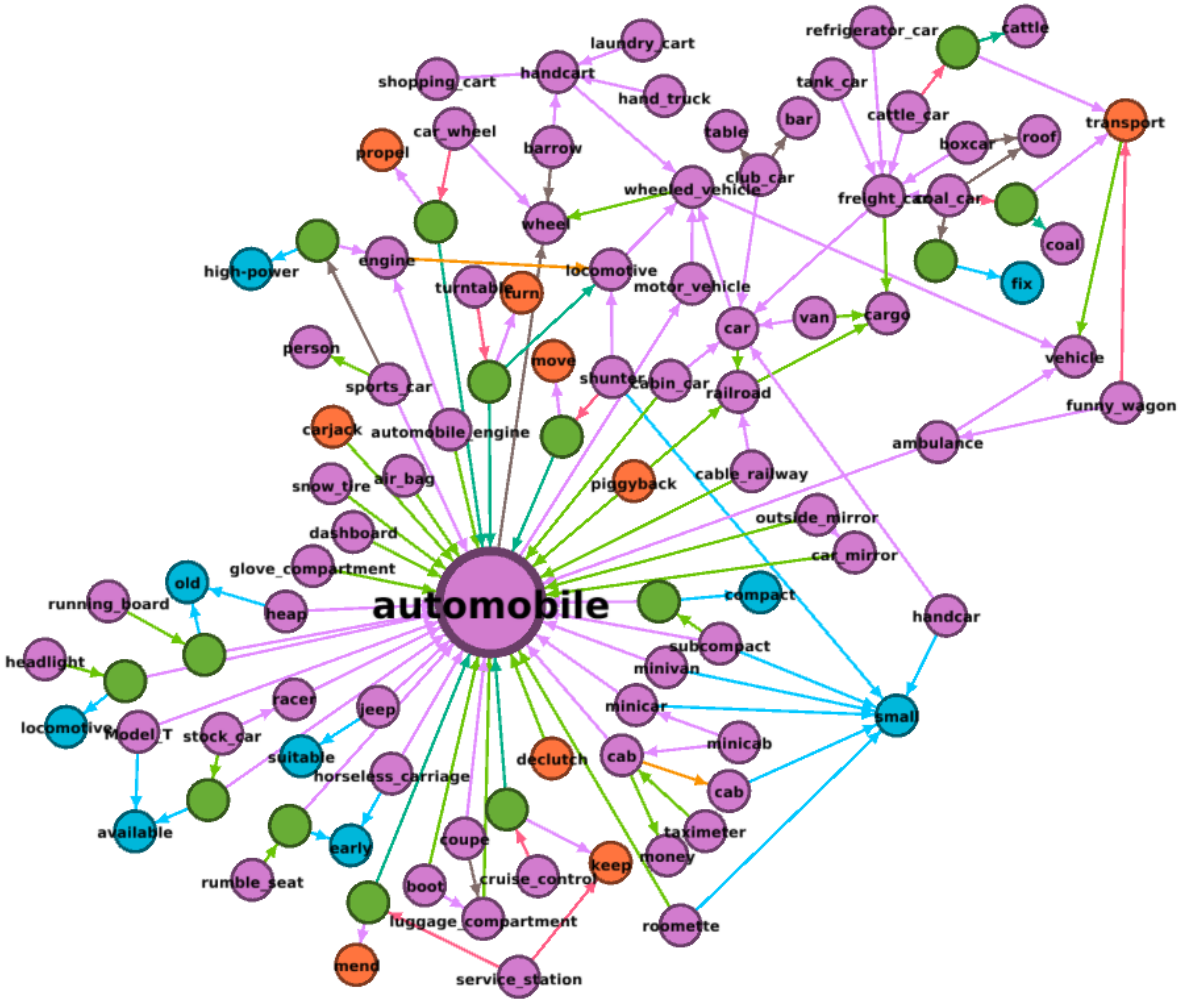


FIGURE 3.21 – Sous-graphe centré sur *automobile-n02958343*. Nous avons ici choisi de ne pas afficher tous les nœuds reliés par une relation *linkedTo* au concept central afin de préserver la lisibilité du graphe

automobile-n02958343 : (figure 3.21) Ce graphe nous montre un simple point d'amélioration à envisager dans la construction de notre ontologie. En effet, un certain nombre de relations **linkedTo** avec le concept central correspondent en fait à des relations *hasA* inversées que l'on pourrait nommer *isPartOf*. C'est le cas de *linkedTo(glove compartment, automobile)*, *linkedTo(air bag, automobile)* ou encore *linkedTo(car mirror,automobile)*. Le principal problème de la relation *hasA* comme nous la considérons actuellement est que seuls les éléments apparaissant dans la définition d'un concept pourront être rattachés à celui-ci par cette relation.

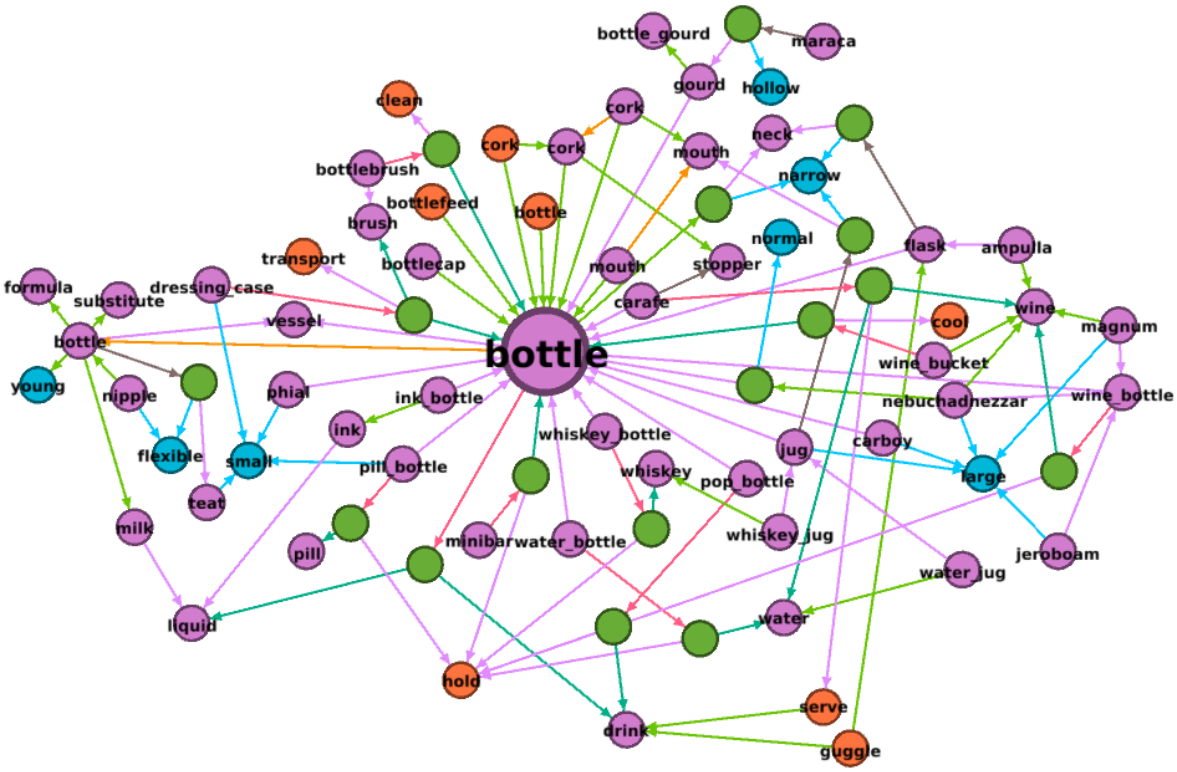


FIGURE 3.22 – Sous-graphe centré sur *bottle-n02876657*

bottle-n02876657 : (figure 3.22) Une fois de plus, le contexte entourant le concept central apparaît ici aussi clairement via les nœuds *water*, *liquid*, *drink*, *wine*, *milk*, *whyskey*. Tous ces nœuds sont en fait des hyponymes de *liquid*. Les arêtes correspondantes n'apparaissent pas dans ce graphe limité aux nœuds à distance 1 du concept central mais elles sont présentes dans l'ontologie.

Pour conclure cette analyse, on peut voir que l'ontologie obtenue par notre méthode représente un ensemble relativement riche de relations. Elle permet de faire émerger les contextes liés aux différents concepts eg. la nourriture pour la fourchette ou les liquides pour la bouteille. De plus, elle permet aussi d'inférer directement les concepts intervenant dans des requêtes utilisateurs. Par exemple, considérons la requête suivante : *Donne moi un récipient pour servir du vin*. Le robot doit tout d'abord inférer quels concepts sont susceptibles de correspondre à cette définition. En notant C le concept de *récipient (vessel) pour servir du vin*, il s'exprime en terme de relations par :

$$isA(C, vessel) \wedge usedFor(C, F) \wedge isA(F, serve) \wedge on(F, wine) \tag{3.17}$$

où F est un nœud facteur. Comme l'on verra plus en détails dans la section 5.5, la requête Prolog définie par (3.17) renverra la solution *carafe* (cf figure 3.22). Le robot peut ensuite se mettre à rechercher des instances de ce concept.

3.4.3 Analyse quantitative de l'ontologie générée

L'analyse quantitative consiste à inspecter manuellement les différentes relations extraites suivant notre méthodologie. Pour chaque type de relation un échantillon aléatoire contenant 200 arêtes est extrait. Deux degrés d'erreur sont considérés : les relations erronées et celles qui sont dans l'absolu correctes mais insuffisamment complètes pour pouvoir être exploitable. La table 3.5 résume les résultats de cette évaluation. La précision est définie ici par la proportion de relations correctes sur l'ensemble testé.

Cette évaluation montre que notre méthode a permis de découvrir des relations intéressantes. Les relations *isA* présentées sont disjointes de celle de WordNet. On voit par exemple que *vinegar* (n07828987) est un sous-concept de *liquid*. Cette information apporte une nouvelle description du concept qui n'apparaissait pas dans les hyperonymes issus de WordNet (que sont *condiment, flavorer, ingredient, foodstuff*). De même, notre méthode a trouvé que *clabber*[12] était une sous-classe de *milk* alors que ses hyperonymes associés (*dairy product, foodstuff*) ne donnent pas directement cette relation. Dans d'autres cas, le gain d'information est moins important car l'on trouve une relation avec un hyperonyme à distance supérieure à 1 du concept. Cela s'illustre avec *CD_player* : son hyperonyme directe est *electronic_equipment* d'hyperonyme *equipment*. Autrement dit :

$$CD\_player \xrightarrow{isA} electronic\_equipement \xrightarrow{isA} equipment$$

Notre méthode a extrait la relation *isA(CD_player, equipment)*, information redondante avec la relation d'hyperonymie directement donnée par WordNet. Ces relations pourraient être simplement retirées de l'ontologie en vérifiant que l'hyperonyme trouvé n'appartient pas à l'ensemble des hyperonymes (en remontant jusqu'au concept racine) de l'ontologie WordNet.

Les performances pour la relation *hasA* sont moins bonnes. Cela était prévisible car cette relation est destinée surtout à la description de parties d'objets. Une majeure partie des relations incomplètes proviennent d'expression du type *"has a *** of ***"*. Par exemple, on a pour le concept *module-n03778817* :

"consisting of an assembly of electronic components"

12. lait caillé

Notre méthode actuelle considère *A consisting of a B of C* comme *hasA(A,B)*, alors qu'on devrait avoir *hasA(A,C)*. Certaines erreurs proviennent également de l'analyseur SyntaxNet considérant certains adjectifs ou participe passé (eg. *shallow, bent*) comme des noms. Enfin, il y a des erreurs pour lesquelles nous ne voyons pas de lien lié directement à la méthodologie et semblerait provenir d'erreurs d'implémentation eg. *barbed_wire-n02790823* :

"strong wire with barbs at regular intervals used to prevent passage"

On arrive cependant à avoir des éléments intéressants comme

$$hasA(knife, handle),\ hasA(French\_dressing, mustard)$$

Les relations *prop* ne soulèvent pas de remarque particulière. Les erreurs rencontrées n'en sont pas vraiment car *other, only, small* peuvent être effectivement des adjectifs. Cependant, dans notre ontologie, ils n'apportent pas d'informations exploitables. On pourrait donc simplement envisager de les filtrer en post-traitement.

Les relations *usedFor* et *on* sont très importantes dans notre ontologie orientée sur les applications robotiques. On obtient les usages d'objets tels que

drinking_vessel-n03241496 : a vessel intended for drinking
food_processor-n03378174 : kitchen appliance with interchangeable blades ;
used for shredding or blending or chopping or slicing food

L'incomplétude des relations provient de divers facteurs. Il peut y avoir différents niveaux d'usage comme dans

chase-n03010283 : a rectangular metal frame used in letterpress printing to hold together
the pages or columns of composed type that are printed at one time

Il n'est pas faux de dire que l'object *chase* est utilisé pour *print*, mais à un niveau de détail supérieur il est utilisé pour *hold pages*. Enfin les erreurs sont issues de définitions dans lesquelles l'usage

Relation	Précision	Exemples correctes	Exemples incomplets	Exemples erronés
isA	94% (94%)	(vinegar-n07828987,liquid) (Tuileries-n04496173,palace) (CD_player-n02988304,equipment) (clabber-n07850219, milk) (dust-n14840092,particule) (paraboloid-n13897002,surface)		(central_nervous_system-n05480794,part) (sublimate-n15062284,product) (main-n09345932,body) (katharometer-n03609147,measure)
hasA	90% (84.5%)	(revolver-n04086273,cylinder) (knife-n03623556,handle) (French_dressing-n07833816,mustard) (hammer-n03481172,head) (Emmenthal-n07854982,hole) (marmite-n03722827,leg)	(stacks-n04295571,arrangement) →[of bookshelves] (module-n03778817,assembly) →[of electronic component] (strap-n04333709,loop) →[of leather]	(triangle-n04480853,bent) (mangosteen-n07763987,juicy) (spoon-n04284002,shallow) (Roman_nose-n05599501,prominent) (barbed_wire-n02790823,regular)
prop	96% (96%)	(wafer-n07695012,thin) (wheel-n04575723,circular) (sapphirine-n15012810,blue) (jawbreaker-n07599161,hard) (headpiece-n03504205,protective) (dolmen-n03220237,large) (painting-n03876519,artistic)		(loft-n03686470,other) (hydrocarbon-n14911057,only) (synchromesh-n04375241,same) (encephalogram-n03285730,X)
usedFor	96.5% (94.5%)	(drinking_vessel-n03241496,drink) (peavey-n03903133,handle) (clothesbrush-n03050453,clean) (chessboard-n03014317,play) (estradiol-n14750316,treat) (food_processor-n03378174,blend)	(chase-n03010283,print) →[used in letterpress printing to hold(.)] (oscillograph-n03857687,make) →[(.)for making a record]	(tin-n14658855,alloy) (making-n03714899,perform) (steam_engine-n04309049,raise)
on	98% (85.5%)	(wallet-n04548362,money) (mannequin-n03717921,clothes) (butcher_knife-n02927053,meat) (bowl-n02881193,liquid) (parer-n03890093,fruit) (spoon-n04284002,food)	(nephoscope-n03819047,velocity) →[(.)for measuring(.) velocity of mouvement of clouds] (indicator-n14917208,completion) →[(.)indicate the completion of a chemical reaction] (metal_detector-n03751757,presence) →[(.)used to detect the presence of stray bits of metal]	(stealth_fighter-n04308397,bomb) (liqueur_glass-n03676623,amount) (tire_iron-n04441093,shell)

TABLE 3.5 – Résultat de l'extraction de relations à partir d'un sous-ensemble de définitions de WordNet . Nous avons mis entre parenthèses la précision lorsque l'on considère les relations incomplètes comme erronées. Ces chiffres ne sont donnés qu'a titre indicatif étant donné le faible nombre (200) d'échantillons contrôlés.

d'élément intermédiaire est explicité, comme dans

steam_engine-n04309049 : external-combustion engine in which heat is used to raise steam which either turns a turbine or forces a piston to move up and down in a cylinder

Ici c'est le fonctionnement et non l'utilisation de l'objet qui est décrit. Ce genre d'information n'est actuellement pas pris en compte dans notre méthode.

L'inspection des relations *on* montre une proportion plus élevée de relations incomplètes. La raison est qu'actuellement nous ne considérons que les noms. Nous rencontrons le même problème qu'avec *hasA* avec les expressions du type *the *** of ****.

3.5 Notion de contexte

En plus des utilisations directes permettant de rechercher un concept à partir d'une description comme dans l'exemple (3.17), notre ontologie permet d'avoir un accès au "contexte" sous-jacent des différents concepts. La notion de contexte, qui informellement représente un ensemble de concepts reliant d'autres concepts entre eux, n'a pas de définition exacte. Nous proposons ici deux façons de la définir. L'intérêt dans une application robotique est de trouver des relations non explicites entre les objets constituant l'environnement. Ceci permet de répondre à la raison de la présence de l'objet et surtout de pouvoir inférer la présence d'autres objets liés au même contexte.

3.5.1 Contexte hiérarchique basé hyperonymie

Le contexte C_c^n d'un concept c peut se définir par un sous-graphe ontologique centré sur le concept c regroupant ses concepts voisins. Il y a plusieurs façon de le définir. La plus simple consiste à le construire selon les relations d'hypero/hyponymies (*isA*) à une profondeur n. Ce sous-graphe se construit de la manière suivante. On considère les concepts parents (hyperonymes) à distance d'au plus n du concept c. Pour chacun de ces concepts, en notant $d < n$ leur distance au concept c, on récupère leur concepts fils (hyponymes) à distance d'au plus d. Le principe de construction est illustré à la figure 3.24 : on "remonte" aux concepts parents puis l'on "redescend" au niveau du concept c.

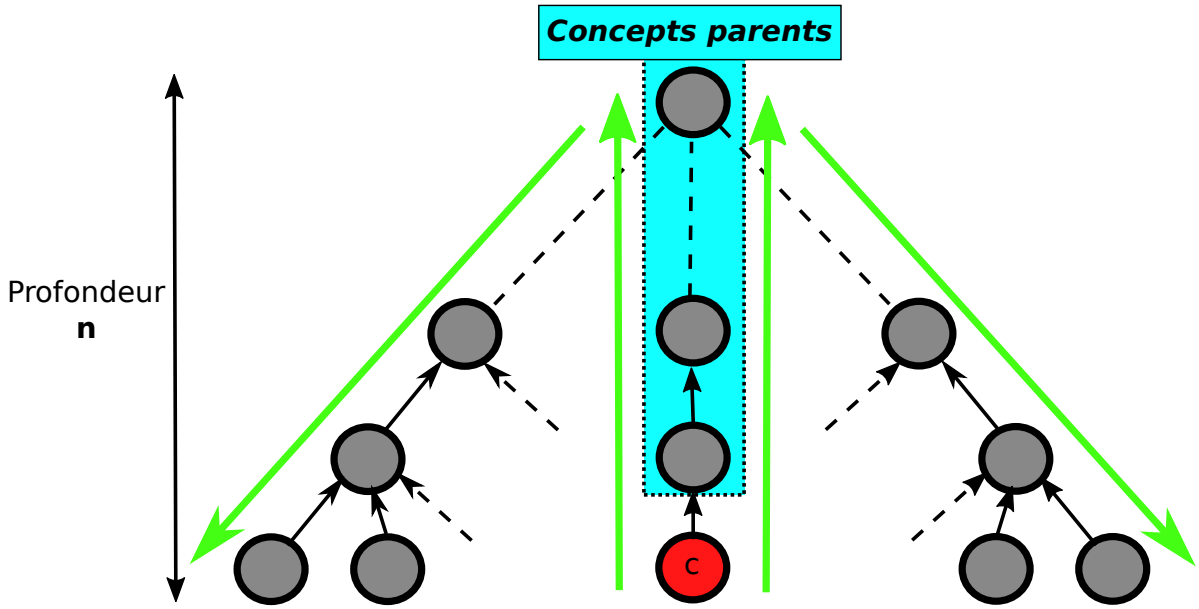
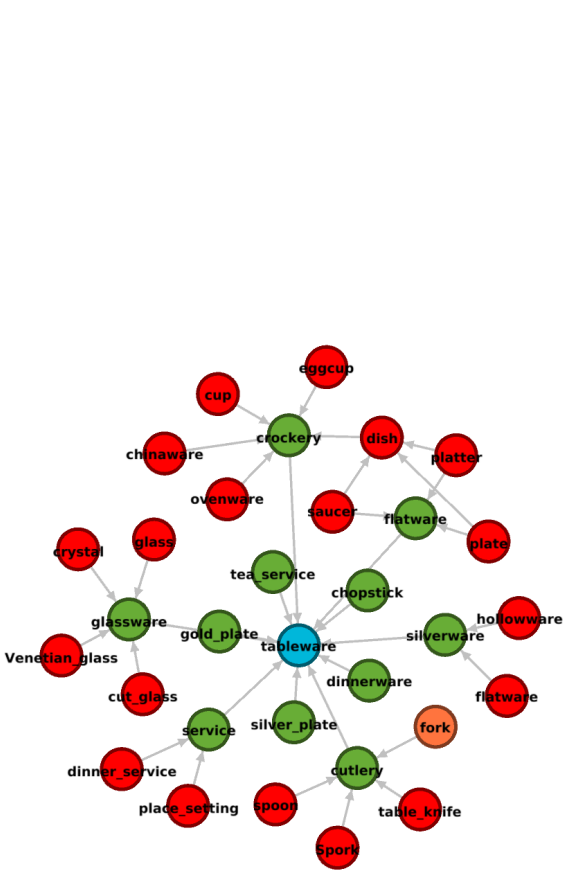
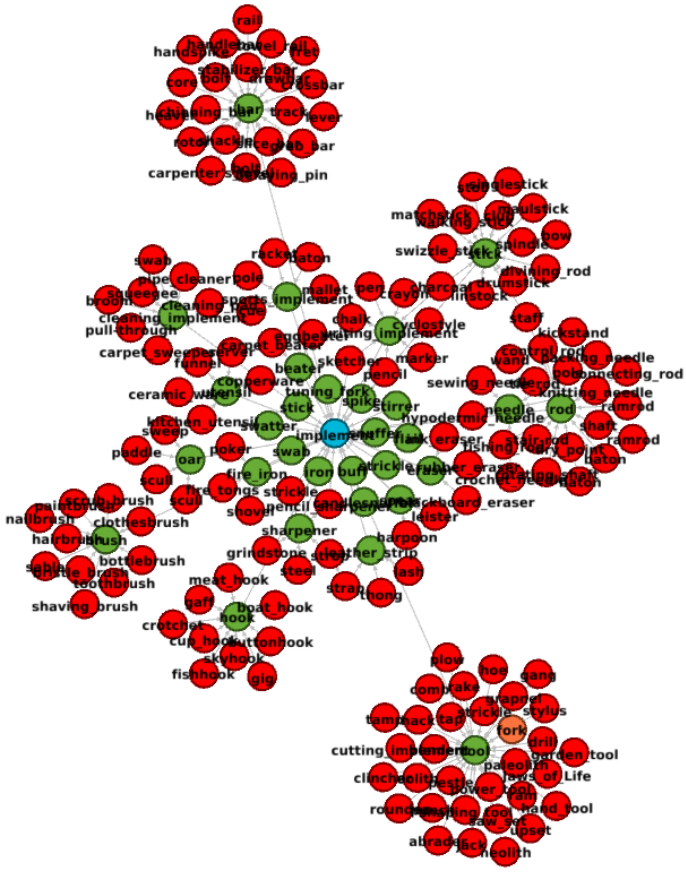


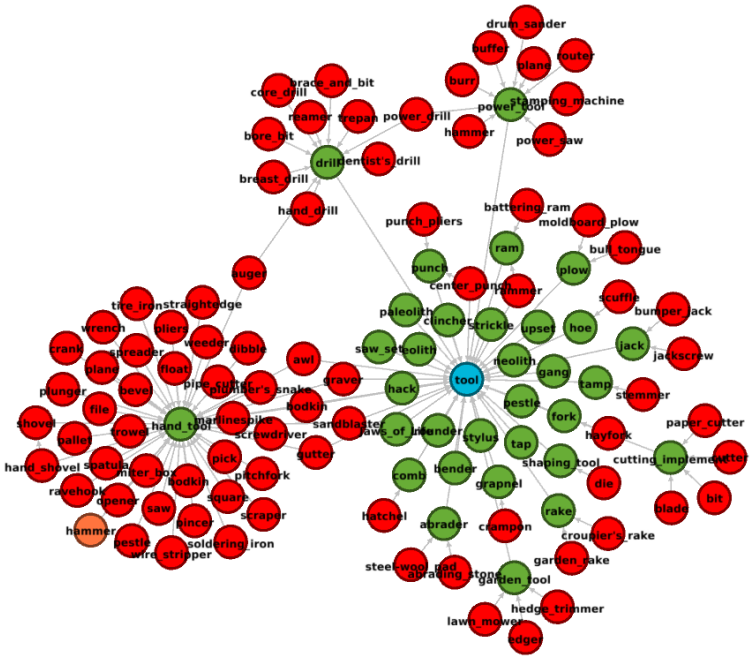
FIGURE 3.24 – Schéma du sous-graphe de contexte d'un concept.



(a) fork-n03383948



(b) fork-n03384167



(c) hammer-n03481172

FIGURE 3.23 – Sous-graphes de contexte (profondeur $n = 2$) pour différents concepts. Les concepts de même niveau sont indiqués par une couleur commune : 0 (rouge), 1 (vert) et 2 (bleu).

Dans le cas $n = 2$, le sous graphe de contexte est constitué des concepts "grand-parents" et de chacun de leur concepts "fils" et "petit-fils". Les figures 3.23a, 3.23b et 3.23c sont trois exemples de sous-graphes de contexte pour les concepts *fork-n03383948*, *fork-n03384167* et *hammer-n03481172*.

Cette définition du "contexte" permet ainsi de précharger les classifieurs de concepts potentiellement présents dans le voisinage du robot. Cependant, tous les concepts du contexte n'ont pas la même pertinence. La section suivante présente une façon de mesurer celle-ci.

3.5.2 Contexte basé sur la répresentation vectorielle de concepts

La pertinence est mesurée à l'aide des représentations vectorielles obtenues par l'algorithme non supervisée GloVe [PSM14], entraîné sur les corpus de Wikipedia et Gigaword 5. Lorsqu'aucun descripteur ne correspond à un concept, on utilise le descripteur de son hyperonyme. La pertinence d'un concept est évaluée par la similarité cosinus entre sa représentation vectorielle et celle du concept central c. En notant $G : V_K \mapsto \mathbb{R}^d$ [13] la fonction renvoyant le vecteur GloVe correspondant à un concept du graphe ontologique $G_K = \{V_K, E_K\}$, la pertinence contextuelle $p_{GloVe} : V_K \times V_K \mapsto [0, 1]$ entre deux concepts c_i et c_j se définit par :

$$p_{GloVe}(c_i, c_j) = \frac{G(c_i)^T G(c_j)}{||G(c_i)||_2 ||G(c_j)||_2} \tag{3.18}$$

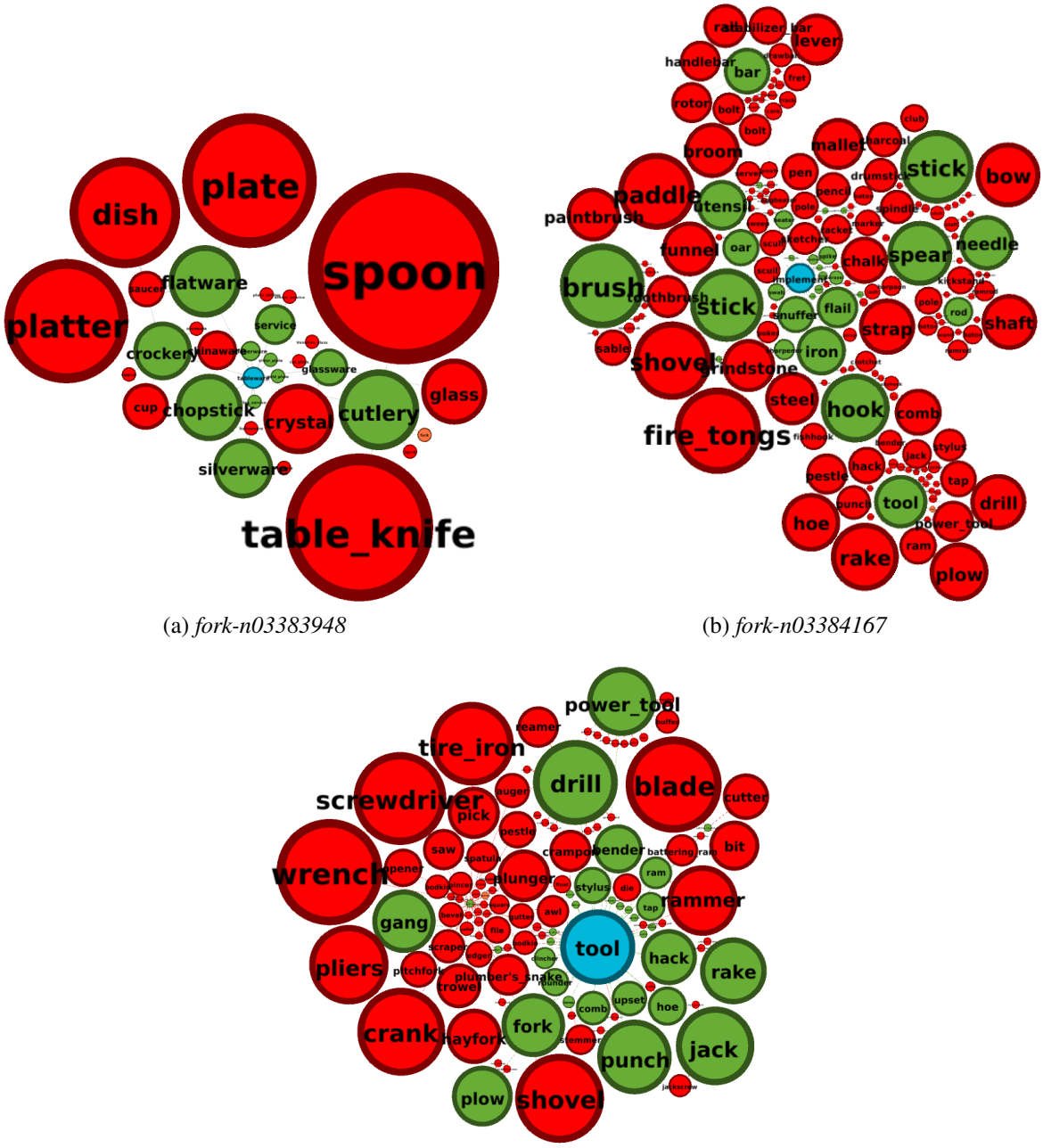
Les figures 3.25a, 3.25b et 3.25c représentent les mêmes sous-graphes de contexte avec la taille des nœuds proportionnelle à leur pertinence relative. Il faut noter que GloVe donne une représentation unimodale c'est à dire que les différents sens d'un mot sont représentés par un seul et même vecteur. On peut également appliquer cette notion de contexte sur un sous-graphe complet, c'est à dire non limité aux relations d'hypero/hyponymies.

Cette approche a l'avantage de la simplicité et de renvoyer une liste ordonnée de concepts susceptibles d'être présents dans la scène courante. Toutefois, elle ne rend pas compte des contextes plus abstraits ni des actions. Par exemple, le concept de nourriture (food) n'apparaît pas dans le contexte de *fork-n03383948* (figure 3.23a). Notre seconde approche cherche explicitement à trouver ces concepts.

3.5.3 Contexte abstrait basé sur la centralité intermédiaire

Revenons sur les figures 3.7 et 3.20 des sous-graphes centrés sur les concepts *fork-n03383948* et *spoon-n04284002*. On voit assez clairement que le concept $food$ à une importance particulière

13. En pratique, nous utilisons $d = 300$



(a) *fork-n03383948*

(b) *fork-n03384167*

(c) *hammer-n03481172*

FIGURE 3.25 – Sous-graphes de contexte (profondeur $n = 2$) avec mesure de pertinence. La taille des noeuds est proportionnelle à la distance au concept central calculée à partir des représentations vectorielles GloVe.

et correspond à la notion intuitive de contexte. Il faut maintenant trouver un moyen systématique de détecter ces concepts de contexte. Parmi l'ensemble des mesures d'importance de nœud dans un

graphe, la mesure de centralité intermédiaire [Fre77] [Bra01] nous semble adaptée à la détection des concepts contextes. La centralité intermédiaire mesure la proportion d'un noeud à être sur le plus court chemin entre deux autres nœuds. Formellement, en considérant un graphe $G = \{V, E\}$, la centralité intermédiaire d'un nœud $v \in V$ est donné par

$$ci(v) = \sum_{s,t \in V, s \neq \ t \neq v} \frac{\sigma(s,t|v)}{\sigma(s,t)} \tag{3.19}$$

où $\sigma(s,t)$ est le nombre de plus court chemin entre $s-t$ et $\sigma(s,t|v)$ le nombre de plus court chemin entre $s-t$ passant par v. Le résultat peut être normalisé en divisant par le nombre de paires s,t (v exclu) soit $(|V|-1)(|V|-2)$ dans le cas d'un graphe dirigé et $\frac{(|V|-1)(|V|-2)}{2}$ pour un graphe non dirigé.

Dans le cadre de ces mesures, notre graphe ontologique est considéré comme non dirigé. De plus, les nœuds correspondant à des adjectifs sont ignorés car ils peuvent avoir une centralité intermédiaire élevée sans nécessairement représenter un contexte. Typiquement, les adjectifs $small$ et $large$ vont relier entre eux de nombreux concepts qui n'ont aucun rapport les uns avec les autres. On se limite également ici à un sous graphe (de profondeur n) créé à partir du concept pour lequel on souhaite obtenir le contexte. On utilisera généralement $n = 2$ ou $n = 3$, en limitant les liens d'hyperonymie à une distance de 1. En effet, plus on remonte les liens d'hyperonymie, plus les concepts rencontrés deviennent génériques. Au contraire, on peut s'attendre à voir apparaître le contexte comme étant l'ensemble commun aux hyponymes même éloignés.

Une application directe de la centralité intermédiaire sur un sous-graphe de profondeur $n = 2$ comme décrit ci-dessus nous donne le graphe de la figure 3.26. La taille et la couleur des nœuds est proportionnelle à la valeur de centralité intermédiaire calculée.

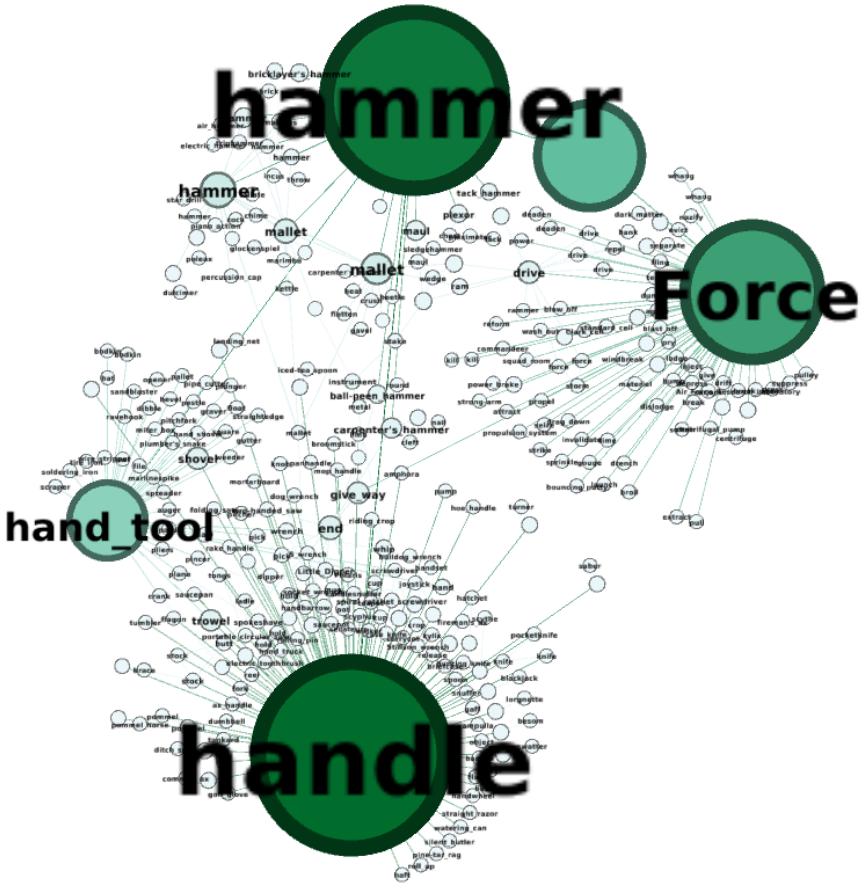


FIGURE 3.26 – Centralité intermédiaire sur le sous-graphe de profondeur 2 centré sur *hammer-n03481172*

Cette figure illustre les deux modifications à apporter à cette mesure sur notre ontologie :

– La centralité intermédiaire ne nous intéresse pas sur les nœuds facteurs. On peut voir un nœud facteur avec une forte centralité intermédiaire proche du nœud $hammer$. On doit donc "redistribuer" sa centralité intermédiaire sur les concepts qui lui sont liés. Soit v_f un nœud facteur. On note par ci la fonction qui renvoie la valeur de centralité intermédiaire d'un nœud. En notant $C(v_f)$ l'ensemble des voisins de ce nœud on aura alors une nouvelle valeur de centralité intermédiaire ci' :

$$\forall v \in C(v_f),\ ci'(v) = ci(v) + \frac{ci(v_f)}{deg(v_f)} \tag{3.20}$$

$$ci'(v_f) = 0 \tag{3.21}$$

En prenant en compte ces nouvelles valeurs, on obtient le graphe suivant

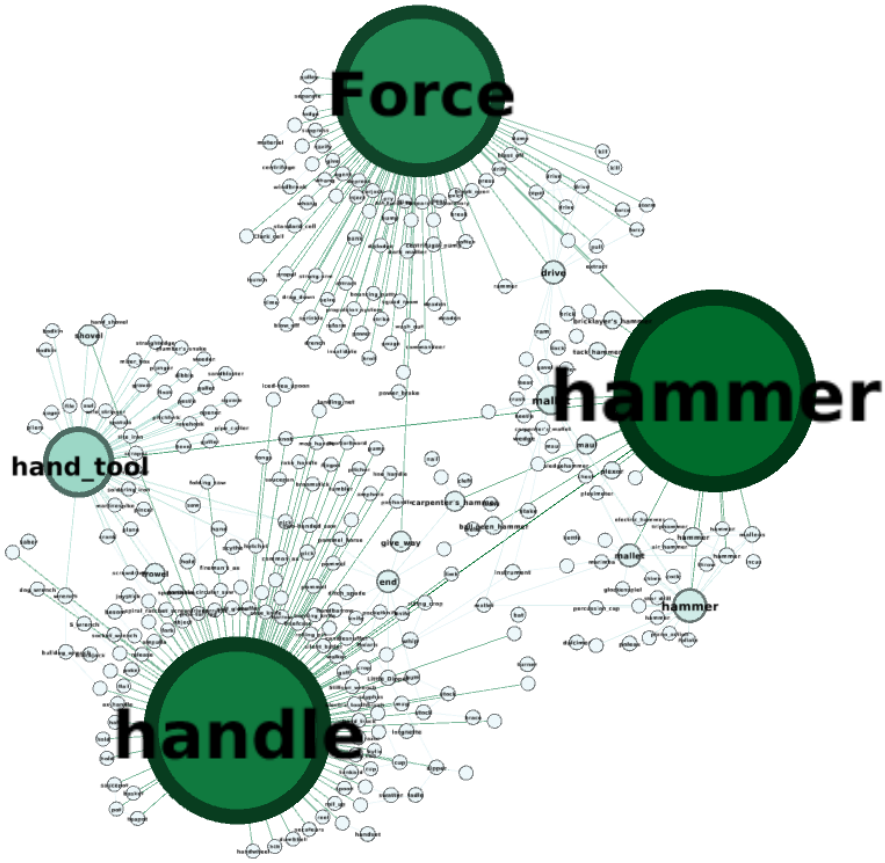
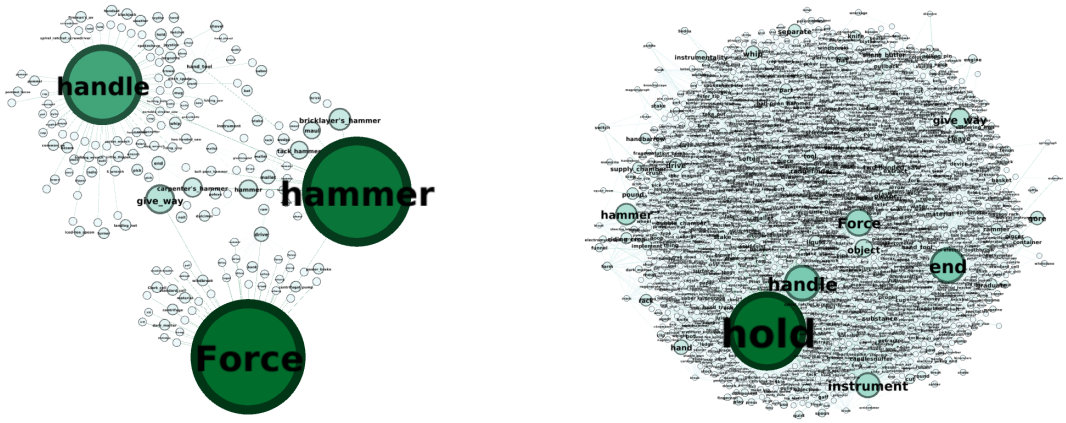


FIGURE 3.27 – Centralité intermédiaire sur le sous-graphe de profondeur 2 centré sur *hammer-n03481172*, après distribution des valeurs des nœuds facteur

– Avoir une centralité intermédiaire élevée est une condition nécessaire mais non suffisante : les nœuds ayant un degré élevé auront tendance à avoir des centralités intermédiaires élevées. Ainsi sur la figure 3.27, est-ce que le nœud *handle* doit sa valeur de centralité intermédiaire uniquement par son degré élevé ? Pour corriger ce biais, nous modifions le calcul de la centralité intermédiaire en normalisant le score final par le degré du nœud :

$$ci(v) = \frac{1}{deg(v)} \sum_{s,t \in V, s \neq\ t \neq v} \frac{\sigma(s,t|v)}{\sigma(s,t)} \tag{3.22}$$

De plus, le sous-graphe sur lequel s'effectue le calcul est préfiltré en retirant les nœuds de degré 1 qui ne sont pas liés à des nœuds facteurs. En effet, ces nœuds n'apportent aucune information sur le contexte car ils ne relient pas d'autres nœuds entre eux. On obtient finalement les graphes suivants (de profondeur 2 et 3) :



(a) Profondeur 2

(b) Profondeur 3

FIGURE 3.28 – Centralité intermédiaire avec normalisation

On remarque sur la figure 3.28a que la valeur relative du concept *handle* a diminué mais reste élevée : autrement dit, la centralité intermédiaire de ce concept est due à sa relation avec le concept *hammer* et non à son degré. Au contraire, la valeur du concept *hand_tool* a fortement diminué comparé à la figure 3.27. En fait, le lien entre *hand_tool* et *hammer* est explicité par l'intermédiaire du concept *handle*. La figure 3.29 présente des résultats pour quelques autres concepts.

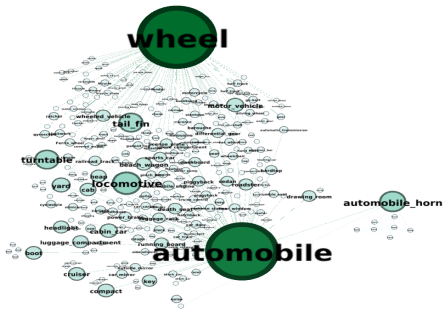
3.5.4 Extension au contexte commun d'un ensemble de concepts

On s'intéresse désormais à une extension naturelle de la notion de contexte proposée précédemment. Considérons non plus un seul mais un ensemble de concepts $\mathcal{C} = \{c_i\}_{1,\dots,m}$. On peut définir le contexte de $\mathcal{C}$ comme l'union des contextes de chaque $c_i \in \mathcal{C}$:

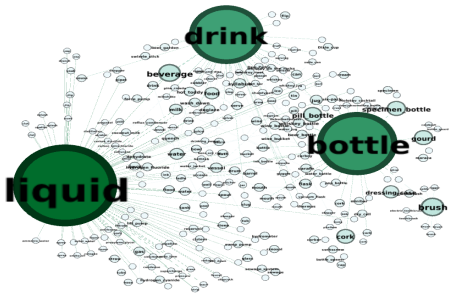
$$C^n_{\mathcal{C}} = \bigcup_{i=1}^{m} C^n_{c_i} \tag{3.23}$$

Dans le cas d'un ensemble de concepts, on cherche à trouver les concepts étant "proche" de chaque concept de l'ensemble. Le contexte basé sur la centralité intermédiaire peut directement s'appliquer sur le sous-graphe $C^n_{\mathcal{C}}$. Voyons maintenant comment adapter l'autre définition du contexte basé sur la représentation vectorielle GloVe des concepts.

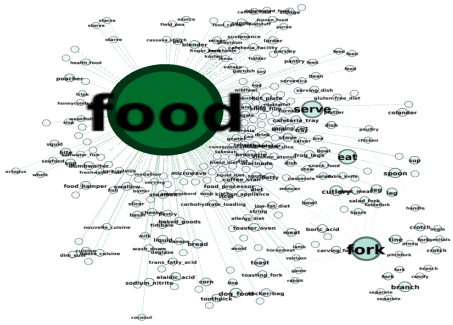
En notant par $P(E)$ l'ensemble des parties (sous-ensemble) d'un ensemble E, on définit une nouvelle pertinence contextuelle $p^e_{GloVe} : \mathcal{P}(V_K) \times V_K \mapsto [0, 1]$ par la moyenne géométrique des



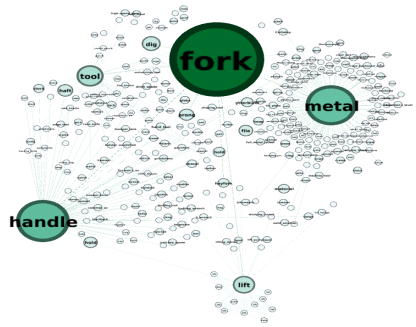
(a) *automobile-n02958343*



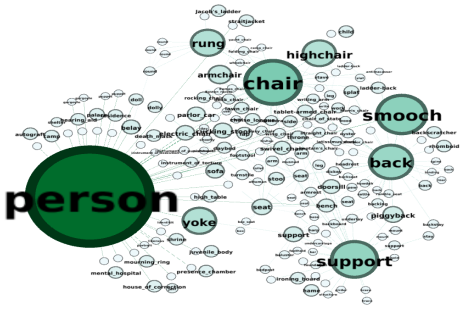
(b) *bottle-n02876657*



(c) *fork-n03383948*

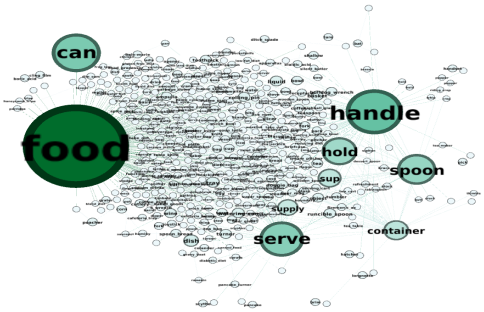


(d) *fork-n03384167*

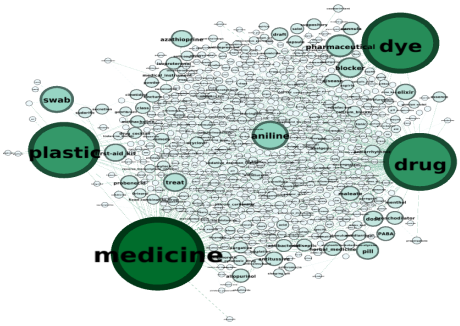


(e) *chair-n03001627*

(f) *handbag-n02774152*



(g) *spoon-n04284002*



(h) *medicine-n03740161*

FIGURE 3.29 – Exemples de sous-graphes proportionnés par les valeurs de centralités intermédiaires

FIGURE 3.30 – Sous-graphe centré sur les concepts *cup-n03147509* et *spoon-n04284002*, proportionné par la mesure de pertinence définie par (3.24)

distances d'un concept $c \in C^n_{\mathcal{C}}$ à chaque concept de l'ensemble $\mathcal{C}$:

$$p^e_{GloVe}(\mathcal{C}, c) = \sqrt[m]{\prod_{i=1}^{m} p_{GloVe}(c_i, c)} \tag{3.24}$$

L'intérêt de la moyenne géométrique est de fortement pénaliser une faible pertinence même sur un seul concept de l'ensemble. La figure 3.30 montre le contexte commun ainsi obtenu pour $\mathcal{C} = \{cup\text{-}n03147509,\ spoon\text{-}n04284002\}$. Parmi les concepts ayant les pertinences les plus élevées, on retrouve *bowl*, *tablespoon*, *teaspoon*, *cream* ou encore *milk*.

3.5.5 Discussion

Cette méthode d'extraction de contexte est difficile à évaluer autrement que par l'analyse qualitative. En effet, la notion de contexte est assez floue et nécessite la définition d'une métrique. Les résultats obtenus nous montrent que notre méthode permet de faire ressortir des concepts de contexte pertinents, en évitant le piège des concepts "généralistes" ayant de nombreux concepts fils. Cependant, en ne se basant finalement que sur des définitions de dictionnaire, on ne peut accéder qu'à un certain type de contexte. Notre méthode ne permet pas, par exemple, d'obtenir *hammer* dans le contexte de *nail*. De plus, il y a toujours le problème de la désambiguïsation qui perturbe de façon non négligeable les contextes obtenues avec la seconde méthode comme dans l'exemple de la figure 3.31 pour le concept *cup-n03147509*.

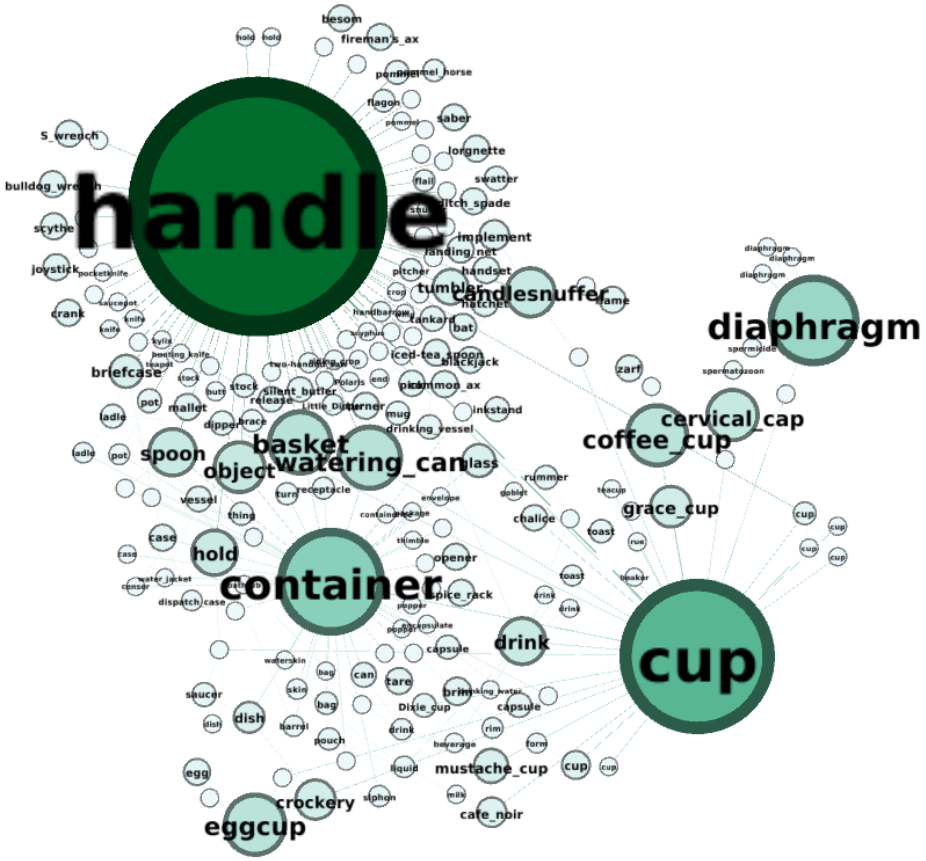


FIGURE 3.31 – Sous-graphe contextuel basé sur la centralité intermédiaire pour le concept *cup-n03147509*. La présence des concepts *diaphragm*, *cervical_cap* est due à une erreur de désambiguïsation lors de l'analyse de la définition du concept *diaphragm-n03189461*.

3.6 Conclusion et perspectives

Cette section a décrit notre modèle de représentation de connaissance basé sur la sémantique. Tout d'abord, un tour d'horizon sur les principales ontologies existantes a montré qu'elle pouvait manquer de détails dans la description d'objets physiques, éléments nécessaires à la compréhension du monde entourant un robot. De manière générale, elles sont souvent générées manuellement sur un petit ensemble de concepts. D'autres méthodes exploitent les avancées récentes dans l'apprentissage automatique profond sur de grande bases de données (Wikipedia) mais aucune garantie n'est donnée sur les informations qui vont être extraites. En pratique, ces méthodes donnent beaucoup de relations bruitées et finalement peu de détails sur la description/fonctionnalité des objets. C'est pourquoi nous proposons une méthode de construction d'ontologie reposant sur l'analyse de définitions, qui ont l'avantage de décrire un concept de façon relativement simple et structurée. Notre graphe de connaissance représente des concepts reliés entre eux par des relations choisies afin de correspondre aux critères permettant la description d'objets. Nous initialisons notre ontologie sur celle de WordNet et la complétons par l'analyse syntaxique de ses définitions. Les premières évaluations ainsi que l'inspection visuelle des données montrent la pertinence des relations extraites.

Cependant, nous sommes conscients des limites de notre approche et avons plusieurs pistes d'améliorations. Certaines sont immédiates :

- Ajout de règles d'extraction se basant sur les erreurs vues lors de l'évaluation ainsi que la correction de problèmes liés à l'implémentation.
- Détection de relations présentes mais exprimées sous forme "passive". Par exemple :

$$\textit{partOf(A,B)} = \textit{hasA(B,A)} \text{ ou } \textit{usedBy(A,B)}.$$

- Etendre l'ontologie en analysant d'autres dictionnaires comme *Oxford Dictionaries*
- S'attaquer au problème de la désambiguïsation en ré-analysant les définitions à l'aide du graphe ontologique précédemment construit.

Ces corrections nous permettront d'avoir une meilleur précision mais le problème reste que nous définissons manuellement nos règles. Une approche orientée sur l'apprentissage automatique nous permettrait :

- de découvrir de nouvelles règles/prédicats en analysant à nouveau les définitions de concepts ayant un contexte commun.
- de filtrer les informations pertinentes provenant d'autres ontologies et de les intégrer à la nôtre. En particulier, ConceptNet [SH12] représente des relations relativement similaires à

celles que nous proposons.

Enfin, il faudrait pouvoir récupérer de possibles erreurs. L'idée générale serait de s'inspirer du filtre à particules utilisé pour la localisation de robot mobile. Chaque particule représente une trajectoire affectée d'un poids proportionnel à sa vraisemblance. Ici, on représenterait notre ontologie par une distribution de probabilité. Chaque extraction de relation mettrait à jour la vraisemblance de chacune des particules, éliminant ainsi celles qui ne sont pas consistantes.

Chapitre 4

Modèle de Perception

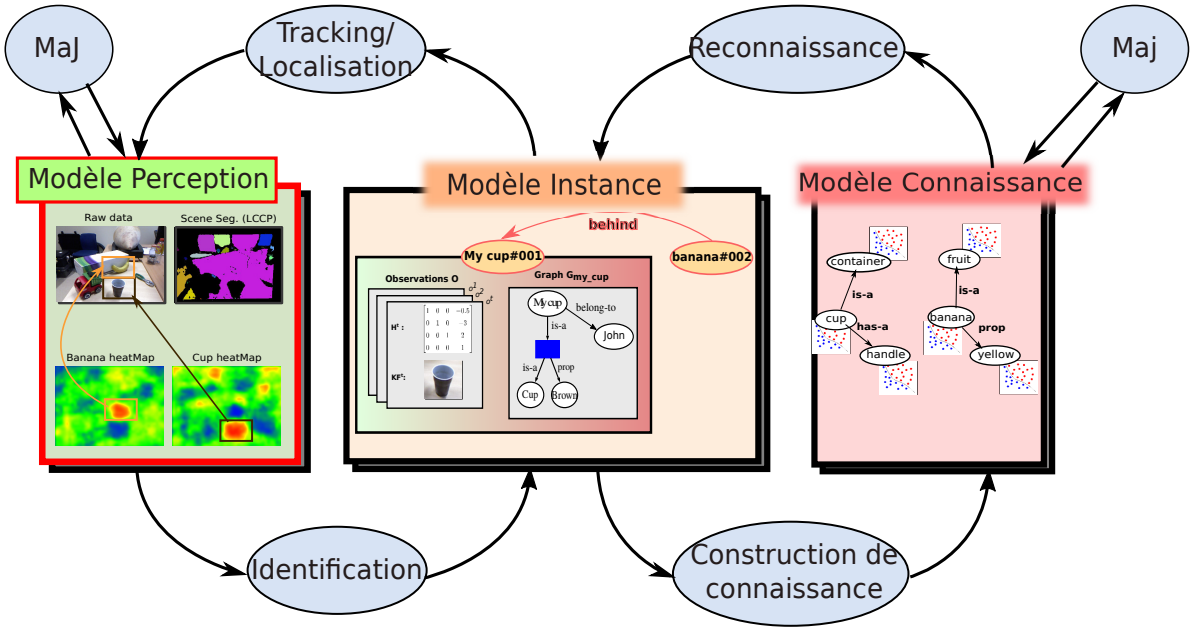


FIGURE 4.1 – Modèle de perception dans notre architecture

4.1 Introduction

Cette section est dédiée à la première unité de notre modèle à 3 couches, à savoir le modèle de perception. Il est chargé de structurer les données capteurs bas niveau (comme par exemple une caméra RGBD) en un ensemble d'objets, qui seront ensuite associés à des instances dans le modèle suivant (section 5.1). Ce modèle est donc une représentation sensorielle de l'environnement visible

par le robot **à l'instant courant**. La segmentation des scènes observées peut être faite de manière *passive*, ce qui correspond au problème de *la segmentation de scène non supervisée* et à la *saillance*. Il est également possible de rechercher des objets d'une certaine classe par des méthodes de détection/localisation ou de matching/tracking si l'on cherche des instances particulières. L'hypothèse de monde ouvert dans laquelle se placent nos travaux restreint les approches possibles :

- Méthodes non supervisées ne requérant pas de données d'entraînement.
- Classifieurs binaires.
- Utilisation d'approches supervisées par le *Transfert learning*.

Il faut également noter qu'en pratique, certains processus sont communs avec le modèle d'instance comme par exemple la recherche d'une instance dans une image ($Instance \rightarrow Perception$) et la reconnaissance d'une instance ($Perception \rightarrow Instance$).

4.2 Une approche basée sur les descripteurs profonds

Nous devons distinguer la détection d'objet active de la segmentation de scène passive. Ce dernier problème, difficile, est toujours un sujet de recherche en cours. Nous employons actuellement l'algorithme LCCP [SSPW14] [1] (cf section 2.1.1) pour la segmentation de scène non supervisée. Compte-tenu de la variabilité d'échelle des objets que l'on peut rencontrer dans l'environnement, les résultats sont loin de correspondre à la segmentation voulue comme le montre la figure 4.2. Afin de se concentrer sur notre modèle et non sur la méthode de segmentation à proprement parler, nous supposons par la suite une scène correctement segmentée.

Dans le cadre de la détection active, il nous faut choisir une représentation visuelle des objets à partir de laquelle nous pourrons effectuer la classification, la détection/localisation et la reconnaissance d'instance. Notre choix s'est porté sur les représentations intermédiaires construites par un réseau de neurones convolutifs (CNN). Tout d'abord, nous commencerons par rappeler la définition de ces CNNs. On introduira également une extension permettant de les appliquer à des tâches de localisation et de détection.

La littérature récente a montré que ces descripteurs sont très génériques et utilisables hors du cadre dans lequel ils ont été entraînés [RASC14]. Nous étudions plus en profondeur cette propriété en vue de leur utilisation pour la représentation d'instance. Cette analyse nous mènera ensuite vers l'étude de la continuité de ces descripteurs par des transformations de similarité 2D [2] ainsi qu'à des exemples d'applications possibles.

1. Nous utilisons l'implémentation de PCL [RC11]
2. Groupe composé de rotation, translation et de mise à l'échelle noté $\mathbf{SIM}(\mathbf{2})$

(a) Image original RGB (b) Segmentation obtenue

FIGURE 4.2 – Exemple de segmentation erronée obtenue avec LCCP [SSPW14]

4.2.1 Réseau de neurones convolués

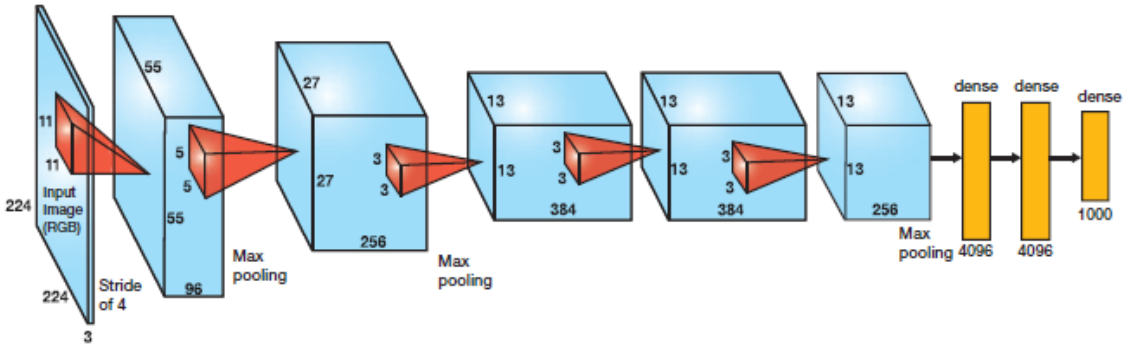


FIGURE 4.3 – Structure du réseau de neurones convolutifs AlexNet

Nous allons décrire ici la structure classique d'un CNN en se basant sur deux exemples que sont AlexNet [KSH12] et VGGNet [SZ14]. Ces réseaux relativement simples contiennent l'essentiel des briques constituant le récent succès de ces méthodes. Ce domaine évolue rapidement avec une amélioration constante des architectures. Plus de détails sont donnés par [GBC16].

Un CNN est une succession de couches de neurones interconnectés, chacune définie par un tenseur 3D $Largeur \times Hauteur \times Profondeur$ comme on peut le voir sur la figure 4.3 représentant AlexNet. L'image donnée en entrée est également vue comme une couche de dimension $227 \times 227 \times 3$, chaque pixel étant un neurone et la profondeur étant le nombre de couleurs. La couche suivante, nommée *conv1*, a elle pour dimension $55 \times 55 \times 96$.

Chaque couche considère comme entrées les sorties de la couche précédente. Il y a principa-

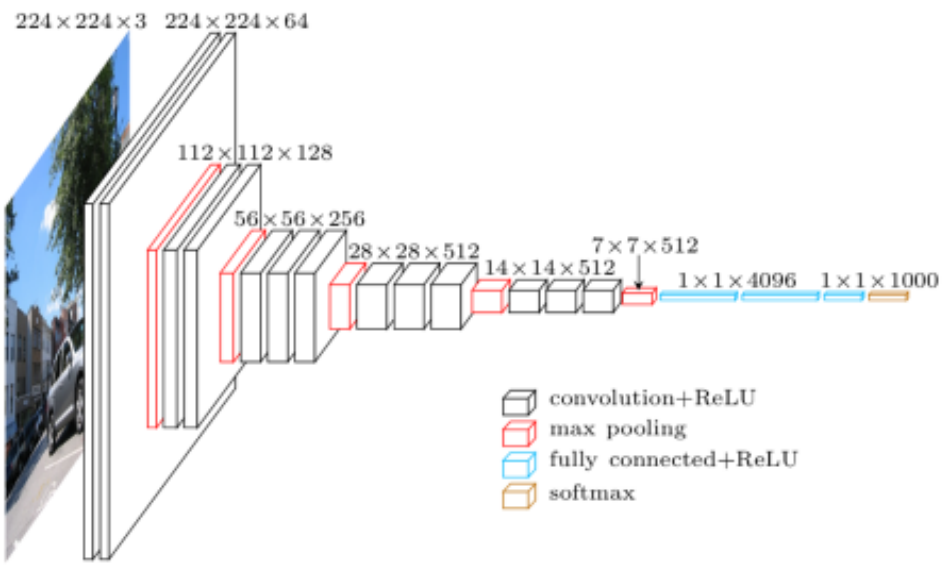
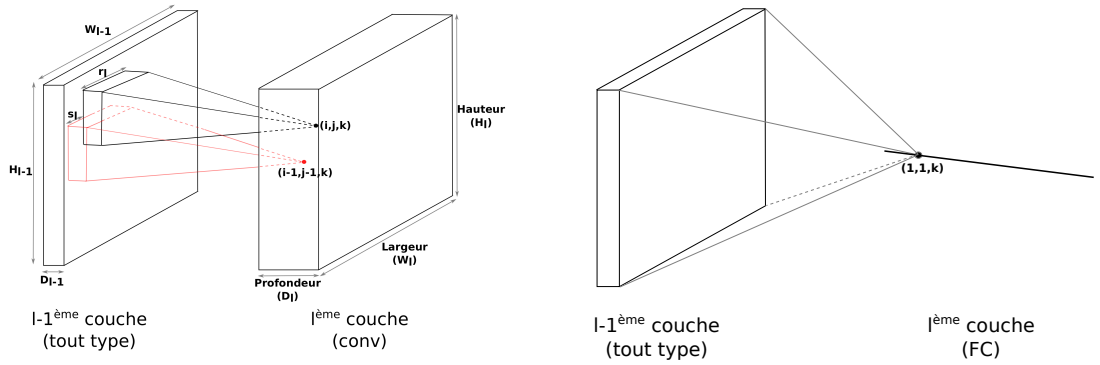


FIGURE 4.4 – Structure du réseau de neurones convolutifs VGG16 (version avec 16 couches)

lement 3 types de couches : convolution (*conv*), unité de rectification linéaire (*ReLU*) et regroupement (*pool*). Bien sûr, il en existe d'autres, mais celles-ci sont suffisantes pour comprendre le fonctionnement global du réseau. Nous employons ici indifféremment le terme *couche* pour désigner une couche seule ou une composition de couches. On peut voir sur la Table 4.1 que *conv1* est une couche composée de trois sous-couches, à savoir dans l'ordre convolution-ReLU-pool. Les dernières couches des deux réseaux considérés ici sont dites *entièrement connectées* (FC), ce qui veut dire que chaque neurone est connecté à tous les neurones de la couche précédente. Ces couches sont en fait de type convolution avec des noyaux de convolution de taille unitaire. Nous allons rapidement décrire chaque type de couche :

– **Couche de convolution :** Comme leur nom l'indique, ces couches servent à appliquer des filtres de convolutions aux neurones de la couche précédente. La profondeur correspond au nombre de filtres de convolution (ou noyau) de même taille *r* appliqués à la couche d'entrée avec un décalage de *s* neurones (ou pixels si la couche d'entrée est l'image RGB). Nous avons représenté sur la figure 4.5a deux neurones de la couche l qui ont pour valeur le résultat de la convolution du $k^{\text{ème}}$ noyau de la couche l appliquée sur des régions de la couche précédente indiquées ici par des cônes. Chacune de ces régions est de taille r_l (la taille des filtres de convolution) et elles se superposent en fonction du décalage s_l : le décalage d'un neurone en largeur (de $j-1$ à j) sur la couche l correspond à un décalage de s_l sur la région considérée dans la couche d'entrée. Nous supposons ici les noyaux de convolutions, ainsi que les



(a) Couche de convolution (b) FC

FIGURE 4.5 – Description d'une couche de convolution et du cas particulier entièrement connecté. Dans les deux cas, la couche gauche est l'entrée de la couche droite et peut être de n'importe quel type. La largeur (resp. hauteur) de ces couches peut être calculée selon $W_l = 1 + \frac{W_{l-1} - r_l}{s_l}$

décalages, carrés mais ils peuvent être rectangulaires. La largeur et la hauteur dépendent de la largeur/hauteur de l'entrée, de la taille des noyaux *r* et du décalage *s*. Reprenons l'exemple de *conv1* : nous avons la largeur W_{conv1} telle que

$$s(W_{conv1} - 1) + r = W_{image}$$

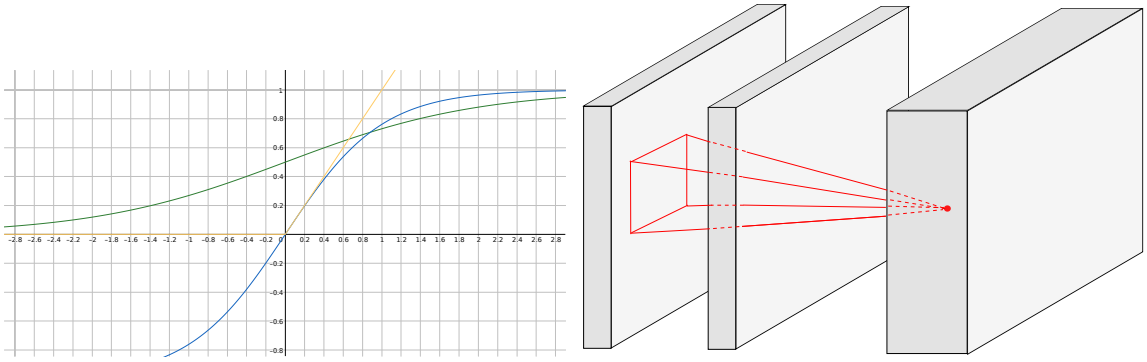
La valeur du neurone en position (i, j, k) de la couche l est le résultat de la convolution par le $k^{\text{ème}}$ filtre de la région de la couche d'entrée centrée sur la position (i', j') :

$$(i', j') = (s(i-1) + 0.5(r-1),\ s(j-1) + 0.5(r-1)). \tag{4.1}$$

– **Couche ReLU :** Elle fait partie de la classe plus générale des couches d'activation qui permettent d'introduire de la non-linéarité dans le réseau. En effet, si un réseau de neurones n'était constitué que d'un enchaînement de couches de convolution, sa sortie ne serait alors qu'une combinaison linéaire des entrées. En introduisant de la non-linéarité, cela permet au réseau d'élargir la famille de fonction qu'il est en capacité d'approximer. Ces couches ont exactement la même dimension que leur couche d'entrée et chaque élément x_{ijk}^{sortie} est défini par

$$x_{ijk}^{sortie} = f(x_{ijk}^{entree}) \tag{4.2}$$

avec f une fonction variant selon la couche envisagée. Les premiers réseaux utilisaient la



(a) fonctions d'activation d'un CNN (b) Champ réceptif

FIGURE 4.6 – (a) Graphique des principales fonctions employées dans les couches d'activation : sigmoïde (vert), tangente hyperbolique (bleu) et ReLU (orange) (b) Illustration du champ réceptif d'un neurone

fonction sigmoïde σ [3] ou la fonction tangente hyperbolique $tanh$.

Comme on peut le voir à la figure 4.6a qui compare les principales fonctions f rencontrées, le défaut de ces deux fonctions est qu'elles ont un faible gradient pour des valeurs d'activation (x_{ijk}^{entree}) élevées en valeur absolue. L'apprentissage des réseaux de neurones étant basé sur la propagation de gradient, elles ont fait place à la fonction ReLU qui est définie simplement par :

$$ReLU : x \rightarrow max(0, x) \tag{4.3}$$

L'avantage est de toujours avoir un gradient important lorsque qu'un neurone est actif (c'est-à-dire avec une activation positive). L'inconvénient est que l'apprentissage, basé sur la descente de gradient, ne peut pas se faire sur des données pour lesquelles l'activation est nulle (point de non dérivabilité).

– **Couche Pool :** Les couches pool sont là pour renforcer l'invariance à la translation et permettent également de réduire la taille des couches du réseau. Chacune de leur valeur correspond à une statistique locale [4] calculée sur une région de la couche d'entrée. Par exemple, AlexNet et VGGNet utilisent des couches pool conservant le maximum local. D'autres réseaux (dont AlexNet sur ces deux premières couches) sont également composés d'une couche de normalisation locale (LNR) qui, comme son nom l'indique, normalise les valeurs d'entrée localement.

3. $\forall x,\ \sigma(x) = \frac{1}{1+e^{-x}}$
4. Moyenne, médiane ou encore maximum

Couche	Sous-couche	Dimension	Taille Kernel / Décalage
Conv1	C + R	$55 \times 55 \times 96$	$11, 4$
	P + L	$27 \times 27 \times 96$	$3, 2$
Conv2	C + R	$27 \times 27 \times 256$	$5, 1$
	P + L	$13 \times 13 \times 256$	$3, 2$
Conv3	C + R	$13 \times 13 \times 384$	$3, 1$
Conv4	C + R	$13 \times 13 \times 384$	$3, 1$
Conv5	C + R	$13 \times 13 \times 256$	$3, 1$
	P	$6 \times 6 \times 256$	$3, 2$
FC6	FC + R	$1 \times 1 \times 4096$	$6, 1$
FC7	FC + R	$1 \times 1 \times 4096$	$1, 1$
FC8*	FC	$1 \times 1 \times 1000$	$1, 1$

(a) AlexNet

Couche	Sous-couche	Dimension	Taille Kernel / Décalage
Conv1	C + R (conv1_1)	$224 \times 224 \times 64$	$3, 1$
	C + R (conv1_2)	$224 \times 224 \times 64$	$3, 1$
	P	$112 \times 112 \times 64$	$2, 2$
Conv2	C + R (conv2_1)	$112 \times 112 \times 128$	$3, 1$
	C + R (conv2_2)	$112 \times 112 \times 128$	$3, 1$
	P	$56 \times 56 \times 128$	$2, 2$
Conv3	C + R (conv3_1)	$56 \times 56 \times 256$	$3, 1$
	C + R (conv3_2)	$56 \times 56 \times 256$	$3, 1$
	C + R (conv3_3)	$56 \times 56 \times 256$	$3, 1$
	P	$28 \times 28 \times 256$	$2, 2$
Conv4	C + R (conv4_1)	$28 \times 28 \times 512$	$3, 1$
	C + R (conv4_2)	$28 \times 28 \times 512$	$3, 1$
	C + R (conv4_3)	$28 \times 28 \times 512$	$3, 1$
	P	$14 \times 14 \times 512$	$2, 2$
Conv5	C + R (conv5_1)	$14 \times 14 \times 512$	$3, 1$
	C + R (conv5_2)	$14 \times 14 \times 512$	$3, 1$
	C + R (conv5_3)	$14 \times 14 \times 512$	$3, 1$
	P	$7 \times 7 \times 512$	$2, 2$
FC6	FC + R	$1 \times 1 \times 4096$	$7, 1$
FC7	FC + R	$1 \times 1 \times 4096$	$1, 1$
FC8*	FC	$1 \times 1 \times 1000$	$1, 1$

(b) VGG

TABLE 4.1 – Topologie des réseaux AlexNet et VGG. Nous utilisons la notation suivante pour les couches : "C" pour convolution, "R" pour ReLU, "L" pour LNR et "P" pour pool.
* : dans les deux cas, FC8 est la couche de classification dont chaque sortie est la probabilité de la classe correspondante. Nous n'utilisons pas cette couche.

Revenons sur des caractéristiques plus globales des CNN. On définit le *champ réceptif* d'un neurone comme étant l'ensemble des neurones de la couche d'entrée (pixels de l'image ici) auxquels il est indirectement connecté. Ceci est illustré à la figure 4.6b. Il augmente avec la profondeur des couches de convolution du réseau. Cette notion peut être exploitée dans des approches multi-échelles. Un exemple est proposé par Ma [MHYY15] dans une application de tracking basée sur les filtres de corrélations. L'utilisation de couches supérieures (champ réceptif important) du réseau permet d'avoir une estimation grossière qui sert ensuite d'initialisation à l'estimation basée sur la couche précédente ayant un champ réceptif plus réduit.

Il est à noter que les couches FC, ayant comme particularité d'avoir chaque neurone entièrement connecté à la couche précédente, ont leur champ réceptif égal aux dimensions de la couche d'entrée. Ceci entraîne la perte des propriétés liées à la spatialité. Les descripteurs issus de ces couches sont donc des représentations plus abstraites et robustes aux variations de positions spatiales. Cela est confirmé par les travaux de Simonyan [SVZ13] et Mahendran [MV15] qui proposent une méthode permettant de visualiser la représentation de chaque classe apprise aux niveaux des différentes couches (figure 4.7)

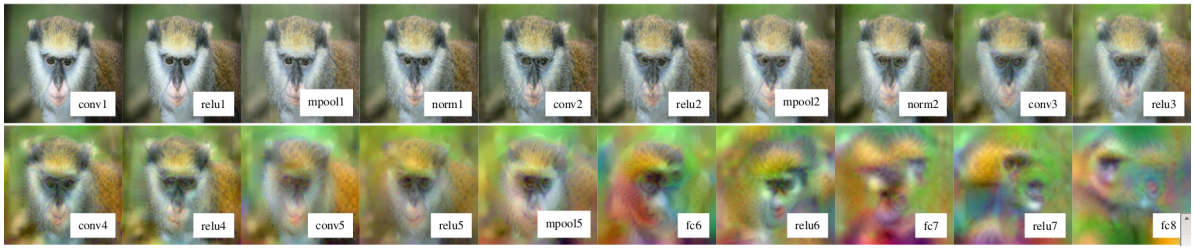


FIGURE 4.7 – Interprétation des couches de CNN par reconstruction de l'image d'entrée. Extrait de [MV15]

Une autre propriété intéressante soulevée par Szegedy [SZS+13] est que la base canonique associée à chaque sortie de couche n'est pas meilleure qu'une base choisie aléatoirement. Autrement dit, les axes de la base canonique ne démêlent pas les facteurs de variation : chaque direction de l'espace correspond à une certaine variation dans l'espace d'entrée qu'est l'image.

Les explications précédentes concernent la structure et l'inférence dans les réseaux de neurones. L'apprentissage des nombreux poids θ (éléments des filtres de convolution) se fait comme pour la plupart des méthodes de machine learning par une descente de gradient sur une fonction coût J. On cherche à maximiser le maximum de vraisemblance :

$$\prod_{i=1}^{m} p(f(x_i;\theta) = y_i|x_i;\theta) \tag{4.4}$$

où x_i, y_i désigne les données d'entraînement accompagnées de leur label et f la fonction représentant le réseau. C'est équivalent à vouloir minimiser J défini par

$$J(\theta) = -\sum_{i=1}^{m} L(f(x_i;\theta), y_i) = -\sum_{i=1}^{m} log(p(f(x_i;\theta) = y_i|x_i;\theta))$$

Dans le cas des réseaux présentés ici, la probabilité $p(f(x_i;\theta) = y_i|x_i;\theta)$ est donnée par la dernière couche $softmax$.

La difficulté de l'optimisation vient de la non linéarité du réseau ainsi que du nombre important de données d'entraînement. La mise à jour des poids en elle-même est effectuée par un algorithme de descente de gradient stochastique (SGD). A chaque itération, l'espérance du gradient est estimée (sans biais) à partir de la moyenne des gradients sur un minibatch de données choisi aléatoirement (taille n)

$$\nabla_\theta J(\theta) = -\frac{1}{n}\sum_{i=1}^{n} \nabla_\theta L(f(x_i;\theta), y_i)$$

Chaque gradient $\nabla_\theta L(f(x_i;\theta), y_i)$ par rapport aux poids du réseau est calculé à l'aide de l'algo-

rithme de rétropropagation de gradient, qui applique récursivement la dérivation des fonctions composées (*chain rule*) dans un ordre optimal. C'est l'élément clé qui permet l'exploitation de larges données d'entraînement. Nous renvoyons le lecteur sur le livre de Goodfellow *et al.* [GBC16] pour une description détaillée.

Dans la suite, nous ferons également référence au réseau GoogLeNet [SLJ+15]. Il possède deux principales différences avec les réseaux précédents :

– L'utilisation de module d'*inception*, en place des couches de convolutions simples, dont la sortie est la concaténation de convolutions par des filtres de différentes tailles (1, 3 et 5) (figure 4.8). En faisant ainsi, on évite le choix arbitraire de la taille des filtres. A noter la présence de filtre 1x1 avant chacun des autres filtres : ils sont là pour réduire la dimension de l'entrée à convoluer (limitation du nombre de calcul).

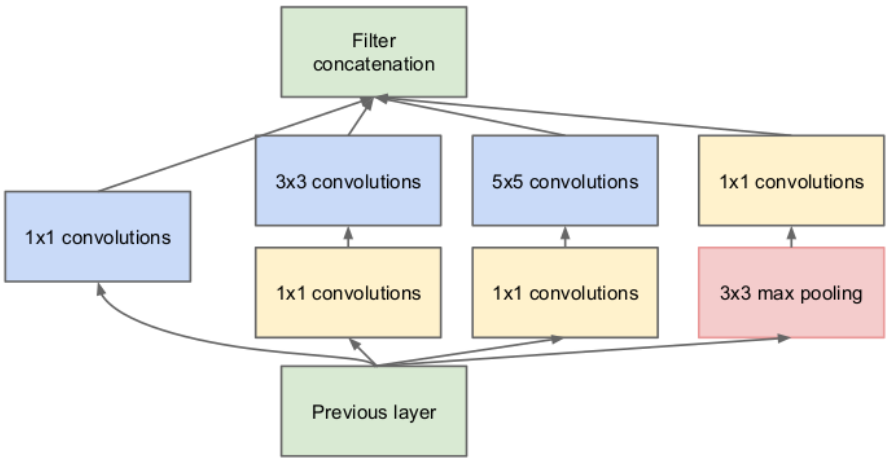


FIGURE 4.8 – Schéma d'un module d'Inception employé par le réseau GoogLeNet. Extrait de [SLJ+15]

– L'ajout de classifieurs auxilliaires connectés à des couches intermédiaires. Ces classifieurs sont constitués de deux couches entièrement connectées suivi par une couche d'activation *softmax*. Ceux ci ne sont présents que lors de la phase d'entraînement, afin d'encourager les poids des couches intermédiaires à devenir plus discriminants. Le tableau ci-dessous présente la structure du réseau. Les couches auxquelles ont été rattaché un classifieur sont encadrées en rouge.

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

FIGURE 4.9 – Structure de GoogLeNet. Extrait de [SLJ+15]

4.2.2 Classification binaire et reconnaissance d'instance

Après la sélection des descripteurs, il nous faut choisir la méthode de classification. La plupart des travaux actuels se font sur la base d'un ensemble **fermé** de classes. Par exemple, de nouveaux réseaux de plus en plus performant dans les tâches de classification, de localisation et de détection ont été conçus dans le cadre de challenges (eg. ILSVRC), basés sur des datasets fixés : c'est le cas d'AlexNet et de VGG qui ont été initialement entraînés sur 1000 classes issues d'ImageNet. Nous nous plaçons dans nos travaux dans un monde ouvert où le nombre de classes n'est ni constant ni borné. Il n'est donc pas envisageable d'utiliser des classifieurs multi-classes à cause de la mise à l'échelle et de la nécessité de les ré-entraîner lors de l'ajout d'une nouvelle classe.

Nous avons choisi les Random Forests (RF) [Bre01] comme classifieur binaire de base associé à chaque concept de notre base de connaissance. Ce classifieur est un ensemble d'arbres de décision aléatoire (d'où le terme de "forêt"), avec pour hyperparamètres principaux le nombre d'arbres n et leur profondeur d. Chaque arbre est construit indépendamment des autres de la façon suivante : à chaque nœud, un sous-ensemble aléatoire des données d'entraînements est sélectionné ainsi qu'un sous-ensemble des descripteurs (ie des coordonnées des vecteurs de données). Un classifieur linéaire est appris en choisissant la séparation selon les axes des coordonnées maximisant le gain d'information ou de manière équivalente réduisant au maximum l'entropie après la classification. Autrement dit, le but est d'avoir dans les nœuds fils des distributions de points les plus homogènes

possible (ne contenant qu'une seule classe). A la fin, on associe à chaque nœud feuille une distribution de probabilité égale aux proportions de chaque classe dans celui-ci. La probabilité qu'une donnée test x^* soit de la classe c est alors la moyenne des probabilités p_i données par chaque arbre de décision :

$$p(c|x^*) = \frac{1}{n}\sum_{i=1}^{n} p_i(c|x^*) \tag{4.5}$$

Il serait envisageable utiliser des classifieurs SVM (Machine à vecteurs de support) mais les classifieurs RFs offrent plusieurs avantages :

- Ils donnent un résultat directement sous forme de probabilité. Dans le cas des SVMs, on doit se contenter de la distance à l'hyperplan séparateur comme indicateur de confiance.
- Ils permettent de trouver des séparations non linéaires. Cela est possible également avec les SVM en utilisant des noyaux (kernel), mais là encore se pose le problème du choix du noyau ainsi que de ses hyperparamètres.

Les RFs sont entraînés à partir d'images obtenues sur Internet, en particulier provenant de la base de données ImageNet [DDS+09]. Les images négatives sont prises à partir de classes sémantiquement éloignées et de classes ayant un parent commun.

Maintenant intéressons nous non plus à la tâche de classification mais à celle de mise en correspondance d'instances ("template matching"). On cherche à reconnaître une instance particulière à partir d'une observation précédemment faite. Ici nous n'avons accès à aucune source extérieure d'entraînement. Nous utilisons simplement la similarité cosinus S_C entre les descripteurs image $\phi_1,\ \phi_2$ de deux instances pour mesurer leur correspondance :

$$S_C(\phi_1, \phi_2) = cos(\theta) = \frac{\phi_1^T \phi_2}{||\phi_1||_2||\phi_2||_2} \in [0, 1] \tag{4.6}$$

où θ correspond à l'angle entre les vecteurs ϕ_1 et ϕ_2.

4.2.3 Application à la localisation et à la détection

Les réseaux de neurones convolutifs ont été conçus à la base pour les tâches de classification. Des travaux récents ont étendu leur application aux tâches de localisation et de détection [5] avec

5. La localisation consiste à fournir la position d'un objet en plus de sa classe. La détection étend cette tâche en supposant un nombre quelconque (pouvant être nul) d'objets dans l'image, pénalisant les faux positifs.

principalement OverFeat [SEZ+13], R-CNN [GDDM14] [Gir15] [RHGS15] ou encore FCN (Fully Connected Network) [LSD15].

Sermanet [SEZ+13] propose le système OverFeat, qui étend l'application d'un réseau de neurones convolutifs en l'appliquant sur des fenêtres glissantes à différentes échelles. Les régions délimitant un objet (bounding box) sont inférées à partir d'une couche intermédiaire. Les descripteurs issus de cette couche sont ensuite propagés en entrée d'un réseau constitué de deux couches entièrement connectées et d'une couche de sortie à quatre unités (coordonnées du rectangle de détection). Il y a un régresseur de ce type par classe.

R-CNN [GDDM14] décompose lui le problème en deux étapes : une proposition de région correspondant à un objet puis l'application d'un CNN sur ces régions. Leur méthode générale ne dépend pas de l'algorithme de proposition de régions. Ils utilisent un CNN préentraîné sur ImageNet, qui peut ensuite être raffiné en se basant seulement sur des régions d'images. La détection se fait non pas directement à partir de la couche finale de leur CNN mais en entraînant un classifieur SVM binaire pour chaque classe avec pour descripteur image la sortie d'une couche intermédiaire. Ils utilisent pour cela les régions de la réalité terrain comme positifs et les régions suffisamment disjointes de celles-ci comme négatifs. Dans [Gir15], le même auteur propose de partager les descripteurs issus des couches du CNN à toutes les régions objets. Ceci est fait en considérant la région d'une couche intermédiaire correspondant à la région au niveau de l'image (cf la notion de champ réceptif). Une dernière amélioration [RHGS15] propose également d'intégrer la proposition de région objet dans un réseau de neurones, en partageant les couches de convolution avec le réseau responsable de la classification.

Enfin Long [LSD15] remarque que l'on peut convertir les couches entièrement connectées FC en couche de convolution [6], permettant de voir n'importe quel CNN comme un filtre de convolution de même taille que ses entrées (227 pour AlexNet par exemple). Le fait de ne pas recalculer les descripteurs des couches de convolution à chaque région permet une meilleure efficacité. On obtient donc une classification sur une grille régulièrement séparée et recouvrante de l'image.

Une classification dense par pixel peut être obtenue en fusionnant la prédiction de couches inférieures (échelle plus petite) avec la prédiction finale. Pour se faire, il faut sur-échantillonner les couches inférieures avec des couches de *déconvolution* (opération inverse de la convolution). Les poids associés à ces couches peuvent être fixés (eg. suréchantillonnage bilinéaire) ou appris.

6. En fait, ce sont des couches de convolution de noyau 1×1

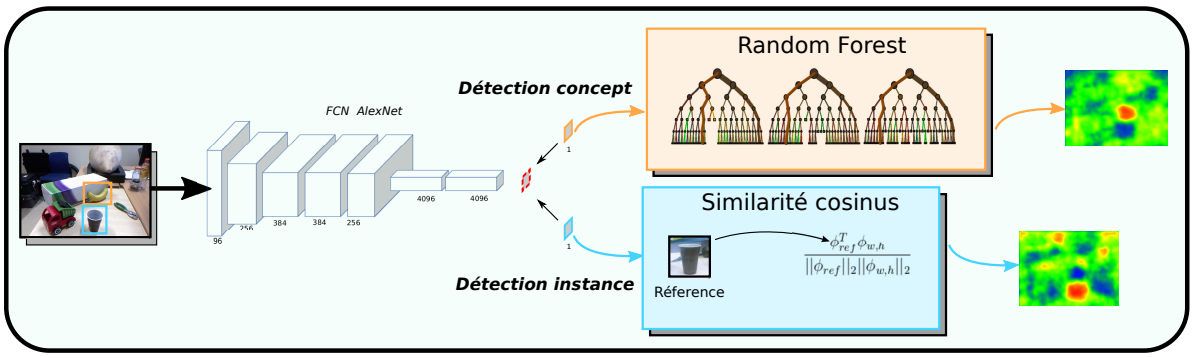


FIGURE 4.10 – Adaptation d'un FCN avec classifieur binaire RF et similarité cosinus

Nous notons par la suite $fc*\_conv$ les couches $fc*$ après conversion en couches de convolution classique.

Il faut noter également que les dimensions de ces couches ne sont plus fixes : elles dépendent de la taille de l'image d'entrée. Deux éléments voisins (± 1 en largeur/hauteur) de ces couches correspondent, sur l'image d'origine, à deux régions de tailles égales à l'image d'entrée du CNN d'origine (227 pour AlexNet, 224 pour VGG). Le décalage s entre ces régions dépend des décalages des couches de convolutions précédentes. Par exemple, pour la couche $fc7$ d'AlexNet, on a $s = 32$. De là, on peut en déduire le nombre de régions minimum recouvrant l'image et donc la taille finale de la couche. Ainsi, pour un FCN basé sur AlexNet, une image 611×483 donnera une couche $fc7\_conv$ de dimensions $12 \times 8 \times 4096$. Du zéro-padding est utilisé lorsqu'une région "sort" des frontières de l'image. Le zéro-padding est également employé dans les couches de convolution afin d'agrandir la couche d'entrée, permettant ainsi de contrôler indépendamment la taille des noyaux et de la sortie. Sans cela, la taille des couches du réseau diminuerait trop rapidement imposant alors soit de limiter fortement le nombre de couches, soit de n'utiliser que de petits noyaux.

Parmi les méthodes présentées ci-dessus, le FCN nous a paru la plus appropriée à notre système. La raison principale est qu'elle ne nécessite pas de données d'entraînement supplémentaires : elle est directement applicable à n'importe quel CNN préentraîné. Les méthodes basées sur de la proposition d'objet non supervisée (comme la première version de R-CNN) semblent moins robuste. En effet, la proposition d'objet dans une scène se rapproche de la problématique de saillance et de segmentation de scène. Comme discuté plus tôt dans cette section, ce sont des problèmes difficiles et les résultats, lorsque le cadre d'application est ouvert (pas d'hypothèse sur les scènes observées), sont très variables.

Nous avons adapté l'approche FCN afin de pouvoir détecter un concept via les classifieurs binaires RF associés à chacun des nœuds de notre modèle de connaissance (figure 4.10). De la même manière, nous pouvons également détecter la présence d'une instance particulière à partir d'un descripteur de référence lui étant attaché. Dans un FCN classique, la dernière couche ($fc8$ pour AlexNet et VGG) correspond à des régions recouvrantes auxquelles sont associé un vecteur de probabilité, chaque coordonnée correspondant à une classe. Nous la remplaçons par une couche Random Forest de même hauteur et largeur que la couche précédente $fc7\_conv$ mais de profondeur 1 (la probabilité d'être de la classe représentée par le RF). On peut ensuite modifier le Random Forest employé en fonction de la classe à rechercher sans avoir à recharger le FCN en mémoire GPU.

On utilise exactement le même principe pour la recherche d'instance. On remplace la couche $fc8$ par une couche de similarité cosinus, également de même hauteur et largeur que $fc7\_conv$ et de profondeur 1. Le seul paramètre de cette couche est le descripteur de référence. En pratique, il est modifiable sur un FCN préchargé en mémoire.

Dans la suite, l'utilisation des descripteurs CNN dans nos travaux sera justifiée en analysant certaines de leur propriétés. Tout d'abord, nous montrerons que ces descripteurs sont bien adaptés à la représentation d'instance (réalisation physique d'une classe). Pour cela, différentes causes de variations de ces descripteurs dans le cas d'objets supposés rigides seront analysées

- Dans un premier temps, les variations entre objets d'une même classe seront analysées à partir des similarités intra-classe.
- Dans un second temps, nous inspecterons une autre cause de variation de ces descripteurs liée aux transformations de similarité $\mathbf{SIM}(\mathbf{2})$ [7]. Malgré la forte non-linéarité des CNN, on observera que la variation des descripteurs reste relativement régulière. Nous proposerons ensuite des exploitations de cette propriété. Nous finirons sur une formalisation sous forme matricielle dans le but de justifier et de mieux exploiter cette propriété des réseaux.

Chacun de ces résultats sera comparé en fonction de la profondeur à laquelle sont extraits les descripteurs et pour différents réseaux (AlexNet, VGGNet et GoogLeNet).

4.2.4 Descripteur image générique pour la représentation d'instance

Un certain nombre de nos processus (cf figure 1.1) nécessitent l'utilisation de descripteurs image (classification, tracking, représentation d'instance). On peut trouver un ensemble riche de méthodes pour chacun de ces processus avec des descripteurs image différents tels que SIFT (Scale Invariant

7. $\mathbf{SIM}(\mathbf{2})$ est composé des translations, rotations et mise à l'échelle isotropique du plan.

	WordnetId	Nombre d'images
Paper-cup (Gobelet)	n03216710	125
Cup (Tasse)	n03147509	153
Car (Voiture)	n02958343	839
Dog (Chien)	n02084071	609
Apple (Pomme)	n07739125	409
Cat (Chat)	n02121620	381
Plate (Assiette)	n03960374	151
Knife (Couteau)	n03623556	138

TABLE 4.2 – Classes utilisées pour nos expériences sur la représentation d'instance par descripteurs issus de CNN. On se limite à un sous-ensemble de $n = 120$ images pour les similarités extra-classes. Les images de la classe *Plate* contiennent un certain nombre d'outliers correspondant à un homonyme de *Plate* (sens de plaque d'immatriculation)

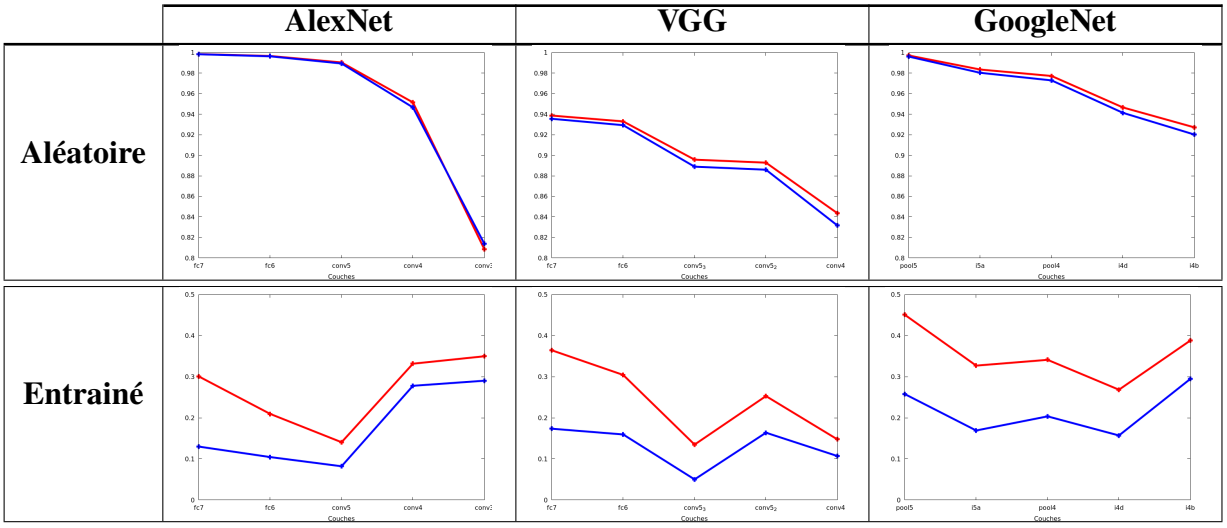


FIGURE 4.11 – Moyenne des similarités intra-classe (—) et extra-classes(—) pour chaque couche des trois CNNs AlexNet, VGG et GoogLeNet

Feature Transform) [Low99] ou HoG (Histogram of Oriented Gradients) [DT05]. Plus récemment, les descripteurs extraits de couches intermédiaires de CNN ont prouvé être suffisamment génériques pour être utilisés dans des tâches pour lesquels ils n'ont pas été entraînés [DJV+13] [RASC14] [HBKM15]. Cependant, nous sommes intéressés en particulier par leur usage pour la représentation d'instance. L'efficacité de tels descripteurs pour l'apprentissage de classifieur est un fait reconnu : les descripteurs ainsi appris maximisent la distance inter-classes tout en minimisant la distance intra-classe.

Notre intérêt se porte ici sur les distances intra-classe. Sont-elles suffisamment faibles pour per-

mettre la classification mais également suffisamment élevées pour pouvoir séparer deux instances d'une même classe ? Comment ces distances varient-elles selon la profondeur des couches et selon l'architecture des réseaux ? C'est ce à quoi nous allons chercher à répondre dans cette partie. Pour cela, nous considérons un ensemble de classes avec pour chaque image la vraie position (bounding box) de l'objet à partir d'ImageNet (cf Table 4.2). Nous considérons ensuite les trois CNNs suivants (avec leurs dernières couches) :

- AlexNet : fc7, fc6, conv5, conv4, conv3
- VGG : fc7, fc6, conv5_3, conv5_2, conv4_3
- GoogleNet : pool5, inception_5a, pool4, inception_4d, inception_4b

Pour chaque image, le descripteur issu de chaque couche est extrait et normalisé. Les similarités cosinus sont ensuite calculées entre les pairs de descripteurs de même classe et de classes différentes. La figure 4.11 représente sous forme de graphique la moyenne des similarités obtenues, dans le cas où les réseaux ont leur poids initialisés aléatoirement et dans le cas entraîné.

Le principal résultat contre-intuitif de l'analyse de ces courbes est que **la similarité minimale est atteinte pour la dernière couche de convolution**. On voit que pour chaque réseau entraîné, la similarité minimale est atteinte pour la dernière couche de convolution à savoir *conv5* pour AlexNet et *conv5_3* pour VGG. GoogLeNet est un peu différent, car il a deux couches intermédiaires (inception_4a et inception_4d) qui, lors de la phase d'entraînement, sont rattachées à des couches entièrement connectées et à une fonction de coût. On voit que dans ce cas aussi, le minimum est atteint pour les deux couches *inception_5a* et *inception_4b*.

Ce résultat est surprenant. En effet, on peut s'attendre à avoir des valeurs de similarités qui diminuent avec la réduction du champ réceptif en descendant dans les couches inférieures d'un réseau. C'est ce qu'on a cherché à confirmer en reprenant les mêmes calculs mais avec des réseaux dont les poids ont été fixés aléatoirement (1$^{\text{ère}}$ ligne de la figure 4.11). On observe bien la diminution de similarité en descendant vers les couches inférieures.

Ceci semble donc indiquer que la dernière couche de convolution d'un réseau est la plus appropriée pour la représentation d'instance. De plus, parmi les trois réseaux considérés ici, VGG est celui pour lequel la similarité est minimum. GoogLeNet, quant à lui, donne des similarités plus élevées. Dans les applications de la littérature pour lesquelles il est souhaitable d'avoir des similarités maximums entre objets de même classe, la performance des descripteurs est liée à la profondeur du réseau [GBC16] [PLR$^+$16]. Or ici on souhaite minimiser la similarité entre instances d'une même classe tout en profitant de la robustesse des descripteurs CNNs. Autrement dit, il est cohérent d'obtenir un meilleur compromis avec des réseaux n'ayant pas une trop grande capacité. Ce constat sera

également vérifié dans les sections 4.2.5 et 4.2.6 étudiant la variation de ces descripteurs selon des transformations de similarité.

Nous ne pouvons cependant qu'émettre des hypothèses quant à la justification théorique de ce phénomène. La singularité de la dernière couche de convolution est qu'elle est la dernière ayant encore une information de spatialité. Il ne serait donc pas absurde d'y observer des phénomènes particuliers lors de la rétropropagation du gradient durant la phase d'entraînement. Les descripteurs, et donc leur similarité, ne sont pas que fonction de l'instance considérée. Si l'on considère une instance fixe mais avec une pose différente, on obtiendra un descripteur différent. En particulier, il a été montré dans la littérature la sensibilité des CNNs à la rotation et leur relative invariance à la translation et au changement d'échelle via les opérations de convolutions et de pooling. Nous nous intéressons donc dans la suite aux variations de ces descripteurs selon des transformations de similarité.

4.2.5 Variation des descripteurs CNN relativement à SIM(2)

Dans la littérature récente, un certain nombre de travaux se sont intéressés aux variations des descripteurs CNNs via les notions d'équivariance et d'invariance [LV15] [CW16] [HV16]. Un descripteur $\phi \in \mathbb{R}^d$ est dit équivariant à une transformation g s'il existe une application $M_g : \mathbb{R}^d \to \mathbb{R}^d$ telle que pour toute image d'entrée x

$$\phi(gx) \approx M_g(\phi(x)) \tag{4.7}$$

L'invariance à une transformation g est simplement un cas particulier où $M_g = Id$.

Lenc [LV15] montre l'équivariance des descripteurs CNNs d'AlexNet à partir de méthodes empiriques. Il propose pour cela d'estimer l'application M_g, supposée affine, pour des ensembles de transformations g fixés : ensemble discret de rotation de 0 à $90°$, retournement d'image verticale. Cohen [CW16] généralise l'opération de convolution (équivariant pour la translation) en l'élargissant aux groupes de symétrie constitués des translations, rotations à $90°$ et réflexions. Dans ces travaux, les transformations considérées sont limitées à de petits ensembles discrets. Pour passer outre cette limitation, Henriques [HV16] propose une version modifiée de la convolution (*warped convolution*) (figure 4.12). Elle consiste à appliquer une déformation fixe, dépendante de la famille de transformation considérée, sur l'image d'entrée de telle sorte qu'après une simple convolution la sortie soit équivariante à ces transformations. La méthode reste cependant limitée à des transformations à 2 degrés de liberté (rotation/échelle ou échelle/rapport d'aspect par exemple).

Notre analyse va complémenter les propriétés citées en montrant expérimentalement la régu-

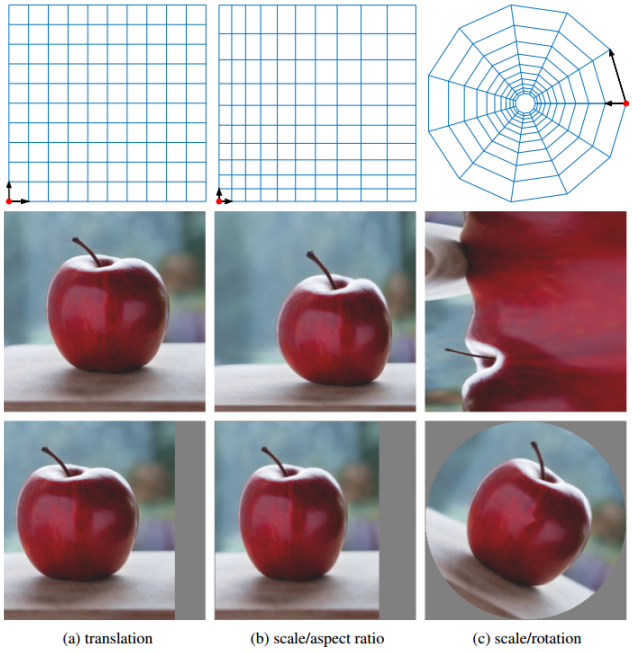


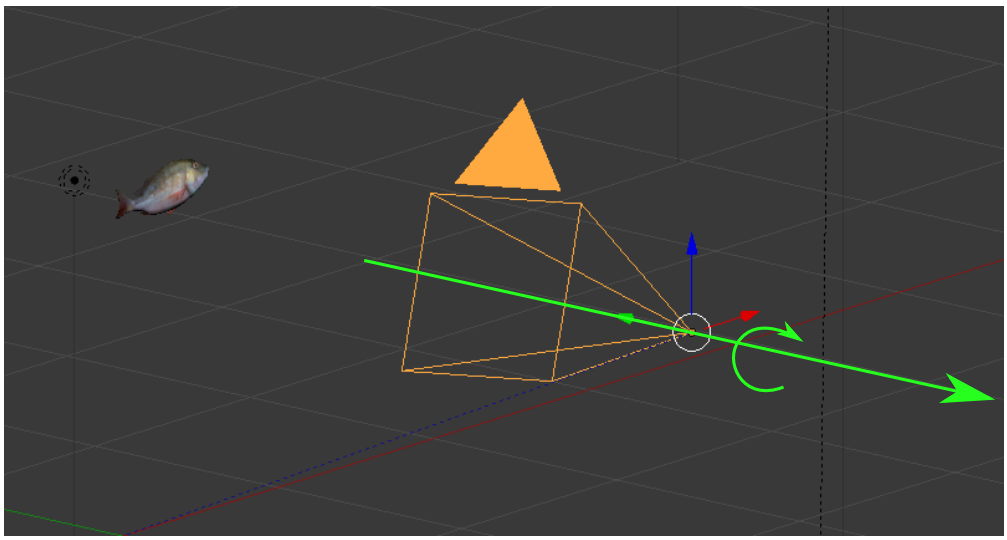
FIGURE 4.12 – Illustration des *warped convolution*. Extrait de [HV16]

larité des variations des descripteurs par les transformations de similarité qui forment le groupe $\mathbf{SIM}(\mathbf{2})$. L'exploitation de cette régularité dans une application de suivi d'objet sera présentée à la section suivante.

Pour notre analyse, nous considérons quatre objets observés par une caméra en mouvement de telle sorte que leur projection dans l'image subisse une transformation de similarité (voir figure 4.13). Nous extrayons à chaque image de ces séquences les descripteurs image correspondants à la région vérité terrain de l'objet (bounding box rectangulaire). Nous effectuons ensuite une décomposition en composantes principales (ACP) et traçons les courbes engendrées par leur projection sur les deux premiers axes. Les résultats sont visibles à la figure 4.14.

En première analyse, ces courbes confirment les résultats obtenus par Poole [PLR$^+$16] qui montrent la complexification des variétés propagées dans un CNN. On remarque que les couches de convolution inférieures sont plus régulières contrairement aux couches entièrement connectées, qui n'ont plus de propriétés spatiales. Dans le cas de GoogleNet, les différentes couches sont relativement irrégulières. Cela pourrait s'expliquer non seulement par la profondeur (qui est proche de celle de VGG) mais par l'ajout de fonction coût intermédiaire contraignant les couches inférieures, lors de l'apprentissage, à représenter les frontières de décisions (complexe).

On remarque aussi que la variété générée par les descripteurs obtenus après transformation est



(a) La caméra se déplace selon l'axe -y (en vert) tout en tournant autour de ce même axe

(b) Tête (c) Livre (d) Poisson (e) Tasse

FIGURE 4.13 – Configuration générale de la simulation

de dimension 1. En effet, bien que le groupe des transformation de similarité soit une variété de dimension 4, les transformations sont ici fonctions d'une unique variable : l'indice t des images dans la séquence.

La propriété remarquable des courbes produites à partir des couches de convolution est leur régularité. Ce constat empirique semble indiquer que les paramètres de la transformation pourrait être inférés à partir de ces descripteurs : c'est le sujet de la section suivante.

4.2.6 Applications

Les résultats précédents semblent montrer que la régularité par rapport aux transformations de similarité au niveau de l'image se conserve lors de la propagation dans le réseau (avec une dégradation attendue lors du passage aux couches FC entièrement connectées). Nous allons maintenant confirmer ces résultats à travers des applications simples. Le but ici n'est pas d'obtenir des appli-

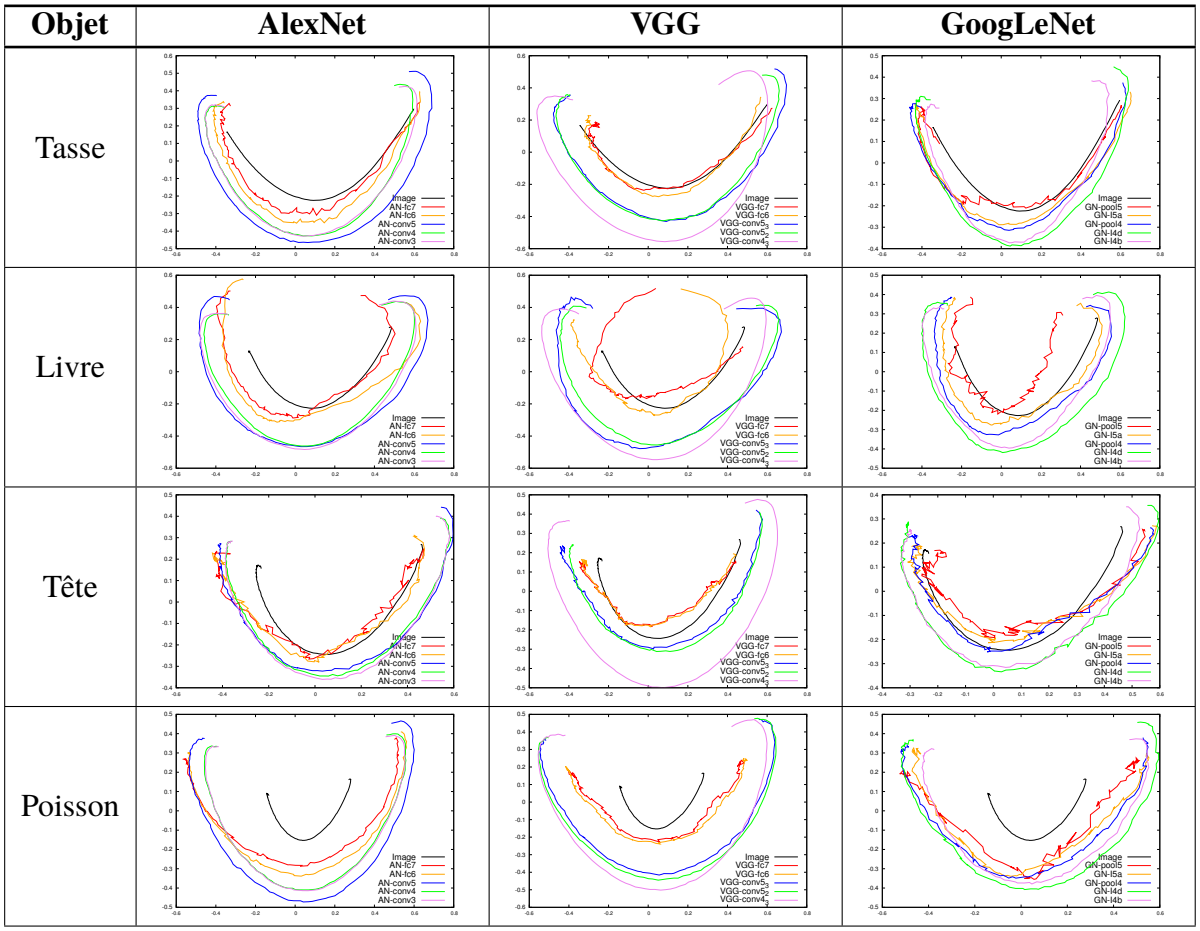


FIGURE 4.14 – Projections sur les deux premières composantes principales de l'ACP pour chaque séquence d'images

cations en compétition avec les méthodes de l'état de l'art mais de mettre en avant les observations de la section 4.2.5. Le développement de méthodes plus approfondies fait cependant partie de nos perspectives.

Estimation de petites transformations et suivi d'objet

La régularité observée des descripteurs CNN est une propriété intéressante pour l'inférence des paramètres de transformations de similarité et par conséquence pour le suivi d'instance. En effet, en notant $T_t \in \mathbf{SIM}(\mathbf{2})$ la position de l'objet au temps t et $\boldsymbol{\phi}_l(T_t)$ le descripteur correspondant extrait de la $l^{\text{ème}}$ couche, un développement de Taylor au premier ordre donne

$$\boldsymbol{\phi}_l(T_{t+1}) = \boldsymbol{\phi}_l(T_t) + \frac{\partial \boldsymbol{\phi}_l}{\partial T}(T_t)\Delta(T) + o(\Delta(T)),\ \Delta(T) = T_{t+1} - T_t \tag{4.8}$$

On pourrait alors estimer le mouvement d'objet sous condition de petits mouvements via l'algorithme de Gauss-Newton.

Le problème est que les méthodes d'optimisations (Gauss-Newton, Descente de gradient) ont été prévues pour travailler sur des espaces vectoriels euclidiens, ce que n'est pas $\mathbf{SIM}(\mathbf{2})$. Cependant, $\mathbf{SIM}(\mathbf{2})$ étant un groupe de Lie, on a une paramétrisation parfaitement adaptée à ce genre d'optimisation : les coordonnées dans la base canonique de son algèbre $\mathfrak{sim}(2)$. Contrairement à d'autres approches comme par exemple les quaternions, elle n'introduit aucun degré de liberté supplémentaire et ne nécessite l'ajout d'aucune contrainte.

Nous allons revenir sur des définitions élémentaires afin de définir précisément le groupe de similarité $\mathbf{SIM}(\mathbf{2})$ en tant que groupe de Lie. Nous invitons le lecteur à se référer à [LS14] [Hal15] pour plus de détails sur les notions abordées.

Définition 1. ***(Variété)*** *Une variété M de dimension N (ou N-variété) est un espace topologique tel que tous les voisinages V_p d'un point $\mathbf{p} \in M$ sont homéomorphiques à $\mathbb{R}^N$. On peut intuitivement voir les variétés comme une généralisation des surfaces en dimension quelconque ([LS14], chapitre 9).*

Définition 2. ***(Espace tangent à une variété)*** *A tout point $\mathbf{x}$ non singulier d'une N-variété M de $\mathbb{R}^D$ ($D > N$), on associe un espace vectoriel de dimension N dit espace tangent et noté T_xM([LS14], section 9.4). On peut le voir comme l'espace vectoriel engendré par les vecteurs tangent en $\mathbf{x}$ à toutes les courbes différentiables de M passant par $\mathbf{x}$. La figure 4.15 en est une illustration, avec l'espace tangent au point $\mathbf{e}$ dont une base est générée par les vecteurs tangents à φ et ψ.*

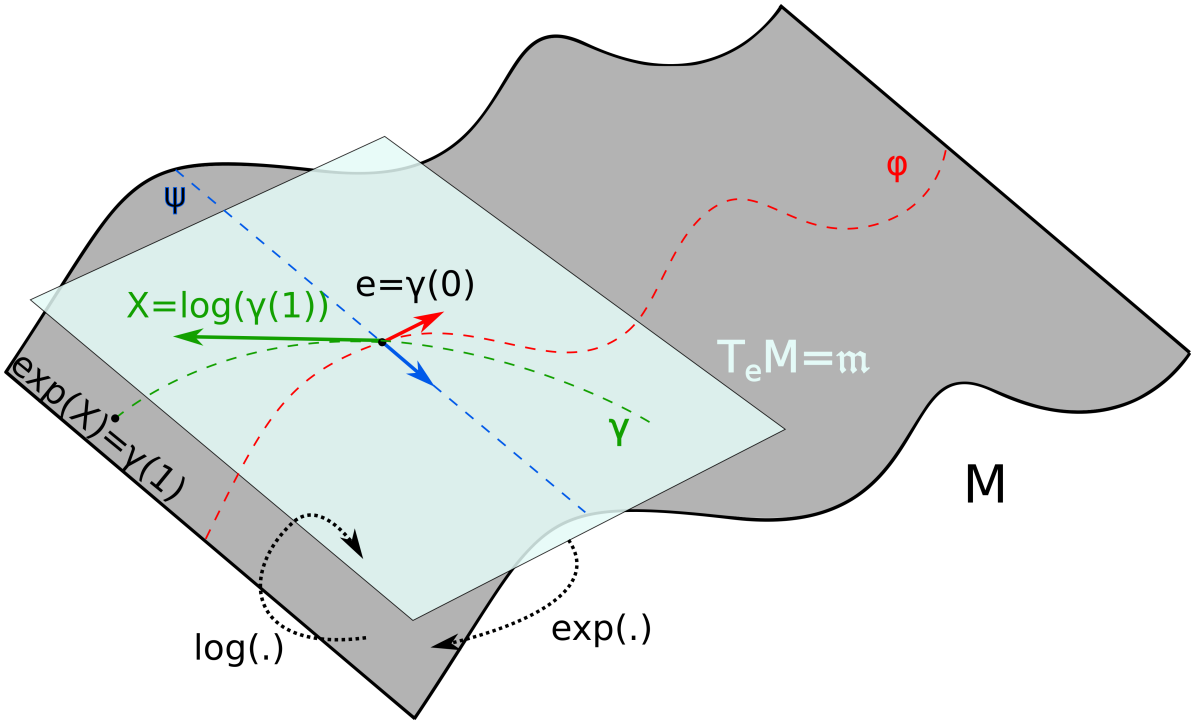


FIGURE 4.15 – Représentation d'une 2-variété M de $\mathbb{R}^3$ avec l'espace tangent T_eM en un point $\mathbf{e}$. φ et ψ sont deux courbes de M passant par $\mathbf{e}$. Leur vecteur tangent en $\mathbf{e}$ forment ici une base de T_eM. Si l'on suppose que M est un groupe de Lie dont $\mathbf{e}$ est l'élément neutre, on a alors que $T_eM = \mathfrak{m}$ est son algèbre de Lie associé. Le passage de l'un à l'autre se fait via les applications exponentielle $exp : \mathfrak{m} \mapsto M$ et logarithme $log : M \mapsto \mathfrak{m}$. γ représente le chemin associé à X défini dans (4.10)

Définition 3. ***(Variété différentielle et lisse)*** *Une N-variété dans l'espace ambiant $\mathbb{R}^D$, $D > N$, est dite différentielle si chaque point $\mathbf{p} \in M$ peut être localement paramétré par une fonction $\phi : \Omega \to U$ avec $\mathbf{p} \in U$, $\boldsymbol{O}_N \in \Omega$ et (Ω, U) étant respectivement des sous-ensembles ouverts de $\mathbb{R}^N$ et M. ϕ doit être un difféomorphisme. Si c'est un difféomorphisme C^∞, on parle alors de variété* ***lisse****.*

Définition 4. ***(Groupe de Lie)*** *Un groupe de Lie G est défini comme une variété lisse ayant une structure de groupe dont l'opérateur produit et son inverse sont différentiables.*

Définition 5. ***(Algèbre de Lie)*** *A tout groupe de Lie G est associé une algèbre [8] de Lie $\mathfrak{g}$. Son produit bilinéaire (antisymétrique) $[.,.] : \mathfrak{g} \times \mathfrak{g} \mapsto \mathfrak{g}$ est nommé crochet de Lie. Un fait remarquable est que $\mathfrak{g}$ est l'espace tangent à l'élément identité $\mathbf{e}$ de G, c'est-à-dire $\mathfrak{g} = T_eG$.*

Nous allons maintenant définir les groupes de Lie matriciels (ou linéaires) qui vont intervenir par la suite.

8. On rappelle qu'une algèbre est un espace vectoriel muni d'une loi de composition interne bilinéaire

Définition 6. ***(Théorème de Cartan)*** *On note $\mathbf{M_n}(\mathbb{R})$ l'ensemble des matrices réelles carrées de dimension $n \times n$ et $\mathbf{GL_n}(\mathbb{R})$ le sous-groupe de $\mathbf{M_n}(\mathbb{R})$ des matrices inversibles. Tout sous-groupe fermé G de $\mathbf{GL_n}(\mathbb{R})$ est un groupe de Lie (ie vérifie les conditions de la définition 4) pour $n \in \mathbb{N}$. En particulier, on les nomme groupe de Lie matriciel.* ***SO(n), SE(n), SIM(n)*** *en sont des exemples.*

Le principal intérêt des groupes de Lie est que l'on peut travailler sur l'algèbre de Lie correspondante qui est plus simple (structure d'espace vectoriel). On ramène ensuite les résultats obtenus sur l'algèbre au groupe en utilisant *l'application exponentielle*. On peut également passer du groupe à l'algèbre avec son inverse : *l'application logarithme*. Ceci est une fois de plus illustré à la figure 4.15.

Définition 7. ***(Application exponentielle)*** *L'application exponentielle envoie les éléments d'une algèbre de Lie $\mathfrak{m}$ vers le groupe de Lie M correspondant*

$$exp : \mathfrak{m} \mapsto M \tag{4.9}$$

On peut déterminer ainsi la structure locale du groupe de Lie. Dans le cas des groupes de Lie matriciels, les algèbres de Lie associées peuvent se définir ainsi

$$\mathfrak{m} = \left\{ X \in M_n(\mathbb{R}) \mid \forall t \in \mathbb{R},\ \gamma(t) = e^{tX} \in M \right\} = T_e M \tag{4.10}$$

avec $\mathbf{e}$ l'élément identité de M. On remarque que $\gamma : \mathbb{R} \mapsto M$ est un chemin de M tel que $X = \frac{d\gamma}{dt}(0)$ ie X est le vecteur tangent à γ en $\mathbf{e}$ [9] comme illustré à la figure 4.15. On rappelle que l'exponentielle d'une matrice carrée A quelconque est définie par la série convergente

$$e^A = \sum_{k=0}^{+\infty} \frac{1}{k!} A^k \tag{4.11}$$

Définition 8. ***(Application logarithme)*** *L'application logarithme envoie des éléments du groupe vers l'algèbre*

$$log : M' \mapsto \mathfrak{m} \tag{4.12}$$

où $M' = Im(exp)$ est le sous-ensemble image de M par l'application exponentielle.

Il est important de noter que l'application exponentielle est de classe C^1. Cependant, il n'y a pas de résultat général sur l'injectivité/surjectivité de cette application et de son inverse. Nous

9. Plus précisément, γ est l'unique sous-groupe à un paramètre (ie vérifiant $\forall s, t \in \mathbb{R},\ \gamma(s+t) = \gamma(s)\gamma(t)$) dont la tangente en $\mathbf{e}$ est X.

allons désormais définir les groupes liés à différentes transformations dont le groupe de similarité $\mathbf{SIM}(\mathbf{2})$.

Définition 9. ***(Groupe Spécial Orthogonal SO)*** *On note* $\mathbf{O}(\mathbf{n})$ *le groupe orthogonal en dimension n. C'est un sous-groupe de* $\mathbf{GL_n}(\mathbb{R})$ *et donc un groupe de Lie matriciel en vertu du théorème 6. Il représente l'ensemble des matrices* $n \times n$ *orthogonales* [10]*. Il en découle que leur déterminant vaut* ± 1*. Les matrices de déterminant* -1 *sont des combinaisons de réflexions et de rotations. Celles de déterminant* 1 *forment le sous-groupe* $\mathbf{SO}(\mathbf{n})$ *qui représente donc les rotations. C'est de même un groupe de Lie matriciel (variété de dimension* $n-1$*). En dimension 2, les éléments* $R_\theta \in \mathbf{SO}(\mathbf{2})$ *sont de la forme*

$$R_\theta = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix},\ \theta \in \mathbb{R} \tag{4.13}$$

avec θ *correspondant à l'angle de la rotation.*

L'algèbre associée $\mathfrak{so}(2)$ *est définie par l'ensemble des matrices antisymétriques* 2×2 *:*

$$\Omega_\omega = \begin{bmatrix} 0 & -\omega \\ \omega & 0 \end{bmatrix} \tag{4.14}$$

Définition 10. ***(Groupe Spécial Euclidien SE)*** *On note* $\mathbf{E}(\mathbf{n})$ *le groupe euclidien en dimension n représentant l'ensemble des isométries de* $\mathbb{R}^n$*. Ce n'est pas un sous-groupe de* $\mathbf{GL_n}(\mathbb{R})$ *mais est isomorphique a un sous-groupe de* $\mathbf{GL_{n+1}}(\mathbb{R})$ *par l'application* [11]

$$\begin{aligned} \mathbf{O}(\mathbf{n}) \times \mathbb{R}^n &\to \mathbf{GL_{n+1}}(\mathbb{R}) \\ R, \mathbf{t} &\mapsto \begin{bmatrix} R & \mathbf{t} \\ \mathbf{0^T} & 1 \end{bmatrix} \end{aligned} \tag{4.15}$$

et est donc également un groupe de Lie matriciel (variété de dimension n*).* $\mathbf{SE}(\mathbf{n})$ *est un sous-groupe de* $\mathbf{E}(\mathbf{n})$ *(et donc un groupe de Lie matriciel) composé des isométries directes (rotation pure et translation). En particulier, (4.15) nous donne l'expression d'un élément* $T \in \mathbf{SE}(\mathbf{2})$

$$T = \begin{bmatrix} R & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix},\ R \in \mathbf{SO}(\mathbf{2}),\ t \in \mathbb{R}^2 \tag{4.16}$$

avec $\mathbf{t}$ *représentant le vecteur de translation.*

10. Matrices qui vérifient $A^T A = I_n$

11. En tant qu'ensemble, $\mathbf{E}(\mathbf{n}) = \mathbf{O}(\mathbf{n}) \times \mathbb{R}^n$ mais ce n'est pas le cas en tant que groupe : c'est un produit semi-direct $\mathbf{E}(\mathbf{n}) = \mathbf{O}(\mathbf{n}) \rtimes \mathbb{R}^n$

L'algèbre associée $\mathfrak{se}(2)$ *est définie par*

$$\forall \Xi \in \mathfrak{se}(2),\ \Xi = \begin{bmatrix} \Omega_\omega & \mathbf{u} \\ \mathbf{0^T} & 0 \end{bmatrix},\ \Omega_\omega \in \mathfrak{so}(2),\ \mathbf{u} \in \mathbb{R}^2 \tag{4.17}$$

Définition 11. ***(Groupe de similarité SIM)*** *On note* $\mathbf{SIM(n)}$ *le groupe de similarité en dimension* n*. Il représente l'ensemble des isométries composées avec une mise à l'échelle isotropique. Tout comme* $\mathbf{E(n)}$*,* $\mathbf{SIM(n)}$ *est isomorphique à un sous-groupe de* $\mathbf{GL_{n+1}}(\mathbb{R})$ *par l'application*

$$\begin{aligned} \mathbf{SE(n)} \times \mathbb{R} &\to \mathbf{GL_{n+1}}(\mathbb{R}) \\ \{R, \mathbf{t}\},\ s &\mapsto \begin{bmatrix} sR & \mathbf{t} \\ \mathbf{0^T} & 1 \end{bmatrix} \end{aligned} \tag{4.18}$$

C'est ainsi un groupe matriciel de Lie (variété de dimension $n+1$*).*

En dimension 2, l'algèbre associée $\mathfrak{sim}(2)$ *est définie par ([LABG15])*

$$\forall \Pi \in \mathfrak{sim}(2),\ \Pi = \begin{bmatrix} \lambda I_2 + \Omega_\omega & \mathbf{u} \\ \mathbf{0^T} & 0 \end{bmatrix},\ \Omega_\omega \in \mathfrak{so}(2),\ \mathbf{u} \in \mathbb{R}^2,\ \lambda \in \mathbb{R} \tag{4.19}$$

L'application exponentielle pour les groupes matriciels de Lie est simple : c'est exactement l'exponentielle d'une matrice. Par soucis de simplification, on utilisera l'abus de notation suivant

$$\forall \Pi \in \mathfrak{sim}(2),\ \Pi \equiv \begin{bmatrix} \lambda & \omega & \mathbf{u}^T \end{bmatrix}^T = \boldsymbol{\varepsilon}^T \tag{4.20}$$

où ε correspond aux coordonnées dans la base canonique de $\mathfrak{sim}(2)$ dont les vecteurs sont :

$$G_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{4.21}$$

$$G_2 = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{4.22}$$

$$G_3 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{4.23}$$

$$G_4 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \tag{4.24}$$

Leonardos *et al.* [LABG15] donnent une preuve de la surjectivité de exp pour $n \geq 1$, ce qui permet de définir une fonction logarithme $log : \mathbf{SIM(n)} \mapsto \mathfrak{sim}(\mathbf{n})$. De plus, ils donnent une expression générale de l'application exponentielle qui donne pour $n = 2$

$$\forall \Pi \in \mathfrak{sim}(2),\ exp_{\mathbf{SIM(2)}}(\Pi) = e^{\Pi} \equiv e^{\varepsilon} = \begin{bmatrix} e^{\lambda} e^{\Omega_\omega} & V\mathbf{u} \\ \mathbf{0}^T & 1 \end{bmatrix} \tag{4.25}$$

avec

$$V = \int_0^1 e^{\lambda t I_2} e^{t\Omega_\omega} dt \tag{4.26}$$

Il nous faut calculer l'expression (4.26). Elle est donnée dans le cas de $\mathbf{SIM(3)}$ par [LABG15] en utilisant la formule de Rodrigue, non disponible en dimension 2. En s'inspirant de la preuve pour $\mathbf{SIM(3)}$, nous obtenons :

Théorème 1. *L'expression de V (eq 4.26) est déterminée par :*

- *Si $\omega = \lambda = 0$ alors $V = I_2$.*
- *Si $\omega = 0$ et $\lambda \neq 0$ alors $V = \frac{e^{\lambda}-1}{\lambda} I_2$.*
- *Sinon*

$$V = \frac{1}{\lambda^2 + \omega^2} \left[\left(\lambda(e^{\lambda} \cos\omega - 1) + \omega e^{\lambda} \sin\omega \right) I_2 \ + \left(\frac{\omega(1 - e^{\lambda}\cos\omega) + \lambda e^{\lambda} \sin\omega}{\omega} \right) \Omega_\omega \right] \tag{4.27}$$

De plus, V est inversible si $\lambda \neq 0$ ou $\omega \neq k2\pi$, $k \in \mathbb{Z}^*$ (résultat général en dimension n démontré dans [LABG15]).

Démonstration. Si $\omega = \lambda = 0$ alors $V = \int_0^1 e^0 e^0 dt = I_2$. Si $\omega = 0$ et $\lambda \neq 0$ alors $V = \int_0^1 e^{\lambda t} I_2 dt = \frac{1}{\lambda}\left[e^{\lambda t}\right]_0^1 = \frac{e^\lambda - 1}{\lambda} I_2$. Intéressons nous désormais aux autres cas. Nous remarquons que les puissances de Ω_ω vérifient

$$\forall k \in \mathbb{N},\ \begin{cases} \Omega_\omega^{2k} & = (-1)^k \omega^{2k} I_2 \\ \Omega_\omega^{2k+1} & = (-1)^k \omega^{2k} \Omega_\omega \end{cases} \tag{4.28}$$

Nous avons ainsi par l'expression de l'exponentielle de matrice (4.11) et le regroupement des termes pairs/impairs

$$\forall t \in [0,1],\ e^{t\Omega_\omega} = \sum_{i=0}^{+\infty} \frac{\Omega_{t\omega}^i}{i!} \tag{4.29}$$

$$= \sum_{i=0}^{+\infty} \frac{(-1)^i (t\omega)^{2i}}{(2i)!} I_2 + \sum_{i=0}^{+\infty} \frac{(-1)^i (t\omega)^{2i}}{(2i+1)!} \Omega_\omega \tag{4.30}$$

en utilisant le fait que $\forall t \forall \omega,\ t\Omega_\omega = \Omega_{t\omega}$. On reconnaît dans (4.29) le développement en série entière classique des fonctions trigonométriques $\cos$ et $\sin$

$$\forall \theta,\ \cos\theta = \sum_{i=0}^{+\infty} \frac{(-1)^i \theta^{2i}}{(2i)!},\ \sin\theta = \sum_{i=0}^{+\infty} \frac{(-1)^i \theta^{2i+1}}{(2i+1)!} \tag{4.31}$$

qui nous donnent après arrangement

$$e^{t\Omega_w} = \cos(t\omega) I_2 + \frac{\sin(t\omega)}{\omega} \Omega_\omega \tag{4.32}$$

On remplace (4.32) dans (4.26)

$$V = \int_0^1 e^{\lambda t} \cos(t\omega) dt I_2 + \frac{1}{w} \int_0^1 e^{\lambda t} sin(t\omega) dt \Omega_\omega \tag{4.33}$$

Ces deux intégrales sont calculées avec la formule d'Euler en prenant la partie réelle et imaginaire de

$$\int_0^1 e^{\lambda t} e^{i\omega t} dt = \frac{(e^{\lambda + i\omega t})(\lambda - i\omega)}{\lambda^2 + \omega^2} \tag{4.34}$$

ce qui nous donne le résultat énoncé par (4.27). □

Nous avons une expression simple de l'application logarithme (en utilisant l'abus de notation

(4.20))

$$\forall T = \begin{bmatrix} sR_\omega & \mathbf{t} \\ \mathbf{0^T} & 1 \end{bmatrix} \in SIM(2),\ log_{SIM(2)}(T) = \begin{bmatrix} ln(s) & \omega & (V^{-1}\mathbf{t})^T \end{bmatrix}^T \tag{4.35}$$

Nous considérons des angles à valeur dans $[-\pi, \pi[$ donc V est bien inversible.

Revenons maintenant à l'objectif de notre application. Nous souhaitons trouver la pose au temps t d'un objet représenté par $T_t \in \mathbf{SIM(2)}$ de tel sorte que le descripteur CNN correspondant à $\phi_l(T_t)$ se rapproche au maximum d'un descripteur de référence ϕ_l^{ref}. Formellement nous cherchons à résoudre

$$T_t = arg \max_{T \in \mathbf{SIM(2)}} \|\phi_l^{ref} - \phi_l(T)\|_2^2 \tag{4.36}$$

On reformule (4.36) en optimisant directement sur l'algèbre $\mathfrak{sim}(2)$:

$$\boldsymbol{\varepsilon}_t = arg \max_{\boldsymbol{\varepsilon} \in \mathfrak{sim}(2)} \|\phi_l^{ref} - \phi_l(\boldsymbol{\varepsilon})\|_2^2 + \tau(\boldsymbol{\varepsilon})^T C \tau(\boldsymbol{\varepsilon}) \tag{4.37}$$

en ajoutant au passage une matrice diagonale 4×4 de régularisation C sur les paramètres τ de $\mathbf{SIM(2)}$:

$$\begin{aligned} \forall \boldsymbol{\varepsilon} = \begin{bmatrix} \lambda & \omega & \mathbf{u}^T \end{bmatrix}^T \in \mathfrak{sim}(2),\ \tau(\boldsymbol{\varepsilon}) &= \begin{bmatrix} s & \theta & \mathbf{t}^T \end{bmatrix}^T \\ &= \begin{bmatrix} e^\lambda & \theta & (Vu)^T \end{bmatrix} \end{aligned} \tag{4.38}$$

(4.37) se résout classiquement avec l'algorithme de Gauss-Newton tel qu'à la $q^{\text{ème}}$ itération

$$\boldsymbol{\varepsilon}_t^{(q)} = \boldsymbol{\varepsilon}_t^{(q-1)} + \left({J_{\phi_l}^s}^T J_{\phi_l}^s + J_\tau^T C J_\tau \right)^{-1} \left[{J_{\phi_l}^s}^T \left(\phi_l^{ref} - \phi_l(\boldsymbol{\varepsilon}_t^{(q-1)}) \right) - J_\tau^T C \boldsymbol{\varepsilon}_t^{(q-1)} \right] \tag{4.39}$$

$$\boldsymbol{\varepsilon}_t^{(0)} = \boldsymbol{\varepsilon}_{t-1} \tag{4.40}$$

J_τ est la matrice jacobienne de τ évaluée en $\boldsymbol{\varepsilon}_t^{(q-1)}$. Elle s'obtient à partir de (4.38)

$$J_\tau = \begin{bmatrix} e^\lambda & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ \frac{\partial V}{\partial \lambda}\mathbf{u} & \frac{\partial V}{\partial \omega}\mathbf{u} & V & \end{bmatrix} \tag{4.41}$$

$J_{\phi_l}^s$ est la matrice jacobienne de ϕ_l relative à $\mathfrak{sim}(2)$ évaluée en $\boldsymbol{\varepsilon}_t^{(q-1)}$. Pour la calculer, nous

passons par la règle de dérivation en chaîne :

$$J^{s}_{\phi_l} = J^{\tau}_{\phi_l} J_{\tau} \tag{4.42}$$

où $J^{\tau}_{\phi_l}$ est la matrice jacobienne de ϕ_l relative à τ. La façon la plus simple d'obtenir une approximation de $J^{\tau}_{\phi_l}$ est de la calculer numériquement par différence centrale. En effet, nous pouvons simuler une transformation sur l'objet en fixant sa position et en appliquant la transformation inverse sur l'image. Cela signifie en pratique que nous appliquons indépendamment huit transformations (rotation, mise à l'échelle, translation x/y) à l'image et utilisons le descripteur CNN correspondant à la bounding box fixée de l'objet. Nous aborderons dans la section 4.2.8 une autre façon d'approximer cette matrice jacobienne.

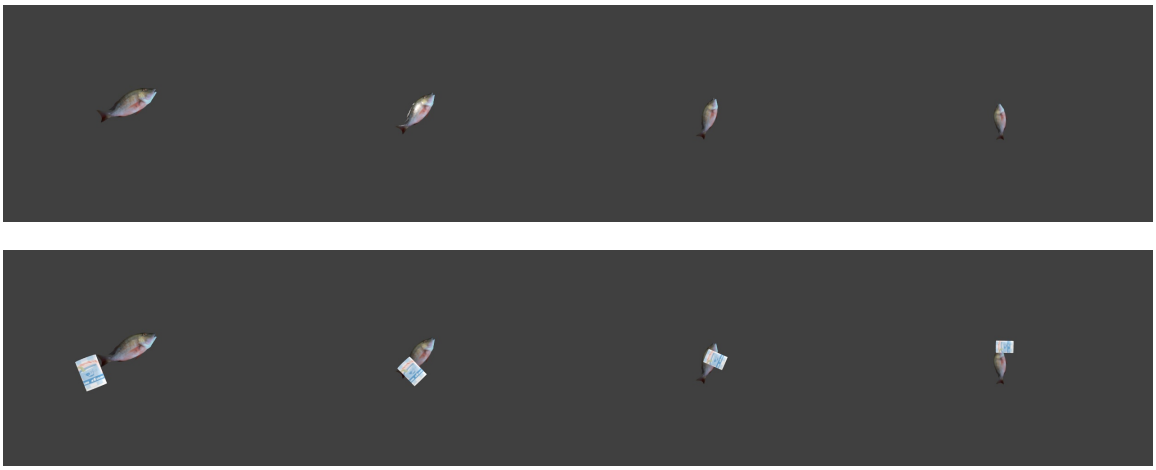
4.2.7 Expérimentation

Inférence des paramètres par simulation

L'algorithme présenté précédemment peut s'appliquer avec un descripteur image quelconque. Nous avons donc voulu comparer les résultats obtenus selon les couches et réseaux utilisés. En effet, on peut supposer que dans le cas idéal sans perturbation les couches inférieures seront plus performantes car avec des champs réceptifs plus petits. Au contraire, les couches FC devraient être peu performantes. Cependant, en pratique, un objet est soumis à de nombreuses perturbations : changement d'illumination, occlusion, changement de point de vue. Dans ces cas-là, les couches inférieures devraient être plus sensibles aux variations locales. Le problème est donc de savoir quelles couches apportent un bon compromis entre robustesse face aux perturbations et sensibilité aux transformations de similarité.

Afin d'avoir une vérité terrain sur les paramètres à estimer, notre expérience se base sur des simulations 3D déjà présenté à la figure 4.13. La caméra se déplace linéairement tout en ayant une rotation selon ce même axe. Le centre de chaque objet est placé de manière identique. A noter que celui-ci ne se trouve pas sur l'axe optique de la caméra, ce qui va entraîner une translation de l'objet dans l'image. Nous avons ensuite trois scénarii : un idéal avec seulement l'objet sans aucune perturbation, un avec une illumination variante et le dernier avec de l'occlusion (cf figure 4.16). Les résultats sont visibles pour chaque objet aux figures 4.17, 4.18, 4.19 et 4.20. On a noté par des segments verticaux bleu et rouge respectivement l'entrée de l'objet occlusif dans la bounding box de l'objet suivi et le début de l'occlusion.

Avant de commencer notre analyse, il faut noter que le faible nombre d'échantillons nous permet



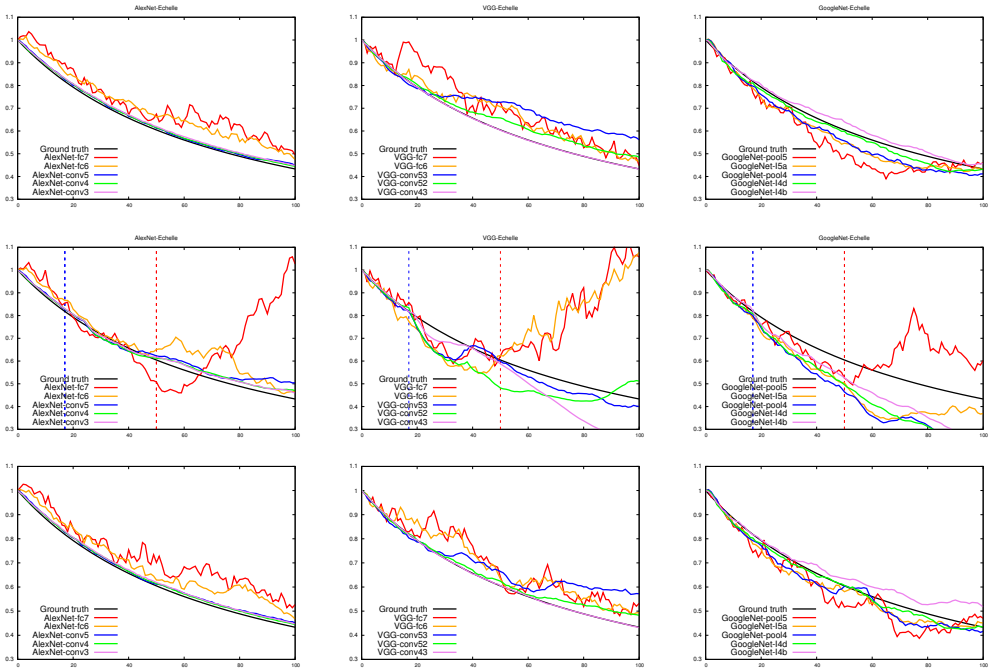
(e) Img 25 (f) Img 50 (g) Img 75 (h) Img 100

FIGURE 4.16 – Images de l'expérience avec occlusion

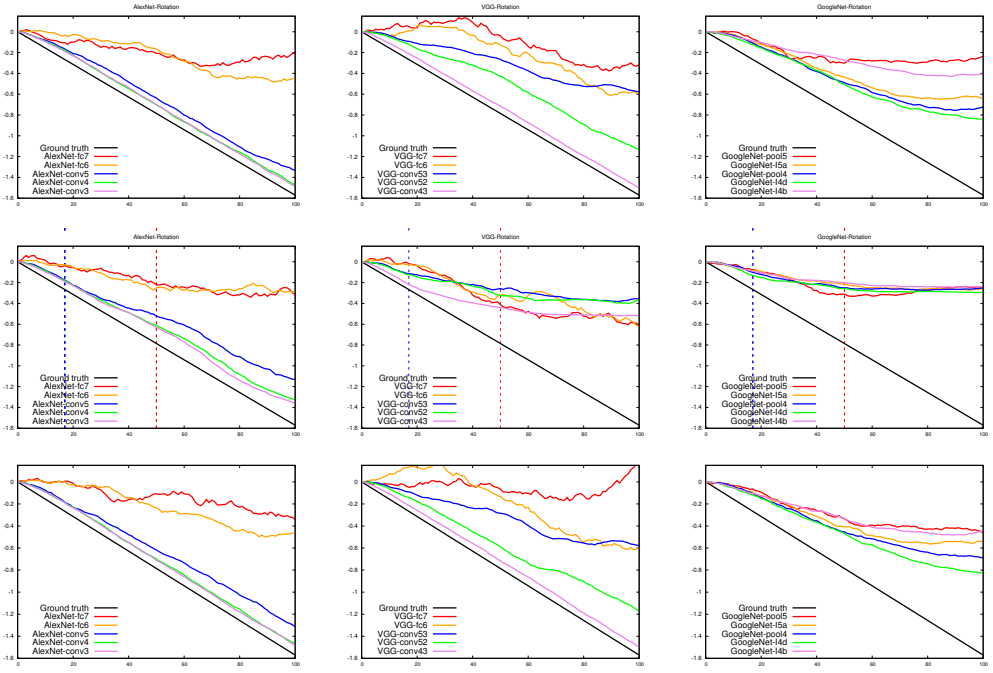
au mieux de proposer certaines hypothèses ainsi que de valider empiriquement l'approche décrite dans cette section. Tout d'abord, considérons le cas idéal. Il est clair que toutes les couches de GoogLeNet donnent des résultats très médiocres, notamment pour l'estimation de la rotation. Cela pourrait être du aux fonctions de coûts intermédiaires utilisées lors de l'entraînement, qui partent des couches *Inception_4a / Inception_4d*. En effet, ces couches sont reliées à des couches FC qui n'ont plus de notion de spatialité. La profondeur des réseaux a également un rôle dans la déformation de la variété générée par les transformations de similarité sur l'image. Poole [PLR$^+$16] montre qu'au fur et à mesure de la propagation dans le réseau, les variétés ont tendance à se complexifier. Ce résultat s'observe encore ici avec VGG, qui a une structure similaire à AlexNet mais avec une plus grande profondeur : cela se traduit par une plus grande instabilité des estimations.

Les changements d'illuminations ne changent que peu les résultats. Le plus notable s'observe pour l'estimation de la rotation dans la séquence *poisson* pour AlexNet. On remarque également concernant l'occlusion partielle des objets une plus grande robustesse du réseau AlexNet, là où VGG décroche dès l'apparition de l'objet responsable de l'occlusion dans la région de détection de l'objet. Ceci est attendu du fait de la plus grande capacité de réseau, de façon équivalente aux résultats de la section 4.2.4 sur la représentation d'instance.

Nous avons regroupé les erreurs moyennes dans les tables 4.3 et 4.4.

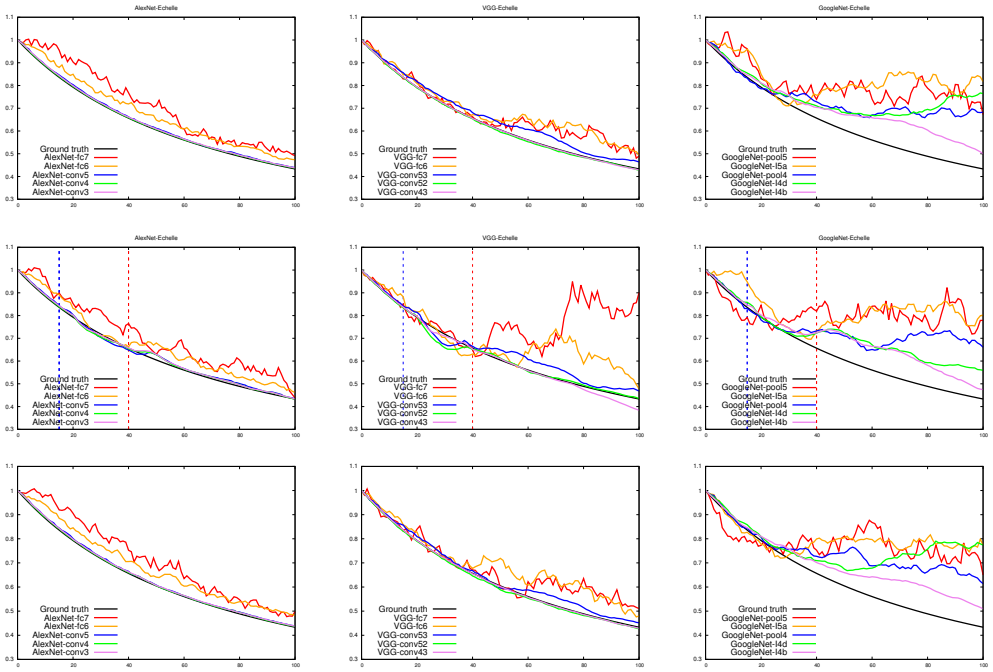


(a) Erreur d'échelle

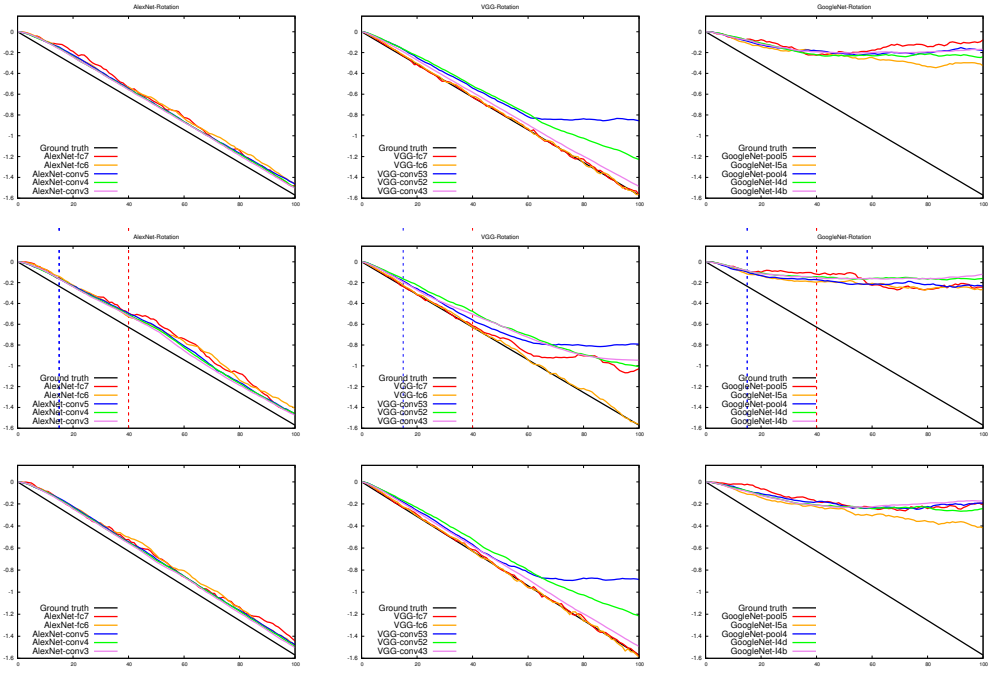


(b) Erreur de rotation

FIGURE 4.17 – Erreur d'échelle et de rotation pour l'objet tête. La première ligne correspond au cas idéal, la seconde au changement d'illumination et la dernière avec occlusion. Les colonnes correspondent aux différents CNNs avec de gauche à droite : AlexNet, VGG et GoogLeNet

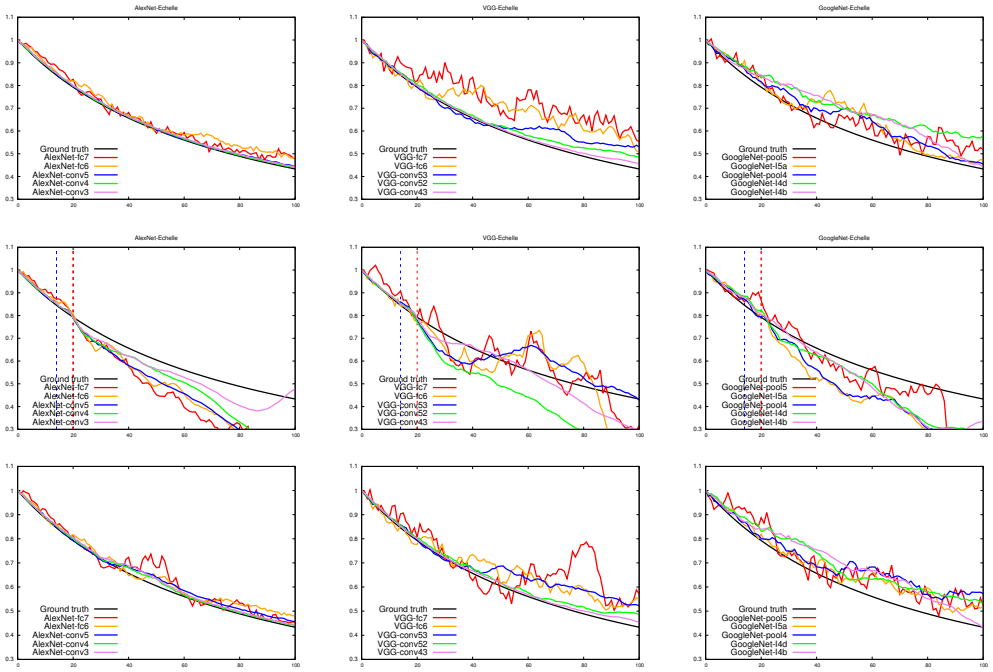


(a) Erreur d'échelle

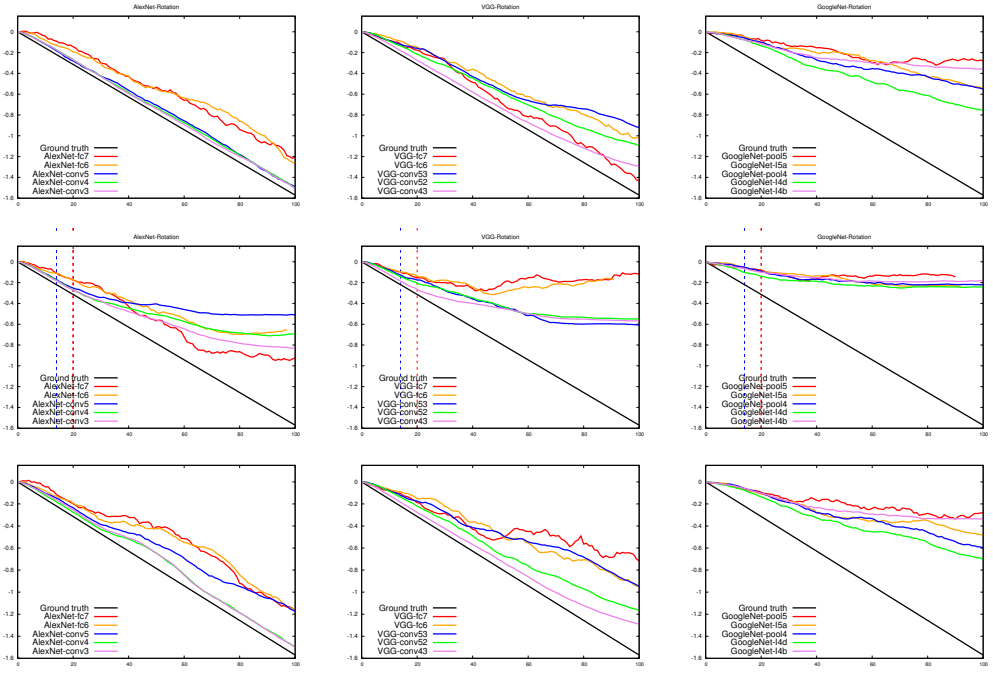


(b) Erreur de rotation

FIGURE 4.18 – Erreur d'échelle et de rotation pour l'objet livre. La première ligne correspond au cas idéal, la seconde au changement d'illumination et la dernière avec occlusion. Les colonnes correspondent aux différents CNNs avec de gauche à droite : AlexNet, VGG et GoogLeNet

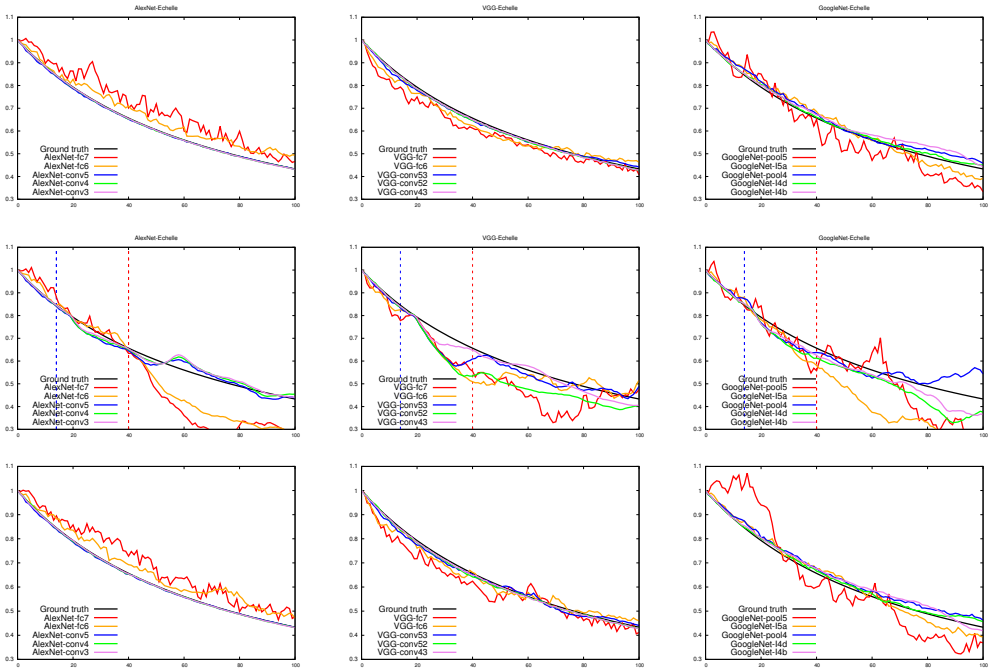


(a) Erreur d'échelle

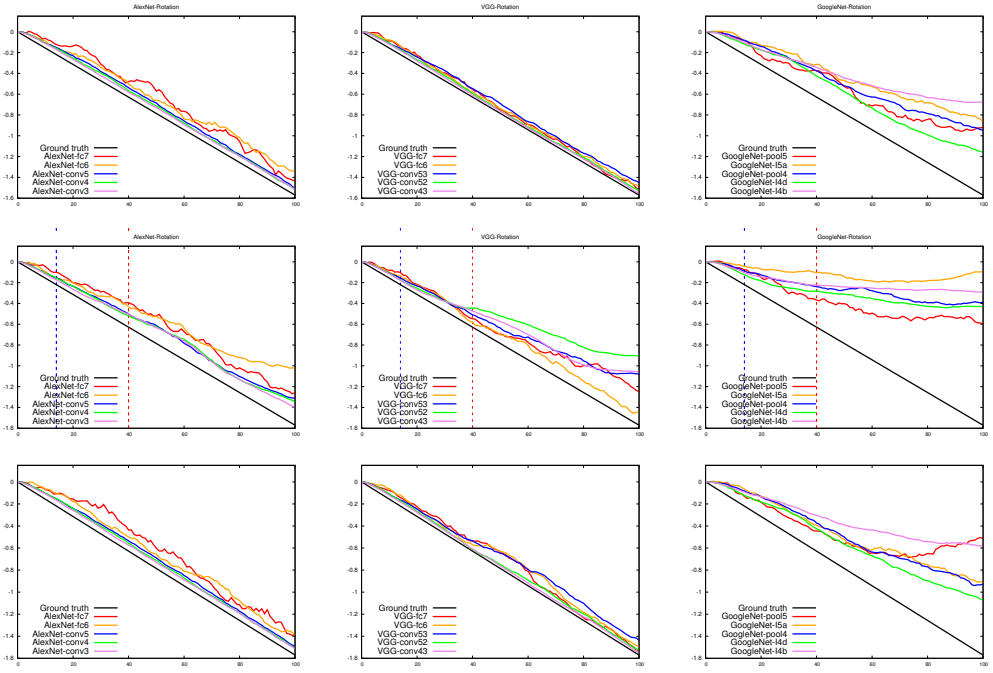


(b) Erreur de rotation

FIGURE 4.19 – Erreur d'échelle et de rotation pour l'objet poisson. La première ligne correspond au cas idéal, la seconde au changement d'illumination et la dernière avec occlusion. Les colonnes correspondent aux différents CNNs avec de gauche à droite : AlexNet, VGG et GoogLeNet



(a) Erreur d'échelle



(b) Erreur de rotation

FIGURE 4.20 – Erreur d'échelle et de rotation pour l'objet tasse. La première ligne correspond au cas idéal, la seconde au changement d'illumination et la dernière avec occlusion. Les colonnes correspondent aux différents CNNs avec de gauche à droite : AlexNet, VGG et GoogLeNet

		Tête		Livre		Poisson		Tasse	
		S.O.	A.O.	S.O.	A.O.	S.O.	A.O.	S.O.	A.O.
AlexNet	fc7	87.34(35.82)	133.17(89.97)	86.30(32.40)	75.58(26.75)	26.42(16.9)	140.52(87.60)	72.68(32.98)	131.83(99.40)
	fc6	66.3(17.65)	65.32(40.65)	47.40(11.14)	42.72(20.46)	33.82(17.58)	132.91(88.84)	40.8(17.49)	105.78(77.35)
	conv5	10.65(4.12)	31.61(23.73)	6.63(2.22)	10.49(9.35)	6.62(3.46)	130.95(82.85)	4.33(2.61)	19.39(18.55)
	conv4	10.49(2.36)	18.25(14.27)	3.02(0.94)	8.69(8.00)	5.20(1.63)	104.95(72.68)	2.42(1.25)	19.22(19.11)
	conv3	14.07(2.80)	16.15(10.56)	5.06(1.05)	9.02(7.18)	7.92(1.83)	44.16(30.45)	2.08(0.94)	19.68(19.08)
VGG	fc7	87.04(44.63)	68.91(65.08)	59.95(41.84)	215.44(159.52)	132.45(64.93)	69.81(69.80)	43.56(23.66)	90.15(55.74)
	fc6	62.70(37.40)	60.16(51.43)	69.03(41.22)	93.06(70.69)	103.52(60.01)	82.76(82.71)	27.81(23.39)	66.50(53.24)
	conv5_3	98.66(57.94)	110.02(81.24)	28.90(12.22)	31.76(25.44)	50.12(38.61)	54.43(53.59)	9.90(8.25)	41.08(33.47)
	conv5_2	37.49(20.13)	70.48(60.28)	4.84(2.52)	18.42(17.72)	27.20(16.34)	133.88(72.03)	6.96(3.03)	73.91(38.69)
	conv4_3	12.19(11.22)	11.31(10.87)	2.18(1.74)	15.42(13.29)	10.55(6.14)	55.82(46.52)	5.47(2.05)	23.55(17.96)
GoogleNet	pool5	75.56(47.05)	85.26(71.62)	195.42(99.23)	223.31(14.38)	53.59(37.66)	66.87(62.94)	57.12(48.51)	88.93(77.38)
	Inception_5a	47.35(24.04)	34.12(26.13)	228.58(131.29)	229.40(129.06)	58.68(40.83)	129.73(80.96)	26.01(25.98)	138.22(85.84)
	pool4	45.05(19.92)	58.30(31.32)	131.80(83.57)	142.47(94.44)	48.93(25.49)	115.31(68.95)	20.79(11.98)	43.85(43.31)
	Inception_4d	21.00(11.72)	31.98(26.40)	150.01(100.15)	90.06(51.62)	93.13(36.99)	92.88(67.36)	8.83(4.91)	55.97(33.68)
	Inception_4b	19.07(10.95)	63.06(54.36)	75.62(40.02)	75.03(41.43)	77.04(36.34)	93.00(70.60)	26.00(15.7)	37.85(23.84)

TABLE 4.3 – Erreurs d'estimations du paramètre d'échelle. Echelle : 10^{-3}

		Tête		Livre		Poisson		Tasse	
		S.O.	A.O.	S.O.	A.O.	S.O.	A.O.	S.O.	A.O.
AlexNet	fc7	70.63(37.71)	78.35(42.48)	10.13(2.82)	142.61(5.07)	25.12(9.25)	29.25(14.84)	16.60(5.36)	21.87(7.42)
	fc6	63.29(28.81)	70.78(36.61)	10.33(2.61)	13.65(4.32)	26.97(12.15)	40.78(22.31)	15.88(6.26)	27.42(13.37)
	conv5	14.47(4.49)	26.63(12.39)	8.33(1.74)	10.87(3.38)	6.60(2.16)	52.45(32.49)	8.05(1.82)	14.20(5.47)
	conv4	8.17(1.73)	15.55(5.69)	7.43(1.39)	9.92(2.86)	5.74(1.63)	40.85(25.68)	6.24(1.21)	13.52(5.36)
	conv3	7.98(1.59)	12.96(4.52)	6.83(1.17)	8.59(2.31)	4.83(1.30)	34.14(21.39)	5.19(0.92)	12.33(5.00)
VGG	fc7	76.25(31.65)	56.63(27.56)	1.17(1.08)	21.71(16.85)	14.35(4.53)	76.53(44.33)	6.38(2.1)	18.96(9.87)
	fc6	63.57(25.83)	61.70(25.60)	1.33(1.07)	1.96(1.27)	32.61(14.53)	64.67(37.33)	6.62(1.61)	9.80(3.01)
	conv5_3	56.17(27.25)	62.72(30.45)	29.56(20.34)	32.04(22.50)	34.94(18.58)	47.83(26.83)	8.59(2.31)	22.08(12.03)
	conv5_2	30.89(11.74)	43.50(22.56)	17.36(8.58)	26.69(14.97)	24.91(12.00)	50.97(29.52)	3.99(0.93)	32.68(18.87)
	conv4_3	6.15(1.25)	14.09(5.92)	5.12(1.42)	26.96(16.92)	10.87(6.52)	49.21(30.80)	2.86(0.75)	23.12(12.79)
GoogleNet	pool5	68.80(37.96)	79.97(41.56)	79.14(44.40)	73.91(38.35)	70.13(36.62)	70.74(38.7)	30.00(15.12)	50.83(28.20)
	Inception_5a	46.35(23.96)	69.02(35.80)	68.06(36.97)	71.87(39.42)	61.57(30.08)	67.25(34.94)	41.36(18.91)	78.54(41.27)
	pool4	40.44(20.71)	54.52(27.61)	75.10(41.77)	73.41(40.27)	57.91(29.72)	74.12(39.62)	34.43(15.91)	63.42(33.70)
	Inception_4d	36.78(18.34)	57.02(30.09)	72.50(39.87)	77.74(42.15)	45.70(22.84)	71.83(39.76)	22.67(8.84)	59.61(33.58)
	Inception_4b	61.94(32.09)	70.03(38.05)	75.28(41.65)	78.23(42.79)	65.67(35.57)	75.97(40.93)	45.55(24.24)	69.58(38.58)

TABLE 4.4 – Erreurs d'estimations de l'angle de rotation. Unité : 10^{-2} radians

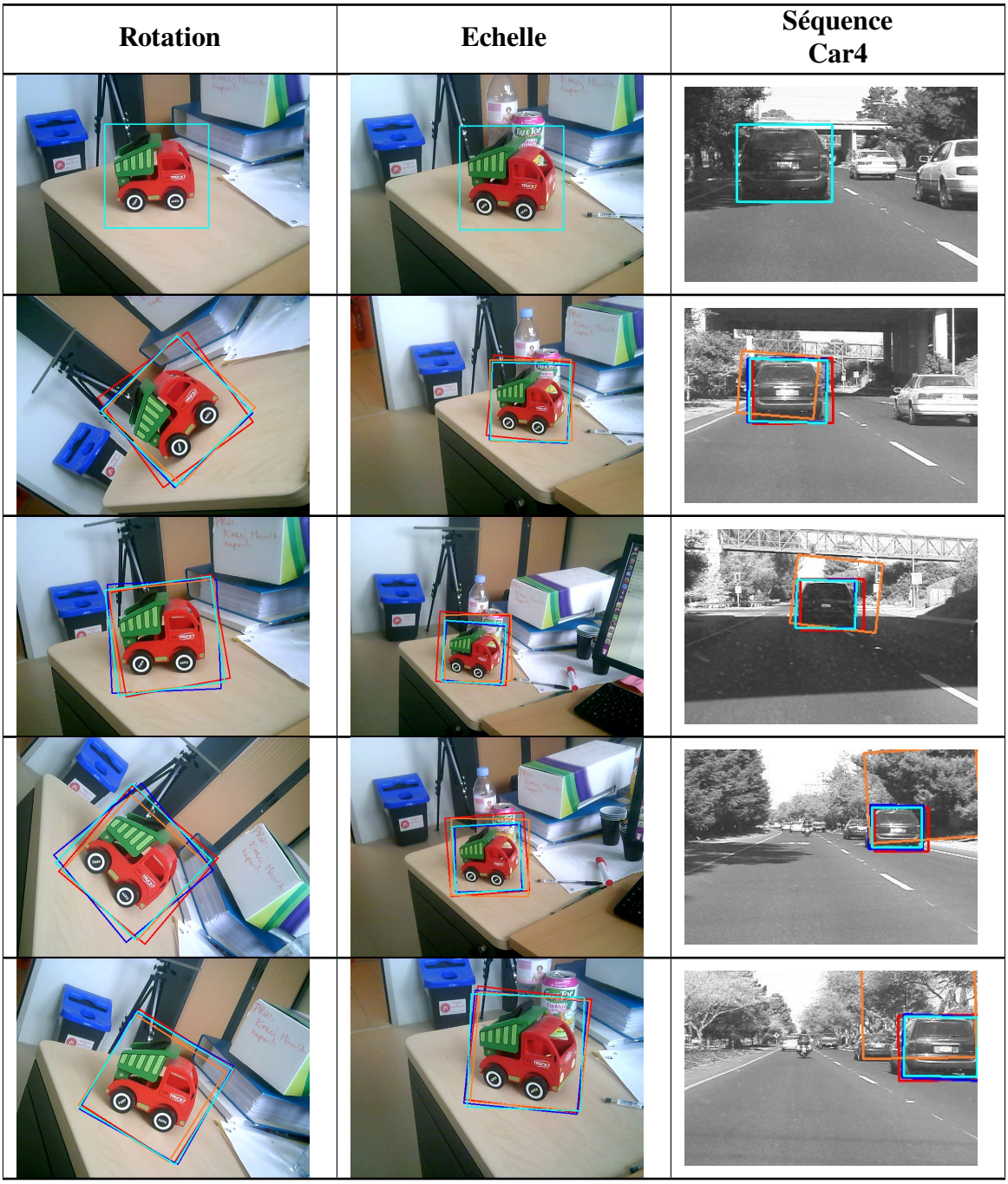


FIGURE 4.21 – Exemple de tracking selon la méthode présentée à la section 4.2.6. Légende : *fc7-AlexNet*(—), *fc7-VGG*(—), *conv5-AlexNet*(—) et *conv5_3-VGG*(—). La première ligne correspond à la position initiale

Application sur des séquences vidéo

Afin de confirmer l'exploitation de notre méthode en pratique, notre algorithme a été appliqué sur des vidéos prises manuellement par une Webcam. Les résultats qualitatifs sont visibles à

la figure 4.21 en utilisant différents descripteurs images. A chaque image, une seule itération de Gauss-Newton est effectuée. Conformément à l'expérience précédente, les descripteurs issus de couches entièrement connectées sont moins performants que les couches de convolutions les précédant. Nous l'avons également testé sur des séquences du dataset [WLY13], avec les résultats sur la séquence *Car4* à la dernière ligne de la figure 4.21. Bien que l'on obtienne des résultats corrects sur certaines séquences, notre algorithme reste limité au petit mouvement et aux objets rigides. Il est toutefois possible, en effectuant plusieurs itérations de la mise à jour de Gauss-Newton, de suivre de plus grands mouvements mais au prix d'un temps de calcul élevé (calcul numérique des jacobiennes). De plus, pour avoir un véritable algorithme de suivi, il faudrait intégrer une mise à jour du descripteur de référence. L'idée est de rester *data driven* c'est à dire de représenter l'instance suivie par un ensemble de descripteurs et non par un modèle unique que l'on met à jour au fur à mesure. La poursuite de ce travail sera donc de trouver un critère permettant de savoir quand ajouter ou retirer un descripteur représentant un "mode" de l'instance. La formalisation qui va suivre a pour but d'obtenir des expressions permettant d'éviter le calcul numérique des jacobiennes à chaque itération. De plus, elle pourrait également permettre, en utilisant les outils d'analyse matricielles, de mieux comprendre théoriquement les propriétés des descripteurs évoqués dans cette section.

4.2.8 Formalisation

L'objectif de cette section est d'obtenir une expression matricielle de la variation des descripteurs CNN en fonction de transformations de similarité. L'intérêt est double : disposer des outils d'analyses matricielles afin de pouvoir mieux appréhender les comportements des descripteurs CNN mais également, dans le cadre de la section précédente, de trouver une approximation de la jacobienne des descripteurs relativement aux transformations de similarité.

Afin de ne pas surcharger les notations et de simplifier les expressions, nous considérons tout d'abord que toutes les couches ont une profondeur de 1. Comme précédemment, on notera $\phi_l \in \mathrm{M}_{H_l,W_l}$ la $l^{\text{ème}}$ couche de hauteur H_l et largeur W_l du CNN. L'image d'entrée est notée $I \in \mathrm{M}_{H,W}$ et la région représentant l'objet observé $\phi_0 \subset I$. La position de l'objet dans l'image est représentée par $\varepsilon \in \mathfrak{sim}(2)$. Nous faisons l'hypothèse que le nombre de pixels dans la région d'entrée est constant, même si la taille de la bounding box peut varier lors de changements d'échelle [12].

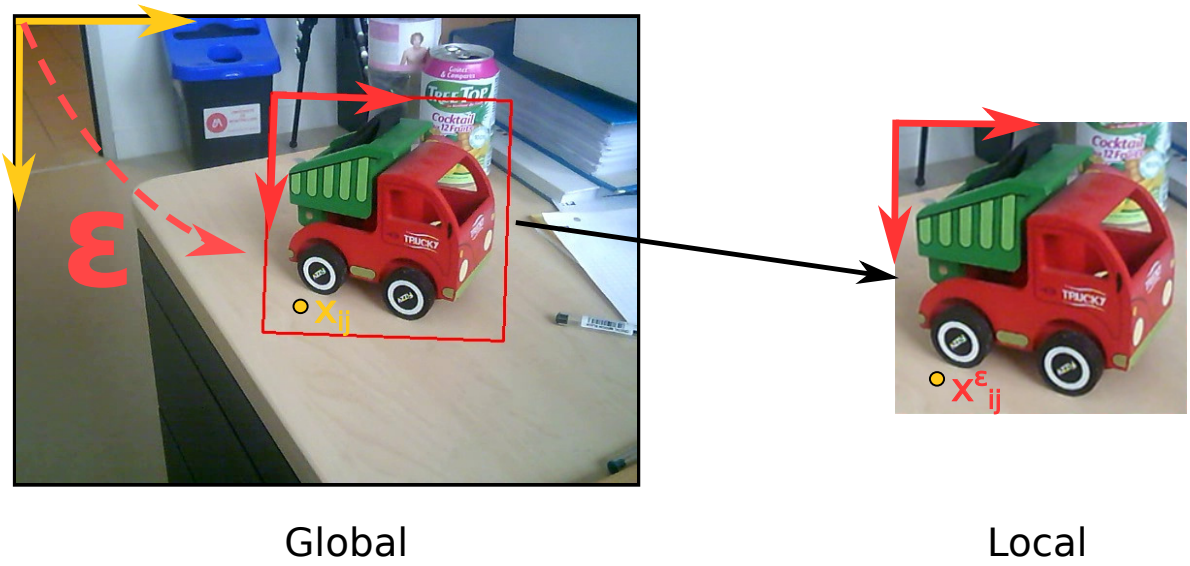


FIGURE 4.22 – Définition des coordonnées globales et locales pour la formalisation

Expression du gradient d'un élément ϕ_0^{ij}

Nous notons par $\mathbf{x}_{ij}^I = \begin{bmatrix}(\tilde{\mathbf{x}}_{ij}^I)^T & 1\end{bmatrix}^T$ les coordonnées homogènes du pixel en position (i,j) dans l'image I avec $\tilde{\mathbf{x}}_{ij}^I = \begin{bmatrix}i & j\end{bmatrix}^T$. De même, $\mathbf{x}_{ij}^\varepsilon$ ses coordonnées dans le repère local courant (figure 4.22).

Définissons la fonction ψ_ε de changement de repère de la région objet (local) vers l'image (global)

$$\psi_\varepsilon : \begin{cases} [1,H_0]\times[1,W_0] \to [1,H]\times[1,W] \\ \mathbf{x}^\varepsilon \mapsto \mathbf{x}^I = e^\varepsilon \mathbf{x}^\varepsilon \end{cases} \tag{4.43}$$

avec $e = exp_{\mathbf{SIM(2)}}$ l'application exponentielle de $\mathbf{SIM(2)}$.

Nous définissons la fonction c qui retourne la valeur d'un pixel à partir de ses coordonnées globales

$$c : \begin{cases} [1,H]\times[1,W] \to \mathbb{R} \\ \mathbf{x}^I \mapsto c(\mathbf{x}^I) \end{cases} \tag{4.44}$$

A partir de (4.43) et (4.44), nous pouvons écrire la valeur d'un pixel dans son repère local

$$\forall (i,j) \in [1,H_0]\times[1,W_0]\,,\ \phi_0^{ij}(\boldsymbol{\varepsilon}) = (\phi_0(\boldsymbol{\varepsilon}))_{ij} = (c\circ\psi_\varepsilon)(\mathbf{x}_{ij}^\varepsilon) \tag{4.45}$$

Nous en déduisons, en notant ε_k la k-ème coordonnée de $\boldsymbol{\varepsilon}$ et en utilisant la règle de dérivation

12. En pratique, l'image est redimensionnée par interpolation bilinéaire.

en chaîne

$$\frac{\partial \phi_0^{ij}}{\partial \varepsilon_k}(\boldsymbol{\varepsilon}) = \left(\frac{\partial c}{\partial \mathbf{x}}\left(\psi_{\boldsymbol{\varepsilon}}\left(\mathbf{x}_{ij}^{\boldsymbol{\varepsilon}}\right)\right)\right)^T \left(\frac{\partial exp_{\mathbf{SIM(2)}}}{\partial \varepsilon_k}(\boldsymbol{\varepsilon})\mathbf{x}_{ij}^{\boldsymbol{\varepsilon}}\right) \tag{4.46}$$

Remarquons que le premier terme à droite est relié au gradient de l'image ∇I au pixel $\psi_{\boldsymbol{\varepsilon}}\left(\mathbf{x}_{ij}^{\boldsymbol{\varepsilon}}\right)$.

Nous pouvons alors réécrire (4.46) de la façon suivante

$$\frac{\partial \phi_0^{ij}}{\partial \varepsilon_k}(\boldsymbol{\varepsilon}) = \left(\Gamma_{k,\boldsymbol{\varepsilon}}^T \widetilde{\nabla I}\right)\left(\mathbf{x}_{ij}^{\boldsymbol{\varepsilon}}\right) \tag{4.47}$$

avec

$$\forall (i,j) \in [1,H_0]\times[1,W_0]\,,\ \Gamma_{k,\boldsymbol{\varepsilon}}\left(\mathbf{x}_{ij}^{\boldsymbol{\varepsilon}}\right) = \frac{\partial exp_{\mathbf{SIM(2)}}}{\partial \varepsilon_k}(\boldsymbol{\varepsilon})\mathbf{x}_{ij}^{\boldsymbol{\varepsilon}} \in \mathbb{R}^3 \tag{4.48}$$

$$\forall (i,j) \in [1,H]\times[1,W]\,,\ \nabla I(\mathbf{x}_{ij}^I) = \begin{bmatrix}\frac{\partial c}{\partial x}(\mathbf{x}_{ij}^I) & \frac{\partial c}{\partial y}(\mathbf{x}_{ij}^I) & 0\end{bmatrix}^T \in \mathbb{R}^3 \tag{4.49}$$

$$\widetilde{\nabla I} = \nabla I \circ \psi_{\boldsymbol{\varepsilon}} \tag{4.50}$$

Expression du gradient de ϕ_0 sous forme vectorielle

Nous cherchons maintenant à obtenir l'expression vectorielle de $\frac{\partial \phi_0}{\partial \varepsilon_k}$, à savoir $Vec\left(\frac{\partial \phi_0}{\partial \varepsilon_k}\right)$ (cf Définition 19, annexe B). Après arrangement, nous arrivons au résultat suivant

Proposition 1. *Le gradient vectorisé d'une région $\phi_0(\boldsymbol{\varepsilon}) \subset I$, de pose $\boldsymbol{\varepsilon} \in \mathfrak{sim}(2)$, peut se factoriser sous la forme*

$$Vec\left(\frac{\partial \phi_0}{\partial \epsilon_k}\right) = M_k(\boldsymbol{\varepsilon})\hat{\nabla}\mathbf{I}\left(\phi_0(\boldsymbol{\varepsilon})\right) \tag{4.51}$$

$$M_k(\boldsymbol{\varepsilon}) = R_{H_0W_0}^T\left(I_{H_0W_0}\otimes D_{\Gamma_{k,\boldsymbol{\varepsilon}}}\right)\left(R_{H_0W_0}\otimes I_3\right) \tag{4.52}$$

avec

$$D_{\Gamma_{k,\boldsymbol{\varepsilon}}} = \left[D_x\left(I_{H_0W_0}\otimes \frac{\partial exp_{\mathbf{SIM(2)}}}{\partial \varepsilon_k}(\boldsymbol{\varepsilon})\right)\right],\ D_x = \begin{bmatrix}\mathbf{x}_{11}^T & & \\ & \ddots & \\ & & \mathbf{x}_{H_0W_0}^T\end{bmatrix} \tag{4.53}$$

$$\hat{\nabla}\mathbf{I}\left(\phi_0(\boldsymbol{\varepsilon})\right) = \begin{bmatrix}\widetilde{\nabla I}^T(\mathbf{x}_{11}^{\boldsymbol{\varepsilon}}) & \dots & \widetilde{\nabla I}^T(\mathbf{x}_{H_01}^{\boldsymbol{\varepsilon}}) & \widetilde{\nabla I}^T(\mathbf{x}_{12}^{\boldsymbol{\varepsilon}}) & \dots & \widetilde{\nabla I}^T(\mathbf{x}_{H_0W_0}^{\boldsymbol{\varepsilon}})\end{bmatrix}^T \tag{4.54}$$

avec $R_{H_0W_0}$ défini au théorème 11, annexe B.1.

Démonstration. Voir annexe B.3.1 □

Avec l'utilisation du produit de Kronecker, on aboutit à une expression relativement concise. Un point intéressant de cette factorisation est que seul le second terme est dépendant de l'image, le second n'étant que fonction de la pose $\boldsymbol{\varepsilon}$. Nous allons maintenant voir comment propager ce gradient à travers les différentes couches du réseau.

Expression du gradient d'un élément d'une couche supérieur ϕ_l^{ij}

En général, chaque couche de convolution d'un CNN est en fait une composition de "sous-couches" : convolution ϕ_l^c, unité de rectification linéaire ReLU ϕ_l^{ReLU} et pooling ϕ_l^p. Pour les premières couches, certains réseaux utilisent également une couche de normalisation locale LRN ϕ_l^{LRN}. Nous ignorons ici l'effet du pooling, ce qui nous donne une expression générale :

$$\phi_l = \phi_l^{lnr} \circ \phi_l^{relu} \circ \phi_l^c \tag{4.55}$$

Dans la suite, nous allons développer les expressions du gradient à travers chaque type de sous-couche.

Couche de convolution : Soient $k_l,\ s_l$ respectivement la taille et le décalage du filtre de convolution de la couche ϕ_l^c. Afin de ne pas trop complexifier nos expressions , nous supposons ici $s_l = 1$. Nous supposons également des noyaux carrés. Soit $W_l \in M_{k_l,k_l}$ la matrice des poids du filtre de convolution [13]. On a alors :

Proposition 2. *L'expression vectorielle d'une couche de convolution fonction de la couche précédente est donnée par*

$$\begin{aligned} Vec\left(\phi_l^c\right) &= \Omega_l Vec\left(\phi_{l-1}\right) && (4.56)\\ \Omega_l &= \left(I_{H_l W_l} \otimes Vec\left(W_l\right)^T\right)\left(\hat{S}_\beta * \hat{S}_\alpha\right) && (4.57) \end{aligned}$$

avec

$$\hat{S}_\beta = \begin{bmatrix} S_{\beta_1} \\ \vdots \\ S_{\beta_{W_l}}, \end{bmatrix},\ \beta_i = \{i - \frac{k_l - 1}{2},\ \ldots,\ i + \frac{k_l - 1}{2}\} \tag{4.58}$$

$$\hat{S}_\alpha = \begin{bmatrix} S_{\alpha_1} \\ \vdots \\ S_{\alpha_{H_l}} \end{bmatrix},\ \alpha_j = \{j - \frac{k_l - 1}{2},\ \ldots,\ j + \frac{k_l - 1}{2}\} \tag{4.59}$$

13. Nous considérons un biais nul

$S_{\alpha_j}, S_{\beta_i}$ sont des matrices de permutation partielle (Définition 18, annexe B.1) et $*$ le produit de Khatri-Rao (cf Définition 16, annexe B.1).

Démonstration. Voir annexe B.3.2. □

On en déduit :

Proposition 3. *Le gradient vectorisé d'une couche de convolution en fonction du gradient de la couche précédente est donné par*

$$Vec\left(\frac{\partial \phi_l^c}{\partial \varepsilon_k}\right) = \Omega_l Vec\left(\frac{\partial \phi_{l-1}}{\partial \varepsilon_k}\right) \tag{4.60}$$

Démonstration. On utilise le fait que pour toute matrice A fonction d'un réel x :

$$Vec\left(\frac{dA}{dx}\right) = \frac{dVec\,(A)}{dx}$$

□

Couche ReLU

On note par $\mathcal{H}$ la fonction de Heaviside [14]. Par définition, on a

$$\phi_l^{relu,ij} = \max(0, \phi_l^{c,ij}) \tag{4.61}$$

Proposition 4. *Le gradient vectorisé d'une couche ReLU en fonction du gradient de la couche précédente est donné par*

$$Vec\left(\frac{\partial \phi_l^{relu}}{\partial \varepsilon_k}\right) = \mathbf{H}_l \odot Vec\left(\frac{\partial \phi_l^c}{\partial \varepsilon_k}\right) \tag{4.62}$$

où $\odot$ est le produit de Hadamard et avec

$$\mathbf{H}_l = \left[\mathcal{H}\left(\phi_l^{c,11}\right) \dots \mathcal{H}\left(\phi_l^{c,H_l W_l}\right)\right]^T \tag{4.63}$$

Démonstration. Voir annexe B.3.3. □

14. $\mathcal{H} : \mathbb{R} \mapsto \{0, 1\}$ qui vaut 1 pour les valeurs positives et 0 ailleurs.

Couche LNR

Soient $n_l > 0$ la taille des régions pour la normalisation et $\alpha_l, \beta_l > 0$ deux paramètres. La couche LNR est définie par

$$\phi_l^{lnr,ij} = \frac{\phi_l^{relu,ij}}{d_{ij}^{\beta_l}} \tag{4.64}$$

avec le dénominateur :

$$d_{ij} = 1 + \frac{\alpha_l}{n_l} \sum_{p=i-\frac{n_l-1}{2}}^{i+\frac{n_l-1}{2}} \sum_{r=j-\frac{n_l-1}{2}}^{j+\frac{n_l-1}{2}} \phi_l^{relu,pr\,2} \tag{4.65}$$

Proposition 5. *De manière similaire à la couche précédente, le gradient de la couche LNR a pour expression*

$$Vec\left(\frac{\partial \phi_l^{lnr}}{\partial \varepsilon_k}\right) = \frac{\partial Vec\left(\phi_l^{lnr}\right)}{\partial Vec\left(\phi_l^{relu}\right)} \frac{\partial Vec\left(\phi_l^{relu}\right)}{\partial \varepsilon_k} \tag{4.66}$$

avec

$$\begin{aligned} \frac{\partial Vec\left(\phi_l^{lnr}\right)}{\partial Vec\left(\phi_l^{relu}\right)} &= \mathbf{diag}\left(K_{lnr}^{\odot \beta_l}\right) - 2\frac{\alpha_l}{n_l}\beta_l \,\mathbf{diag}\left(Vec\left(\phi_l^{relu}\right) \odot K_{lnr}^{\odot \beta_l + 1}\right) \\ &\quad \left(\left[I_{W_l H_l} \otimes Vec\left(\phi_l^{relu}\right)^T\right] (\Lambda_\beta * \Lambda_\alpha)\right) \\ &= L_l \end{aligned} \tag{4.67}$$

et

$$K_{lnr}(\phi_l^{relu}) = \begin{bmatrix} \frac{1}{d_{11}} \\ \vdots \\ \frac{1}{d_{W_l H_l}} \end{bmatrix} \in \mathbb{R}^{W_l H_l} \tag{4.68}$$

$$\Lambda_\beta = \begin{bmatrix} S_{\beta_1}^{lnr\,T} S_{\beta_1}^{lnr} \\ \vdots \\ S_{\beta_{W_l}}^{lnr\ T} S_{\beta_{W_l}}^{lnr} \end{bmatrix},\ \Lambda_\alpha = \begin{bmatrix} S_{\alpha_1}^{lnr\,T} S_{\alpha_1}^{lnr} \\ \vdots \\ S_{\alpha_{H_l}}^{lnr\ T} S_{\alpha_{H_l}}^{lnr} \end{bmatrix} \tag{4.69}$$

Démonstration. Voir annexe B.3.4. □

Nous avons donc maintenant l'expression recherchée :

$$\frac{\partial Vec\,(\phi_l)}{\partial \epsilon_k} = L_l \left(\Omega_l \frac{\partial Vec\,(\phi_{l-1})}{\partial \epsilon_k} \odot \mathbf{H}_l\right) \tag{4.70}$$

Remarque : Les matrices $\mathbf{H}_l,\ L_l$ sont des fonctions de ϕ_{l-1} et donc de ϕ_0.

Expression du gradient de ϕ_l fonction de ϕ_0

L'équation 4.70 nous donne une expression du gradient de ϕ_l en fonction du gradient de la couche précédente ϕ_{l-1}. Cependant, elle ne permet pas de remonter jusqu'au gradient de la couche d'entrée ϕ_0. Nous introduisons un nouvel opérateur $\boxdot$ et réexprimons 4.70 de façon à pouvoir remonter en cascade au gradient initial.

Définition 12. *Soient* $A = \begin{bmatrix} \mathbf{a}_1^T \\ \vdots \\ \mathbf{a}_n^T \end{bmatrix} \in M_{n,m}$ *et* $\mathbf{x} \in \mathbb{R}^m$. *On note* $\boxdot$ *le produit entre* A *et* $\mathbf{x}$ *défini par*

$$A \boxdot \mathbf{x} = A \odot \left(\mathbf{1}_n \otimes \mathbf{x}^T\right) = \begin{bmatrix} \mathbf{a}_1^T \odot \mathbf{x}^T \\ \vdots \\ \mathbf{a}_n^T \odot \mathbf{x}^T \end{bmatrix} \tag{4.71}$$

avec $\mathbf{1}_n \in \mathbb{R}^n$ *un vecteur de* 1.

Théorème 2. *Soient* $A \in M_{n,m}$ *et* $\mathbf{x},\ \mathbf{y} \in \mathbb{R}^m$. *Alors*

$$A\left(\mathbf{x} \odot \mathbf{y}\right) = \left(A \boxdot \mathbf{y}\right)\mathbf{x} = \left(A \boxdot \mathbf{x}\right)\mathbf{y} \tag{4.72}$$

Démonstration. La propriété est immédiate en développant les expressions de chaque côté du signe égal. La seconde égalité est une conséquence directe de la commutativité du produit de Hadamard. □

Théorème 3. *Soit* $\mathbf{u} \in \mathbb{R}^n$. *On a*

$$\mathbf{diag}(\mathbf{u}) = I_n \boxdot \mathbf{u} \tag{4.73}$$

L'application du théorème 2 à (4.70) nous donne

$$\begin{aligned} \frac{\partial Vec\left(\phi_l\right)}{\partial \varepsilon_k} &= \left(L_l \boxdot \mathbf{H}_l\right)\Omega_l \frac{\partial Vec\left(\phi_{l-1}\right)}{\partial \varepsilon_k} \\ &= \hat{\Omega}_l \frac{\partial Vec\left(\phi_{l-1}\right)}{\partial \varepsilon_k} \end{aligned} \tag{4.74}$$

D'où l'on déduit par itération

$$\frac{\partial Vec\,(\phi_l)}{\partial \varepsilon_k} = \left(\prod_{i=1}^{l} \hat{\Omega}_{l-i+1}\right) \frac{\partial Vec\,(\phi_0)}{\partial \varepsilon_k} \tag{4.75}$$

En posant $A_l = \left(\prod_{i=1}^{l} \hat{\Omega}_{l-i+1}\right)$ et $M = \begin{bmatrix} M_1 & M_2 & M_3 & M_4 \end{bmatrix}$ on obtient l'expression finale de la jacobienne

$$\begin{aligned} \frac{\partial Vec\,(\phi_l)}{\partial \boldsymbol{\varepsilon}} &= \begin{bmatrix} A_l M_1 \hat{\nabla} I & \dots & A_l M_4 \hat{\nabla} I \end{bmatrix} && (4.76) \\ &= A_l M (\hat{\nabla} I \otimes I_4) && (4.77) \end{aligned}$$

Application

Une première application de cette formalisation va nous permettre, en ajoutant des couches appropriées au réseau, de directement calculer le plan tangent courant $\frac{\partial Vec(\phi_l)}{\partial \varepsilon}$ au descripteur $Vec\,(\phi_l)$. Nous ignorons pour l'instant les couches LNR. En reprenant l'équation (4.74) avec $L_l = I_{H_l W_l}$, on obtient

$$\begin{aligned} \frac{\partial Vec\,(\phi_l)}{\partial \varepsilon_k} &= \left(I_{H_l W_l} \boxdot \mathbf{H}_l(\phi_{l-1})\right) \Omega_l \frac{\partial Vec\,(\phi_{l-1})}{\partial \varepsilon_k} && (4.78) \\ &= \mathbf{diag}(H_l(\phi_{l-1}))\Omega_l \frac{\partial Vec\,(\phi_{l-1})}{\partial \varepsilon_k} && (4.79) \end{aligned}$$

On doit cependant également prendre en compte en pratique le pooling P qui consiste dans la plupart des réseaux à conserver le maximum sur une région autour d'un point

$$\forall x,\ P(x) = \max_{i,j} \phi^{ij}(x) \tag{4.80}$$

On a alors ses dérivées

$$\frac{\partial P}{\partial \varepsilon_k}(\phi) = \frac{\partial \phi^{i^* j^*}}{\partial \varepsilon_k} \text{ avec } i^*, j^* = arg\max_{i,j} \phi^{ij} \tag{4.81}$$

Finalement, on passe du plan tangent à la couche $l-1$ à la couche l par

$$\frac{\partial Vec\,(\phi_l)}{\partial \varepsilon_k} = \frac{\partial P}{\partial \varepsilon_k} \left(\mathbf{diag}(H_l(\phi_{l-1}))\Omega_l \frac{\partial Vec\,(\phi_{l-1})}{\partial \varepsilon_k}\right) \tag{4.82}$$

Comme on le verra par la suite, ceci peut se calculer directement dans le réseau de neurones en

rajoutant certaines couches. Maintenant il reste à calculer la jacobienne initiale $\frac{\partial Vec(\phi_0)}{\partial \varepsilon}$. Partons de l'équation (4.47). Sans perte de généralité, on considère que la position courante représentée par ε_0 est l'origine, c'est à dire que $\varepsilon_0 = \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix}^T$ et par conséquence $\mathbf{x}_{ij}^{\varepsilon} = \mathbf{x}_{ij}^{I}$ (autrement dit, la transformation de similarité considérée ici est relative à la position précédente et non exprimée par rapport au repère global de l'image). On a alors

$$\Gamma_{0,\varepsilon_0}(\mathbf{x}_{ij}^{\varepsilon}) = \mathbf{x}_{ij}^{\varepsilon} = \mathbf{x}_{ij}^{I} \tag{4.83}$$

$$\Gamma_{1,\varepsilon_0}(\mathbf{x}_{ij}^{\varepsilon}) = \begin{bmatrix} -j & i & 0 \end{bmatrix}^T \tag{4.84}$$

$$\Gamma_{2,\varepsilon_0}(\mathbf{x}_{ij}^{\varepsilon}) = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^T \tag{4.85}$$

$$\Gamma_{3,\varepsilon_0}(\mathbf{x}_{ij}^{\varepsilon}) = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}^T \tag{4.86}$$

Finalement on obtient donc

$$\frac{\partial \phi_0^{ij}}{\partial \boldsymbol{\varepsilon}} = \begin{bmatrix} \frac{\partial c}{\partial i}(\mathbf{x}_{ij}^{I})i + \frac{\partial c}{\partial j}(\mathbf{x}_{ij}^{I})j \\ \frac{\partial c}{\partial j}(\mathbf{x}_{ij}^{I})i - \frac{\partial c}{\partial i}(\mathbf{x}_{ij}^{I})j \\ \frac{\partial c}{\partial i}(\mathbf{x}_{ij}^{I}) \\ \frac{\partial c}{\partial j}(\mathbf{x}_{ij}^{I}) \end{bmatrix}^T \tag{4.87}$$

Le plan tangent nécessite donc seulement le calcul du gradient spatial de l'image à la position (échelle et rotation comprises) actuelle.

Voyons maintenant son implémentation pratique. Nous utilisons la bibliothèque Caffe [JSD+14] mais le principe est adaptable de manière générale à d'autres frameworks. On rappelle qu'une couche est définie par un tenseur 3D $D \times H \times W$ (profondeur, hauteur, largeur). Dans Caffe, ce sont en fait des tenseurs 4D $N \times D \times H \times W$ avec le premier axe correspondant au batch. En effet, il est plus efficace de traiter plusieurs images d'un coup que de répéter le traitement pour chaque image. La figure 4.23 présente un schéma des couches nécessaires pour le calcul de la jacobienne initiale. On part de l'image d'entrée qui est une couche de dimension $1 \times 3 \times H \times W$. Le gradient spatial

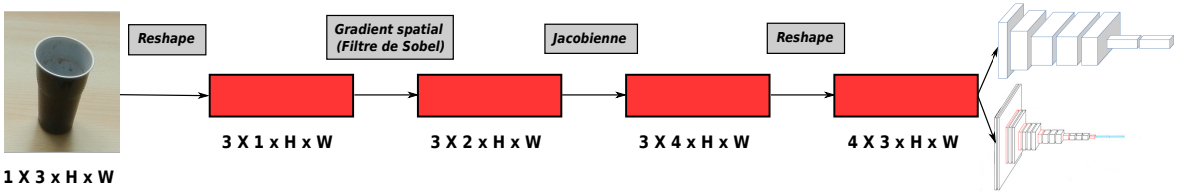


FIGURE 4.23 – Schéma des couches à ajouter à un CNN afin de calculer le plan tangent initial

de l'image va être calculé indépendamment pour chaque canal (la profondeur, ici, correspond aux couleurs). L'astuce consiste à considérer chaque canal comme une donnée et de les traiter en bloc comme un batch. Autrement dit, il faut permuter l'axe correspondant au batch et aux canaux d'où l'utilisation d'une couche *Reshape*, qui comme son nom l'indique, permet de permuter les données. On obtient alors en sortie un tenseur $3 \times 1 \times H \times W$. Celui-ci est envoyée ensuite dans une couche qui va calculer le gradient spatial de l'image à partir d'un filtre de Sobel (couche de convolution dont on a fixé le noyau pour être celui de Sobel). On a deux filtres (dérivée horizontale et verticale) d'où une sortie de dimension $3 \times 2 \times H \times W$. La jacobienne $\frac{\partial Vec(\phi_0)}{\partial \varepsilon}$ est ensuite calculée à partir des équations (4.87) où chaque canal va correspondre à la dérivée par rapport à un paramètre de similarité. La sortie obtenue a donc une profondeur égale à 4. Enfin, pour pouvoir propager cette jacobienne dans la suite du réseau, il faut repasser les canaux de couleurs (actuellement en tant que batch) en tant qu'axe de profondeur. C'est pourquoi nous réutilisons une seconde couche *Reshape* pour effectuer l'opération inverse de la première transformation.

Le réseau CNN original est ensuite modifié selon la figure 4.24.

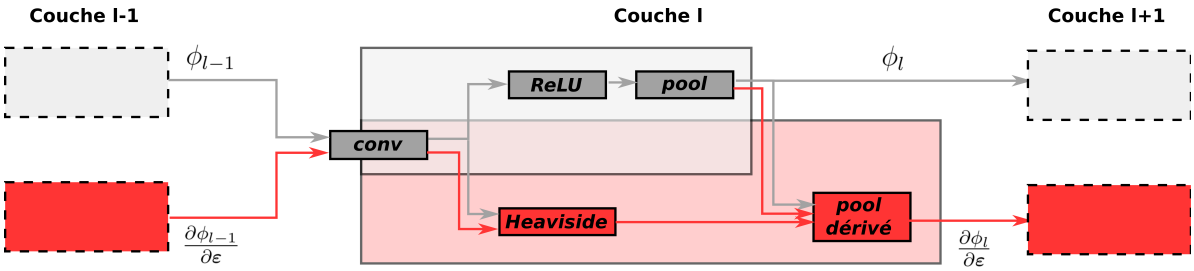


FIGURE 4.24 – Adaptation des couches de convolutions pour le calcul de la jacobienne du descripteur à la couche l

Chaque couche est constituée de deux parties :

- La première est exactement celle du réseau original, à savoir l'enchaînement des couches *conv* + *ReLU* + *pool* calculant le descripteur ϕ_l (en gris dans la figure 4.24)
- La seconde s'occupe de calculer la jacobienne de ϕ_l suivant l'équation (4.82). Elle prend en entrée la jacobienne de la couche précédente $\frac{\partial \phi_{l-1}}{\partial \varepsilon}$ qui passe à travers la même couche de convolution. Les sorties de la couche *conv* pour ϕ_l et $\frac{\partial \phi_{l-1}}{\partial \varepsilon}$ sont ensuite passées en entrée de la couche *Heaviside*. La dernière couche *pool dérivé* implémente l'équation (4.81) : elle nécessite en entrée un masque indiquant les index i^*, j^* correspondant aux maximum locaux de ϕ_l, la sortie de la couche *Heaviside* ainsi que les dimensions (hauteur et largeur) de la sortie.

Il est également possible d'obtenir la jacobienne, autrement dit l'espace tangent au descripteur, de manière dense en utilisant la méthode FCN.

Opérateur de K-dérivation

On définit ici l'opérateur de K-dérivation (ou dérivation de Kronecker) [Ter03] [DB15]

Définition 13. *Soit* $f : \mathbb{R}^n \to \mathbb{R}^m$ *telle que*

$$\forall \mathbf{x} = [x_1 \dots x_n]^T \in \mathbb{R}^n,\ \mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x}) \dots f_m(\mathbf{x})]^T$$

L'opérateur différentiel est noté $D_{\mathbf{x}} = \left[\frac{\partial}{\partial \mathbf{x_1}} \dots \frac{\partial}{\partial \mathbf{x_n}}\right]^T$. *L'opérateur de K-dérivation noté* $D_{\mathbf{x}}^{\otimes}$ *est défini par*

$$D_{\mathbf{x}}^{\otimes}\mathbf{f}(\mathbf{x}) = \mathbf{f}(\mathbf{x}) \otimes D_{\mathbf{x}} = Vec\left(J_{\mathbf{f}}^T(\mathbf{x})\right) \in M_{nm,1} \tag{4.88}$$

avec $J_{\mathbf{f}} = \left[\frac{\partial f_i}{\partial x_j}\right]$ *la matrice jacobienne de* f. *En utilisant la notation (B.4), la dérivation à l'ordre* k *est définie par*

$$D_{\mathbf{x}}^{\otimes k}\mathbf{f} = D_{\mathbf{x}}^{\otimes}\left(D_{\mathbf{x}}^{\otimes k-1}\mathbf{f}\right) \in M_{n^k m,1} \tag{4.89}$$

Quelques nouvelles propriétés de cet opérateur et leur démonstration sont données en annexe B.2. L'application principale de la K-dérivation est de permettre une expression de série de Taylor multivariée [DB15] similaire au cas univarié

Théorème 4. *Soient* $f : \mathbb{R}^n \to \mathbb{R}^m$ *et* $\mathbf{x_0} \in \mathbb{R}^n$. *La série de Taylor de* f *au point* $\mathbf{x_0}$ *est donnée par*

$$\forall \mathbf{x} \in \mathbb{R}^n,\ f(\mathbf{x}) = \sum_{i=0}^{+\infty} \frac{1}{i!}\left(D_{\mathbf{x}}^{\otimes i}\mathbf{f}(\mathbf{x_0})\right)^T (\mathbf{x} - \mathbf{x_0})^{\otimes i} \tag{4.90}$$

On obtient donc une approximation du premier ordre en $\boldsymbol{\varepsilon}_0$ à partir de l'expression (4.76)

$$\begin{aligned}
Vec\left(\phi_l\right)(\boldsymbol{\varepsilon}) &= Vec\left(\phi_l\right)(\boldsymbol{\varepsilon}_0) + \left(D_{\boldsymbol{\varepsilon}}^{\otimes} Vec\left(\phi_l\right)(\boldsymbol{\varepsilon}_0)\right)^T (\boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}_0) && (4.91)\\
&= Vec\left(\phi_l\right)(\boldsymbol{\varepsilon}_0) + \left(Vec\left(\frac{\partial Vec\left(\phi_l\right)}{\partial \boldsymbol{\varepsilon}}(\boldsymbol{\varepsilon}_0)^T\right)\right)^T (\boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}_0) && (4.92)\\
&= Vec\left(\phi_l\right)(\boldsymbol{\varepsilon}_0) + \left(Vec\left(\hat{\nabla} I^T(\boldsymbol{\varepsilon}_0) M^T(\boldsymbol{\varepsilon}_0) A_l^T\right)\right)^T (\boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}_0) && (4.93)\\
&= Vec\left(\phi_l\right)(\boldsymbol{\varepsilon}_0) + \left(\left(A_l \otimes \hat{\nabla} I^T \otimes I_4\right) Vec\left(M^T\right)\right)^T (\boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}_0) && (4.94)\\
&= Vec\left(\phi_l\right)(\boldsymbol{\varepsilon}_0) + \left(Vec\left(M^T\right)^T \left(A_l^T \otimes \hat{\nabla} I \otimes I_4\right)\right)(\boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}_0) && (4.95)
\end{aligned}$$

Les approximations d'ordres supérieures sont également calculables à partir de la règle de dérivation en chaîne et de la mise à l'échelle présentées et démontrées en annexe B.2.

Discussion : La formalisation proposée ici a permis d'obtenir des expressions matricielles concises en évitant la complexité engendrée par l'écriture tensorielle. On a notamment montré que la jacobienne du descripteur à la couche l peut se factoriser sous la forme $A_l(\phi_0)M(\boldsymbol{\varepsilon})\nabla I(\phi_0, \boldsymbol{\varepsilon})$. Bien que les expressions obtenues soient difficilement exploitables en pratique dans leur forme actuelle, elles sont un point de départ pour de futurs développements. La première application de cette formalisation a permis de modifier un CNN afin qu'il calcule également directement la jacobienne. L'idée derrière ceci est de séparer les variations du descripteur dues à un changement non rigide d'apparence $\Delta\phi^{NR}$ et celles dues à une transformation de similarité $\Delta\phi^{SIM2}$. La figure 4.25 illustre ce point.

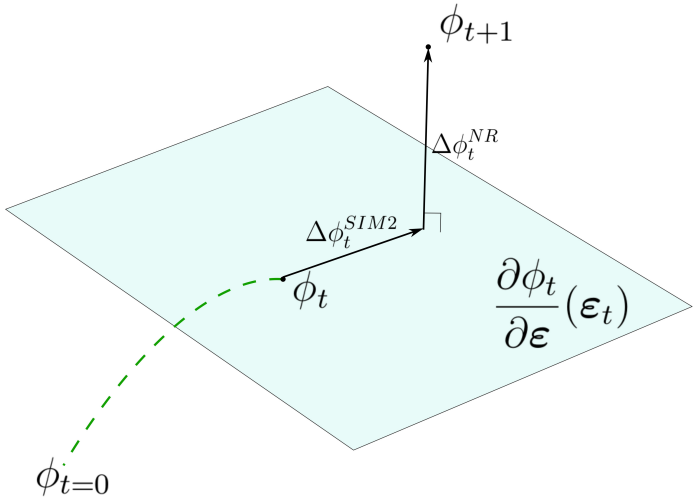


FIGURE 4.25 – Décomposition de la variation d'un descripteur

4.3 Conclusion et perspectives

Cette section a présenté notre modèle de perception. Il est responsable du traitement des données sensorielles au travers des tâches de classifications et de detections. Comme nous nous plaçons dans un monde ouvert, les descripteurs images considérés sont issus de couches intermédiaires de réseau de neurones convolutifs (CNN). Bien qu'appris sur un ensemble d'entraînement limité pouvant être totalement disjoint des données rencontrées, leur grande généralité permet leur utilisation à la fois pour la classification de concepts et pour la représentation d'instance. Un défaut de ces

descripteurs souvent relevé dans la littérature est leur sensibilité à la rotation et, dans une moindre mesure, aux changements d'échelle. Nous avons montré que les CNNs propagent les variétés générées par des transformations de similarité (**SIM(2)**) de façon relativement lisse, tout du moins jusqu'aux couches entièrement connectées. De là notre idée d'exploiter cette régularité pour des applications de suivis d'objets : on bénéficie ainsi à la fois de la robustesse naturelle de ces descripteurs tout en inférant les rotations et les changements d'échelles. Bien que notre application reste simpliste en considérant un modèle d'objet fixe sans mécanisme de mise à jour, nos premiers résultats sont encourageants et indiquent une nouvelle piste intéressante pour le suivi d'objet. Enfin, nous proposons une formalisation sous forme matricielle de la propagation des variations selon **SIM(2)**. L'objectif final de cette formalisation est de pouvoir :

- obtenir une justification théorique de la régularité observée.
- approximer la jacobienne des descripteurs images relativement à **SIM(2)** car le calcul numérique de celle-ci à chaque itération est coûteux.

Pour le deuxième point, nous envisageons l'utilisation de l'opérateur de K-dérivation qui permet d'obtenir une expression de la formule de Taylor dans le cas multivarié et multidimensionnel sous forme matricielle. En particulier, nous démontrons de nouvelles propriétés de cet opérateur.

La suite logique de ces travaux est de développer une méthode pour la mise à jour du descripteur référence. En particulier, un objet peut avoir selon les angles de vues des apparences extrêmement variables. Il serait plus intéressant dans notre cas, dans l'optique de reconnaissance d'instance, d'avoir un descripteur multimodal "data-driven". La difficulté principale est d'être capable de séparer les variations dues à un changement de point de vue de celle occasionnées par de l'occlusion : nous ne voulons pas inclure ces dernières dans le descripteur final de l'objet suivi.

Chapitre 5

Modèle d'instance et processus inter-modèles

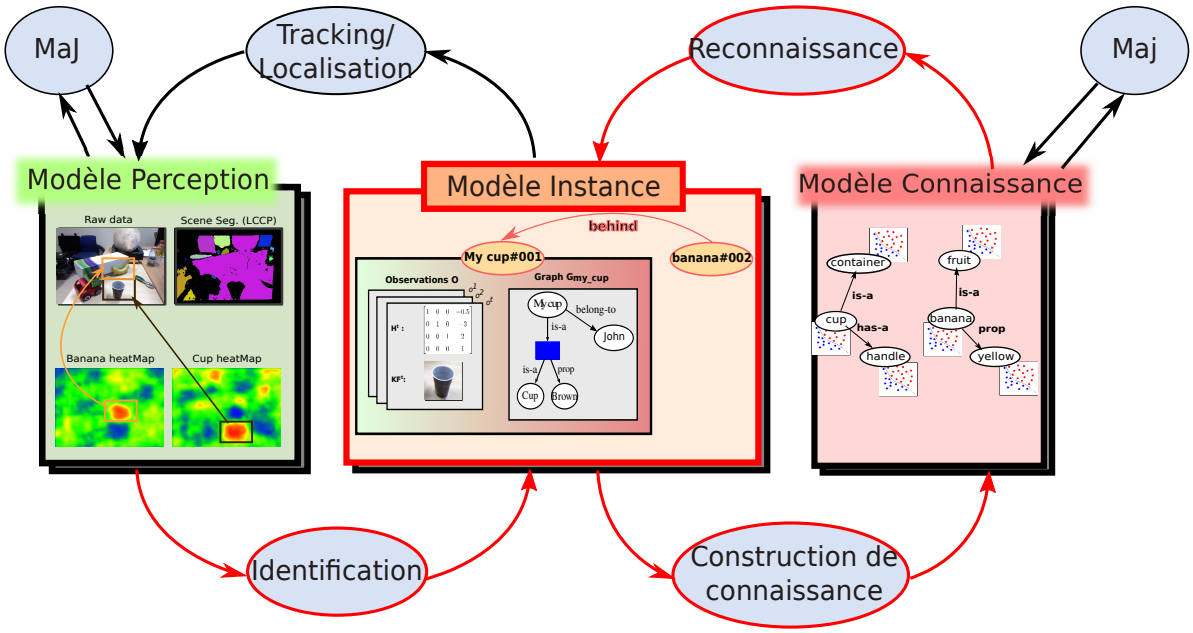


FIGURE 5.1 – Modèle d'instance et processus dans notre architecture

Dans le chapitre précédent, nous nous sommes attachés à décrire séparément les modalités perceptuelles et sémantiques. Nous allons donc maintenant voir les interactions entre ces unités à mesure que le robot acquiert de nouvelles données sémantiques et/ou sensorielles. On introduit pour cela un modèle d'instance qui va faire la jonction entre les modalités précédentes. Nous donnerons également la structure générale de l'implémentation. En effet, de nombreux processus concurrents

ont lieu simultanément : l'application finale est parallélisée en s'appuyant notamment sur les propositions d'implémentations de [Wil12].

5.1 Modèle d'instance

Les instances sont définies comme des réalisations physiques de concepts sémantiques. Le modèle d'instance est donc ainsi un modèle intermédiaire permettant de faire le pont entre données sensorielles et connaissances sémantiques. De plus, il n'est pas seulement intermédiaire qu'en termes d'abstraction mais également en termes temporels. En effet, le modèle de perception travaille à la fréquence d'acquisition des capteurs (ie un point dans le temps) alors que le modèle de connaissance travaille sur un espace de temps "infini". Le modèle d'instance, lui, n'est valide que sur un intervalle de temps fini. Cette propriété devient apparente lorsque l'on cherche à estimer des quantités nécessitant la notion de temps (ou de façon équivalente des séquences temporelles de données) comme par exemple les relations cinématiques entre objets (cf section 2.1.2). Dans le cadre de la robotique de manipulation, ce genre d'information est crucial et difficilement accessible par des données perceptuelles statiques [1]. De même, l'analyse de bases de connaissance comme Wikipedia ne peut donner au mieux qu'une vague description du type de relation cinématique : la page concernant le mot *"Door"* nous donne par exemple *[...]Doors normally consist of a panel that swings on hinges on the edge[...]*. Cette information est trop abstraite pour pouvoir être exploitée en vue d'un modèle orienté robotique.

La plupart des travaux précédents qui se basent sur des représentations multi-couches (section 2.3.3) ont tendance à considérer les instances et les concepts au sein d'une même couche. Dans notre modèle, nous séparons clairement les concepts des instances par cette différence fondamentale : un concept est **"intemporel"** et **générique** alors qu'une instance est **spécifique** au contexte courant du robot et **temporaire**. Dans la suite nous définissons formellement notre modèle d'instance.

5.1.1 Définition d'une instance

Une instance I est définie selon la figure 5.2. Elle est composée d'une description sémantique représentée par un graphe partageant le formalisme de notre modèle de connaissance (section 3.2.2). A cela peut s'ajouter des types d'arêtes et de nœuds spécifiques aux instances. Actuellement, nous considérons les personnes et la relation d'appartenance *belong-to* reliant une instance à son propriétaire eg. *belong-to(my_cup,John)*. La représentation explicite des utilisateurs et des relations

1. Sous-entendu sans données d'entraînement extérieures

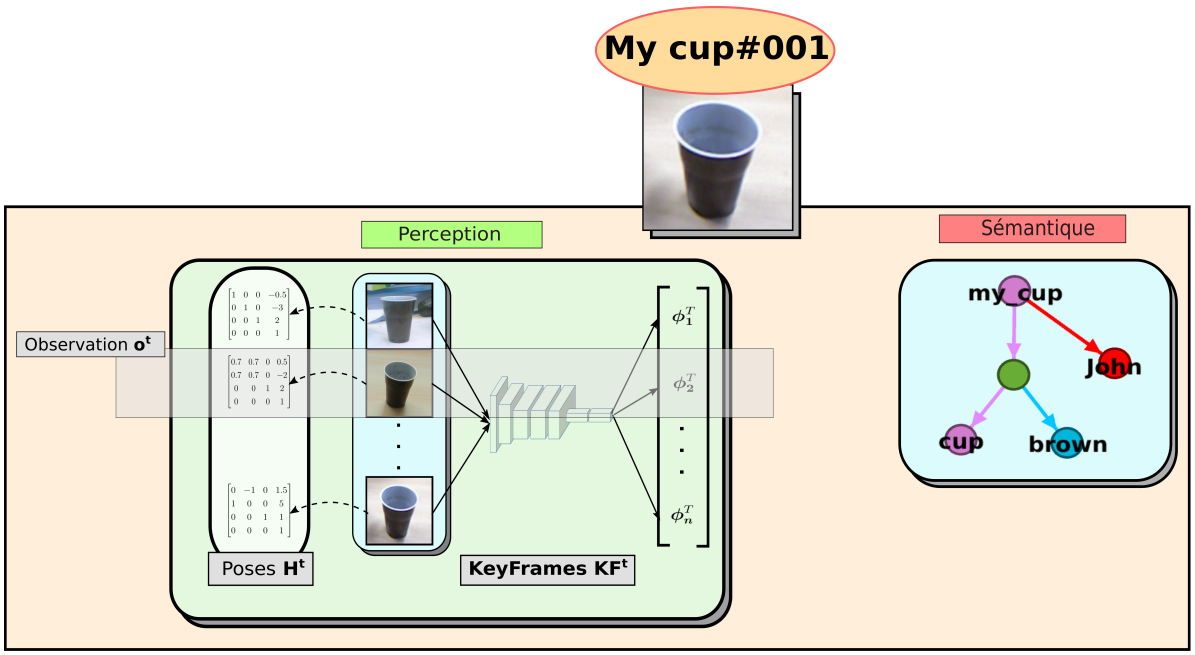


FIGURE 5.2 – Définition d'une instance

d'appartenance permet, entre autres, de contraindre l'espace de recherche d'instances dans une requête. Si un utilisateur $user$ effectue la requête *Bring me my cup*, le système va tout d'abord chercher les instances se nommant my_cup puis va étendre sa recherche aux instances héritant de cup. La recherche d'instance commence par filtrer selon l'adéquation des descriptions sémantiques. On se limitera donc à la recherche d'instance I vérifiant :

$$isA(I, cup) \wedge belong\text{-}to(I, user) \tag{5.1}$$

On verra également dans la section 5.1.3 d'autres utilisations envisageables de ces relations.

A l'aspect sémantique s'ajoute une description perceptuelle. Celle-ci est constituée des différentes observations de l'instance. Plus précisément, une observation est constituée d'une position, d'une date et d'une image correspondant à la détection. De ces images sont extraits des descripteurs images (chapitre 4) formant les lignes d'une matrice qui "résume" l'aspect visuel de l'instance.

Formalisation

Nous formalisons la définition d'une instance donnée précédemment. Une instance I est définie par

$$I = \{\mathbf{O_I},\ G_I\} \tag{5.2}$$

où $\mathbf{O_I}$ est un ensemble d'observations et G_I un graphe de description sémantique.

Une observation $o \in O_I$ est de la forme :

$$o = \{t,\ H,\ R,\ \phi(R)\} \text{ avec } \begin{cases} t \in \mathbb{R} \\ H \in \mathbf{SE(3)} \\ R \in \mathbb{R}^{w \times h \times 3} \\ \phi(R) \in \mathbb{R}^d \end{cases} \tag{5.3}$$

avec t la date d'observation, H la matrice de transformation euclidienne représentant la position de l'instance, R la région $w \times h$ de l'image correspondant à l'instance et $\phi(R)$ le descripteur image de dimension d. On construit une matrice M constituée des descripteurs de l'ensemble des observations :

$$M = \left[\phi(R)_1 \dots \phi(R)_{|\mathbf{O_I}|}\right]^T \tag{5.4}$$

avec $|\mathbf{O_I}|$ le cardinal de O_I.

Le graphe $G_I = \{V_I, E_I\}$ relie l'instance aux concepts du modèle de connaissance sémantique. Les nœuds $V_I \in V_K \cup V_I^P$ correspondent aux mêmes catégories que le graphe ontologique (voir table 3.1) avec l'ajout des nœuds "personne" V_I^P. Le nœud racine v_r du graphe, représentant le "concept" de l'instance, est considéré comme appartenant à $V_K^{C,O}$ (nœuds représentant des concepts objets). De même, les arêtes E_I peuvent représenter les relations de l'ontologie avec en plus une nouvelle relation *belong-to*.

5.1.2 Définition du modèle d'instance

Maintenant que nous avons vu comment décrire une instance, nous allons aborder la définition du modèle d'instance en lui-même. Il consiste en un graphe illustré à la figure 5.3. Les nœuds de ce graphe sont constitués des instances mais également des requêtes. En effet, elles sont faites par un utilisateur et font intervenir des instances. Il est donc naturel de représenter explicitement les liens les reliant. Cela permet :

- d'obtenir des relations (indirectes) entre les utilisateurs et les instances, et entre les instances elles-même. Comme les instances n'ont intrinsèquement pas de lien entre elles (hors relation spatiale), les requêtes servent d'intermédiaires.
- d'augmenter la capacité d'inférence du robot. En effet, en incluant ces relations sous forme de clauses Prolog, le robot peut "raisonner" sur ce que lui ont demandé les utilisateurs précédemment.

La gestion interne des requêtes sous forme de tâche est traitée à la section 5.5 et nous ne rentrerons pas plus dans les détails ici.

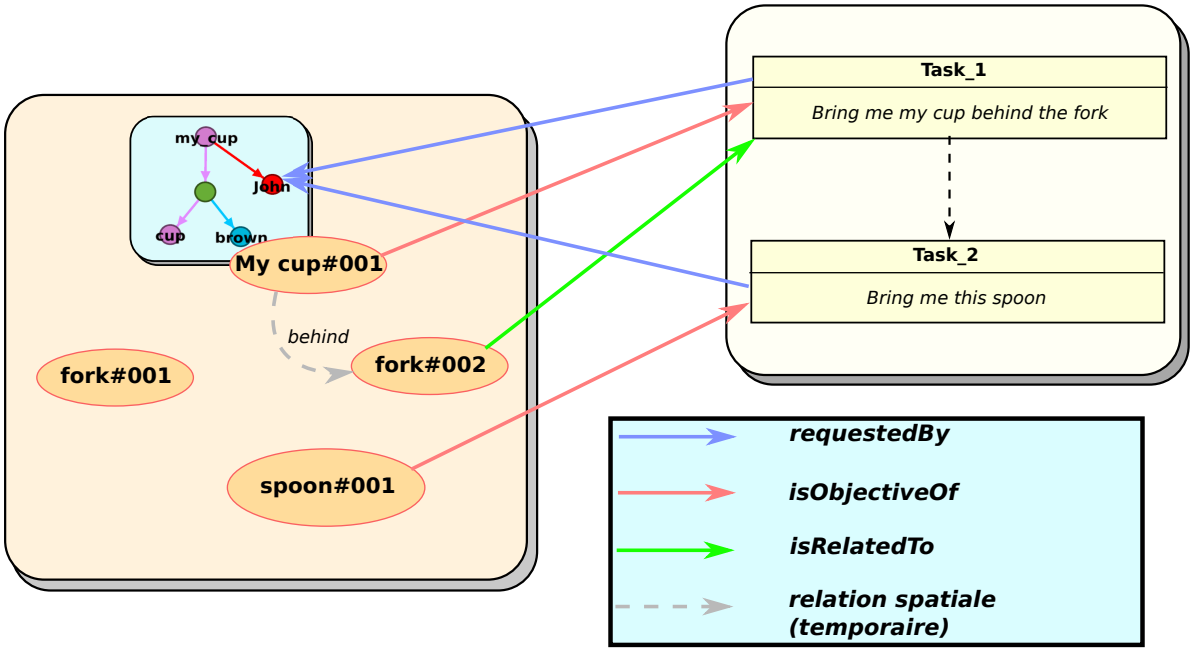


FIGURE 5.3 – Schéma du modèle d'instance

Dans l'exemple de la figure 5.3, la requête *Bring me the cup behind the fork.* fait référence à deux instances : *my_cup* et *fork*, contraints par une relation spatiale. Les arêtes du graphe vont essentiellement relier les tâches aux instances. On considère trois types de relations :

- **requestedBy** : indique l'auteur de la requête eg. *requestBy(task_1,john)*.
- **isObjectiveOf** : indique un objectif de la requête. Par exemple, l'objectif de la requête *Bring me the cup behind the fork* est l'instance *my_cup* et en conséquence on aura la relation *isObjectiveOf(my_cup,task_1)*. A noter qu'une requête peut avoir plusieurs objectifs comme dans *Bring me a knife and a fork*. Dans ce cas, on aura simplement les deux relations *isObjectiveOf(knife,task_1)* et *isObjectiveOf(fork,task_1)*.
- **isRelatedTo** : indique une instance, autre qu'un objectif, intervenant dans une requête. Avec notre exemple de requête, on a *isRelatedTo(fork,task_1)*.
- **behind/in_front_of, to_the_left/right_of, under/above** : ensemble de relations spatiales entre instances. Celles ci sont particulières car elles sont **temporaires**. En effet, on verra à la section 5.5 qu'elles sont estimées à la volée lorsqu'une requête nécessite ce genre d'information.

Formalisation

Le modèle d'instance est défini par un graphe $G_{MI} = \{V_{MI}, E_{MI}\}$. Les nœuds $V_{MI} = V_{inst} \cup V_m$ se regroupent en deux types :

- les nœuds instances $V_{inst} \in V_I$ qui correspondent aux nœuds racine v_r des graphes G_{I_i} des instances et aux nœuds représentant les personnes (utilisateurs).
- les nœuds tâches V_m qui correspondent aux tâches générées par les requêtes utilisateurs.

Les arêtes E_{inst} représentent les quatre relations décrites précédemment, à savoir *requestBy*, *isObjectiveOf*, *isRelatedTo* et les relations spatiales temporaires. La table 5.1 regroupe ces relations et les types de nœuds qu'elles relient.

Relation	$\mathbf{V_{depart}}$	$\mathbf{V_{arrive}}$
requestedBy	V_m	V_I^P
isObjectiveOf	$V_K^{C,O}$	V_m
isRelatedTo	$V_K^{C,O}$	V_m
behind,...	$V_K^{C,O}$	$V_K^{C,O}$

TABLE 5.1 – Relations représentées dans le graphe d'instances

5.1.3 Discussion

Cette section décrit notre définition d'une instance et par extension du modèle d'instance. Celles-ci font le pont entre perception sensorielle et connaissance sémantique au travers de leur descriptions bimodales. De plus, elles sont également reliées aux utilisateurs humains par la relation d'appartenance et les requêtes qu'ils effectuent. Actuellement, nous n'exploitons pas toutes ces relations à leur plein potentiel. En sortant du cadre purement robotique, il serait envisageable d'exploiter ces liens afin d'en déduire des habitudes ou des indications sur l'utilisateur permettant de gérer des informations implicites. Cela se rapprocherait alors des problématiques de préférences utilisateurs, fortement exploitées par exemple dans la publicité sur Internet qui s'adapte en fonction des recherches (requêtes) passées. Le robot pourrait inférer des informations non données dans une requête utilisateur. Plus précisément, il pourrait utiliser les requêtes passées pour définir une probabilité *a-priori* sur les instances requises. Considérons donc un utilisateur A demandant souvent

une tasse accompagnée d'une cuillère particulière. A la prochaine requête du type *"Bring me a cup and a spoon"* par l'utilisateur A, le robot va donc rechercher en priorité la cuillère particulière.

Une autre perspective serait de pouvoir transférer la notion de contexte associée aux concepts aux requêtes successives faites par les utilisateurs. On pourrait alors rapprocher notre approche avec celle proposée par KnowRob qui s'attache à la description de tâches séquentielles (exécution de recettes de cuisine). L'idée serait donc, à partir des relations de notre ontologie, que notre robot trouve *pourquoi* certaines instances sont requises afin d'effectuer une certaine tâche.

5.2 Sémantique vers instance

5.2.1 Interaction avec l'humain

Concentrons nous maintenant sur l'autre extrémité de notre modèle concernant les données sémantiques. Nous allons considérer trois cas d'interactions avec un utilisateur humain :

- Transmission de connaissance générale eg. *A cup has an handle*.
- Transmission d'informations sur une instance eg. *My cup is red*.
- Une requête eg. *Bring me the cup behind the banana*.

L'ajout d'informations sémantiques par les deux premières interactions est représenté sur la figure 5.4. Les entrées sémantiques sont stockées dans une file d'attente qui est ensuite dépilée par un thread d'analyse sémantique similaire au chapitre 3, qui permet également de classifier l'entrée selon les trois catégories précédentes. Les données textuelles sont ensuite converties en un ensemble de faits Prolog et sont ajoutées soit au modèle de connaissance soit à la description de l'instance considérée. Ce processus est décrit par l'algorithme 5. Le type de l'entrée est actuellement simplement basé sur l'analyse du premier mot : un verbe à l'impératif indique une requête, un pronom possessif une instance et un article indéfini une connaissance générale. Nous pourrions aussi considérer l'emploi d'un article défini *the* comme indiquant une instance. Lorsque plusieurs instances peuvent être associées à l'information donnée, nous envisageons l'ajout d'un retour vers l'utilisateur par une question simple permettant de lever au maximum l'ambiguïté. Le cas d'une requête sera abordé à la section 5.5.

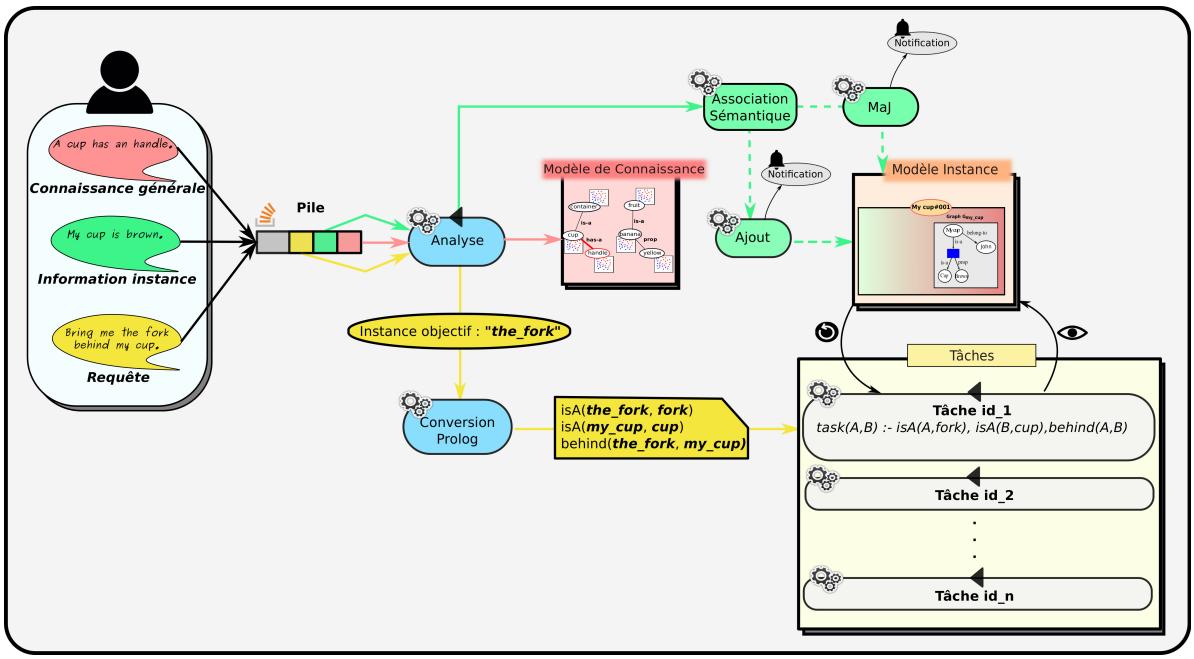


FIGURE 5.4 – Schéma des processus entre le modèle de connaissance sémantique et le modèle d'instance

Algorithme 4 Extraction du type d'entrée sémantique

Entrée: Phrase $\mathbf{s} = [w_1 \dots w_n]^T, w_i = [u_i\ d_i\ x_i\ h_i] \in W$ ▷ cf section 3.3.3

Sortie: Type d'information $T \in \{General,\ Instance,\ Requete,\ Autre\}$

function ENTRÉETYPE(S)
  if $d_1 = \mathbb{ROOT} \wedge x_1 = \mathbb{VB}$ **then**
    $T \leftarrow Requete$
  else if $u_1 = \mathbb{DET}$ **then**
    if $d_1 = \mathbb{POSS}$ **then**
      $T \leftarrow Instance$
    else
      $T \leftarrow General$
    end if
  else
    $T \leftarrow Autre$
  end if
end function

Algorithme 5 Algorithme d'analyse d'entrée sémantique

Entrée: Phrase $\mathbf{s} = [w_1 \dots w_n]^T, w_i = [u_i\ d_i\ x_i\ h_i] \in W$

Sortie: Ensemble de faits Prolog $\mathcal{P} = \{P_i\}$,
Type d'information $T \in \{General,\ Instance,\ Requete,\ Autre\}$

1: $T \leftarrow EntreeType(\mathbf{s})$ ▷ Algorithme 4
2: **if** $T = Requete$ **then**
3: $\quad AnalyseSemantiqueRequete(\mathbf{s})$ ▷ Algorithme 7
4: **else if** $T \neq Autre$ **then**
5: $\quad \mathcal{P} \leftarrow AnalyseSyntaxique(s)$ ▷ cf section 3.3.3
6: **end if**

5.2.2 Détection de concepts

Il y a deux façons de rattacher les concepts du modèle de connaissance aux instances détectées : soit considérer une instance et la rattacher à des concepts sémantiques, soit considérer un concept et vérifier si une instance est une réalisation de celui-ci. Le premier point sera abordé à la section 5.3. Le second point est un problème de classification classique. La plupart des classifieurs sont entraînés sur un ensemble de classes et retournent la probabilité d'appartenance à l'une des ces classes. Comme nous nous plaçons dans l'hypothèse d'un monde ouvert, on doit se restreindre à des classifieurs binaires associés à chacune des classes. Cette discussion a déjà été abordée à la section 4.2.2. Lors de la détection d'objets dans une scène à partir d'une requête (section 5.5), le classifieur doit être appliqué de manière dense sur l'image. Pour cela, on remplace la couche de classification du FCN de base par une couche calculant la prédiction du Random Forest binaire associé à un concept. On obtient en sortie une *heatMap* représentant la probabilité d'appartenance à une classe en chaque pixel de l'image. Un exemple est donné à la figure 5.5.

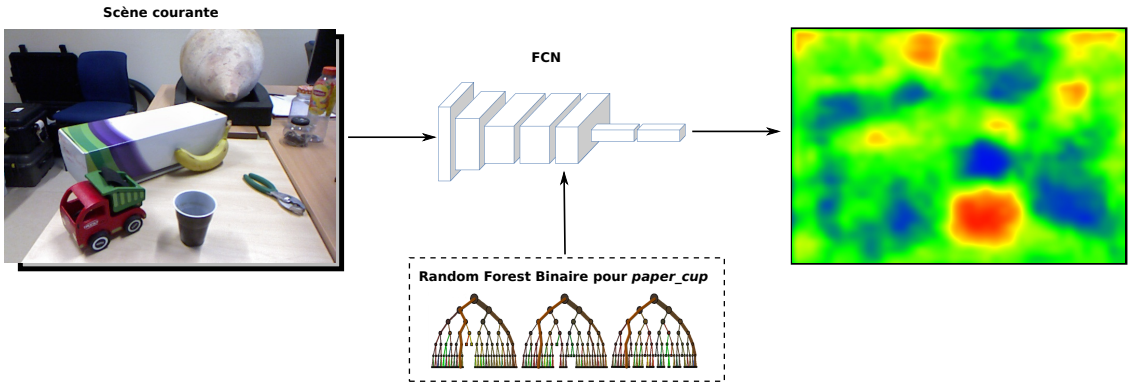


FIGURE 5.5 – Exemple de heat map obtenue par un FCN doté d'une couche Random Forest

5.3 Instance vers sémantique

5.3.1 Problématiques

On va aborder ici le problème de la remontée d'information du modèle d'instance au modèle de connaissance sémantique. Le problème est d'inférer à quel concept peut être associée une instance, sachant que le concept dont elle est une réalisation peut ne pas encore être représenté dans l'ontologie. Une solution "brute force" serait d'appliquer l'ensemble des classifieurs RF associés aux concepts de l'ontologie mais cela pose le problème de la mise à l'échelle et du temps d'exécution. De plus, le cas où le concept représentant l'instance n'est pas dans l'ontologie ne serait pas géré. On se retrouve face à une problématique similaire au *zero-shot learning* (section 2.3.1) à la différence qu'ici nous ne supposons même pas la connaissance des classes non vues via des sources extérieures. Le problème se décompose ainsi en trois parties, avec des hypothèses différentes :

1. **Hypothèse** : Les instances sont des réalisations de concepts connus.

   Objectif : Inférer l'attachement d'une instance aux concepts sémantiques dans le cas d'une ontologie large : mise à l'échelle, performance en termes de temps d'exécution

2. **Hypothèse** : Les instances sont des réalisations d'un concept non présent dans l'ontologie (similaire à l'hypothèse du *zero-shot learning*).

   Objectif : Le problème est d'inférer le plus proche parent auquel l'on pourrait associer l'instance. Par exemple, rattacher une instance de *bouteille* au concept de *conteneur* lorsque le concept *bouteille* n'est pas présent dans l'ontologie.

3. **Hypothèse** : Les instances peuvent être des réalisations de concepts connus et inconnus (similaire au *zero-shot learning généralisé* [CCGS16]). L'appartenance de l'instance à l'une ou l'autre de ces catégories est inconnue.

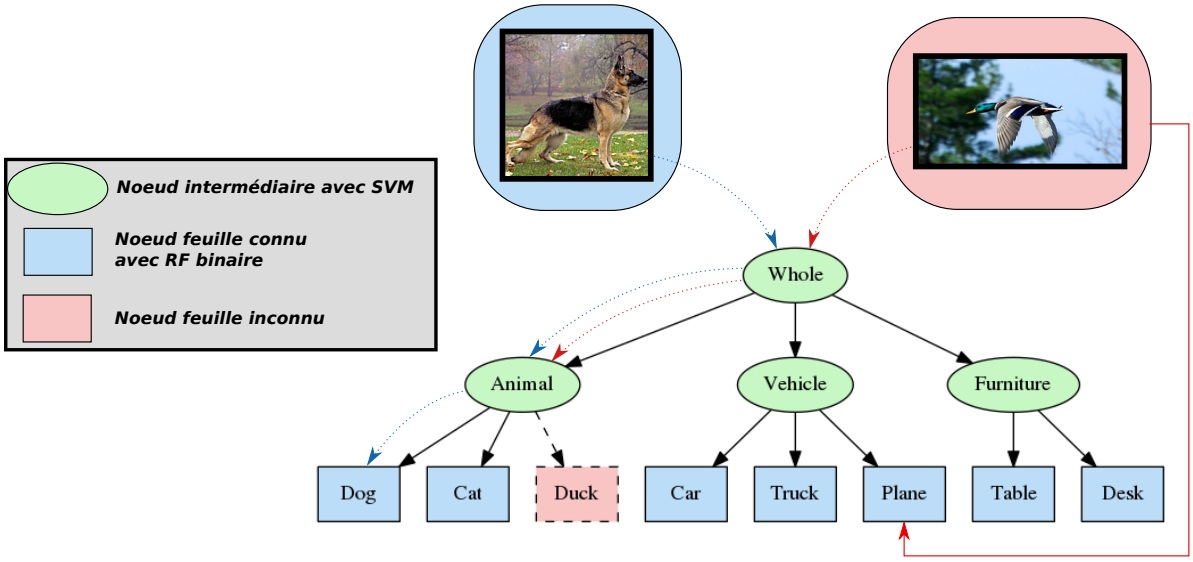


FIGURE 5.6 – Schéma du problème de rattachement d'instance à l'ontologie avec un exemple d'instance de classe connue (*dog*) et de classe inconnue (*duck*). A chaque nœud feuille a été entraîné un classifieur binaire RF. Notre solution consiste à considérer l'ontologie comme un arbre de classification, où la fonction de décision est un classifieurs SVM entraîné sur les classes filles.

> **Objectif** : Inférer si l'instance est issue d'un concept connu ou d'un concept inconnu. Dans le premier cas, on pourra donc le rattacher à un nœud feuille du graphe de connaissance, alors que dans le second cas on l'attachera au plus proche parent contenu dans l'ontologie.

La figure 5.6 en est une illustration. On suppose une ontologie donnée avec un classifieur binaire associé à chaque nœud feuille connu. Considérons maintenant une nouvelle instance observée qui peut être la réalisation d'un concept connu (*dog*) ou inconnu (*duck*). La méthode "brute force" appliquerait chaque classifieur binaire au descripteur visuel de l'instance (soit au total 8 dans notre exemple) en associant l'instance au concept donnant la probabilité maximum. Lorsqu'une classe non vue est observée, elle est forcément rattachée à la classe la plus similaire visuellement. Dans la figure 5.6, ceci est illustré par l'instance de *duck* détectée comme une instance de *plane*.

Dans la suite, nous allons présenter notre solution originale et l'évaluer au regard des hypothèses de travail précédentes.

5.3.2 Arbre de classification ontologique

Notre solution originale, que l'on nomme **OCT** [2], consiste à utiliser l'ontologie comme un arbre de classification en entraînant sur chaque nœud intermédiaire (possédant au moins deux fils) un classifieur SVM multiclasses. Actuellement, nous nous limitons aux seules relations isA. Ceci permet de répondre directement aux deux premières problématiques (1) et (2) décrites précédemment : la classification de l'instance de *dog* ne nécessite plus que 2 estimations et l'instance de $duck$ est rattachée au dernier noeud n'étant pas un noeud fils (ici *Animal*).

Nous allons maintenant montrer la pertinence de cette approche pour les problèmes (1) et (2) au travers d'évaluations quantitatives. Celles-ci se basent sur une ontologie construite de la façon suivante. 44 concepts ont été sélectionnés et constituent les noeuds feuilles. Nous avons intégré ensuite récursivement l'ensemble des nœuds parents selon l'ontologie WordNet (hypernyme) jusqu'au nœud racine. Nous avons retiré les concepts intermédiaire n'ayant qu'un seul fils car ils ne sont pas exploités dans notre méthode. Pour chaque concept feuille, nous avons récupéré l'ensemble des images disponibles avec sa bounding box sur ImageNet. Celui-ci a été ensuite séparé en deux ensembles d'entraînement et de test selon un ratio 80/20. L'utilisation des bounding box diminue cependant le nombre d'images disponibles d'autant plus qu'un nombre non négligeable de bounding box sont associées à des images inexistantes. Ces bounding box invalides sont gérées au préalable par une étape de prétraitement vérifiant l'existence de l'image associée et leur conformité (bounding box ne sortant pas de l'image).

Cette ontologie est visible à la figure 5.8, suivant la même notation qu'à la figure 5.6. Les concepts feuilles sont représentés par des rectangles bleus avec leur WordnetId.

Evaluation sur concepts connus

Complexité : Tout d'abord, nous allons justifier le fait que notre méthode OCT ait une complexité nettement inférieure à la méthode "brute force". En notant $N_{feuille}$ le nombre de concepts feuilles ayant un classifieur binaire associé, cette méthode a une complexité en $O(N_{feuille})$. Pour avoir une approximation de la complexité dans le pire des cas et en moyenne d'OCT, il nous faut connaître la borne supérieur d_{max} et la moyenne des distances $\bar{d}$ séparant les nœuds feuilles du nœud racine de l'ontologie, c'est à dire le nombre de nœuds intermédiaires auxquels seront associés un classifieur SVM. On aura alors respectivement pour complexité $O(d_{max})$ et $O(\bar{d})$. On calcule pour cela la répartition de ces distances pour l'ensemble des concepts feuilles de WordNet, en se limitant aux catégories utilisées dans le cadre de ces travaux (elles apparaissent soulignées à la table

2. Abréviation de *Ontological Classification Tree* ou *Arbre de classification ontologique* en français

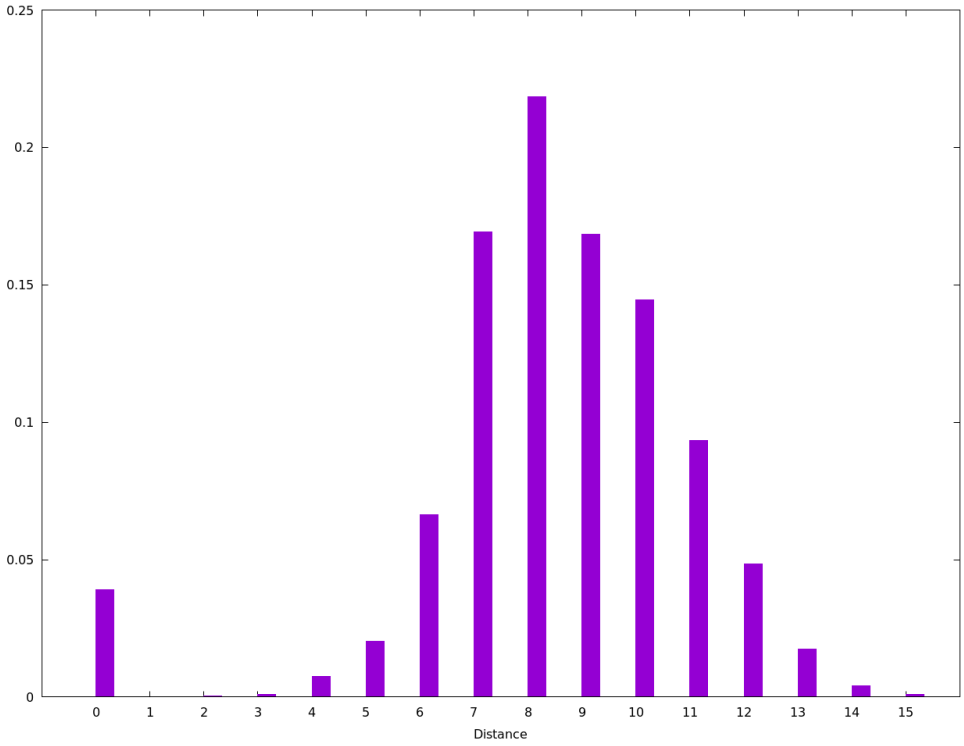


FIGURE 5.7 – Répartition relative des distances entre les nœuds feuilles et le nœud racine de WordNet (limité aux catégories soulignées à la table 3.2). Nombre de nœud feuille : 23330

3.2). L'histogramme résultant est présenté à la figure 5.7 avec ici $N_{feuille} = 23330$. En excluant les nœuds "isolés" (distance de 0), on obtient quasiment une gaussienne avec $\bar{d} = 8$ et $d_{max} = 15$.

Il est clair que $N_{feuille} \gg d_{max}$, validant ainsi notre propos.

Erreurs de classification : Afin d'évaluer la précision de notre approche, nous la comparons à celle obtenue lorsqu'un classifieur SVM multiclasse (que l'on note MSVM) unique est appris sur l'ensemble des concepts feuilles. Nous ne considérons pas ici des classifieurs binaires pour les raisons suivantes :

- Un classifieur binaire nécessite la définition, relativement arbitraire, d'un ensemble de négatifs qui influe sur la précision.
- L'analyse menée par Daniely [DSS12] compare les erreurs théoriques d'estimation ε_{MSVM} d'un SVM multiclasse et d'un ensemble de classifieurs binaires ε_{OvA} (*one-vs-all*(OvA)). Il montre que $\varepsilon_{OvA} \geq \varepsilon_{MSVM}$. Autrement dit nous comparons notre méthode **OCT**, par abus de langage, à la "borne supérieure" obtenue par un ensemble de classifieurs binaires.

Les erreurs de classifications sont présentées à la table 5.2 et détaillées par classe à la table 5.3.

FIGURE 5.8 – Ontologie employée pour l'évaluation de notre méthode OCT. Elle comporte 44 classes feuilles (bleu) et 27 nœuds intermédiaires (vert) auxquels la méthode OCT associe un classifieur SVM

	AN_fc7	VGG_fc7	GN_pool5
MSVM	12.14%	7.99%	7.96%
OCT	13.85%	8.97%	9.16%

TABLE 5.2 – Erreurs de classifications selon la méthode MSVM et OCT

L'utilisation de l'ontologie en tant qu'arbre de classification diminue le nombre de classifieurs à entraîner (27 contre 44 ici) mais surtout le nombre d'inférences nécessaires (égal à la distance du concept inféré au noeud racine). On observe cependant en contrepartie une dégradation attendue des performances, avec une grande variation selon les classes considérées. Les résultats obtenus à partir des dernières couches de VGG et GoogLeNet sont semblables alors qu'AlexNet donne un taux d'erreur plus élevé. Cela semble indiquer l'importance de la capacité de représentation du réseau qui doit être suffisante. Une fois de plus, tout comme dans les résultats de la section 4.2.5, les réseaux donnant de meilleurs performances de classification ne sont pas meilleur ici. Il faut en effet prendre en compte l'arrière plan qui est également capturé et qui peut rendre deux concepts différents "proches" s'ils apparaissent sur des lieux similaires (eau, savane, table).

Evaluation sur concepts inconnus

Nous nous intéressons dorénavant au point (2), à savoir le rattachement d'instances de concepts non représentés dans l'ontologie. Nous considérons un ensemble de concepts, présentés à la table 5.4, qui n'apparaissent pas dans l'ontologie de la figure 5.8. Cependant, nous pouvons les lier à des concepts parents ou, à défaut, aux catégories les plus proches. Nous dénommons donc par "lignée" la trajectoire attendue lorsqu'une instance d'une de ces classes traverse l'arbre de décision ontologique via notre méthode OCT. La dernière colonne donne les concepts feuilles que l'on considère le plus proche de ces concepts inconnus. Afin de montrer la pertinence de notre approche, nous allons comparer l'association des concepts selon deux méthodes :

- L'approche MSVM du point (1). Une instance est correctement associée si elle est classifiée comme un concept voisin (méthode des plus proches voisins).
- Notre approche OCT. Une instance est correctement associée si sa trajectoire dans l'arbre de décision correspond à la lignée attendue (deuxième colonne de la table 5.4).

Les résultats sont présentés à la table 5.5.

Ces premiers résultats montrent qu'OCT permet de rattacher des concepts inconnus avec un taux d'erreur légèrement plus faible qu'avec MSVM. Ceci se constate pour les trois réseaux considérés. VGG semble présenter le meilleur compromis, avec une capacité suffisamment élevée afin

Concept	**WordnetId**	**MSVM**			**OCT**		
		AN_fc7	*VGG_fc7*	*GN_pool5*	*AN_fc7*	*VGG_fc7*	*GN_pool5*
kit fox	n02119789	7.69%	3.08%	1.54%	7.69%	1.54%	1.54%
dog	n02084071	16.39%	3.28%	9.84%	21.31%	6.56%	9.02%
bear	n02131653	15.28%	11.11%	11.11%	16.67%	12.5%	11.11%
tigress	n02129923	12.24%	4.08%	2.08%	10.20%	4.08%	2.08%
persian cat	n02123394	4.60%	1.15%	2.30 %	4.60%	2.30%	2.30%
cougar	n02125311	17.33%	10.67%	8%	21.33%	6.67%	6.67%
lynx	n02127052	11.46%	4.17%	8.33%	10.42%	3.12%	8.33%
wood mouse	n02336641	10.64%	8.51%	4.25%	14.89%	10.64%	6.38%
grey whale	n02066245	4.76%	6.67%	5.71%	4.76%	3.81%	7.62%
dolphin	n02068974	15.85%	15.85%	12.19%	14.63%	13.41%	10.97%
sea lion	n02077923	15.45%	12.19%	16.26%	19.51%	13.82%	19.51%
bull	n02403325	29.27%	2.44%	7.5%	29.27%	7.32%	12.50%
pony	n02382437	17.65%	9.8%	11.76%	21.57%	7.84%	13.72%
zebra	n02391049	1.06%	2.13%	2.13%	1.06%	3.19%	2.13%
chimpanzee	n02481823	6.00%	4.00%	6.00%	5.00%	5.00%	6.00%
gorilla	n02480855	8.65%	2.88%	3.85%	8.65%	2.88%	4.81%
pangolin	n02461830	22.45%	12.25%	14.28%	26.53%	20.41%	20.41%
frog	n01639765	2.86%	2.86%	0.00%	8.57%	5.71%	2.86%
soccer ball	n04254680	6.06%	1.01%	1.01%	11.11%	3.03%	6.06%
revolver	n04086273	8.64%	4.94%	1.23%	19.75%	8.64%	9.88%
watch	n04555897	3.37%	3.37%	1.12%	4.49%	5.62%	1.12%
laptop	n03642806	5.66%	1.89%	1.89%	5.66%	5.66%	2.83%
airbus	n02686121	10.00%	11.43%	7.14%	10.00%	10.00%	8.57%
biplane	n02842573	20.00%	18.33%	15%	16.67%	13.33%	11.67%
bomber	n02867715	16.39%	9.84%	14.75%	16.39%	9.84%	9.84%
drone	n03245889	31.43%	28.57%	31.43%	34.29%	31.43%	34.29%
helicopter	n03512147	18.52%	16.67%	12.96%	22.22%	14.81%	14.81%
ferry	n03329663	8.51%	6.38%	8.51%	12.76%	6.38%	10.64%
motorboat	n03790230	16.48%	7.69%	10.99%	15.38%	8.79%	12.09%
small boat	n04244997	38.09%	23.81%	30.95%	38.09%	30.95%	30.95%
bicycle	n02834778	8.70%	0.00%	8.70%	10.87%	6.52%	8.69%
truck	n04490091	22.22%	11.11%	11.11%	22.22%	11.11%	11.11%
car	n02958343	5.36%	2.98%	3.57%	8.93%	4.17%	4.17%
plate	n03960374	25.81%	9.68%	9.68%	25.81%	6.45%	19.35%
umbrella	n04507155	14.00%	11.00%	8%	19.00%	14.00%	10.00%
cup	n03147509	12.90%	12.90%	6.45%	19.35%	16.13%	16.13%
knife	n03623556	32.14%	46.43%	25%	42.86%	50.00%	32.14%
dresser	n03237340	7.55%	5.66%	5.66%	5.66%	7.55%	5.66%
floor lamp	n03367059	17.39%	21.74%	21.74%	17.39%	17.39%	17.39%
table lamp	n04380533	12.37%	4.12%	5.15%	10.31%	4.12%	4.12%
chair	n03001627	31.11%	24.44%	31.11%	40.00%	26.67%	28.89%
sofa	n04256520	16.28%	15.91%	9.30%	13.95%	15.12%	13.95%
table	n04379243	16.48%	8.79%	8.79%	19.78%	10.99%	9.89%
apple	n07739125	4.88%	3.66%	3.66%	6.10%	4.88%	4.88%

TABLE 5.3 – Erreurs de classification par classe selon la méthode MSVM et OCT

Concept	Lignée attendue	Concepts connus voisins
gazelle-n02423022	*whole-vertebrate-placental-ungulate-**equine***	bull,pony,zebra
lion-n02129165	*whole-vertebrate-placental-carnivore-feline*	cougar,lynx,tigress, persian_cat
fish-n02512053	*whole-vertebrate-placental-**aquatic_mammal***	sea_lion,grey_whale, dolphin
bus-n02924116	*whole-artifact-instrumentality-**vehicle-wheeled_vehicle-motor_vehicle***	car,truck
spoon-n04284002	*whole-artifact-**instrumentality***	knife
car_seat-n02970685	*whole-artifact-instrumentality-**furniture-seat***	chair,sofa
duck-n01846331	*whole-vertebrate-placental-**aquatic_mammal***	sea_lion,grey_whale, dolphin

TABLE 5.4 – Concepts employés dans l'évaluation du rattachement d'instances inconnues. La colonne *Lignée* correspond à la trajectoire attendue des instances dans la méthode OCT. On a noté en gras les concepts qui ne sont pas correctes en terme de sémantique pure (ie diffèrent des hyperonymes donnés par WordNet). La dernière colonne donne les concepts voisins appartenant au même groupe. Dans le cas de l'évaluation par MSVM, on considère une instance correctement rattachée lorsqu'elle est classifiée parmi ces concepts

de pouvoir séparer les différentes classes mais pas trop pour ne pas sur-représenter l'arrière plan.

Pour conclure, les résultats obtenus incitent à investiger plus en profondeur cette approche simple et naturelle. Une extension envisagée est de remplacer les classifieurs SVM par des CNNs partageant leur couches inférieures et initialisés par le CNN attaché au concept parent.

Concepts	MSVM			OCT		
	AN_fc7	*VGG_fc7*	*GN_pool5*	*AN_fc7*	*VGG_fc7*	*GN_pool5*
gazelle-n02423022	25.71%	8.74%	16.30%	36.13%	21.34%	16.97%
lion-n02129165	60.11%	72.76%	82.02%	77.34%	88.43%	92.7%
fish-n02512053	23.53%	49.02%	23.04%	29.90%	52.94%	32.35%
bus-n02924116	96.26%	96.68%	86.72%	95.85%	97.51%	92.94%
spoon-n04284002	48.17%	45.98%	32.84%	47.44%	49.63%	35.03%
car_seat-n02970685	69.35%	70.97%	83.06%	62.09%	71.77%	74.19%
duck-n01846331	46.81%	48.58%	32.98%	52.13%	51.77%	38.83%

TABLE 5.5 – Proportion d'instances rattachées aux concepts de fin de lignée dans le cas d'OCT (deuxième colonne de la table 5.4), ou étant classifiées comme un des concepts voisins (dernière colonne de la table 5.4) dans le cas MSVM.

5.4 Perception vers instance / instance vers perception

Les objets détectés par la segmentation non supervisée sont transmis au modèle d'instance qui est responsable de leur traitement. Ainsi, il faut donc tout d'abord contrôler si ces objets correspondent à une instance déjà existante. Si ce n'est pas le cas, on va alors créer une nouvelle instance définie seulement par ses données sensorielles (cf section 5.1). La figure 5.9 illustre ce processus. Les objets sont détectés dans un thread séparé et sont représentés en tant qu'instances. Celles-ci sont ajoutées à une file d'attente sur laquelle attend le processus d'association. Ce processus est également exécuté dans un thread propre et est chargé de trouver les correspondances entre les instances détectées et celles déjà existantes. Il y a deux possibilités : si une instance détectée est associée à une instance existante alors elle est mise à jour, sinon nous créons une nouvelle instance (cf algorithme 6). Nous utilisons comme dans le cas de la recherche d'instances (section 4.2.2) la similarité cosinus entre deux instances I_i et I_j :

$$S_C(I_i, I_j) = \frac{\phi(I_i)^T \phi(I_j)}{||\phi(I_i)||_2 ||\phi(I_j)||_2} \in [0, 1] \tag{5.5}$$

avec ϕ une fonction donnant une représentation vectorielle de l'instance. Actuellement, nous gardons distinctes les représentations sémantiques et sensorielles. ϕ correspond au descripteur image extrait des couches d'un CNN.

Afin d'optimiser la recherche dans le modèle d'instance, nous commençons par les instances qui ont été détectées aux instants précédents. Jusqu'à présent, nos expériences sont limitées à un nombre relativement faible d'instances. Dans le cas où le nombre d'instances devient important, il est nécessaire de stocker les descripteurs image de façon à pouvoir les parcourir de manière efficace.

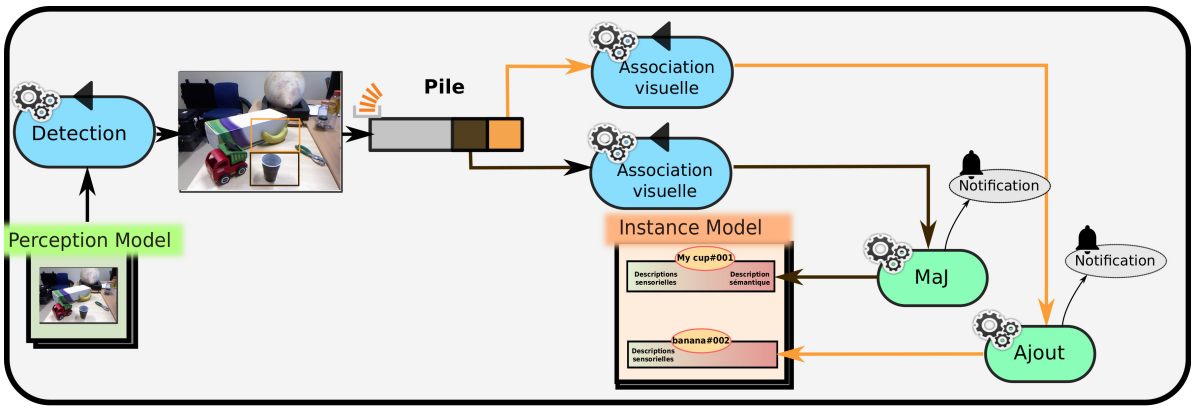


FIGURE 5.9 – Schéma des processus entre le modèle de perception et le modèle d'instance, en supposant l'instance *My cup* déjà connue

Algorithme 6 Processus d'association visuelle des instances détectées avec les instances connues

Entrée: Instance détectée I_d, Ensemble des instances connues $\mathcal{I} = \{I_{c,i}\}_i$, seuil de similarité σ

for $I_c \in \mathcal{I}$ **do**
  if mesureSimilarité(I_d,I_c) $< \sigma$ **then** ▷ cf (5.5)
    $maJInstance(I_c, I_d)$ ▷ cf section 5.1
    $notification\_MaJ(I_c)$
  else
    $ajoutNouvelleInstance(I_d)$
    $notificationNouvelleInstance(I_d)$
  end if
end for

Nous envisageons pour cela de les représenter dans une structure d'arbre k-d.

Les processus dans le sens Instance-Perception ont déjà été abordés à la section 4.2.2 avec la recherche d'instance en adaptant un FCN. Ceci est représenté à la figure 5.10 en utilisant différents descripteurs image.

On remarque au passage que les résultats de l'analyse faite à la section 4.2.4 se retrouve sur les différentes heat map. En effet, le contraste est moins marqué dans le cas de la couche *fc7* de VGG car sa plus grande capacité lui permet de capturer également l'arrière plan. AlexNet, au contraire, doit se limiter aux éléments les plus saillants. Ces résultats apparaissent sur les graphiques de la figure 4.11 où l'on voit que les similarités inter et extra classes sont plus importantes pour VGG que pour AlexNet. On verra par la suite que les tâches requérant la détection de certaines classes d'objets se font directement avec le modèle d'instance.

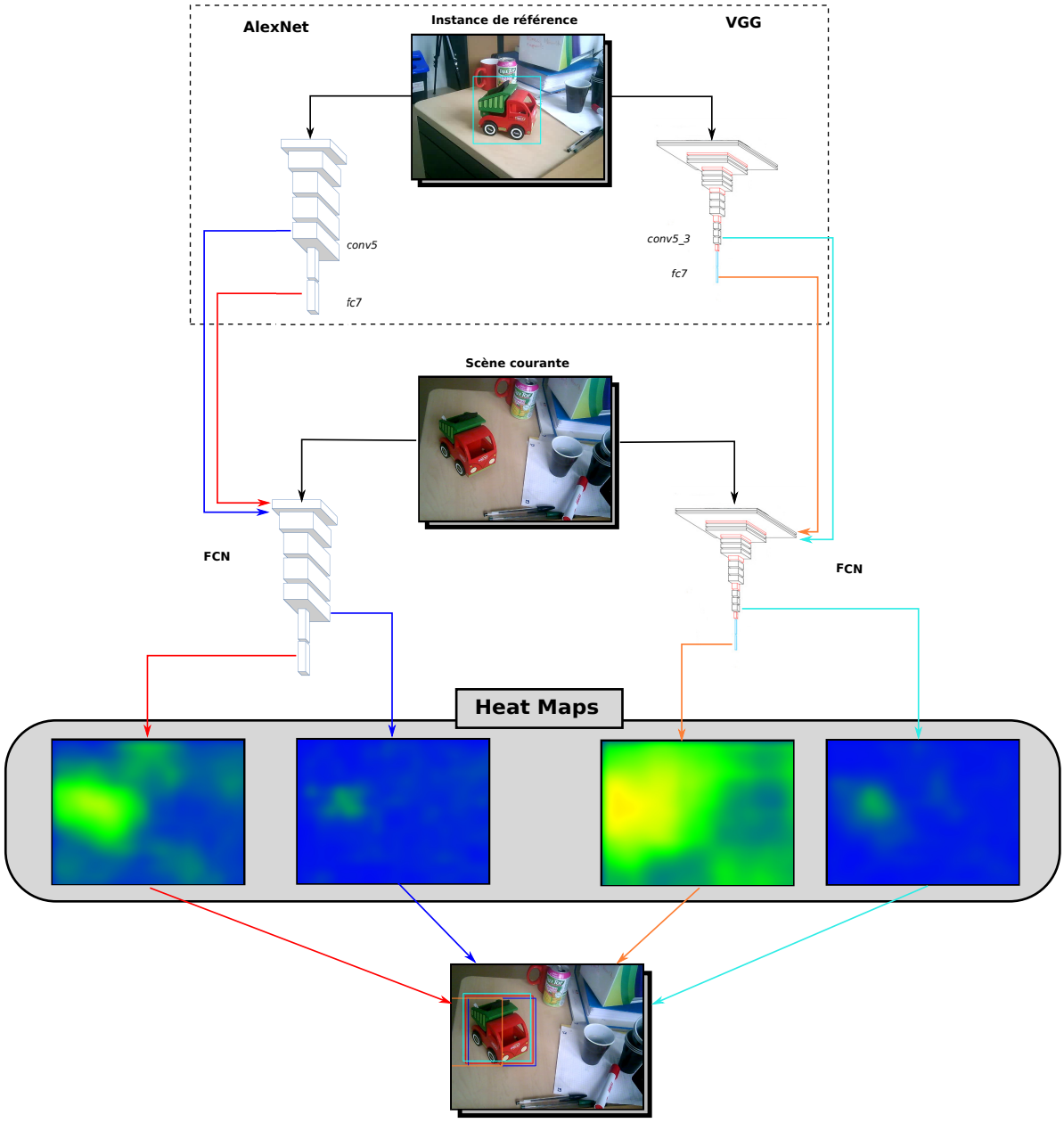


FIGURE 5.10 – Heat map pour la détection d'instance

Algorithme 7 Algorithme d'analyse de requête [3]

Entrée: Phrase $S = [\mathbf{w_1} \dots \mathbf{w_n}]^T, \mathbf{w_i} = [u_i\ d_i\ x_i\ h_i] \in W$
Sortie: Règle Prolog R_m/k où k est le nombre d'instances non déterminées apparaissant dans la requête, indice p de l'objet de la requête

$p \leftarrow 0,\ CPL \leftarrow \emptyset$ ▷ Ensemble de clauses Prolog
$GN \leftarrow$ extractionGroupeNominaux(s) ▷ cf alg. 1, sec. 3.3.3
for $gn = \{idx_{det}, idx_{nom}, I_{comp}\} \in GN$ **do**
    if $d_{idx_{nom}} = \mathbb{DOBJ} \vee d_{idx_{nom}} = \mathbb{POBJ}$ **then**
        $CPL \leftarrow$ conversionEnClausePrologGN(gn) ▷ cf alg. 8
        if $h_{idx_{nom}} = 1$ **then** ▷ Pointe vers le verbe à l'impératif
            $p \leftarrow idx_{nom}$
        end if
    end if
end for
$GR \leftarrow$ extractionGroupeRelation(s) ▷ cf alg. 2, sec. 3.3.3
for $gr \in GR$ **do**
    $CPL \leftarrow$ conversionEnClausePrologGR(gr) ▷ cf alg. 8
end for
$R_m/k \leftarrow creationRegleTâche(CPL)$ ▷ cf alg. 9

5.5 Tâches

Nous abordons enfin le dernier type d'entrée sémantique : les requêtes utilisateur. Elles vont être fortement dépendantes de l'application finale du robot. KnowRob [Ten11], par exemple, se concentre sur la réalisation de recettes de cuisine. Dans notre cas, on se restreint à des requêtes basiques montrant le fonctionnement général de notre modèle : elles seront limitées à la recherche d'instances. En interne, les requêtes vont générer une ou des tâches qui peuvent elles-même être composées de sous-tâches. Le processus global est indiqué en figure 5.4.

Par la suite nous considérons l'exemple type suivant : *Donne moi la fourchette derrière ma tasse*.

3. On considère ici uniquement le cas où l'objet de la requête est limité à une instance. Nous ne traitons pas non plus le cas où il y a des disjonctions ("ou")

5.5.1 Interprétation de requête

Conversion de requête en clauses Prolog

La requête est analysée sémantiquement selon l'algorithme 7. Le verbe d'action à l'impératif (eg. *Bring*) va définir le type de la mission. L'objet de la requête (eg une instance de classe *fork* dans notre exemple) est ensuite extrait ainsi que les autres groupes de relations. Elle est ensuite convertie en clauses Prolog (algorithme 8). Nous définissons ici une clause Prolog F de la façon suivante :

$$F \quad = \quad P\left(\boldsymbol{\alpha} = \{a_1, \ldots, a_n\}\right)\left[\boldsymbol{\alpha}^i,\ \boldsymbol{\alpha}^c,\ \boldsymbol{\alpha}^f\right] \tag{5.6}$$

avec P le prédicat et $\boldsymbol{\alpha}$ l'ensemble des arguments. Les ensembles seront notés en gras pour éviter toute confusion. Nous rappelons également que les constantes Prolog (termes atomiques) sont écrites en minuscule et les variables en majuscule. $\boldsymbol{\alpha}^i$ représente ici le sous ensemble des arguments correspondant à des instances, $\boldsymbol{\alpha}^c$ les arguments décrivant une classe et $\boldsymbol{\alpha}^f$ sont des arguments servant d'intermédiaires (équivalent des nœuds facteur du modèle de connaissance cf chapitre 3).

Notre exemple de requête sera donc converti en l'ensemble de clauses Prolog suivant :

Clause	**P**	$\boldsymbol{\alpha}^i$	$\boldsymbol{\alpha}^c$	$\boldsymbol{\alpha}^f$
isA(the_fork, fork)	isA	{the_fork}	{fork}	$\emptyset$
isA(my_cup, cup)	isA	{my_cup}	{cup}	$\emptyset$
behind(the_fork, my_cup)	behind	{the_fork, my_cup}	$\emptyset$	$\emptyset$

Une requête plus complexe faisant intervenir des arguments intermédiaires comme *Bring me the fork with a red handle behind my small cup* donne, en notant *t.f.w.a.r.h* pour *the fork with a red handle* :

Clause	**P**	$\boldsymbol{\alpha}^i$	$\boldsymbol{\alpha}^c$	$\boldsymbol{\alpha}^f$
isA(t.f.w.a.r.h, fork)	isA	{t.f.w.a.r.h}	{fork}	$\emptyset$
isA(my_small_cup, cup)	isA	{my_small_cup}	{cup}	$\emptyset$
isA(red_handle, handle)	isA	$\emptyset$	{handle}	{red_handle}
prop(red_handle, red)	prop	$\emptyset$	{red}	{red_handle}
hasA(t.f.w.a.r.h, red_handle)	hasA	{t.f.w.a.r.h}	$\emptyset$	{red_handle}
prop(my_small_cup, small)	prop	{my_small_cup}	{small}	$\emptyset$
behind(the_fork, my_cup)	behind	{the_fork, my_cup}	$\emptyset$	$\emptyset$

Jusqu'à présent, nous ne considérions que des prédicats verbaux. Dans le cas de l'analyse de requête, nous élargissons ceux-ci avec des prédicats indiquant des positions relatives : *behind/2, in_front_of/2, to_the_left_of/2, to_the_right_of/2, under/2, above/2*. Ce sont ces prédicats qui relient temporairement les instances entre elles dans le modèle d'instance. Les clauses de spatialité ont la particularité d'être dépendantes de l'environnement : un objet peut être déplacé et changer de position relative. C'est pourquoi nous n'intégrons pas ces clauses définitivement dans le modèle

Algorithme 8 Conversion de l'analyse syntaxique en clauses Prolog

Entrée: Groupe nominal $gn = [w_{det},\ w_{nom},\ W_{comp}]$,
Groupe de relation $gr = [gn_s,\ w_{predicat},\ gn_o,\ W_{adv}]$
Sortie: Clauses Prolog $CPL = \{F_j\}$ avec $F_j = P_j(\boldsymbol{\alpha_j})[\boldsymbol{\alpha_j^i}, \boldsymbol{\alpha_j^c}, \boldsymbol{\alpha_j^f}]$

function CONVERSIONENCLAUSEPROLOGGN(gn)
  $w_{cl} = w_{nom}$
  $w_{inst} = w_{det} + \sum_{w \in w_{adj}} w + w_{nom}$
  if $w_{adj} = \emptyset$ **then**
    $CPL \leftarrow isA(w_{inst}, w_{cl})[\{w_{inst}\}, \{w_{cl}\}, \emptyset]$
  else
    $CPL \leftarrow isA(w_{inst}, facteur)[\{w_{inst}\}, \emptyset, \{facteur\}]$
    $CPL \leftarrow isA(facteur, w_{cl})[\emptyset, \{w_{cl}\}, \{facteur\}]$
    for $w \in w_{adj}$ **do**
      $CPL \leftarrow prop(facteur, w)[\emptyset, \{w\}, \{facteur\}]$
    end for
  end if
end function
function CONVERSIONENCLAUSEPROLOGGR(gr)
  $w_{instSujet} = w_{det}^{gn_s} + \sum_{w \in w_{adj}^{gn_s}} w + w_{nom}^{gn_s}$
  $w_{instObjet} = w_{det}^{gn_o} + \sum_{w \in w_{adj}^{gn_o}} w + w_{nom}^{gn_o}$
  $CPL \leftarrow w_{predicat}(w_{instSujet}, w_{instObjet})[\{w_{instSujet}, w_{instObjet}\}, \emptyset, \emptyset]$
end function

d'instance. De manière similaire aux "computable classes" de KnowRob [Ten11], nous calculons ces relations à la volée au moment de la résolution de la clause.

De même, nous introduisons la règle $isDetected/1$ qui est elle aussi vérifiée lors de son appel. Elle est vérifiée lorsque l'instance en argument est détectée dans la scène courante.

Représentation des tâches par règle Prolog

La tâche générée par la requête est une règle Prolog constituée des clauses précédentes. Plusieurs cas peuvent se présenter :

– L'instance my_cup a déjà été vue, on obtient alors la règle :

$$R_m(A_1) : \text{-}\ isDetected(my\_cup),\ isA(A_1, fork),\ behind(A_1, my\_cup) \tag{5.7}$$

– L'instance my_cup n'a pas été vue mais a déjà été définie sémantiquement. On obtient alors,

Algorithme 9 Création d'une règle définissant une tâche à partir de la représentation en clause Prolog d'une requête

Entrée: Clauses Prolog $CPL = \{F_j\}_{1,\dots,n}$ avec $F_j = P_j(\boldsymbol{\alpha_j})[\boldsymbol{\alpha_j^i}, \boldsymbol{\alpha_j^c}, \boldsymbol{\alpha_j^f}]$
Sortie: Règle de la tâche R_m/k avec arguments $\mathbf{A} = \{A_1, \dots, A_k\}$

function CREATIONRÈGLETÂCHE(CPL)
  $R_m \leftarrow \emptyset,\ k \leftarrow 0$
  $\boldsymbol{\alpha_m^i} \leftarrow \bigcup_{j=1}^{n} \boldsymbol{\alpha_j^i}\ \boldsymbol{\alpha_m^c} \leftarrow \bigcup_{j=1}^{n} \boldsymbol{\alpha_j^c},\ \boldsymbol{\alpha_m^f} \leftarrow \bigcup_{j=1}^{n} \boldsymbol{\alpha_j^f}$
  $\sigma \leftarrow Id$ ▷ Mapping argument clauses - argument règle
  for $a^i \in \boldsymbol{\alpha_m^i}$ **do**
    if instanceDéjàVue(a^i) **then**
      $R_m \leftarrow R_m \wedge isDetected(a^i)[\{a^i\}, \emptyset, \emptyset]$
    else
      $k \leftarrow k + 1,\ \sigma(a^i) \leftarrow A_k$
      $CPL_{a^i} \leftarrow descriptionSemantique(a^i)$
      for $F^i = P^i(\boldsymbol{\alpha}) \in CPL_{a^i}$ **do**
        $R_m \leftarrow R_m \wedge P^i(\sigma(\boldsymbol{\alpha}))$
      end for
    end if
  end for
  for $a^c \in \boldsymbol{\alpha_m^c}$ **do**
    $k \leftarrow k + 1,\ \sigma(a^c) \leftarrow A_k$
    $R_m \leftarrow R_m \wedge isA(A_k, a^c)[\{A_k\}, \{a^c\}, \emptyset]$
  end for
  for $F_j \in CPL$ **do**
    $R_m \leftarrow R_m \wedge P_j(\sigma(\boldsymbol{\alpha_j}))$
  end for
end function

en notant $my\_cup/1$ la règle représentant la description sémantique de l'instance :

$$R_m(A_1, A_2) : \text{-}\ my\_cup(A_2),\ isA(A_1, fork),\ behind(A_1, A_2) \tag{5.8}$$

– L'instance est my_cup est inconnue et l'on obtient alors :

$$R_m(A_1, A_2) : \text{-}\ isA(A_1, fork),\ isA(A_2, cup),\ behind(A_1, A_2) \tag{5.9}$$

Formellement, nous définissons une règle de tâche R_m/k de la façon suivante :

$$R_m(\mathbf{A} = A_1, \dots, A_k)[\boldsymbol{\alpha}_m^i, \boldsymbol{\alpha}_m^c, \boldsymbol{\alpha}_m^f] = \bigwedge_{j=1}^{N} F_j(\sigma(\boldsymbol{\alpha}_j))[\boldsymbol{\alpha}_j^i,\ \boldsymbol{\alpha}_j^c, \boldsymbol{\alpha}_j^f] \tag{5.10}$$

avec

$$\sigma : \bigcup_{j=1}^{N} \boldsymbol{\alpha_j} \rightarrow \bigcup_{j=1}^{N} \boldsymbol{\alpha_j} \cup \mathbf{A}$$

A est l'ensemble des arguments (ici forcément **variables**) de la règle et qui peuvent correspondre à un concept ou une instance. $\boldsymbol{\alpha}_m^c, \boldsymbol{\alpha}_m^i, \boldsymbol{\alpha}_m^f$ correspondent respectivement à l'ensemble des **constantes** concepts, instances et facteurs apparaissant parmi les arguments des clauses F_j. Ces ensembles sont définis par

$$\boldsymbol{\alpha}_m^i = \bigcup_{j=1}^{N} \boldsymbol{\alpha}_j^i \tag{5.11}$$

$$\boldsymbol{\alpha}_m^c = \bigcup_{j=1}^{N} \boldsymbol{\alpha}_j^c \tag{5.12}$$

$$\boldsymbol{\alpha}_m^f = \bigcup_{j=1}^{N} \boldsymbol{\alpha}_j^f \tag{5.13}$$

σ est un mapping qui va envoyer certains arguments de clauses F_j sur des variables de l'ensemble A. Ainsi, en reprenant l'expression (5.9), on a

$$F_3 = behind,\ \boldsymbol{\alpha}_3 = \{fork,\ my\_cup\},\ \sigma(\boldsymbol{\alpha}_3) = \{A_1,\ my\_cup\}$$

Ces règles sont construites selon l'algorithme 9. Tout d'abord, on récupère l'ensemble des arguments des différentes clauses par union [4](ligne 3). La ligne 4 initialise un mapping σ entre ces arguments et les arguments de ces même clauses lorsque qu'elles apparaissent dans la règle (lignes **??**,**??**). Nous traitons ensuite les arguments décrivant une instance. Si celle-ci a déjà été précédemment vue, nous ajoutons une clause *isDetected/1* (ligne 7). Dans le cas contraire, elle devient alors une variable de la règle (ligne 9), ce qui se traduit par un mapping de l'argument correspondant sur une nouvelle variable. De plus, les clauses issues de sa représentation sémantique (si disponible) sont ajoutées à la requête (ligne 10).

Les arguments décrivant une classe génèrent automatiquement une variable associée, qui sera l'instance réelle de cette classe satisfaisant la règle (ligne 17). Enfin, on ajoute à la règle la conjonction des clauses en entrée en remplaçant leur arguments par leur image via le mapping σ (ligne 21).

Un point important est que l'on fait une distinction entre les instances (*"my cup"*) et les objets (*"a cup"*). En effet, la recherche d'une instance particulière et d'un objet d'une certaine classe ne

4. Il n'y a donc pas de doublon

Clause Prolog	Type	Processus enclenché
$isA(A, c)$	Active	Détection de concepts c
$isDetected(i)$	Active	Détection de l'instance i
$prop(A, c)$	Active	Détection de concepts c
$hasA(A, c)$	Active	Détection de concepts c
$behind, \ldots, to\_the\_left\_of(A, B)$	Passive	

TABLE 5.6 – Type des clauses Prolog

font pas appel aux même méthodes. La suite de cette section va détailler la résolution de ces tâches.

5.5.2 Résolution de la tâche

Une tâche va se décomposer en un ensemble de sous-tâches élémentaires. Chacune correspond aux clauses Prolog apparaissant dans la règle R_m/k définissant la tâche. Nous classons ici les clauses Prolog selon deux types :

- *Clauses Actives :* Elles enclenchent la mise en route de processus afin de rechercher activement les solutions les vérifiant.
- *Clauses Passives :* Elles ne font que vérifier si les arguments donnés sont solution ou non.

La table 5.6 donne la classification des clauses Prolog que nous employons. La détection de concepts et d'instances se font à partir d'un FCN : dans le premier cas avec une couche appliquant le classifieur Random Forest du concept et dans le second cas avec une couche calculant la similarité cosinus avec un précédent descripteur connu de l'instance (section 4.2.1).

Dans notre exemple de la requête *Bring me the fork behind my cup* définie par (5.7), on va avoir deux processus enclenchés : un pour la détection d'objets de type *fork* via la clause

$$isA(A_1, fork)$$

et un pour la reconnaissance de l'instance *my_cup* via

$$isDetected(my\_cup)$$

Le schéma global du déroulement d'une tâche est visible à la figure 5.13. On y considère une tâche définie selon (5.10). A chaque concept $a^c \in \boldsymbol{\alpha}_m^c$ et instance $a^i \in \boldsymbol{\alpha}_m^i$ va être associé un thread de détection. Ces threads sont dotés d'une pile locale indépendante de pointeurs vers les instances du modèle d'instance. Elles sont initialisées par le modèle d'instance courant. Quand une nouvelle

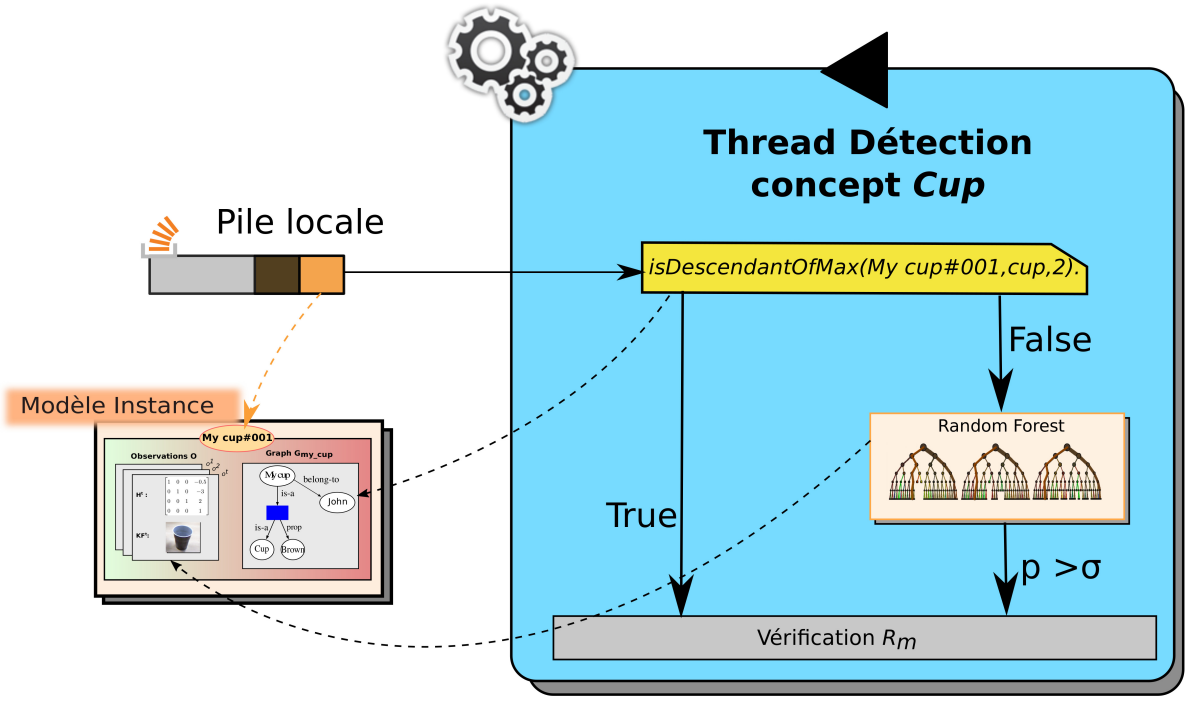


FIGURE 5.11 – Schéma de fonctionnement d'un thread de détection de concept.

instance est ajoutée (ou que les informations d'une instance sont mises à jour) elle est ajoutée aux piles des différents threads en cours d'exécution. Le travail des threads est simple : il est de vérifier si l'instance en haut de pile s^i correspond au concept ou à l'instance recherchée. Les schémas des figures 5.11 et 5.12 décrivent ces processus. Dans les deux cas, nous commençons par la description sémantique :

– Dans le cas d'un concept a^c : nous vérifions si l'instance considérée s^i hérite de a^c via une règle Prolog $isDescendantOf/3$. Cette clause vérifie si une variable A est descendante d'une autre variable B à un degré N :

$$isDescendantOf(A, B, N) \text{ : - } isDescendantAcc(A, B, 0, N). \quad (5.14)$$

$$isDescendantAcc(A, B, T, N) \text{ : - } notFactor(B),\ \textit{Tnew is T+1}, isA(C, B), isDescendantAcc(A, C, Tnew, N). \quad (5.15)$$

$$isDescendantAcc(A, B, T, N) \text{ : - } isA(A, B), \textit{N is T+1}. \quad (5.16)$$

La règle $notFactor/1$ vérifie simplement que l'argument ne correspond pas à un nœud facteur. En effet, ils doivent être ignorés lorsque l'on considère la distance entre deux nœuds. En pratique, nous considérons une variante prenant en compte les descendances jusqu'à un

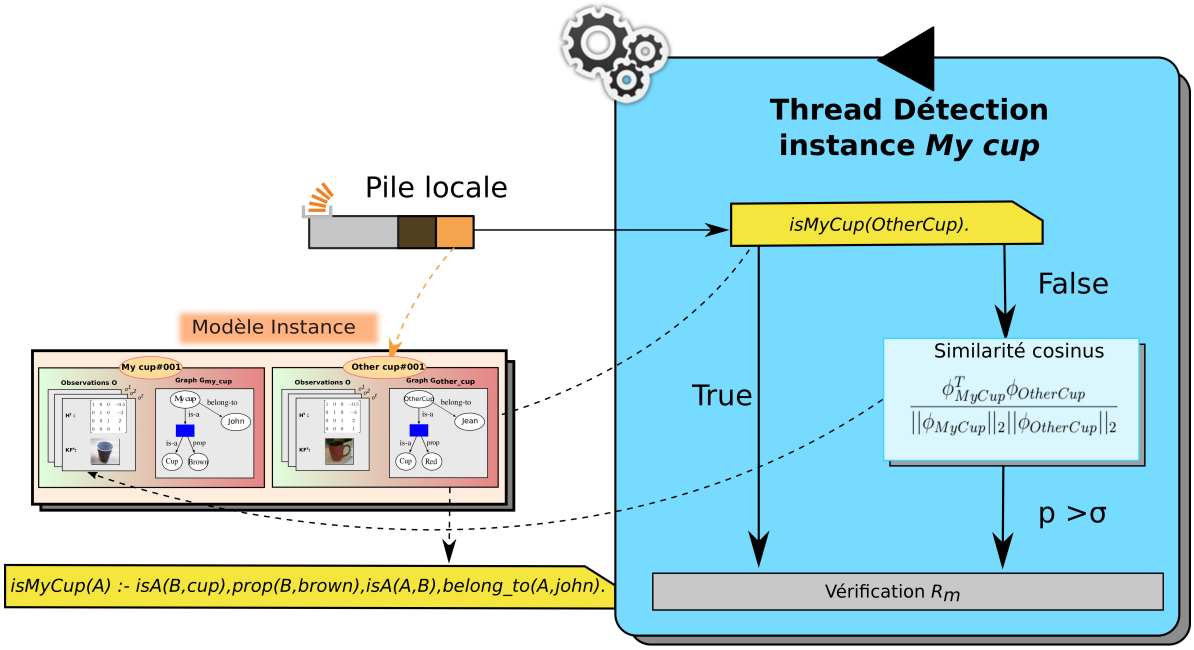


FIGURE 5.12 – Schéma de fonctionnement d'un thread de détection d'instance.

degré max $Nmax$:

$$isDescendantOfMax(A, B, Nmax) : - isDescendantAcc(A, B, 0, N),\ N =< Nmax. \tag{5.17}$$

Finalement, la vérification est donnée par $isDescendantOfMax(s^i, a^c, Nmax)$. Nous utilisons $Nmax = 2, 3$ mais l'estimation d'un seuil adaptatif optimal est à envisager.

– Dans le cas d'une instance a^i : nous vérifions si les deux descriptions sémantiques sont identiques.

Si l'instance ne correspond pas à l'élément recherché, nous passons ensuite aux descripteurs visuels :

– Classification binaire par Random Forest associé à a^c pour les concepts. Nous utilisons actuellement un seuillage fixe (on note $\sigma \in [0, 1]$ la valeur de ce seuil)

– Similarité cosinus avec un descripteur référence de a^i pour les instances

En parallèle de cela, le modèle de perception va également rechercher activement le concept a^c (resp. instance a^i) via l'utilisation de notre version modifiée de FCN. Les instances détectées de cette façon seront ajoutées au modèle d'instance de la même manière que lors de la segmentation de scène passive. Elles seront donc ensuite naturellement poussées dans les piles locales des différents thread de détection.

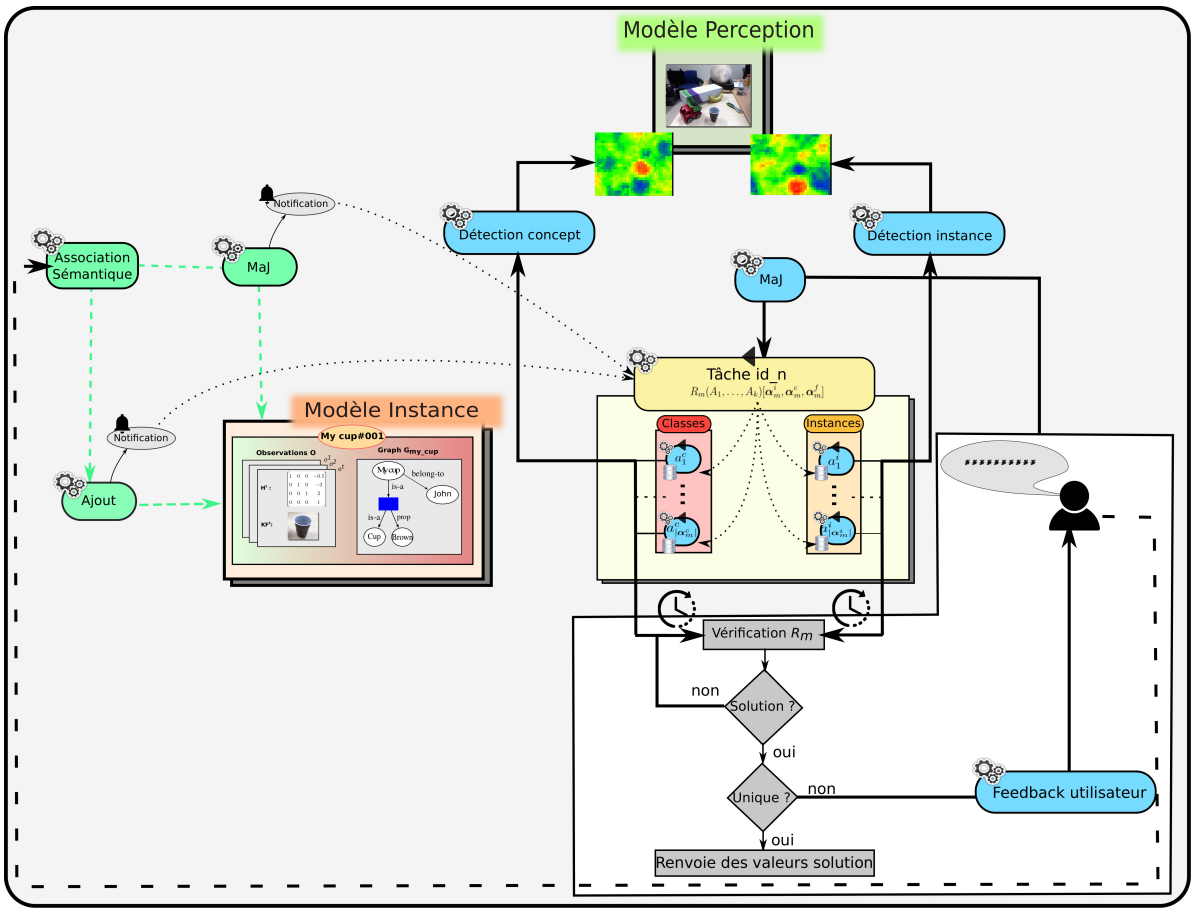


FIGURE 5.13 – Schéma du déroulement interne d'une tâche

A chaque fois qu'un thread détecte un élément recherché, nous vérifions la règle de la tâche R_m/k. Actuellement, le cas où plusieurs (resp. aucune) solutions n'est pas géré. Pour cela, des informations complémentaires doivent être demandées à l'utilisateur. L'ajout de ce processus de feedback est parmi nos perspectives.

5.6 Conclusion et perspectives

Ce chapitre traite des interactions entre les différentes unités de notre modèle. Plus particulièrement, il se concentre sur deux problématiques principales :

– La remontée du modèle d'instance au modèle de connaissance sémantique. Autrement dit, étant donné un descripteur image d'une instance inconnue, nous recherchons à quels concepts de notre ontologie nous pouvons la rattacher. Ce problème peut être encore séparé en trois parties :

- Le cas d'une instance héritant d'un concept connu et pour lequel un classifieur binaire est disponible. Pour un faible nombre de concepts, on peut se permettre d'appliquer l'ensemble des classifieurs disponibles. En pratique, le nombre de concepts sera important et il faut une méthode permettant de trouver le concept père efficacement.
- Le cas d'une instance héritant d'un concept inconnu. Il faut donc trouver le concept le plus proche dont hérite cette instance.
- Les deux premiers cas supposent que l'on sache par avance si l'instance hérite d'un concept connu ou inconnu. En pratique, ce n'est pas le cas.

Pour les deux premiers cas, nous proposons une méthode originale OCT qui emploie l'ontologie comme un arbre de classification où chaque noeud (non feuille) a pour fonction de décision un SVM entraîné sur ses nœuds fils. Les résultats obtenus sur un jeu de test extrait d'ImageNet montrent que notre méthode permet de résoudre ces deux problématiques, au prix d'une baisse modérée de la précision de classification.

Les pistes pour de futurs travaux seraient tout d'abord d'effectuer les même expériences sur un graphe ontologique plus important. Il serait aussi intéressant de remplacer les SVM par d'autres classifieurs. Nous avons tenté de les remplacer par des classifieurs Random Forest en testant sur un certain nombre d'hyperparamètres mais les résultats étaient loin de ceux obtenus par SVM.

– Les interactions entre la sémantique et les instances au travers des interactions utilisateurs. Nous montrons comment l'utilisateur peut communiquer des informations au robot mais également lui faire des requêtes. Leur représentations internes, que l'on dénomme par le terme "tâche", sont obtenues par analyse syntaxique puis une conversion en un ensemble de clauses Prolog formant une règle définissant la tâche.

  La suite de ces travaux sera d'implémenter une méthode permettant le feedback vers l'utilisateur lorsque qu'une tâche possède plusieurs solutions. L'objectif principal est donc de déterminer quelle(s) information(s) (actuellement manquantes) permettrai(en)t de séparer au mieux les différentes solutions.

Chapitre 6

Vers une validation expérimentale

Ce chapitre est consacré à l'application pratique de notre modèle. Différents cas d'usage mettant en avant le fonctionnement et l'intérêt de notre système seront présentés. Tout d'abord, nous décrirons globalement notre architecture logicielle ainsi que la configuration utilisée. Trois cas sont ensuite proposés :

- Un cas d'usage basique basé sur la requête utilisateur *"Bring me the cup"*.
- Un cas introduisant plusieurs instances similaires avec une information donnée par l'utilisateur permettant de lever l'ambiguïté sur la solution.
- Un cas montrant l'utilisation de l'ontologie et l'intérêt de la notion de contexte, introduite à la section 3.5.

Pour chaque cas, nous décrirons chaque étape, allant de l'analyse sémantique à la résolution des requêtes (tâches).

6.1 Description de l'architecture logicielle

Le cœur du système a été implémenté en C++11. Prolog est intégré à partir de l'interface C/C++ proposée par l'environnement SWI-Prolog 7.7.12 [WSTL12]. La partie concernant les réseaux de neurones se base sur le framework Caffe [JSD+14]. Notre implémentation utilise également OpenCV3.1 [PBKE12] dont est issue l'implémentation de Random Forest employée ici. Les images utilisées pour l'entraînement du RF associé à un concept proviennent, si disponibles, de la base de données ImageNet [DDS+09]. Celle-ci est organisée selon l'ontologie WordNet et chaque ensemble d'images est lié au synset du concept correspondant. L'accès à la base de données WordNet se fait à partir de son API en C. Les classifieurs sont généralement entraînés hors ligne mais on

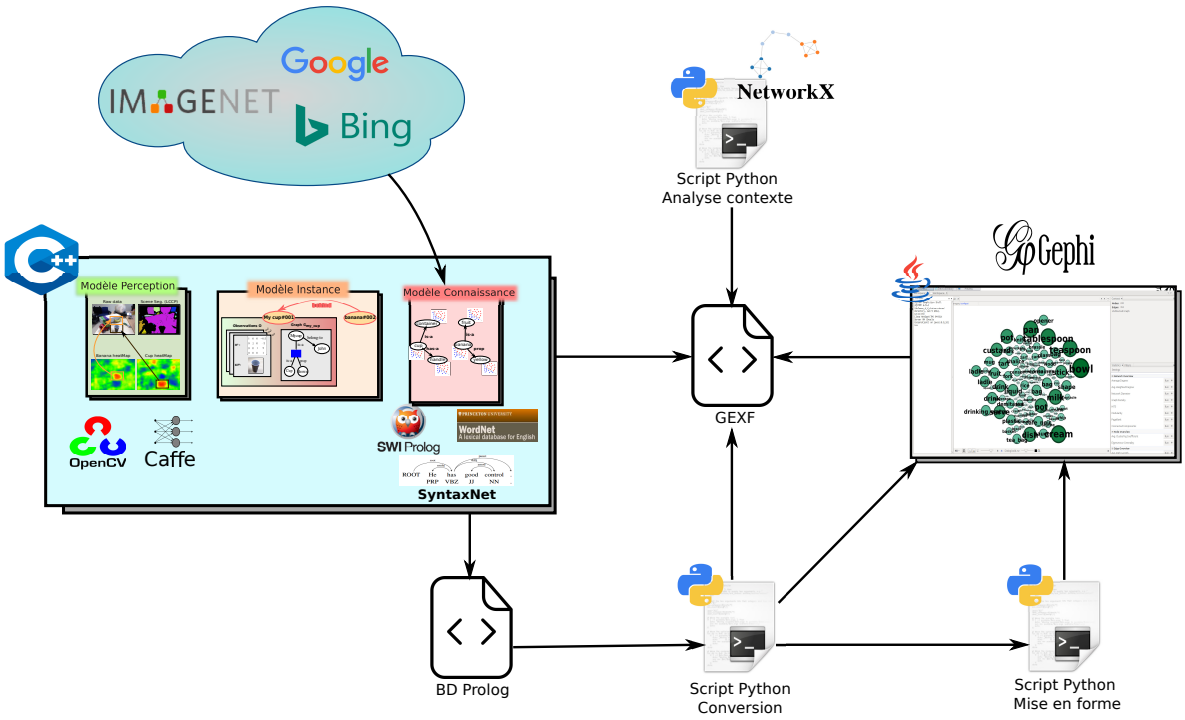


FIGURE 6.1 – Aperçu général des différents élements constituant l'implémentation du système

peut également entraîner un concept en ligne. En effet, l'API d'ImageNet permet le téléchargement des images par url. Si un concept ne possède pas ou peu d'images sur ImageNet, il est possible d'employer d'autres sources telles que les moteurs de recherche.

L'ontologie ainsi que le modèle d'instance sont représentés par une base de données Prolog (ensemble de clauses). Les graphes sont affichés avec le logiciel Gephi 0.9.1 [BHJ09] après avoir converti, si nécessaire, les bases de données Prolog au format gexf [HBJ+07]. A noter également qu'il est possible d'y exécuter des scripts python afin d'automatiser la mise en forme des graphes. Les sous-graphes de contexte et les mesures de pertinences associées (cf section 3.5) sont obtenues à partir d'un script python en utilisant la bibliothèque de traitement de graphes NetworkX [HSSC08].

L'interaction avec l'utilisateur se fait actuellement à partir d'une simple interface graphique. Il est envisagé d'utiliser des API de transcription audio (Google Cloud Speech-to-Text, Microsoft Bing speech) afin de pouvoir directement communiquer avec le système.

La figure 6.1 donne un aperçu général des différents éléments employés dans notre implémentation. Dans l'état actuel du système, les graphes ne sont pas affichés en temps réel mais hors ligne. L'intégration de Gephi (implémenté en Java) ou d'une autre bibliothèque graphique dédiée aux graphes est en cours.

Comme décrit dans le chapitre précédent, l'implémentation repose sur une architecture mul-

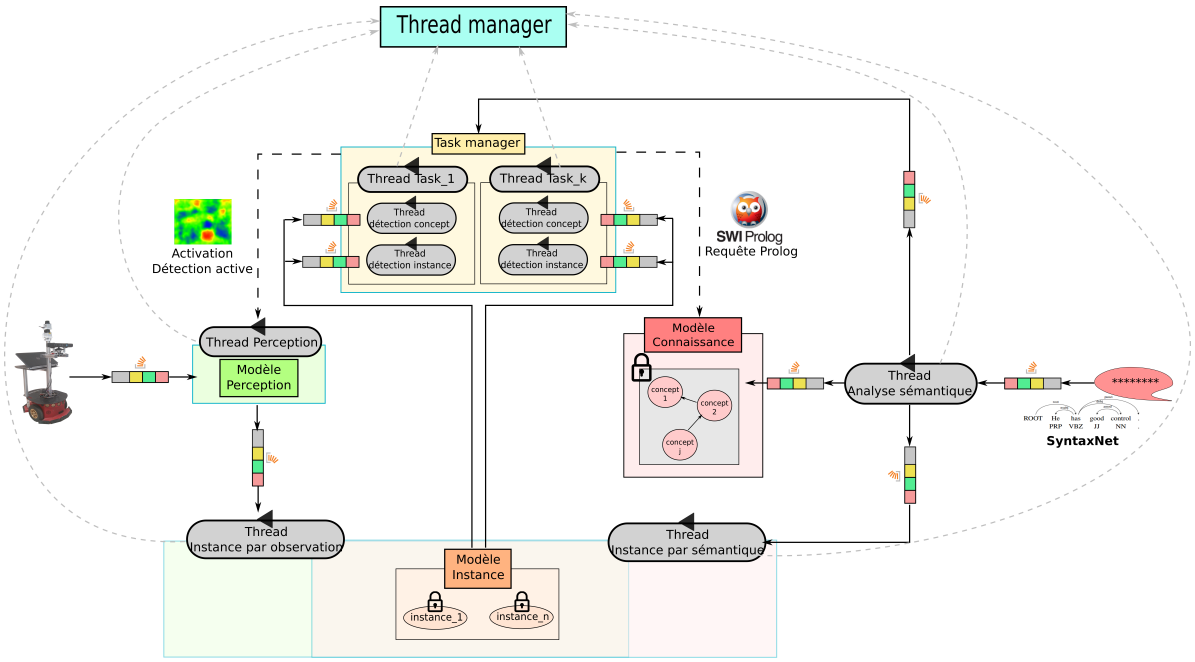


FIGURE 6.2 – Aperçu général de l'approche multithreads de notre architecture

tithread afin de pouvoir gérer parallèlement les données capteurs, les entrées textuelles ainsi que la bonne mise à jour des différents modèles (instance, connaissance) et des tâches. En effet, les instances sont des ressources partagées qui sont à la fois mises à jour au fil des détections et des informations sémantiques reçues, tout en étant analysées par les tâches en cours. Cette architecture se base sur le standard C++11, et plus particulièrement sur les implémentations proposées par [Wil12]. La figure 6.2 illustre notre architecture.

6.2 Dispositif expérimental

Les trois cas d'utilisation du modèle proposé dans ces travaux se basent sur le même dispositif expérimental. On considère une scène composée d'une table sur laquelle sont posés un certain nombre d'objets. Celle-ci est observée par un capteur RGBD Kinect. Comme discuté à la section 4.2, la méthode générale et passive de détection d'instances dans une scène se base sur l'algorithme LCCP. En pratique, on obtient très souvent une segmentation qui ne correspond pas à des instances bien définies. C'est pourquoi, afin de montrer le fonctionnement de notre modèle indépendamment des algorithmes externes employés, nous remplaçons cette méthode par un algorithme plus simple et plus robuste de détection. Il consiste à :

– Estimer le plan de la table à partir du nuage de point 3D de la Kinect. Les quatres coins

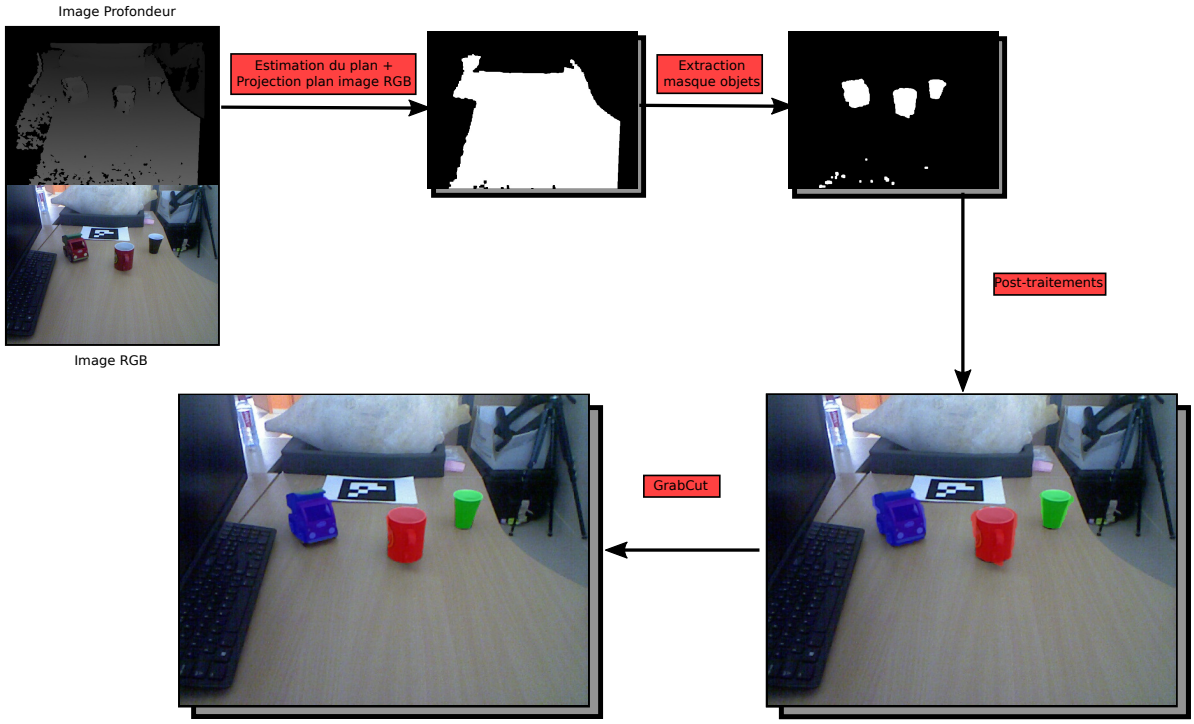


FIGURE 6.3 – Schéma de la détection d'instance passive employée dans ce chapitre

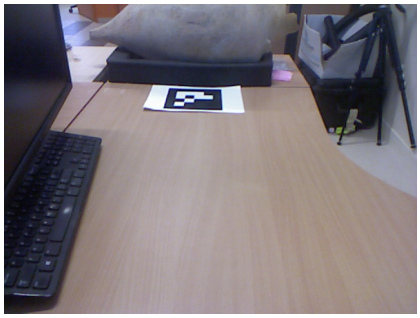
d'un QR code permettent d'obtenir une première estimation grossière du plan qui est ensuite recalculée par moindres carrées selon la méthode RANSAC [FB81].

- Extraire le masque 2D correspondant aux objets se situant au-dessus de la table.
- Segmenter le masque précédent en objets et appliquer quelques post-traitements (érosion, dilatation, seuillage sur la taille).
- Raffinement des masques 2D des objets par l'algorithme GrabCut [RKB04].

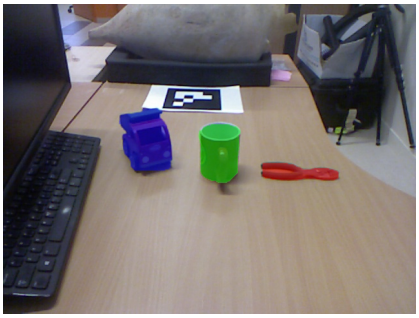
La plateforme utilisée est dotée d'un CPU Intel Core i7-2600@3.40Ghz x 8, de 8 Gb de RAM ainsi que d'un GPU GeForce GTX 970, sous Ubuntu 15.04.

6.3 Cas d'une requête simple

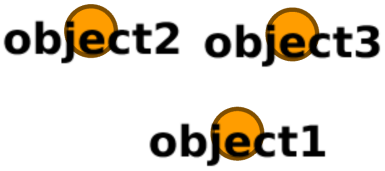
La scène est initialement vide, c'est à dire sans aucun objet sur la table (figure 6.4a). Le modèle d'instance est également vide et le modèle de connaissance initialisé selon la section 3.3. On introduit par la suite trois objets : une tasse, un jouet et un outil. Le modèle de perception, via l'algorithme décrit précédemment, détecte la présence de ces trois instances et les ajoute au modèle d'instance (figures 6.4b et 6.4c).



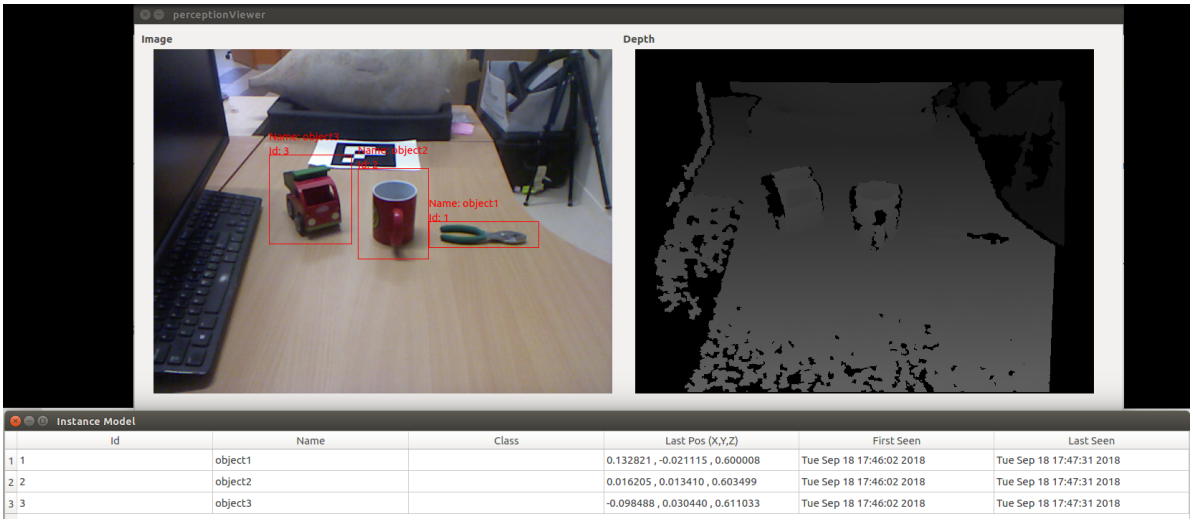
(a) Scène initiale



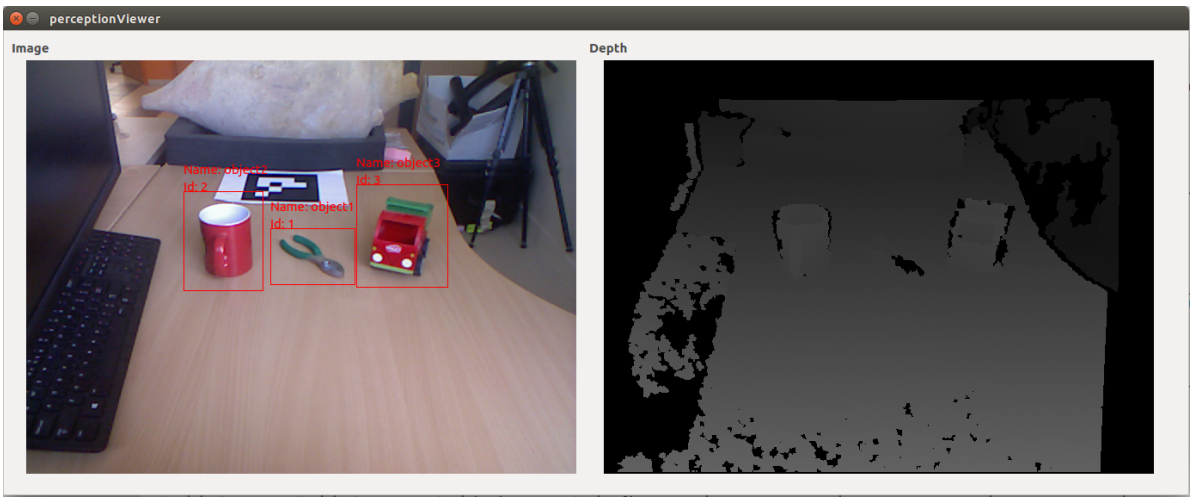
(b) Détection des objets



(c) Graphe du modèle d'instance après détection des instances



(d) Etat du système après détection des instances



(e) Détection des objets après déplacement par l'utilisateur

FIGURE 6.4 – Description du cas d'usage basique avant la requête utilisateur

Il est à noter que ces instances sont "suivies" de manière passive d'image en image. Comme exposé dans la section 5.4, chaque nouvelle détection du modèle de perception est transmise au modèle d'instance afin de vérifier une possible correspondance avec une instance déjà connue. Ceci est illustré à la figure 6.4e après que les trois objets de la scène aient été déplacés. L'utilisateur ("Yohan" ici) va ensuite faire la requête suivante : *"Bring me the cup"*. Cette entrée sémantique est tout d'abord analysée par SyntaxNet qui renvoie la structure de la phrase au format **conLL-X** (figure 6.5).

```
1       Bring   _       VERB    VB      _       0       ROOT    _       _
2       me      _       PRON    PRP     _       1       iobj    _       _
3       the     _       DET     DT      _       4       det     _       _
4       cup     _       NOUN    NN      _       1       dobj    _       _
5       .       _       .       .       _       1       punct   _       _
```

FIGURE 6.5 – Analyse syntaxique de la requête *"Bring me the cup"* obtenue par SyntaxNet

On détermine le type d'entrée sémantique (connaissance générale, information spécifique sur une instance ou requête) comme décrit par l'algorithme 4 (section 5.2.1). Ici, *"Bring"* étant un verbe à l'impératif, l'entrée est donc classifiée comme une requête. Suivant l'algorithme 7 (section 5.5), la requête est décomposée en groupes nominaux et en groupes de relations. On a ici :

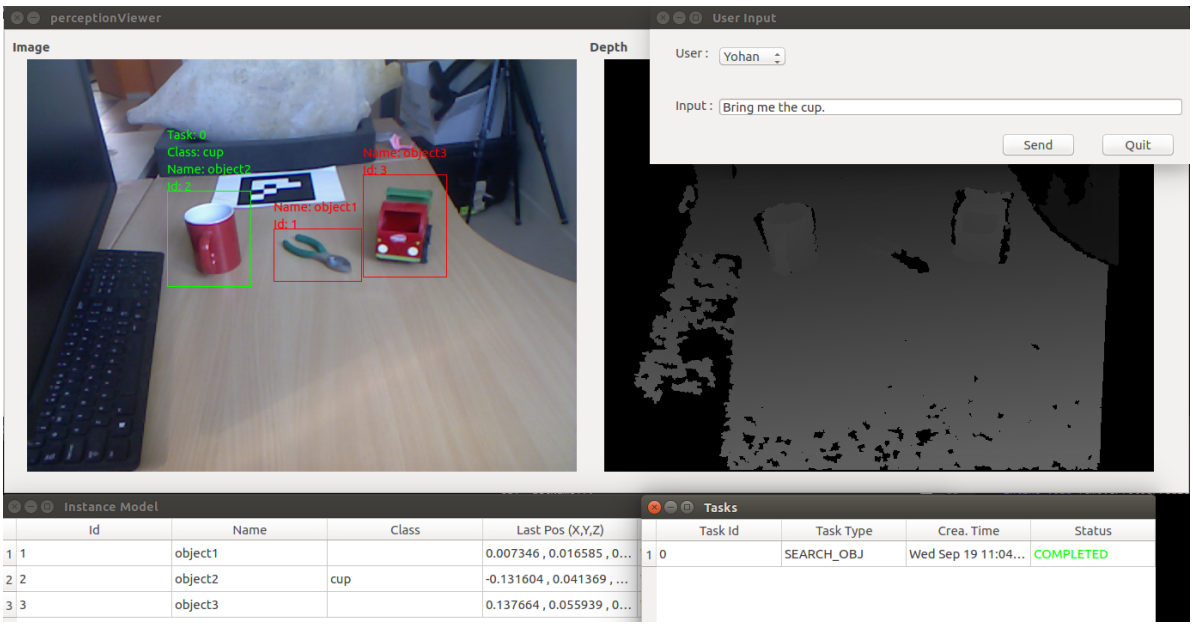
$$\begin{aligned} GN &= \{gn_0 = \{the, cup, \emptyset\}\} \\ GR &= \emptyset \\ GRU &= \emptyset \end{aligned}$$

La règle définissant la tâche générée à partir de la requête est donnée par l'algorithme 9 (section 5.5.1) avec ici simplement :

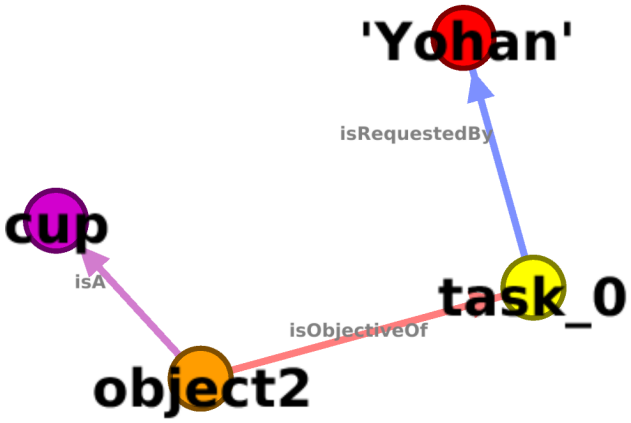
$$R_m(A_1) : \text{- } isA(A_1, cup) \text{ avec } \sigma(the\_cup) = A_1 \tag{6.1}$$

Un thread de détection d'objet de type *cup* est démarré avec le classifieur correspondant chargé. A défaut, s'il n'a pas encore été entraîné, des images d'entraînement sont automatiquement téléchargées à partir de la base ImageNet et/ou Google Image/Bing. Un autre thread est générée pour l'entraînement de ce classifieur en tâche de fond. Le thread de la tâche est mis en attente jusqu'à ce que le classifieur soit disponible.

Comme initialement aucune instance n'est connue comme étant de type *cup*, il teste chaque instance et classifie l'instance *object2* en tant que *cup*. Ainsi, *object2* est solution de la règle définie



(a) Détection de l'objet de la requête



(b) Modèle d'instance mis à jour

FIGURE 6.6 – Etat final du système après résolution de la requête

par (6.1) et la tâche est considérée comme accomplie. Cela est illustré à la figure 6.6 avec également le modèle d'instance mis à jour suite à la résolution de cette requête. On peut donc constater que le modèle permet d'obtenir le résultat attendu de façon interactive.

6.4 Cas multi-instances

Ce cas propose une application un peu plus complexe que précédemment. On se base ici sur une scène initialement vide. L'utilisateur informe le système sur la position de sa tasse, qui n'est

pas visible dans la scène et qui lui est inconnue : *"My cup is to the left of the car"*. La figure 6.7 illustre le résultat de l'analyse syntaxique avant et après correction (cf chapitre 3).

1	my	_	PRON	PRP$	_	2	poss	_	_
2	cup	_	NOUN	NN	_	3	nsubj	_	_
3	is	_	VERB	VBZ	_	0	ROOT	_	_
4	to	_	ADP	IN	_	3	prep	_	_
5	the	_	DET	DT	_	6	det	_	_
6	left	_	NOUN	NN	_	4	pobj	_	_
7	of	_	ADP	IN	_	6	prep	_	_
8	the	_	DET	DT	_	9	det	_	_
9	car	_	NOUN	NN	_	7	pobj	_	_
10	.	_	.	.	_	3	punct	_	_

(a) Analyse obtenue par SyntaxNet

1	my	_	PRON	PRP$	_	2	poss	_	_
2	cup	_	NOUN	NN	_	3	nsubj	_	_
3	is	_	VERB	VBZ	_	0	ROOT	_	_
4	to_the_left_of	_	ADP	IN	_	2	prep	_	_
5	the	_	DET	DT	_	6	det	_	_
6	car	_	NOUN	NN	_	4	pobj	_	_
7	.	_	.	.	_	3	punct	_	_

(b) Analyse après correction

FIGURE 6.7 – Analyse syntaxique de la phrase *"My cup is to the left of the car"*

L'analyse a la décomposition suivante :

$$\begin{aligned} GN &= \{gn_0 = \{my, cup, \emptyset\},\ gn_1 = \{the, car, \emptyset\}\} \\ GR &= \{gn_0,\ to\_the\_left\_of,\ gn_1,\ \emptyset\} \\ GRU &= \emptyset \end{aligned}$$

Cette instance est sémantiquement définie par la règle Prolog :

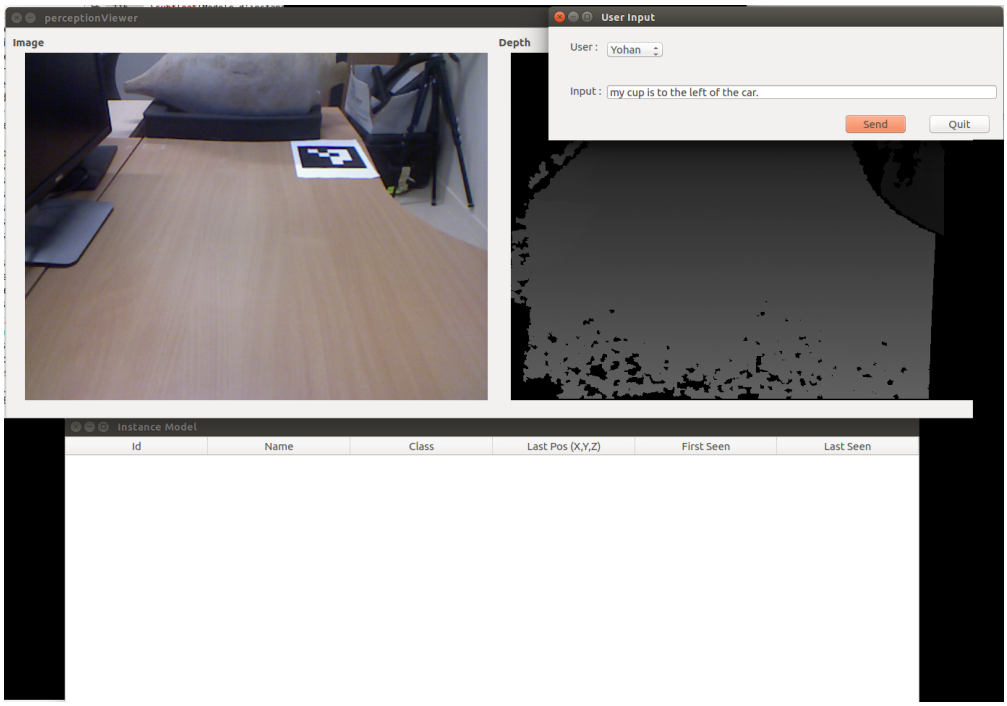
$$my\_cup(A) : \text{-}\ isA(A, cup),\ belongTo(A, 'Yohan'),\ to_{t}he_{l}eft_{o}f(A, B),\ isA(B, car). \quad (6.2)$$

En pratique, la relation $belongTo$ n'est pas directement utilisée pour définir l'instance. En effet, cette relation ne peut être inféré visuellement : une heuristique définie en plus une règle $belongToUser$ (ici $belongToYohan$) :

$$belongToYohan(A) : \text{-}\ my\_cup(A).$$

La figure 6.8 montre l'état du système après cette étape, ainsi que les règles générées. On ajoute

ensuite plusieurs objets à la scène comme on peut le voir à la figure 6.9.

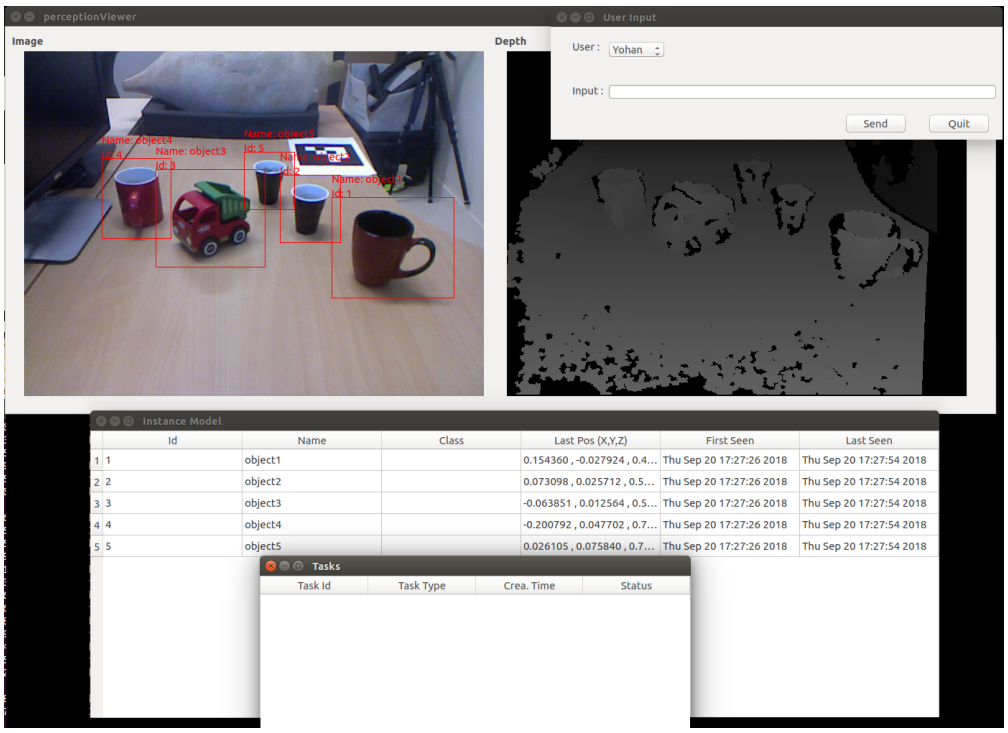


(a) Aperçu du système

```
1 my_cup(A) :- isA(B,car),isA(A,cup),to_the_left_of(A,B).
2 belongToYohan(A) :- my_cup(A).
```

(b) Règles générées

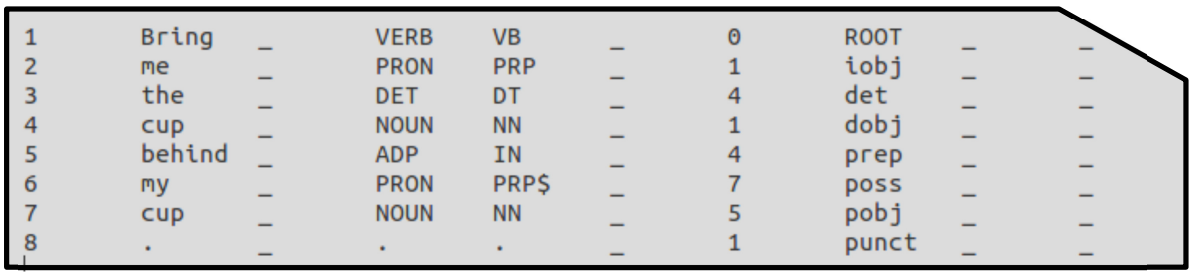
FIGURE 6.8 – Etat du système après que l'utilisateur ait donné une information concernant l'instance *my_cup*



(a) Aperçu du système

(b) Masques de détection

FIGURE 6.9 – Système après l'introduction d'objets dans la scène



1	Bring	_	VERB	VB	_	0	ROOT	_	_
2	me	_	PRON	PRP	_	1	iobj	_	_
3	the	_	DET	DT	_	4	det	_	_
4	cup	_	NOUN	NN	_	1	dobj	_	_
5	behind	_	ADP	IN	_	4	prep	_	_
6	my	_	PRON	PRP$	_	7	poss	_	_
7	cup	_	NOUN	NN	_	5	pobj	_	_
8	.	_	.	.	_	1	punct	_	_

FIGURE 6.10 – Analyse syntaxique de l'information *"Bring me the cup behind my cup."*

L'utilisateur fait ensuite la requête suivante : *"Bring me the cup behind my cup"* avec l'analyse syntaxique correspondante à la figure 6.10. Sa décomposition est

$$\begin{aligned} GN &= \{gn_0 = \{the, cup, \emptyset\},\ gn_1 = \{my, cup, \emptyset\}\} \\ GR &= \{gn_0, behind, gn_1, \emptyset\} \\ GRU &= \emptyset \end{aligned}$$

Toujours selon l'algorithme 9 (section 5.5.1), on obtient la règle correspondant à la tâche :

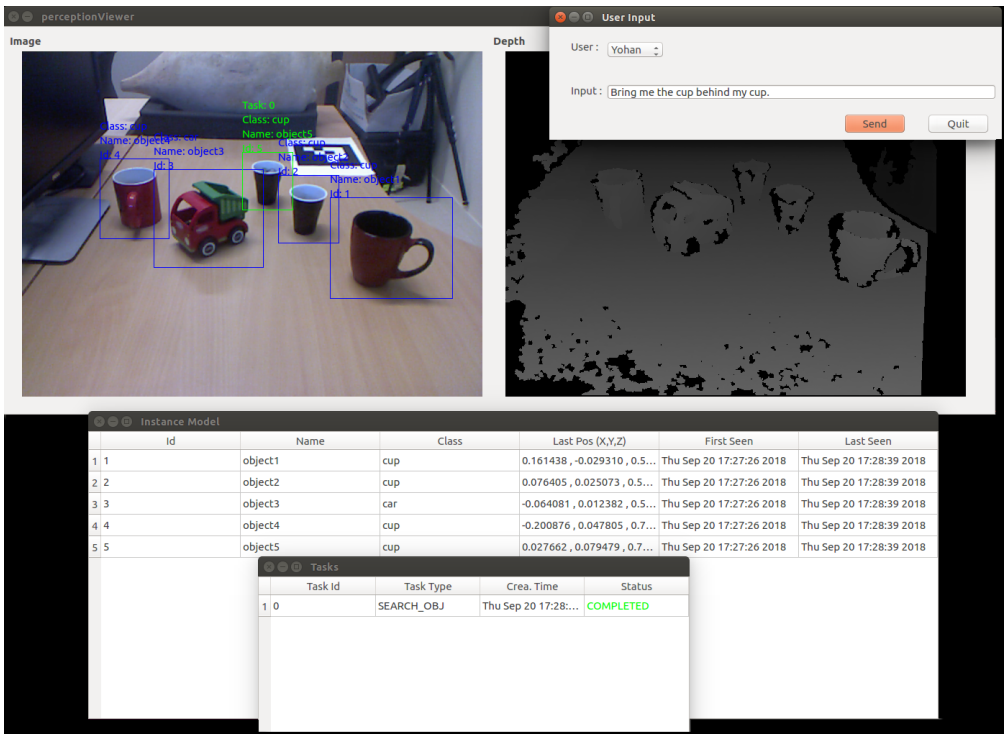
$$R_m(A) : \text{-}\ isA(A, cup), my\_cup(B), behind(A, B). \tag{6.3}$$

Deux threads de détection de classes sont créés. Le premier va rechercher des objets de type *cup* requis à la fois pour déterminer l'instance *my_cup* et l'instance recherchée. Le second va lui chercher un objets de type *car* nécessaire dans la règle *my_cup/1*. La règle de la tâche (6.3) ainsi que les clauses de relations spatiales sont stockées dans un ficher temporaire visible à la figure 6.11.

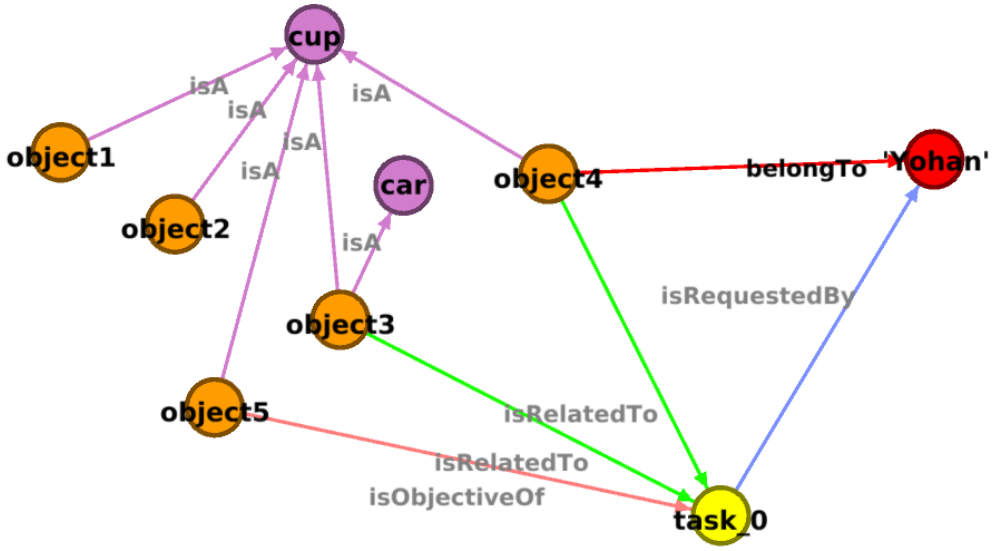
```
constraintPredicate(C,A) :- isA(A,cup),behind(A,C),my_cup(C).
behind(object1,object2).
to_the_left_of(object2,object1).
behind(object3,object1).
to_the_left_of(object3,object1).
behind(object4,object1).
to_the_left_of(object4,object1).
behind(object5,object1).
to_the_left_of(object5,object1).
behind(object3,object2).
to_the_left_of(object3,object2).
behind(object4,object2).
to_the_left_of(object4,object2).
behind(object5,object2).
to_the_left_of(object5,object2).
behind(object4,object3).
to_the_left_of(object4,object3).
behind(object5,object3).
to_the_left_of(object3,object5).
behind(object5,object4).
to_the_left_of(object4,object5).
```

FIGURE 6.11 – Fichier temporaire pour la tâche en cours. Il contient la règle R_m (ici $constraintPredicate$) ainsi que les clauses de relations spatiales.

Une fois la règle résolue, l'instance recherchée est déterminée comme étant *object4* (figure 6.12a). Le modèle d'instance mis à jour est visible à la figure 6.12b. La possibilité d'introduire des relations nécessaires à l'accomplissement et à la définition de la mission illustre bien la nécessité d'un modèle d'instance, qui est plus "local" que le modèle sémantique. Les interactions robots/humain sont donc naturellement associées à ce modèle comme on le voit dans le cas de cette désambiguïsation (figure 6.12a).



(a) Détection de l'instance recherchée



(b) Modèle d'instance après résolution de la requête

FIGURE 6.12 – Etat du modèle après résolution de la requête

6.5 Usage de l'ontologie et de la notion de contexte

Le dernier cas d'utilisation abordé dans cette section a pour but de mettre en avant une utilisation de l'ontologie ainsi que l'intérêt de la notion de contexte développée à la section 3.5. On

considère une scène visible à la figure 6.15. On remarque que la cuillère n'a pas été détectée par l'algorithme décrit à la figure 6.3 car trop proche du capteur Kinect. Jusqu'à présent, les cas présentés se limitaient à la détection passive d'instance (segmentation non supervisée). On va illustrer ici l'utilisation de FCN pour la détection active d'instance.

La requête utilisateur est ici *"Give me something for serving coffee"*. Contrairement aux deux cas précédents, l'objet recherché ici n'est pas directement précisé : l'ontologie du modèle de connaissance va nous donner l'ensemble des classes pouvant satisfaire la requête.

L'analyse syntaxique de la requête donne le résultat à la figure 6.13.

1	give	_	VERB	VB	_	0	ROOT	_	_	
2	me	_	PRON	PRP	_	1	iobj	_	_	
3	something		_	NOUN	NN	_	1	dobj	_	_
4	for	_	ADP	IN	_	3	prep	_	_	
5	serving	_	VERB	VBG	_	4	pcomp	_	_	
6	coffee	_	NOUN	NN	_	5	dobj	_	_	
7	.	_	.	.	_	1	punct	_	_	

FIGURE 6.13 – Analyse syntaxique de la requête *"Give me something for serving coffee"*

La décomposition en GN/GR/GRU est la suivante :

$$\begin{aligned} GN &= \{gn_0 = \{\emptyset, something, \emptyset\},\ gn_1 = \{\emptyset, coffee, \emptyset\}\} \\ GR &= \emptyset \\ GRU &= \{gn_0, serve, \{coffee\}\} \end{aligned}$$

Comme expliqué à la section 5.5.1, la classe recherchée est décrite par une règle Prolog :

$$classDef(A) : \text{-}\ useFor(A, F), on(F, coffee), isA(F, serve) \tag{6.4}$$

avec F ici une variable correspondant à un noeud facteur. La règle définissant la tâche générée à partir de la requête utilisateur est donc

$$R_m(A_1) : \text{-}\ classDef(B), isA(A_1, B)$$

Le système commence par rechercher l'ensemble des concepts (synsetId) pouvant correspondre aux mots *coffee* et *serve* dans l'ontologie, soit :

$coffee$-$n07929519$, $serve$-$v01181295$, $serve$-$v01180351$, $serve$-$v01428011$, $serve$-$v01438681$

Pour chaque pair de concepts, on requête l'ontologie avec la règle définie par (6.4) (figure 6.14)

```
?- useFor(A,B),actOn(B,coffee-n07929519),isA(B,serve-v01181295).
A = demitasse-n03174731,
B = -1303 ;
A = coffee_mug-n03063599,
B = -1012 ;
false.

?- useFor(A,B),actOn(B,coffee-n07929519),isA(B,serve-v01180351).
false.

?- useFor(A,B),actOn(B,coffee-n07929519),isA(B,serve-v01428011).
false.

?- useFor(A,B),actOn(B,coffee-n07929519),isA(B,serve-v01438681).
false.

?-
```

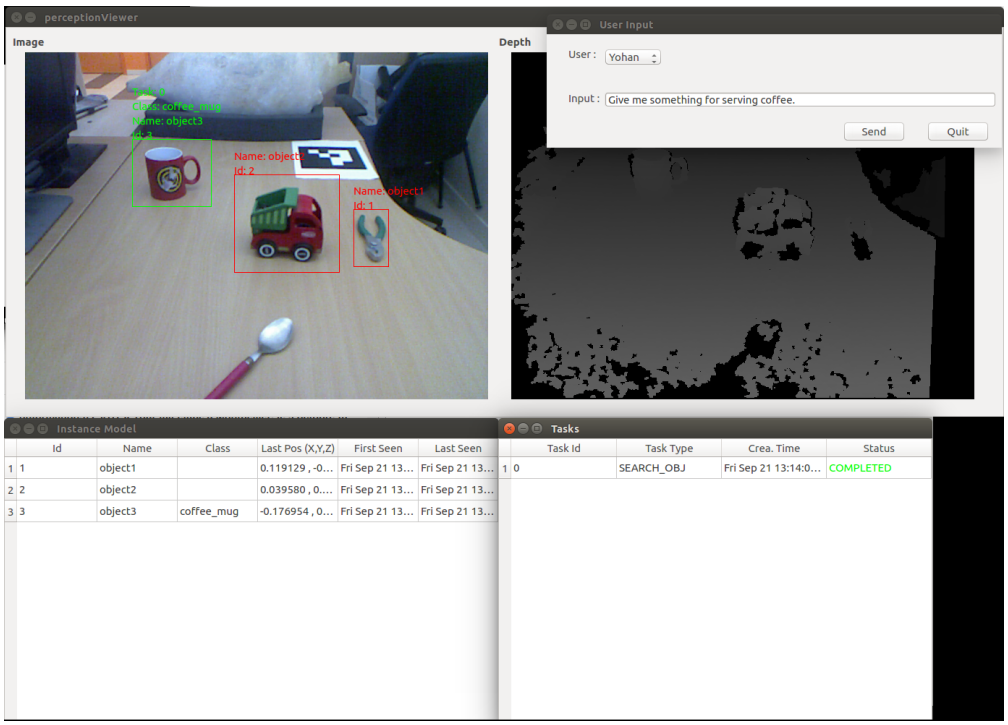
FIGURE 6.14 – Requête sur l'ontologie afin de déterminer les classes correspondants à l'objet recherché. Les concepts -1303 et -1012 correspondent à des nœuds facteurs.

Deux solutions sont trouvées : *demitasse-n03174731* et *coffee_mug-n03063599*. Deux threads de détections pour chacune de ces classes sont créées et l'on obtient la solution à la requête illustrée à la figure 6.15a.

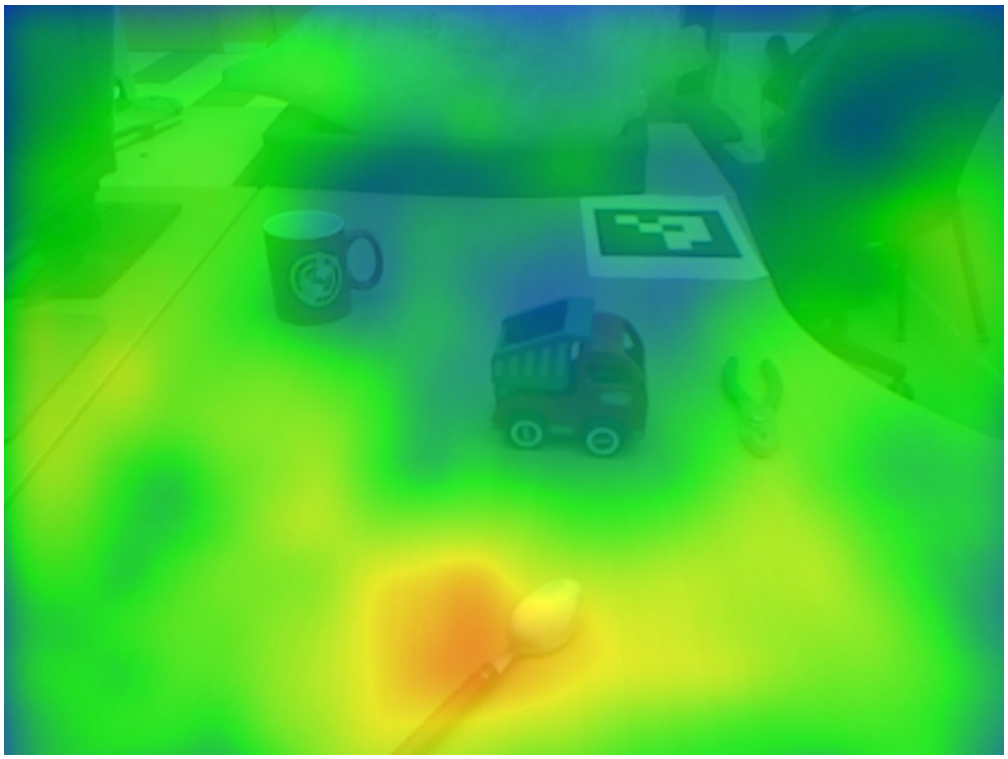
A cet instant, le système sait que la scène est constituée d'une instance de *coffee_mug*. Une stratégie pour détecter de nouvelles instances et/ou classifier les instances déjà connues est d'exploiter le contexte à partir du modèle d'instance. En effet, la présence d'un concept peut indiquer la présence d'autres objets reliés au même contexte. La figure 6.16 présente le sous-graphe ontologique de *coffee_mug* avec les nœuds de taille et couleur proportionnelles à leur pertinence contextuelle, selon la mesure employée (section 3.5).

On a vu à la section 4.2 que notre système a deux façons de détecter des instances : une passive (celle employée principalement dans ce chapitre) et une active à partir de FCNs. Ainsi, on classifie les instances détectées passivement puis l'on recherche de nouvelles instances à partir de FCN Random Forest. On détecte ainsi une instance de type *spoon* (figure 6.15b).

Ce dernier cas d'application illustre l'intérêt d'une approche basée sur notre ontologie. Plutôt que de travailler avec un univers "fermé" (nombre de classes pré-défini), nous avons choisi une approche ouverte qui exclut la possibilité de pouvoir identifier un objet inconnu dans un temps raisonnable. La notion de contexte nous permet alors d'effectuer cette reconnaissance comme une "tâche de fond" en mettant à profit les informations données par l'utilisateur de façon indirecte.

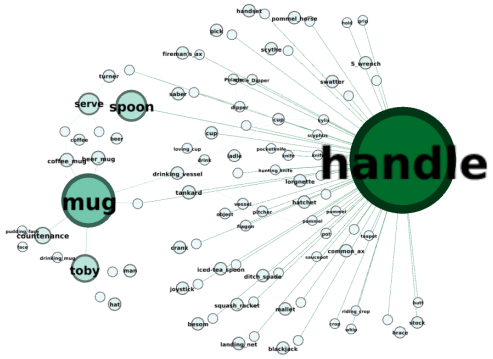


(a) Résolution de la requête

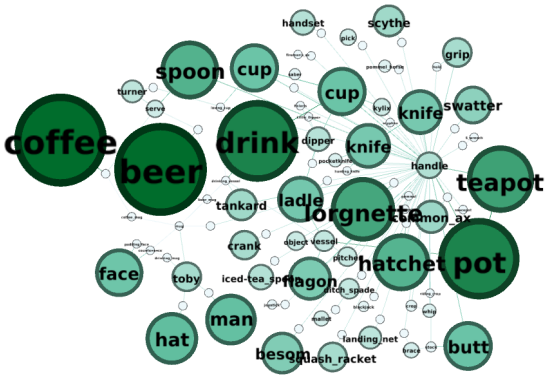


(b) Détection opportuniste d'une instance *spoon* à partir du contexte de *coffee_mug* par un FCN avec une couche Random Forest binaire

FIGURE 6.15 – Résultat de la requête *Give me something for serving coffee.*



(a) Sous-graphe de contexte basé sur la centralité intermédiaire



(b) Sous-graphe de contexte basé sur la similarité GloVe

Id	WnId	Label	POS	bc
1061	n03485997	handle	1	0.176945
3726	n03797390	mug	1	0.080214
5059	n04284002	spoon	1	0.039233
16208	n04443766	toby	1	0.032888
1657	v01181295	serve	2	0.022321
24976	n05601357	countenance	1	0.012591
3725	n02824058	beer_mug	1	0.00708
7817	n03063599	coffee_mug	1	0.00708
10246	n04389430	tankard	1	0.006962
10243	n03241496	drinking_vessel	1	0.005566
8992	n03147901	cup	1	0.00519
3884	n02831894	besom	1	0.005155
4195	n02847461	blackjack	1	0.005155
2427	n03077442	common_ax	1	0.005155
8698	n03127024	crank	1	0.005155
9915	n03214450	ditch_spade	1	0.005155
2429	n03346289	fireman's_ax	1	0.005155
13185	n03488438	handset	1	0.005155
537	n03497657	hat	1	0.005155
2430	n03498962	hatchet	1	0.005155
13853	n03557270	iced-tea_spoon	1	0.005155
8286	n03602883	joystick	1	0.005155
11659	n03639077	landing_net	1	0.005155
15166	n03690168	lorgnette	1	0.005155
8813	n03715892	mallet	1	0.005155
507	n10287213	man	1	0.005155
10556	n03929202	pick	1	0.005155
13656	n03980478	pommel_horse	1	0.005155
16650	n04374521	S_wrench	1	0.005155
11364	n04121511	saber	1	0.005155
10558	n04158250	scythe	1	0.005155
18218	n04292414	squash_racket	1	0.005155
13919	n04369282	swatter	1	0.005155
8360	n04500060	turner	1	0.005011
9799	n03633091	ladle	1	0.004086
3527	n03147509	cup	1	0.003734
4976	n04321804	stock	1	0.003436
3708	n07886849	beer	1	0.002605
7806	n07929519	coffee	1	0.002605

(c) Tableau ordonné selon la pertinence contextuelle (centralité intermédiaire)

Id	WnId	Label	POS	Sim.GloVe
7806	n07929519	coffee	1	0.367353
3708	n07886849	beer	1	0.36721
8991	v01170052	drink	2	0.318169
5828	n03990474	pot	1	0.316549
17859	n04398044	teapot	1	0.254961
15166	n03690168	lorgnette	1	0.242645
5059	n04284002	spoon	1	0.217567
2430	n03498962	hatchet	1	0.213065
537	n03497657	hat	1	0.19101
507	n10287213	man	1	0.186272
5631	n02927296	butt	1	0.177221
3527	n03147509	cup	1	0.175728
8992	n03147901	cup	1	0.175728
207	n05600637	face	1	0.172203
9799	n03633091	ladle	1	0.164429
3480	n03624134	knife	1	0.159968
3130	n03623556	knife	1	0.159968
11695	n03355768	flagon	1	0.153052
3884	n02831894	besom	1	0.146892
13919	n04369282	swatter	1	0.129924
2427	n03077442	common_ax	1	0.109748
10246	n04389430	tankard	1	0.104166
10558	n04158250	scythe	1	0.103972
18218	n04292414	squash_racket	1	0.094573
16208	n04443766	toby	1	0.094453
5439	n00812526	grip	1	0.094426
8698	n03127024	crank	1	0.093627
1061	n03485997	handle	1	0.081229
13853	n03557270	iced-tea_spoon	1	0.076459
13185	n03488438	handset	1	0.07546
9798	n03204306	dipper	1	0.072543
11659	n03639077	landing_net	1	0.064556
8360	n04500060	turner	1	0.063632
2350	n04531098	vessel	1	0.056391
9915	n03214450	ditch_spade	1	0.054278
8726	n03950228	pitcher	1	0.04928
829	n00002684	object	1	0.047141
8987	n03629520	kylix	1	0.040485
1657	v01181295	serve	2	0.039938

(d) Tableau ordonné selon la pertinence contextuelle (similarité GloVe)

FIGURE 6.16 – Sous-graphe de contexte selon les approches proposées à la section 3.5

6.6 Discussion et conclusion

Ce chapitre a présenté quelques cas simples d'applications illustrant les possibilités du modèle de représentation d'environnement proposé. Les résultats obtenus sont comme attendus et suivent les explications données lors des chapitres précédents. Les requêtes/informations données par l'utilisateur sont analysées sémantiquement, converties en règles Prolog qui entraînent la création de thread de détection jusqu'à leur résolution. L'implémentation pratique a permis, en plus de valider notre approche d'un point de vue pratique, de soulever un certain nombre de difficultés :

– La présence d'instances identiques. Une piste serait de lier ces instances par une relation commutative et transitive *same*. On peut la voir comme l'équivalent visuelle de la relation *hyponym* de l'ontologie.

– La gestion des instances définies sémantiquement. Dans le deuxième cas, l'instance *my_cup* n'a pas été directement représentée : elle l'est implicitement par l'objet *object4*. On pourrait la représenter explicitement avec une relation *correspondTo* comme à la figure 6.17.

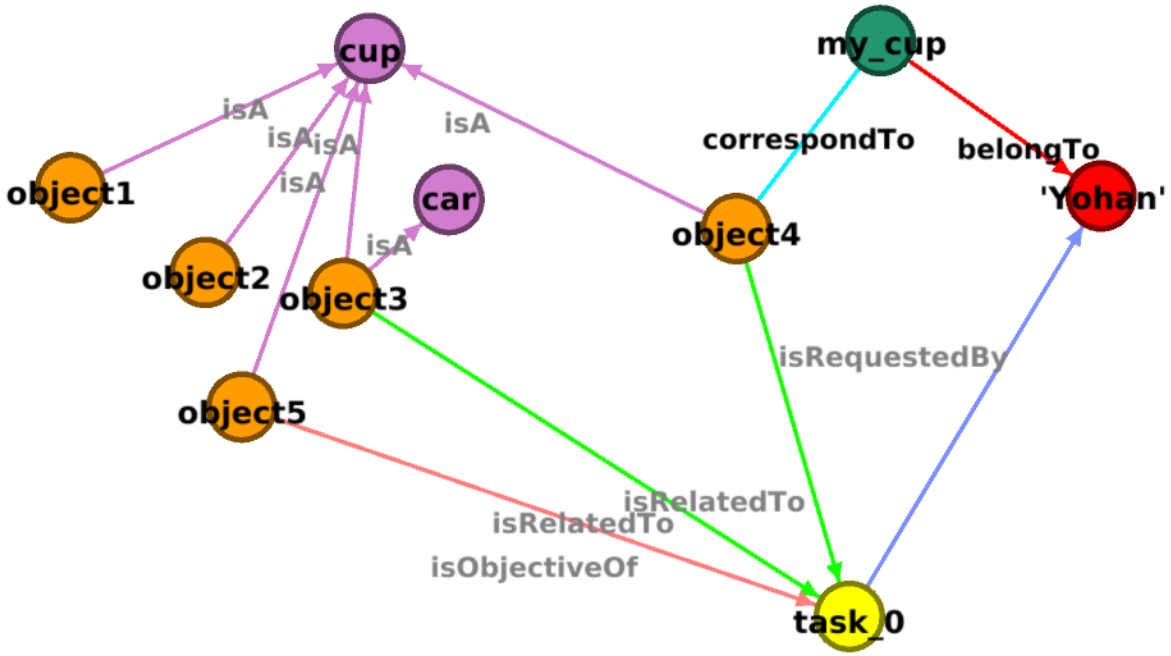


FIGURE 6.17 – Modèle d'instance résultant du cas présenté à la section 6.4 en représentant explicitement *my_cup*

Cela permettrait de gérer le cas où plusieurs instances sont nommées *my_cup*. Cette relation est équivalente à la relation $same$ mais avec un domaine d'entrée différent.

– La prise en compte des incertitudes liées à la classification et à la reconnaissance d'instances.

L'implémentation du modèle à trois couches que nous proposons n'en est encore qu'à l'étape de la preuve de concept. Cependant, elle nous a permis de montrer l'intérêt de l'approche et ses limites actuelles. Le choix des algorithmes de base sur lesquels repose l'implémentation actuelle (segmentation d'image, tracking, etc) peut être amélioré, ce qui devrait nous permettre d'aboutir à une plateforme complète dans les mois à venir. L'étape suivante est, bien entendu, son déploiement sur une plateforme robotisée complète.

Chapitre 7

Conclusion

Dans cette thèse, nous avons présenté un modèle de représentation d'environnements dans le cadre de la robotique autonome. Nous avons vu que le terme "d'environnement", bien que couramment utilisé, est relativement vague. Ce peut être purement géométrique, comme le plan d'un bâtiment, ou plus abstrait comme le fait qu'une cuisine soit une pièce pour cuisiner. Pour un robot interagissant avec des humains, il est nécessaire de comprendre le monde l'entourant de manière similaire à la nôtre. D'une part, il doit pouvoir analyser ce qui l'entoure avec ses capteurs qui lui donnent une information locale et spécifique. D'autre part, il doit également pouvoir utiliser des connaissances sémantiques globales et génériques. En effet, les bases de données textuelles disponibles sur Internet peuvent être vues comme un condensé de la connaissance humaine : l'analyse de ces données permet donc d'éviter un long apprentissage par expérience. Un autre aspect important est également de pouvoir partager un langage commun avec les humains. Ces deux modalités peuvent se nourrir l'une de l'autre, cependant nous nous limitons à la notion d'objet. Les connaissances sémantiques permettent d'obtenir une description générale (définition) d'une classe d'objet alors que les données sensorielles vont pouvoir décrire précisément une instance de cette classe. Autrement dit, la sémantique permet d'extraire des propriétés non observables d'une instance et la perception permet de "corriger" ses attributs spécifiques.

En essayant de relier la perception à la sémantique, on voit donc que l'on fait naturellement appel à la notion d'instance. De façon équivalente à l'instanciation en programmation orientée objet, c'est une réalisation physique d'une classe (concept). L'environnement peut ainsi se définir comme un ensemble d'instances, reliées entre elles par ce qu'on nomme un *contexte sous-jacent*. En effet, en quoi différencions nous un bureau d'une cuisine ? La réponse est simple : par les éléments (objets) composant ces deux types de scènes :

- Dans une cuisine, on va retrouver des ustensiles, des couverts et des produits alimentaires : le contexte associé est celui de la *nourriture* auquel on peut lier les actions de *manger* et *cuisiner*.
- Dans un bureau, on trouve généralement des documents (papiers), des stylos, un ordinateur et ses périphériques. Une notion commune à tous ces concepts est l'action d'*écrire*.

Cette notion de contexte est importante car elle permet d'inférer la possible présence d'objet **sans aucune expérience (données d'entraînement) préalable**. Ceci est à comparer aux méthodes probabilistes basées sur la cooccurrence d'objets, dont le besoin de données d'entraînement croît exponentiellement en fonction du nombre d'objets.

Nous aboutissons ainsi sur un modèle à trois couches (Perception, Instance, Sémantique). Le modèle de connaissance sémantique est représenté par une ontologie reliant les concepts par un ensemble de relations. Ces ontologies sont généralement construites manuellement. Certaines ont des domaines d'applications génériques (Dbpedia [ABK+07], OpenCyc [MCWD06]), d'autres plus spécifiques (CL [MEP+15] concernant les cellules). Cependant, à notre connaissance, aucune ontologie ne décrit précisément les concepts correspondant à des objets physiques. Un objet étant défini par sa fonctionnalité et son apparence physique, nous avons proposé une méthode de construction automatique d'ontologie à partir de définitions du dictionnaire. L'analyse topologique du graphe ontologique obtenu est une première piste permettant de trouver des concepts liés au contexte des différents objets. La suite de ces travaux serait d'améliorer l'ontologie en traitant le problème de la désambiguïsation. Une première idée serait de réanalyser les définitions des concepts en utilisant l'ontologie précédemment créée et de réitérer l'opération. L'ontologie présentée dans ce manuscrit a été construite à partir du dictionnaire WordNet : l'analyse d'autres dictionnaires en ligne permettrait de l'enrichir davantage. Enfin, il serait intéressant d'observer l'effet de ces améliorations sur la recherche de contexte.

L'analyse sémantique employée pour la création d'ontologie est réutilisée pour l'interaction avec un utilisateur humain. En particulier, nous proposons une méthode de conversion de requête en langage naturel en une règle Prolog (logique du 1er ordre). Le concept de "computable class" de KnowRob [TB17] est repris afin de vérifier certaines clauses à la volée, comme par exemple la présence d'une instance particulière ou d'une classe. Une extension naturelle est de considérer un retour à l'utilisateur lorsque la requête possède plusieurs (ou aucune) solutions. La problématique est alors de trouver l'information manquante qui, une fois connue, permettrait de répondre au mieux à la requête. Enfin, seule la tâche de recherche (du type "Donne moi la fourchette") est actuellement implémentée. Les perspectives envisagées incluent donc l'extension à d'autres actions

en s'appuyant sur les travaux de KnowRob.

L'ontologie est également exploitée pour la classification d'objets. En effet, en se plaçant dans l'hypothèse pratique d'un monde ouvert contenant un nombre potentiellement infini de classes, on doit se limiter à l'apprentissage de classifieurs binaires pour chaque concept. Cependant, cela n'est toujours pas satisfaisant pour trois raisons :

- Choix des données d'entraînement "négatifs" arbitraire
- Problème de mise à l'échelle : nombre de classifieurs à appliquer, probabilité d'erreur plus importante.
- Impossibilité de gérer la détection de nouvelles classes (non présentes dans l'ontologie)

L'approche proposée dans cette thèse est d'utiliser l'ontologie comme un arbre de décision dans laquelle chaque noeud est associé à un classifieur local sur les nœuds fils. Nous avons validé l'approche en se plaçant dans le cadre du Zero-shot learning c'est à dire que l'on sait par avance si une donnée est issue d'une classe connue ou non. Il serait intéressant de confirmer l'étude sur un graphe ontologique plus important. La prochaine étape sera d'étendre cette méthode au cas général où l'on ne sait pas si l'instance considérée est de classe connue ou inconnue.

Cette classification d'instances se fait à partir de descripteurs image issus de réseaux de neurones convolutifs (CNN). Bien qu'entraînés sur un ensemble limité de classes, il a été montré dans la littérature que les descripteurs issus de ces réseaux sont suffisamment génériques pour pouvoir être utilisés hors de leur domaine d'entraînement. Dans notre modèle, ces descripteurs sont utilisés pour l'entraînement des classifieurs liés aux concepts de l'ontologie mais également à la reconnaissance d'instances. On se retrouve donc face à deux objectifs en apparence contradictoires. En effet, la classification vise à obtenir des représentations telles que les distances intra-classe soient minimisées tout en maximisant les distances extra-classes. Or pour la reconnaissance d'instances on souhaite au contraire une séparation nette entre les instances d'une même classe, c'est à dire une distance intra-classe suffisante. Nous avons montré empiriquement que la dernière couche de convolution d'un réseau présente le meilleur compromis à cet égard. Ce résultat est d'autant plus intéressant qu'il vient contredire l'intuition basée sur les champs réceptifs des couches du réseau. Plus on monte dans les couches supérieures, plus le champ réceptif augmente et donc devient moins sensible aux changement locaux. Autrement dit, on s'attend à avoir des distances inter-instances plus grandes pour les couches inférieures. Ceci est vérifié pour un réseau dont les poids ont été choisis aléatoirement mais pas dans le cas d'un réseau entraîné. Ce résultat semble indiquer un phéno-

mène particulier lors de l'apprentissage au niveau de la dernière couche de convolution et il serait intéressant de valider théoriquement ce résultat.

Une autre propriété remarquable de ces descripteurs est leur régularité par rapport à des transformations de similarité (rotation, mise à l'échelle isotropique et translation). Il est assez évident, de part la régularité des images naturelles, que l'image elle-même est régulière pour ces transformations. Ce qui l'est moins c'est que la variété générée sur l'image par ces transformations est également propagée de façon relativement régulière jusqu'à la dernière couche de convolution. L'intérêt est donc de pouvoir inférer les paramètres de la transformation tout en profitant de la robustesse des descripteurs. Ceci est vérifié expérimentalement puis exploité dans une application simple de tracking. Les résultats obtenus sont encourageants mais ne sont pas comparables à l'état de l'art actuel. En vue d'améliorer cette approche, nous proposons une formalisation du problème sous forme matricielle. En particulier, nous montrons que l'on peut exprimer le gradient des descripteurs par rapport aux paramètres de la transformation en fonction du gradient de l'image. Nous démontrons quelques propriétés de l'opérateur de dérivation Kronecker qui permet d'obtenir un développement de Taylor d'une fonction multidimensionnelle et multivariée similaire au cas unidimensionnel/univarié. La perspective est d'utiliser l'approximation de Taylor du gradient afin d'éviter son calcul numérique coûteux.

Il est intéressant de faire le parallèle entre notre modèle et *l'échelle de causalité* proposée par Pearl [PM18], créateur des réseaux bayésiens. Elle classe les entités "intelligentes" (robot, animaux, homme) en trois échelons illustrés à la figure 7.1. Chacun de ces niveaux correspond à un degré "d'intelligence" :

- **Association par observation :** Le premier échelon correspond aux entités dont l'activité se limite à l'observation passive (sensorielle au sens large), à partir de laquelle elles infèrent des relations entre les variables de l'environnement. On peut mettre dans cette catégorie la plupart des animaux mais également les méthodes de machine learning (réseaux de neurones compris). En effet, ces méthodes extraient des informations (ex : probabilité de présence d'objet conditionnée sur le type de salle) uniquement à partir de données capteur. La majorité des travaux en robotique et vision par ordinateur se situent à ce niveau.
- **Intervention :** Contrairement au premier échelon passif, ce niveau correspond aux entités modifiant activement leur environnement afin d'obtenir de nouvelles relations. Certaines informations ne sont pas accessibles par l'observation passive et en particulier celles liées à la causalité. Reprenons l'exemple de Pearl sur le baromètre : une tempête va avoir pour effet une baisse du baromètre mais l'inverse n'est pas vrai. Cela se voit d'autant mieux en consi-

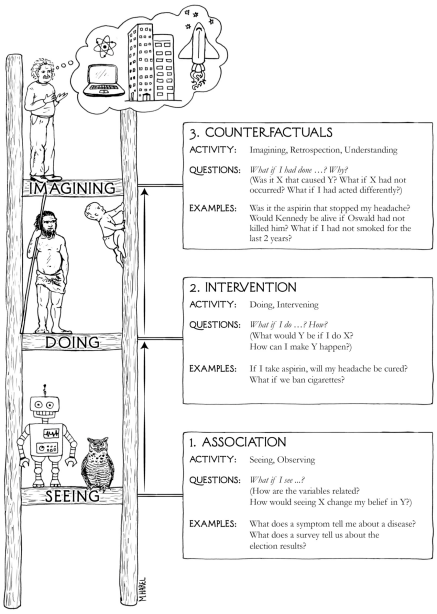


FIGURE 7.1 – Echelle de causalité extraite de [PM18]

dérant la différence entre les deux probabilités *P(tempête | baisse baromètre)* et *P(tempête | do(baisse baromètre))*, où le "do" indique une intervention afin de modifier activement la variable considérée. Il est clair que le fait de baisser intentionnellement le baromètre n'augmentera pas la probabilité d'avoir une tempête. En pratique, l'effet d'une intervention peut évidemment s'obtenir par des expériences. Il est également possible de se passer d'expérience en construisant des modèles intégrant les relations de causalité permettant d'inférer les effets d'intervention. Ce niveau n'a pas vraiment de correspondance dans l'état actuel de notre modèle car nous nous limitons à la représentation d'environnement sans considération des tâches à accomplir. Cependant ce niveau sera particulièrement pertinent dans le cadre de robots manipulateurs en ajoutant explicitement l'interaction avec l'environnement.

– **Contrafactualité :** Alors que l'échelon précédent se base encore sur un monde non observé mais observable, l'échelon final raisonne sur la contrafactualité, c'est à dire sur ce qui aurait pu être si ... mais qui n'est pas. On s'intéresse donc ici au "Pourquoi" permettant des expériences de l'esprit et de raisonner sur des événements non observables. C'est cette faculté d'imagination qui a amené l'homme à concevoir des innovations technologiques n'existant pas dans la nature. Pour un robot, la compréhension des liens entre les objets composant son environnement lui permet d'inférer des relations non observées.

  Dans le cadre de nos travaux orientés plus spécifiquement sur la description d'objets, la notion de causalité est remplacée par la notion de contexte sous-jacent. Considérons l'exemple de la

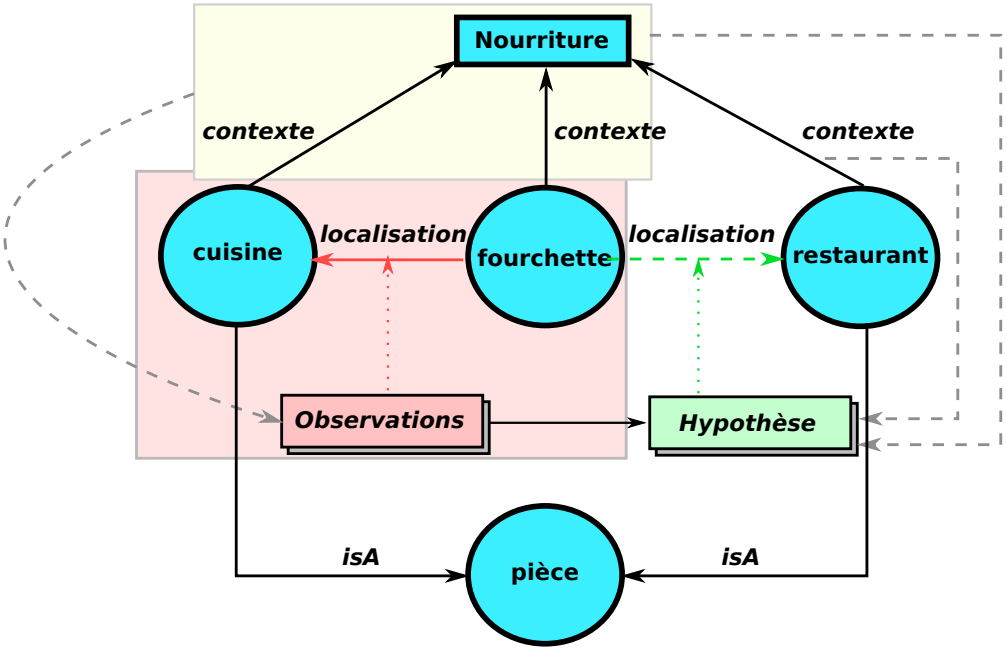


FIGURE 7.2 – Exemple illustrant le dernier niveau de l'échelle de causalité. Le cadre rouge correspond au premier niveau de l'échelle limité à l'observation (méthodes classiques de machine learning). On voit que l'information de contexte, combinée à l'observation, permet d'émettre des hypothèses de relations à savoir ici la présence de fourchette dans les restaurants.

figure 7.2 modélisant les liens entre les concepts de *fourchette*, *cuisine*, *nourriture* et *restaurant*. Le cadre rouge représente la configuration typique des méthodes de machine learning se basant sur les observations (données d'entraînement) afin d'estimer des relations, ici

$$p(fourchette|cuisine)$$

Cela correspond au premier niveau de l'échelle. Nous passons directement au dernier niveau en rajoutant le contexte sous-jacent reliant *fourchette* à *cuisine*, à savoir *nourriture*. Cette information explique la localisation des fourchettes dans la cuisine, permettant ainsi d'extrapoler cette relation à des concepts de même nature et partageant ce même contexte. Cela se traduit ici par l'hypothèse de localisation de fourchette dans les restaurants.

Nous partageons cette approche basée sur la notion fondamentale de *causalité* qui est à mettre en perspective avec les approches de *Data Mining* se basant principalement sur la notion de *corrélation*.

Cette thèse s'est attachée en priorité à construire les fondations d'une approche générique et

multimodale. Il était nécessaire de bien définir les contours de cette architecture avant de rentrer dans les détails des différentes problématiques sous-jacentes. Ce travail est un travail préliminaire montrant le besoin de modèles intégrant la sémantique (le "sens") dans une application robotique. Nous sommes convaincus que l'approche "Task oriented" est fondamentale, cependant elle nécessite une représentation plus large de l'environnement afin que le robot puisse s'y mouvoir et répondre aux besoins de l'humain. Beaucoup de développements restent à faire, mais ce travail nous semble aller dans ce sens.

Annexe A

Part-of-Speech tags

Tag	Signification	Example
VERB	Verbe	*manger, boire*
NOUN	Nom	*bouteille, chien, Lagrange, France*
PRON	Pronom	*je, te, ce, mes, qui, quoi*
ADJ	Adjectif	*petit, grand*
ADV	Adverbe	*après, avant, cependant*
ADP	Préposition/Postposition [1]	*dans, entre, par, pour*
CONJ	Conjonction	*mais, ou, et, quand*
DET	Déterminant	*le, un, des*
NUM	Nombre	*deux, cinq*
PRT	Particule	*up, down* [2]
X	Autre (mot étranger, abréviation)	*ONU, CONLL*
PUNCT	Ponctuation	*'.','?','!','('*

TABLE A.1 – Tags utilisés pour représenter les différentes natures universelles (**UPOS**) des mots [PDM11].

1. Présent dans certaines langues, équivalent d'une préposition mais placé après un nom ou un pronom.
2. Inexistant en français d'où les exemples en anglais.

Tag	Signification	Exemple
CC	Conjonction de coordination	*et, ou, mais*
CD	Nombre cardinal	*cinq, trois, 13%*
DT	Déterminant	*le, un, ces*
EX	Existence	*il y a six garçons*
FW	Mot étranger	*sweet*
IN	Conjonction, subordination ou préposition	*de, sur, avant, à moins que*
JJ	Adjectif	*petit, facile*
JJR	Adjectif (comparatif)	*plus petit, plus facile*
JJS	Adjectif (superlatif)	*le plus petit, le plus facile*
LS	Liste	
MD	Auxiliaire modal (verbe)	*pouvoir*
NN	Nom commun singulier	*tigre, chaise, rire*
NNS	Nom commun pluriel	*tigres, chaises, insectes*
NNP	Nom propre singulier	*Allemagne, Dieu, Alice*
NNPS	Nom propre pluriels	*On s'est vu il y a deux Noëls*
PDT	Prédéterminant	*both his children*
POS	Possession	*'s*
PRP	Pronom personnel	*moi, toi, lui*
RB	Adverbe	*Extremement, bruyamment*
RBR	Adverbe (comparatif)	*mieux*
RBS	Adverbe (superlatif)	*meilleur*
RP	Adverbe (particule)	*up, about, off*
SYM	Symbole	*%*
TO	*to* de l'infinitif	*what to do*
UH	Interjection	*oh, ah*
VB	Verbe (infinitif)	*think*
VBZ	Verbe (3ème personne du singulier présent)	*she thinks*
VBP	Verbe (présent hors 3ème personne du singulier)	*I think*
VBD	Verbe (passé)	*Ils pensaient*
VBN	Verbe (participe passé)	*un trésor caché*
VBG	Verbe (gérondif)	*en marchant*
WDT	wh-déterminant	*which, whatever, whichever*
WP	wh-pronom personnel	*what, who, whom*
WP$	wh-pronom possessif	*whose, whosever*
WRB	wh-adverbe	*where, when*
.	Ponctuation (fin de phrase)	*. ; ?**
,	Ponctuation (virgule)	*,*
:	Ponctuation (double point)	*:*
(	Séparateur contextuel (droit)	*(*
)	Séparateur contextuel (gauche)	*)*

TABLE A.2 – Tags utilisés pour représenter les différentes natures spécifiques (**XPOS**) des mots en anglais. Transcription en français de [San90]. Des exemples en anglais sont employés lorsqu'il n'y a pas d'équivalent en français.

Tag	Signification	Exemple
ACL†	Proposition modifiant un nom	*la difficulté[2] à vivre[1]*
ADVCL	Proposition modifiant un verbe (ou autre prédicat)	*Il faut venir[2] tôt pour avoir[1] de la place*
ADVMOD	Adverbe modifiant le sens d'un mot	*moins[1] souvent[2]*
AMOD	Locution adjectivale modifiant le sens d'un nom	*Marc boit du vin[2] rouge[1]*
APPOS	Apposition	*Sam[2], mon frère[1], arrive*
AUX	Verbe non principal d'une proposition	*On peut[1] nager[1] dans le lac*
AUXPASS	Verbe non principal d'une proposition (forme passive)	*Kennedy a été[1] tué[2]*
CASE†	Préposition attachée à un nom	*le bureau du[1] président[2]*
CC	Relation entre conjonction de coordination et le nom précédent	*Marc est riche[2] et[1] gentil*
CCOMP	Proposition subordonnée complétive	*Il dit[2] que tu aimes[1] nager*
COMPOUND†	Mot composé	*début[2] décembre[1]*
CONJ	Relation entre noms connectés par une conjonction de coordination	*Marc est riche[2] et gentil[1]*
COP	Relation liant l'attribut au sujet d'une proposition	*Marc est[1] riche[2]*
CSUBJ	Subordonnée nominale	*Qu'il vienne[1] m'embête[2]*
CSUBJPASS	Subordonnée nominale (forme passive)	*Qu'il soit venu[1] m'a embêté[2]*
DEP	Relation de type inconnu	
DET	Relation entre un nom et son déterminant	*Le[1] chien[2] est gentil*
DISCOURSE	Relation avec élément indépendant (interjection)	*Il est[2] en Argentine :)[1]*
DISLOCATED†	Relation entre éléments séparés	*Il ne faut pas la manger[2], la plasticine[1]*
DOBJ	Complément d'objet direct	*Il mange[2] une pomme[1]*
EXPL	Mots explétifs	*Il[1] est clair[2] que l'on doit refuser*
FOREIGN†	Séquence de mots étrangers	
GOESWITH	Relation entre partie d'un mot séparé par erreur	*aujourd'[2] hui[1]*
IOBJ	Complément d'objet indirect	*Il m'[1]a donné[2] un chien*
LIST†	Chaîne d'objets comparable	*Marc[1] Dupond Tel[2] : xxxxxxx Email[2] : xxx@xxx*
MARK	Mot introduisant une proposition subordonnée	*Il dit que[1] tu aimes[2] nager*
MWE/FIXED	Expression grammatisée composée	*plutôt[1] que[2]*
NAME/FLAT†	Nom propre composé	*Henri[1] Poincaré[2]*
NEG	Négation	*Je ne fais[2] jamais[1] ça*
NMOD/NN	Nom modifiant un autre nom	*Le résultat[2] de la course[1]*
NPADVMOD	Phrase nominale en tant que modifieur adverbial	*90% des australiens l'aime[2], le plus[1] de tous les pays*
NSUBJ	Sujet nominal d'une proposition	*Marc[1] gagne[2] le match*
NSUBJPASS	Sujet nominal d'une proposition (forme passive)	*Le match[1] est gagné[2] par Marc*
NUMBER	Nombre composé	*trois[2] mille[1]*
NUMMOD/NUM†	Modificateur numérique	*Il a mangé 2[1] pommes[2]*
PARATAXIS	Relation entre un verbe principal et d'autres éléments séparés de la phrase	*C'est l'idée[2] : les enfants sont le futur[1]*
INFMOD/PARTMOD/VMOD	Verbe à l'infinitif (INFMOD) ou participe (PARTMOD) à la racine d'une phrase	*Les champignons[2] ramassés[1] hier sont bon.*
PCOMP	Préposition complétive	*Nous ne savons si[1] les utilisateurs courent[2] un risque.*
POBJ	Objet d'une préposition	*Je m'assois sur[1] la chaise[2]*
POSS	Modifieur de possession	*Leurs[1] bureaux[2]*
POSSESSIVE	Modificateur de possession 's (anglais)	*Bill[1] 's[2] clothes*
PRECONJ	Préconjonction	*Ni[2] Marc[1] Ni Marie n'est présent.*
PREDET	Prédéterminant	*Tous[2] les garçons[1] sont là*
PREP	Modifieur prépositionel	*J'ai vu[2] un chat avec[1] un chapeau*
PRT	Particule verbiale (anglais)	*They shut[2] down[1] the station*
PUNCT	Ponctuation	*Marc est riche[2].[1]*
QUANTMOD	Modifieur quantitatif (peut être compris dans ADVMOD)	*Il y a 200[1] personnes[2]*
RCMOD	Proposition relative modifiant un nom	*J'ai vu l'homme[2] que tu aimes[1]*
REMNANT/ORPHAN†	Ellipse	*Marc a gagné l'or[2] et Marie l'argent[1]*
REPARANDUM†	Indique une disfluence verbale (généralement à l'orale)	*Va à droi[1]...à gauche[2].*
ROOT	Noeud racine (virtuel) d'une phrase	*(ROOT)[2] J'aime[1] les pommes*
TMOD	Modifieur temporel	*La nuit[1] dernière, j'ai vu[2] un chat.*
VOCATIVE†	Vocatif	*Marc[1], comment ça va[2] ?*
XCOMP	Proposition complétive sans sujet propre	*Il dit que tu aimes[2] nager[1]*

TABLE A.3 – Tags utilisés pour représenter les différentes relations de dépendance (**UDR**) [DMM08] [DMDS$^+$14] [NDMG$^+$16]. L'exposant [1](resp. [2]) représente le premier (resp. second) terme de la relation . Pour une raison inconnue, l'ordre des arguments employé par SyntaxNet et par [NDMG$^+$16] est inversé. Ici nous avons choisi l'ordre de SyntaxNet. Les relations non utilisées par SyntaxNet sont indiquées par †.

Annexe B

Opérateurs matriciels

Cette annexe a pour but d'introduire les définitions et les propriétés intervenant dans la formalisation de la section 4.2.8. Pour plus de détails, nous invitons le lecteur à se référer à [AZK07] [LT08].

B.1 Définitions et propriétés

Lorsque la définition ou propriété est une nouvelle contribution de notre part, nous l'indiquons par **(*)**.

Définition 14. *Le produit matriciel de Hadamard (ou produit élément par élément), noté $\odot$, entre deux matrices $A = [a_{ij}] \in M_{m,n}(\mathbb{C}),\ B = [b_{ij}] \in M_{m,n}(\mathbb{C})$ de même dimensions est défini par*

$$A \odot B = \begin{bmatrix} a_{11}b_{11} & \dots & a_{1n}b_{1n} \\ \vdots & & \vdots \\ a_{m1}b_{m1} & \dots & a_{mn}b_{mn} \end{bmatrix} = [a_{ij}b_{ij}] \in M_{m,n} \tag{B.1}$$

Nous définissons également la puissance de Hadamard comme suit

$$A^{\odot n} = A \odot A \dots \odot A \tag{B.2}$$

Définition 15. *Le produit de Kronecker, noté $\otimes$, entre deux matrices $A = [a_{ij}] \in M_{m,n},\ B \in M_{p,q}$*

est défini par la matrice partitionnée

$$A \otimes B = \begin{bmatrix} a_{11}B & \dots & a_{1n}B \\ \vdots & & \vdots \\ a_{m1}B & \dots & a_{mn}B \end{bmatrix} = [a_{ij}B] \in M_{mp,nq} \tag{B.3}$$

La puissance de Kronecker est notée

$$A^{\otimes n} = A \otimes A \dots \otimes A \tag{B.4}$$

Définition 16. *Soient $A = [A_{ij}]_{(i,j)\in[\![1,r]\!]\times[\![1,s]\!]} \in M_{m,n},\ B = [B_{kl}] \in M_{p,q}$ des matrices partitionnées avec $A_{ij} \in M_{m_i,n_j}$ ($\sum m_i = m$, $\sum n_j = n$) et $B_{kl} \in M_{q_k,p_l}$ ($\sum q_k = q$, $\sum p_l = p$). Le produit de Khatri-Rao de ces matrices est une matrice partitionnée $r \times s$ définie par*

$$A * B = \begin{bmatrix} A_{11} \otimes B_{11} & \dots & A_{1s} \otimes B_{1s} \\ \vdots & & \vdots \\ A_{r1} \otimes B_{r1} & \dots & A_{rs} \otimes B_{rs} \end{bmatrix} = [A_{ij} \otimes B_{ij}] \in M_{M,N} \tag{B.5}$$

avec $M = \sum_{i=1}^{r} m_i p_i,\ N = \sum_{j=1}^{s} n_j q_j$.

Définition 17. *Une matrice de permutation $P_{m,n} \in M_{mn,mn}$ est une matrice partitionnée $m \times n$ telle que la $ij^{ème}$ sous-matrice $\hat{P}_{ij} \in M_{n,m}$ a son élément ji valant 1 et 0 partout ailleurs*

$$P_{m,n} = \begin{bmatrix} \hat{P}_{11} & \dots & \hat{P}_{1n} \\ \vdots & & \vdots \\ \hat{P}_{m1} & \dots & \hat{P}_{mn} \end{bmatrix},\ \hat{P}_{ij} = \begin{bmatrix} 0 & \dots & 0 & 0 \\ 0 & \dots & p_{ji} = 1 & 0 \\ 0 & \dots & 0 & 0 \end{bmatrix} \tag{B.6}$$

Remarquons que nous avons également

$$P_{m,n}^T = P_{m,n}^{-1} = P_{n,m} \tag{B.7}$$

Définition 18. *Soient $A \in M_{m,n},\ k \in [\![1,m]\!]$ et $l \in [\![1,n]\!]$. Nous définissons $\alpha = [\![i_1,\dots,i_k]\!] \subseteq [\![1,\dots,m]\!]$ et $\beta = [\![j_1,\dots,j_l]\!] \subseteq [\![1,\dots,n]\!]$. Une matrice de permutation partielle $S_\alpha \in M_{k,m}$ est telle que l'élément en position i_q de la $q^{ème}$ ligne vaut 1 et 0 partout ailleurs. Nous obtenons alors la sous-matrice de A composée des lignes α et des colonnes β, notée $[A]_{\alpha,\beta}$, comme suit*

$$[A]_{\alpha,\beta} = S_\alpha A S_\beta^T \tag{B.8}$$

Définition 19. *Soit $m, n \in \mathbb{N}^*$. L'opérateur vecteur $Vec : M_{m,n} \to \mathbb{R}^{mn}$ déroule les matrices en un vecteur en empilant les colonnes les unes après les autres d'où*

$$\forall A = [a_{ij}] \in M_{m,n},\ Vec(A) = \begin{bmatrix} a_{11} & a_{21} & \dots & a_{m1} & a_{12} & a_{22} & \dots & a_{1n} & \dots & a_{mn} \end{bmatrix}^T \quad \text{(B.9)}$$

Définition 20. *Soit $m \in \mathbb{N}^*$. L'opérateur d'extraction diagonal $Vecd : M_{m,m} \to \mathbb{R}^m$ extrait le vecteur diagonal d'une matrice carrée d'où*

$$\forall A = [a_{ij}] \in M_{m,m},\ Vecd(A) = \begin{bmatrix} a_{11} & a_{22} & \dots & a_{mm} \end{bmatrix}^T \quad \text{(B.10)}$$

Définition 21. *(*) Nous définissons maintenant une nouvelle extension de l'opérateur vecteur à certaines matrices partitionnées. Soient $m, k \in \mathbb{N}^*$ et $A = [\mathbf{a}_{ij}] \in M_{km,m}$ une matrice partitionnée $m \times m$ avec des blocs colonnes $\mathbf{a}_{ij} \in \mathbb{R}^k$. L'opérateur d'extraction de diagonal bloc $Vecd^b$ extrait la diagonale par bloc en empilant les blocs colonne dans un vecteur*

$$Vecd^b(A) = \begin{bmatrix} \mathbf{a}_{11}^T & \mathbf{a}_{22}^T & \dots & \mathbf{a}_{mm}^T \end{bmatrix}^T,\ A = \begin{bmatrix} \mathbf{a}_{11} & & & \\ & \mathbf{a}_{22} & & \\ & & \ddots & \\ & & & \mathbf{a}_{mm} \end{bmatrix} \quad \text{(B.11)}$$

Définition 22. *(*) Soit $\mathbf{u} = [u_1 \dots u_n]^T \in \mathbb{R}^n$. Nous définissons un opérateur **diag** : $\mathbb{R}^n \to M_{n,n}$ qui renvoie une matrice diagonale composée des éléments du vecteur d'entrée*

$$\mathbf{diag}\,(\mathbf{u}) = \begin{bmatrix} u_1 & & \\ & \ddots & \\ & & u_n \end{bmatrix} \quad \text{(B.12)}$$

Nous nous intéressons dorénavant à certaines propriétés des opérateurs définis précédemment.

Théorème 5. *Soient $A \in M_{m,n},\ B \in M_{p,q},\ C \in M_{n,l},\ D \in M_{q,s}$ telles que le produit AC et BD existent. Alors*

$$(A \otimes B)(C \otimes D) = (AC \otimes BD) \quad \text{(B.13)}$$

Théorème 6. *Soient $A \in M_{m,n}$ et $B \in M_{q,p}$. Alors*

$$(A \otimes B)^T = A^T \otimes B^T$$

Théorème 7. *Soit $A \in M_{m,n}$. Nous avons alors*

$$Vec\,(A) = P_{m,n} Vec\left(A^T\right) \tag{B.14}$$

avec $P_{m,n}$ une matrice de permutation définie par (B.6).

Théorème 8. *Soient $\mathbf{u} \in \mathbb{R}^m$ et $\mathbf{v} \in \mathbb{R}^n$. Alors*

$$P_{m,n}\,(\mathbf{u} \otimes \mathbf{v}) = \mathbf{v} \otimes \mathbf{u} \tag{B.15}$$

Théorème 9. *Soient $A \in M_{m,n},\ X \in M_{n,p}$ et $B \in M_{p,q}$. Alors*

$$Vec(AXB) = \left(B^T \otimes A\right) Vec(X) \tag{B.16}$$

Corrolaire 1. *Soient $\mathbf{u} \in \mathbb{R}^m$ et $\mathbf{v} \in \mathbb{R}^n$. Alors*

$$Vec\left(\mathbf{u}\mathbf{v}^T\right) = Vec\left(\mathbf{u} \otimes \mathbf{v}^T\right) = \mathbf{v} \otimes \mathbf{u} \tag{B.17}$$

Théorème 10. *Définissons R_n comme*

$$R_n = \begin{bmatrix} \mathbf{e}_1 & \mathbf{e}_{n+2} & \mathbf{e}_{2n+3} & \dots & \mathbf{e}_{n^2} \end{bmatrix} \in M_{n^2,n} \tag{B.18}$$

avec $\mathbf{e}_k$ le $k^{ème}$ vecteur de la base canonique de $\mathbb{R}^{n^2}$ (sa $k^{ème}$ coordonnée vaut 1 et 0 autrement). Nous avons alors pour toute matrice carrée $A \in M_{n,n}$

$$Vecd\,(A) = R_n^T Vec\,(A) \tag{B.19}$$

Remarquons que $R_n^T R_n = I_n$ et pour toute matrice diagonale X

$$Vec\,(X) = R_n Vecd\,(X) \tag{B.20}$$

Théorème 11. ***(*)** Nouvelle extension du théorème précédent. Soient $k \geq 1$ et $D \in M_{kn,n}$ une*

matrice par blocs colonne définie par

$$D = \begin{bmatrix} \mathbf{a}_1 & & & \\ & \mathbf{a}_2 & & \\ & & \ddots & \\ & & & \mathbf{a}_n \end{bmatrix} \tag{B.21}$$

avec $\mathbf{a}_i \in \mathbb{R}^k,\ \forall i$. *Nous définissons* $R_{n,k} = R_n \otimes I_k$ *avec* R_n *définie par (B.18). Il s'en suit*

$$Vecd^b(D) = R_{n,k}^T Vec(D) \tag{B.22}$$

Les matrices considérées ici étant diagonales par blocs, nous avons de manière similaire à (B.20) :

$$Vec(D) = R_{n,k} Vecd^b(D) \tag{B.23}$$

Théorème 12. *Soient* $A, B \in M_{n,m}$ *deux matrices de même dimensions. Alors*

$$Vec(A \odot B) = Vec(A) \odot Vec(B) \tag{B.24}$$

B.2 Opérateur de K-dérivation

On rappelle la définition de l'opérateur de K-dérivation (ou dérivation de Kronecker) [Ter03] [DB15]

Définition 23. *Soit* $f : \mathbb{R}^n \to \mathbb{R}^m$ *tel que*

$$\forall \mathbf{x} = [x_1 \dots x_n]^T \in \mathbb{R}^n,\ \mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x}) \dots f_m(\mathbf{x})]^T$$

L'opérateur différentiel est noté $D_{\mathbf{x}} = \left[\frac{\partial}{\partial \mathbf{x_1}} \dots \frac{\partial}{\partial \mathbf{x_n}}\right]^T$. *L'opérateur de K-dérivation noté* $D_{\mathbf{x}}^{\otimes}$ *est défini par*

$$D_{\mathbf{x}}^{\otimes}\mathbf{f}(\mathbf{x}) = \mathbf{f}(\mathbf{x}) \otimes D_{\mathbf{x}} = Vec\left(J_{\mathbf{f}}^T(\mathbf{x})\right) \in M_{nm,1} \tag{B.25}$$

avec $J_{\mathbf{f}} = \left[\frac{\partial f_i}{\partial x_j}\right]$ *la matrice jacobienne de* f. *La dérivation à l'ordre* k *est définie par*

$$D_{\mathbf{x}}^{\otimes k}\mathbf{f} = D_{\mathbf{x}}^{\otimes}\left(D_{\mathbf{x}}^{\otimes k-1}\mathbf{f}\right) \in M_{n^k m,1} \tag{B.26}$$

Certaines propriétés de cette opérateur sont données dans [Ter03] [DB15]. Nous allons ici

étendre celles-ci et en démontrer de nouvelles.

Théorème 13 (Généralisation de la propriété de mise à l'échelle)**.** ***(*)*** *Soient $f : \mathbb{R}^n \to \mathbb{R}^m$, $A \in M_{p,m}$ telles que chaque élément a_{ij} de A soit une fonction de $\mathbb{R}^n$ dans $\mathbb{R}$. On définit également la fonction g telle que*

$$\forall \mathbf{x} \in \mathbb{R}^n,\ \mathbf{g}(\mathbf{x}) = A(\mathbf{x})\mathbf{f}(\mathbf{x})$$

Alors

$$\forall \mathbf{x},\ D_{\mathbf{x}}^{\otimes}\mathbf{g}(\mathbf{x}) = (A \otimes I_n)\, D_{\mathbf{x}}^{\otimes}\mathbf{f}(\mathbf{x}) + \left(I_p \otimes \mathbf{f}^T(\mathbf{x}) \otimes I_n\right) D_{\mathbf{x}}^{\otimes} Vec\left(A^T\right)(\mathbf{x}) \tag{B.27}$$

Démonstration. Soit $\mathbf{x} \in \mathbb{R}^n$ et $\mathbf{a}_i^T$ la $i^{\text{ème}}$ ligne de A. Nous avons par définition de l'opérateur de K-dérivation (4.88)

$$\begin{aligned} D_{\mathbf{x}}^{\otimes}\mathbf{g}(\mathbf{x}) &= Vec\left(\left[\frac{\partial \mathbf{a}_i^T\mathbf{f}}{\partial x_j}\right]^T\right) && \text{(B.28)}\\ &= Vec\left(\left[\mathbf{a}_i^T\frac{\partial \mathbf{f}}{\partial x_j}\right]^T\right) + Vec\left(\left[\mathbf{f}^T\frac{\partial \mathbf{a}_i}{\partial x_j}\right]^T\right) && \text{(B.29)}\end{aligned}$$

Le premier terme du membre de droite de (B.29) peut être déduit du théorème (9)

$$Vec\left(\left[\mathbf{a}_i^T\frac{\partial \mathbf{f}}{\partial x_j}\right]^T\right) = Vec\left(J_{\mathbf{f}}^T A^T\right) = (A \otimes I_n)\, D_{\mathbf{x}}^{\otimes}\mathbf{f}(\mathbf{x}) \tag{B.30}$$

Remarquons que lorsque A est une matrice constante, seul ce terme reste et nous obtenons la propriété de mise à l'échelle (scaling) donnée par [DB15].

Nous passons maintenant au second terme

$$\begin{aligned} \left[\mathbf{f}^T\frac{\partial \mathbf{a}_i}{\partial x_j}\right] &= \begin{bmatrix}\mathbf{f}^T & & \\ & \ddots & \\ & & \mathbf{f}^T\end{bmatrix}\begin{bmatrix}\frac{\partial \mathbf{a}_1}{\partial x_1} & \cdots & \frac{\partial \mathbf{a}_1}{\partial x_n}\\ \vdots & & \vdots \\ \frac{\partial \mathbf{a}_p}{\partial x_1} & \cdots & \frac{\partial \mathbf{a}_p}{\partial x_n}\end{bmatrix} && \text{(B.31)}\\ &= \left(I_p \otimes \mathbf{f}^T\right)\left[\frac{\partial \mathbf{a}_i}{\partial x_j}\right] && \text{(B.32)}\end{aligned}$$

De plus, nous pouvons réécrire la transposé de la matrice par blocs colonne $\left[\frac{\partial \mathbf{a}_i}{\partial x_j}\right]$ comme suit

$$\left[\frac{\partial \mathbf{a}_i}{\partial x_j}\right]^T = Vec\left(A^T\right)^T \otimes D_{\mathbf{x}} \tag{B.33}$$

En remplaçant (B.33) dans (B.31) et en appliquant le théorème (9), on obtient

$$Vec\left(\left[\mathbf{f}^T\frac{\partial \mathbf{a}_i^T}{\partial x_j}\right]^T\right) = \left(I_p \otimes \mathbf{f}^T \otimes I_n\right) Vec\left(Vec\left(A^T\right)^T \otimes D_{\mathbf{x}}\right) \tag{B.34}$$

Pour développer (B.34), on utilise le corollaire 1 ainsi que les théorèmes 6, 7 et 8 pour obtenir

$$\begin{aligned} Vec\left(Vec\left(A^T\right)^T \otimes D_{\mathbf{x}}\right) &= P_{mn} Vec\left(Vec\left(A^T\right) \otimes D_{\mathbf{x}}^T\right) && \text{(B.35)}\\ &= P_{mn}(D_{\mathbf{x}} \otimes Vec\left(A^T\right)) && \text{(B.36)}\\ &= Vec\left(A^T\right) \otimes D_{\mathbf{x}} && \text{(B.37)} \end{aligned}$$

d'où en remplaçant dans (B.34)

$$\begin{aligned} Vec\left(\left[\mathbf{f}^T\frac{\partial \mathbf{a}_i^T}{\partial \mathbf{x}_j}\right]^T\right) &= \left(I_p \otimes \mathbf{f}^T \otimes I_n\right)\left(Vec\left(A^T\right) \otimes D_{\mathbf{x}}\right) && \text{(B.38)}\\ &= \left(I_p \otimes \mathbf{f}^T \otimes I_n\right) D_{\mathbf{x}}^{\otimes} Vec\left(A^T\right) && \text{(B.39)} \end{aligned}$$

□

Remarque : La propriété *"Chain Rule"* dans [DB15] n'en n'est pas réellement une : elle concerne l'opérateur de K-dérivation sur une fonction définie par un produit de Kronecker et non par une composition de fonctions.

Théorème 14 (Règle de dérivation par chaîne)**.** ***(*)*** *Soient $f : \mathbb{R}^n \to \mathbb{R}^m$ et $g : \mathbb{R}^p \to \mathbb{R}^n$. Alors*

$$\forall \mathbf{x} \in \mathbb{R}^p,\ D_{\mathbf{x}}^{\otimes}(\mathbf{f} \circ \mathbf{g})(\mathbf{x}) = K\left(I_{n^2} \otimes D_{\mathbf{x}}^{\otimes}\mathbf{g}(\mathbf{x})\right) D_{\mathbf{y}}^{\otimes}\mathbf{f}\left(\mathbf{g}(\mathbf{x})\right) \tag{B.40}$$

$K \in M_{np,n^2mp}$ est une matrice par blocs $1 \times nm$ avec chaque bloc de dimension $np \times np$. Chacun de ces blocs est aussi une matrice par bloc $n \times n$ avec des sous-blocs de dimension $p \times p$. Chaque sous-bloc du bloc en position ij est une matrice nulle à l'exception du sous-bloc ij qui vaut l'identité I_p. Formellement,

$$\begin{aligned} K &= \begin{bmatrix} K_{11} & K_{12} & \dots & K_{nm} \end{bmatrix} && \text{(B.41)}\\ \forall i, j\ K_{ij} &= X_{ji} \otimes I_p && \text{(B.42)} \end{aligned}$$

avec $X_{ji} \in M_{n,n}$ des matrices de zéros avec 1 à la position ji.

Démonstration. Soit $\mathbf{x} \in \mathbb{R}^p$. Nous avons par définition (4.88)

$$\begin{aligned} D^{\otimes}_{\mathbf{x}}(\mathbf{f} \circ \mathbf{g})(\mathbf{x}) &= Vec\left(J^T_{\mathbf{g}}(\mathbf{x}) J^T_{\mathbf{f}}(\mathbf{g}(\mathbf{x}))\right) \\ &= \left(I_m \otimes J^T_{\mathbf{g}}(\mathbf{x})\right) D^{\otimes}_{\mathbf{y}}\mathbf{f}(\mathbf{g}(\mathbf{x})) \end{aligned} \tag{B.43}$$

En utilisant à nouveau le théorème (9), on remarque que

$$\forall X \in M_{n,n},\ Vec\left(J^T_{\mathbf{g}} X\right) = \left(X^T \otimes I_p\right) D^{\otimes}_{\mathbf{x}}\mathbf{g} = \left(I_m \otimes J^T_{\mathbf{g}}(\mathbf{x})\right) Vec\,(X) \tag{B.44}$$

Considérons en particulier les n^2 matrices X_{ij}, avec $i, j \in [\![1, n]\!]$, remplies de zéros avec l'élément ij valant 1. Alors

$$\forall i, j\ Vec\,(X_{ij}) = \mathbf{e}_{(i-1)n+j} \in \mathbb{R}^{n^2} \tag{B.45}$$

Si nous considérons les n^2 équations obtenues en remplaçant X par chaque X_{ij} dans (B.44), nous pouvons les concaténer sous forme matricielle

$$\begin{aligned} \left(I_m \otimes J^T_{\mathbf{g}}(\mathbf{x})\right) I_{n^2} &= \left[\left(X^T_{11} \otimes I_p\right) D^{\otimes}_{\mathbf{x}}\mathbf{g} \quad \left(X^T_{12} \otimes I_p\right) D^{\otimes}_{\mathbf{x}}\mathbf{g} \quad \dots \quad \left(X^T_{nn} \otimes I_p\right) D^{\otimes}_{\mathbf{x}}\mathbf{g}\right] \\ &= \left[\left(X^T_{11} \otimes I_p\right) \quad \left(X^T_{12} \otimes I_p\right) \quad \dots \quad \left(X^T_{nn} \otimes I_p\right)\right] \left(I_{n^2} \otimes D^{\otimes}_{\mathbf{x}}\mathbf{g}(\mathbf{x})\right) \quad \text{(B.46)} \\ &= \left[(X_{11} \otimes I_p) \quad (X_{21} \otimes I_p) \quad \dots \quad (X_{nn} \otimes I_p)\right] \left(I_{n^2} \otimes D^{\otimes}_{\mathbf{x}}\mathbf{g}(\mathbf{x})\right) \quad \text{(B.47)} \end{aligned}$$

Posons

$$K = \left[(X_{11} \otimes I_p) \quad (X_{21} \otimes I_p) \quad \dots \quad (X_{nn} \otimes I_p)\right]$$

(B.43) se réécrit alors

$$D^{\otimes}_{\mathbf{x}}(\mathbf{f} \circ \mathbf{g})(\mathbf{x}) = K\left(I_{n^2} \otimes D^{\otimes}_{\mathbf{x}}\mathbf{g}(\mathbf{x})\right) D^{\otimes}_{\mathbf{y}}\mathbf{f}\left(\mathbf{g}(\mathbf{x})\right) \tag{B.48}$$

□

Nous définissons un nouvel opérateur $\check{\boxdot}$ en lien avec l'opérateur $\boxdot$ précédemment défini à la section 4.2.8.

Définition 24. *Soient* $\mathbf{x} \in \mathbb{R}^{nm}$ *et* $\mathbf{y} \in \mathbb{R}^m$. *L'opération* $\check{\boxdot}$ *entre deux vecteurs* $\mathbf{x}$ *et* $\mathbf{y}$ *est défini par*

$$\mathbf{x} \check{\boxdot} \mathbf{y} = \mathbf{x} \odot (\mathbf{y} \otimes \mathbf{1}_n) \tag{B.49}$$

avec $\mathbf{1}_n \in \mathbb{R}^n$ *un vecteur de* 1.

Théorème 15. *Soient $A \in M_{n,m}$ et $\mathbf{x} \in \mathbb{R}^m$. Alors*

$$Vec\left(A \boxdot \mathbf{x}\right) = Vec\left(A\right) \check{\boxdot} \mathbf{x} \tag{B.50}$$

Démonstration.

$$\begin{aligned} Vec\left(A \boxdot \mathbf{x}\right) = Vec\left(A \odot \left(\mathbf{1}_n \otimes \mathbf{x}^T\right)\right) &= Vec\left(A\right) \odot Vec\left(\mathbf{1_n} \otimes \mathbf{x}^T\right) \\ &= Vec\left(A\right) \odot \left(\mathbf{x} \otimes \mathbf{1_n}\right) \text{ avec (1)} \\ &= Vec\left(A\right) \check{\boxdot} \mathbf{x} \end{aligned}$$

□

Théorème 16 (Dérivation de produit de Hadamard)**.** ***(*)** Soient $f,\ g$ deux fonctions de $\mathbb{R}^n$ dans $\mathbb{R}^m$. La K-dérivation d'un produit de Hadamard de fonctions est donnée par*

$$D^{\otimes}_{\mathbf{x}}\left(\mathbf{f} \odot \mathbf{g}\right) = D^{\otimes}_{\mathbf{x}}\mathbf{f} \check{\boxdot} \mathbf{g} + D^{\otimes}_{\mathbf{x}}\mathbf{g} \check{\boxdot} \mathbf{f} \tag{B.51}$$

Démonstration. Par définition de la K-dérivation,

$$D^{\otimes}_{\mathbf{x}}\left(\mathbf{f} \odot \mathbf{g}\right) = Vec\left(\left[\frac{\partial f_i}{\partial x_j} g_i\right]^T\right) + Vec\left(\left[\frac{\partial g_i}{\partial x_j} f_i\right]^T\right) \tag{B.52}$$

De plus, on peut aisément vérifier que

$$\left[\frac{\partial f_i}{\partial x_j} g_i\right] = J_{\mathbf{f}} \odot \left(\mathbf{1}_n^T \otimes \mathbf{g}\right) \tag{B.53}$$

On remplace (B.53) dans (B.52), ce qui donne avec le théorème (15)

$$\begin{aligned} D^{\otimes}_{\mathbf{x}}\left(\mathbf{f} \odot \mathbf{g}\right) &= Vec\left(J_{\mathbf{f}}^T \odot \left(\mathbf{1}_n \otimes \mathbf{g}^T\right)\right) + Vec\left(J_{\mathbf{g}}^T \odot \left(\mathbf{1}_n \otimes \mathbf{f}^T\right)\right) && \text{(B.54)} \\ &= Vec\left(J_{\mathbf{f}}^T \boxdot \mathbf{g}\right) + Vec\left(J_{\mathbf{g}}^T \boxdot \mathbf{f}\right) && \text{(B.55)} \\ &= Vec\left(J_{\mathbf{f}}^T\right) \check{\boxdot} \mathbf{g} + Vec\left(J_{\mathbf{g}}^T\right) \check{\boxdot} \mathbf{f} && \text{(B.56)} \\ &= D^{\otimes}_{\mathbf{x}}\mathbf{f} \check{\boxdot} \mathbf{g} + D^{\otimes}_{\mathbf{x}}\mathbf{g} \check{\boxdot} \mathbf{f} && \text{(B.57)} \end{aligned}$$

□

B.3 Preuves

B.3.1 Preuve de la proposition 1, section 4.2.8

Démonstration. Soient $D_{\widetilde{\nabla I}} \in M_{3H_0W_0,H_0W_0}$ et $D_{\Gamma_{k,\boldsymbol{\varepsilon}}} \in M_{H_0W_0,3H_0W_0}$ définies par :

$$D_{\widetilde{\nabla I}} = \begin{bmatrix} \widetilde{\nabla I}(\mathbf{x}_{11}^{\boldsymbol{\varepsilon}}) & & & & & & \\ & \ddots & & & & & \\ & & \widetilde{\nabla I}(\mathbf{x}_{H_01}^{\boldsymbol{\varepsilon}}) & & & & \\ & & & \widetilde{\nabla I}(\mathbf{x}_{12}^{\boldsymbol{\varepsilon}}) & & & \\ & & & & \ddots & & \\ & & & & & \widetilde{\nabla I}(\mathbf{x}_{H_0W_0}^{\boldsymbol{\varepsilon}}) \end{bmatrix} \tag{B.58}$$

$$D_{\Gamma_{k,\boldsymbol{\varepsilon}}} = \begin{bmatrix} \left(\frac{\partial exp_{\mathbf{SIM(2)}}}{\partial \varepsilon_k}(\boldsymbol{\varepsilon})\mathbf{x}_{11}^{\boldsymbol{\varepsilon}}\right)^T & & & \\ & \left(\frac{\partial exp_{\mathbf{SIM(2)}}}{\partial \varepsilon_k}(\boldsymbol{\varepsilon})\mathbf{x}_{21}^{\boldsymbol{\varepsilon}}\right)^T & & \\ & & \ddots & \\ & & & \left(\frac{\partial exp_{\mathbf{SIM(2)}}}{\partial \varepsilon_k}(\boldsymbol{\varepsilon})\mathbf{x}_{H_0W_0}^{\boldsymbol{\varepsilon}}\right)^T \end{bmatrix} \tag{B.59}$$

En empilant les colonnes de $\frac{\partial \phi_0}{\partial \varepsilon_k}$ et en utilisant (B.10) nous obtenons

$$Vec\left(\frac{\partial \phi_0}{\partial \epsilon_k}\right) = Vecd\left(D_{\Gamma_{k,\boldsymbol{\varepsilon}}} D_{\widetilde{\nabla I}}\right) \in \mathbb{R}^{H_0W_0} \tag{B.60}$$

Les propriétés (10) et (9) nous donnent

$$Vec\left(\frac{\partial \phi_0}{\partial \epsilon_k}\right) = R_{H_0W_0}^T \left(I_{H_0W_0} \otimes D_{\Gamma_{k,\boldsymbol{\varepsilon}}}\right) Vec\left(D_{\widetilde{\nabla I}}\right) \tag{B.61}$$

On peut pousser le développement de cette expression avec (B.23)

$$Vec\left(\frac{\partial \phi_0}{\partial \epsilon_k}\right) = R_{H_0W_0}^T \left(I_{H_0W_0} \otimes D_{\Gamma_{k,\boldsymbol{\varepsilon}}}\right) R_{H_0W_0,3} Vecd^b\left(D_{\widetilde{\nabla I}}\right) \tag{B.62}$$

Le terme $D_{\Gamma_{k,\boldsymbol{\varepsilon}}}$ peut être développé

$$D_{\Gamma_{k,\boldsymbol{\varepsilon}}} = D_x\left(I_{H_0W_0} \otimes \frac{\partial exp_{\mathbf{SIM(2)}}}{\partial \varepsilon_k}^T(\boldsymbol{\varepsilon})\right),\ D_x = \begin{bmatrix} \mathbf{x}_{11}^T & & \\ & \ddots & \\ & & \mathbf{x}_{H_0W_0}^T \end{bmatrix} \tag{B.63}$$

Nous définissons également $\hat{\nabla}\mathbf{I}$

$$\hat{\nabla}\mathbf{I}\left(\phi_0(\boldsymbol{\varepsilon})\right) = Vecd^b\left(D_{\widetilde{\nabla I}}\right) = \begin{bmatrix} \widetilde{\nabla I}(\mathbf{x}^{\boldsymbol{\varepsilon}}_{11}) \\ \widetilde{\nabla I}(\mathbf{x}^{\boldsymbol{\varepsilon}}_{21}) \\ \vdots \\ \widetilde{\nabla I}(\mathbf{x}^{\boldsymbol{\varepsilon}}_{H_0W_0}) \end{bmatrix} \tag{B.64}$$

Cela nous donne le résultat annoncé, à savoir :

$$Vec\left(\frac{\partial \phi_0}{\partial \epsilon_k}\right) = M_k(\boldsymbol{\varepsilon})\hat{\nabla}\mathbf{I}\left(\phi_0(\boldsymbol{\varepsilon})\right) \tag{B.65}$$

$$M_k(\boldsymbol{\varepsilon}) = R^T_{H_0W_0}\left(I_{H_0W_0} \otimes D_{\Gamma_{k,\boldsymbol{\varepsilon}}}\right)\left(R_{H_0W_0} \otimes I_3\right) \tag{B.66}$$

□

B.3.2 Preuve de la proposition 2, section 4.2.8

Démonstration. On considère un élément $\phi_l^{c,ij}$ de la couche de convolution. Celui-ci est le résultat du filtre de convolution W_l appliqué à la région $U_{ij} \subset \phi_{l-1}$ de taille $k_l \times k_l$ de la couche précédente centrée en ij. Cette région a pour expression (cf Définition 18)

$$U_{ij} = S_{\alpha_j}\phi_{l-1}S^T_{\beta_i} \tag{B.67}$$

$$\beta_i = \left\{i - \frac{k_l - 1}{2}, \ \ldots, \ i + \frac{k_l - 1}{2}\right\} \tag{B.68}$$

$$\alpha_j = \left\{j - \frac{k_l - 1}{2}, \ \ldots, \ j + \frac{k_l - 1}{2}\right\} \tag{B.69}$$

Nous avons alors

$$\phi_l^{c,ij} = Vec\left(W_l\right)^T Vec\left(U_{ij}\right) = Vec\left(W_l\right)^T Vec\left(S_{\alpha_j}\phi_{l-1}S^T_{\beta_i}\right) \tag{B.70}$$

Le théorème 9 permet de développer cette expression

$$\phi_l^{c,ij} = Vec\left(W_l\right)^T \left(S_{\beta_i} \otimes S_{\alpha_j}\right) Vec\left(\phi_{l-1}\right) \tag{B.71}$$

On peut désormais exprimer le gradient vectorisé :

$$\begin{aligned} Vec\left(\phi_l^c\right) &= \begin{bmatrix} Vec\left(W_l\right)^T \left(S_{\beta_1} \otimes S_{\alpha_1}\right) Vec\left(\phi_{l-1}\right) \\ Vec\left(W_l\right)^T \left(S_{\beta_2} \otimes S_{\alpha_1}\right) Vec\left(\phi_{l-1}\right) \\ \vdots \\ Vec\left(W_l\right)^T \left(S_{\beta_{W_l}} \otimes S_{\alpha_{H_l}}\right) Vec\left(\phi_{l-1}\right) \end{bmatrix} \\ &= \begin{bmatrix} Vec\left(W_l\right)^T & & \\ & \ddots & \\ & & Vec\left(W_l\right)^T \end{bmatrix} \begin{bmatrix} S_{\beta_1} \otimes S_{\alpha_1} \\ \vdots \\ S_{\beta_{W_l}} \otimes S_{\alpha_{H_l}} \end{bmatrix} Vec\left(\phi_{l-1}\right) \\ &= \left(I_{H_l W_l} \otimes Vec\left(W_l\right)^T\right) \left(\hat{S}_\beta * \hat{S}_\alpha\right) Vec\left(\phi_{l-1}\right) \\ &= \Omega_l Vec\left(\phi_{l-1}\right) \end{aligned} \tag{B.72}$$

où $*$ est le produit de Khatri-Rao (Définition 16). □

B.3.3 Preuve de la proposition 4, section 4.2.8

Démonstration. On déduit de (4.61) que

$$\frac{\partial \phi_l^{relu,ij}}{\partial \phi_l^{c,ij}} = \mathcal{H} \tag{B.73}$$

La règle de dérivation en chaîne sur (B.73) donne :

$$\begin{aligned} Vec\left(\frac{\partial \phi_l^{relu}}{\partial \varepsilon_k}\right) = \frac{\partial Vec\left(\phi_l^{relu}\right)}{\partial \varepsilon_k} &= \frac{\partial Vec\left(\phi_l^{relu}\right)}{\partial Vec\left(\phi_l^c\right)} \frac{\partial Vec\left(\phi_l^c\right)}{\partial \varepsilon_k} \\ &= \begin{bmatrix} \mathcal{H}\left(\phi_l^{c,11}\right) & & & \\ & \mathcal{H}\left(\phi_l^{c,21}\right) & & \\ & & \ddots & \\ & & & \mathcal{H}\left(\phi_l^{c,H_l W_l}\right) \end{bmatrix} \frac{\partial Vec\left(\phi_l^c\right)}{\partial \varepsilon_k} \\ &= \mathbf{H}_l \odot Vec\left(\frac{\partial \phi_l^c}{\partial \varepsilon_k}\right) \end{aligned} \tag{B.74}$$

avec $\mathbf{H}_l = \left[\mathcal{H}\left(\phi_l^{c,11}\right) \dots \mathcal{H}\left(\phi_l^{c,H_l W_l}\right)\right]^T$ □

B.3.4 Preuve de la proposition 5, section 4.2.8

Démonstration. On s'aperçoit que la somme dans le dénominateur (4.65) peut être exprimée à l'aide de matrices de permutation partielle. On pose $S^{lnr}_{\alpha_j},\ S^{lnr}_{\beta_i}$ avec $\alpha_j = [\![j - \frac{n_l-1}{2}, j + \frac{n_l-1}{2}]\!]$ et $\beta_i = [\![i - \frac{n_l-1}{2}, i + \frac{n_l-1}{2}]\!]$. On obtient alors une expression matricielle de la somme du dénominateur :

$$\sum_{p=i-\frac{n_l-1}{2}}^{i+\frac{n_l-1}{2}} \sum_{r=j-\frac{n_l-1}{2}}^{j+\frac{n_l-1}{2}} \phi_l^{relu,pr\,2} = Vec\left(S^{lnr}_{\alpha_j} \phi_l^{relu} S^{lnr\,T}_{\beta_i}\right)^T Vec\left(S^{lnr}_{\alpha_j} \phi_l^{relu} S^{lnr\,T}_{\beta_i}\right) \tag{B.75}$$

On remarque que

$$\frac{\partial \phi_l^{relu,ij}}{\partial Vec\left(\phi_l^{relu}\right)} = \mathbf{e}^T_{(j-1)W_l+i} \in \mathbb{R}^{H_l W_l} \tag{B.76}$$

où $\mathbf{e}_k$ est le $k^{\text{ème}}$ vecteur de la base canonique de $\mathbb{R}^{H_l W_l}$. Dans la suite, nous avons besoin du résultat ci-dessous, qui se prouve par du calcul matriciel élémentaire [PP+08] :

Lemme 1. *Soit* $u : \mathbb{R}^n \to \mathbb{R}^m$ *et* $a, b > 0$. *Alors*

$$\frac{\partial \left(1 + a\mathbf{u}^T\mathbf{u}\right)^b}{\partial x} = 2ab(1 + a\mathbf{u}^T\mathbf{u})^{b-1}\mathbf{u}^T\frac{\partial \mathbf{u}}{\partial x} \tag{B.77}$$

Ce lemme est utilisé ici pour dériver le dénominateur de l'équation (4.64) en remplaçant $a = \frac{\alpha_l}{n_l}, \beta = \beta_l$ et $\mathbf{u} = Vec\left(S^{lnr}_{\alpha_j} \phi_l^{relu} S^{lnr\,T}_{\beta_i}\right)$. En combinaison avec (9), on aboutit à

$$\frac{\partial \phi_l^{lnr,ij}}{\partial Vec\left(\phi_l^{relu}\right)} = \frac{e^T_{(j-1)W_l+i}}{d_{ij}^{\beta_l}} - 2\frac{\alpha_l}{n_l}\beta_l \frac{\phi_l^{relu,ij} Vec\left(\phi_l^{relu}\right)^T \left(S^{lnr}_{\beta_i} \otimes S^{lnr}_{\alpha_j}\right)^T \left(S^{lnr}_{\beta_i} \otimes S^{lnr}_{\alpha_j}\right)}{d_{ij}^{\beta_l+1}} \tag{B.78}$$

Définissons

$$K_{lnr}(\phi_l^{relu}) = \begin{bmatrix} \frac{1}{d_{11}} \\ \vdots \\ \frac{1}{d_{W_l H_l}} \end{bmatrix} \in \mathbb{R}^{W_l H_l} \tag{B.79}$$

et

$$\Lambda_\beta = \begin{bmatrix} S^{lnr\,T}_{\beta_1} S^{lnr}_{\beta_1} \\ \vdots \\ S^{lnr\ T}_{\beta_{W_l}} S^{lnr}_{\beta_{W_l}} \end{bmatrix},\ \Lambda_\alpha = \begin{bmatrix} S^{lnr\,T}_{\alpha_1} S^{lnr}_{\alpha_1} \\ \vdots \\ S^{lnr\ T}_{\alpha_{H_l}} S^{lnr}_{\alpha_{H_l}} \end{bmatrix} \tag{B.80}$$

Le théorème 5 et la puissance de Hadamard définie par (B.2) nous donnent

$$\begin{aligned}\frac{\partial Vec\left(\phi_l^{lnr}\right)}{\partial Vec\left(\phi_l^{relu}\right)} &= \mathbf{diag}\left(K_{lnr}^{\odot\beta_l}\right) - 2\frac{\alpha_l}{n_l}\beta_l\,\mathbf{diag}\left(Vec\left(\phi_l^{relu}\right)\odot K_{lnr}^{\odot\beta_l+1}\right)\\ &\quad\left(\left[I_{W_lH_l}\otimes Vec\left(\phi_l^{relu}\right)^T\right](\Lambda_\beta * \Lambda_\alpha)\right)\\ &= L_l \in M_{H_lW_l,H_lW_l}\end{aligned} \tag{B.81}$$

□

Bibliographie

[AAW+16] Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. Globally normalized transition-based neural networks. *arXiv preprint arXiv :1603.06042*, 2016.

[ABK+07] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. Dbpedia : A nucleus for a web of open data. In *The semantic web*, pages 722–735. Springer, 2007.

[ABS14] Umar Asif, Mohammed Bennamoun, and Ferdous Sohel. A model-free approach for the segmentation of unknown objects. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4914–4921. IEEE, 2014.

[APHS16] Zeynep Akata, Florent Perronnin, Zaid Harchaoui, and Cordelia Schmid. Label-embedding for image classification. *IEEE transactions on pattern analysis and machine intelligence*, 38(7) :1425–1438, 2016.

[ARDK16] Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Neural module networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 39–48, 2016.

[ASS+12] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Susstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(11) :2274–2282, 2012.

[AZK07] Zeyad Al Zhour and Adem Kiliçman. Some new connections between matrix products for partitioned and non-partitioned matrices. *Computers & Mathematics with Applications*, 54(6) :763–784, 2007.

[BBBB+15] Michael Beetz, Ferenc Bálint-Benczédi, Nico Blodow, Daniel Nyga, Thiemo Wiedemeyer, and Zoltán-Csaba Marton. Robosherlock : Unstructured information pro-

cessing for robot perception. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1549–1556. IEEE, 2015.

[BDZ18] Yohan Breux, Sebastien Druon, and Rene Zapata. From perception to semantics : An environment representation model based on human-robot interactions. In *27th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pages 672–677. IEEE, 2018.

[Bec04] Dave Beckett. Rdf/xml syntax specification. Technical report, W3C, 2004.

[BHB00] Christoph Bregler, Aaron Hertzmann, and Henning Biermann. Recovering non-rigid 3d shape from image streams. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, volume 2, pages 690–696, 2000.

[BHJ09] Mathieu Bastian, Sebastien Heymann, and Mathieu Jacomy. Gephi : An open source software for exploring and manipulating networks. *Icwsm*, 8 :361–362, 2009.

[BK88] Ann L Brown and Mary Jo Kane. Preschool children can learn to transfer : Learning to learn and learning from example. *Cognitive Psychology*, 20(4) :493–523, 1988.

[BM06] Sabine Buchholz and Erwin Marsi. Conll-x shared task on multilingual dependency parsing. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*, pages 149–164, 2006.

[Bra01] Ulrik Brandes. A faster algorithm for betweenness centrality. *Journal of mathematical sociology*, 25(2) :163–177, 2001.

[Bre01] Leo Breiman. Random forests. *Machine learning*, 45(1) :5–32, 2001.

[BRK06] David B Bracewell, Fuji Ren, and Shingo Kuroiwa. Towards knowledge about causal agents in wordnet. In *Proceedings of the 10th WSEAS international conference on Computers*, pages 564–568. World Scientific and Engineering Academy and Society (WSEAS), 2006.

[BSB+15] Ben Bellekens, Vincent Spruyt, Raf Berkvens, Rudi Penne, and Maarten Weyn. A benchmark survey of rigid 3d point cloud registration algorithms. *Int. J. Adv. Intell. Syst*, 8 :118–127, 2015.

[BTW15] Michael Beetz, Moritz Tenorth, and Jan Winkler. Open-ease. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1983–1990, 2015.

[CBK+10] Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka Jr., and Tom M. Mitchell. Toward an architecture for never-ending language learning.

In *Proceedings of the Twenty-Fourth Conference on Artificial Intelligence (AAAI)*, 2010.

[CCGS16] Wei-Lun Chao, Soravit Changpinyo, Boqing Gong, and Fei Sha. An empirical study and analysis of generalized zero-shot learning for object recognition in the wild. In *European Conference on Computer Vision*, pages 52–68. Springer, 2016.

[CK98] João Paulo Costeira and Takeo Kanade. A multibody factorization method for independently moving objects. *International Journal of Computer Vision*, 29(3) :159–179, 1998.

[CM14] Danqi Chen and Christopher Manning. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 740–750, 2014.

[CW16] Taco Cohen and Max Welling. Group equivariant convolutional networks. In *International Conference on Machine Learning*, pages 2990–2999, 2016.

[CXSG13] Chen, Xinlei, Shrivastava, and Gupta. Neil : Extracting visual knowledge from web data. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1409–1416, 2013.

[DB15] C Dharmani Bhaveshkumar. Multivariate generalized gram-charlier series in vector notations. *arXiv preprint arXiv :1503.03212*, 2015.

[DBM02] Marco De Boni and Suresh Manandhar. Automated discovery of telic relations for wordnet. In *Proceedings of the first International WordNet conference*, 2002.

[DDS+09] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet : A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[DFRDW11] Bertrand Douillard, Dieter Fox, F Ramos, and H Durrant-Whyte. Classification and semantic mapping of urban environments. *The international journal of robotics research*, 30(1) :5–32, 2011.

[DJV+13] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf : A deep convolutional activation feature for generic visual recognition. *arXiv preprint arXiv :1310.1531*, 2013.

[DMDS+14] Marie-Catherine De Marneffe, Timothy Dozat, Natalia Silveira, Katri Haverinen, Filip Ginter, Joakim Nivre, and Christopher D Manning. Universal stanford dependencies : A cross-linguistic typology. In *LREC*, volume 14, pages 4585–4592, 2014.

[DMM08] Marie-Catherine De Marneffe and Christopher D Manning. Stanford typed dependencies manual. Technical report, Technical report, Stanford University, 2008.

[DSS12] Amit Daniely, Sivan Sabato, and Shai S Shwartz. Multiclass learning approaches : A theoretical comparison with implications. In *Advances in Neural Information Processing Systems*, pages 485–493, 2012.

[DT05] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 886–893. IEEE, 2005.

[FB81] Martin A Fischler and Robert C Bolles. Random sample consensus : a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6) :381–395, 1981.

[FEHF09] Ali Farhadi, Ian Endres, Derek Hoiem, and David Forsyth. Describing objects by their attributes. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1778–1785. IEEE, 2009.

[FM12] Juan-Antonio Fernández-Madrigal. *Simultaneous Localization and Mapping for Mobile Robots : Introduction and Methods*. IGI Global, 2012.

[FRA11] Joao Fayad, Chris Russell, and Lourdes Agapito. Automated articulated structure and 3d shape recovery from point correspondences. In *IEEE International Conference on Computer Vision*, pages 431–438, 2011.

[Fre77] Linton C Freeman. A set of measures of centrality based on betweenness. *Sociometry*, pages 35–41, 1977.

[FVS09] Brian Fulkerson, Andrea Vedaldi, and Stefano Soatto. Class segmentation and object localization with superpixel neighborhoods. *ICCV*, 9 :670–677, 2009.

[FXKG15] Zhenyong Fu, Tao Xiang, Elyor Kodirov, and Shaogang Gong. Zero-shot object recognition by semantic manifold distance. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2635–2644, 2015.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[GDDM14] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[Gir15] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 1440–1448, 2015.

[Hal15] Brian Hall. *Lie groups, Lie algebras, and representations : an elementary introduction*, volume 222. Springer, 2015.

[HBJ+07] Sebastien Heymann, Mathieu Bastian, Mathieu Jacomy, Camille Maussang, Antonin Rohmer, Julian Bilcke, and Alexis Jacomy. Gexf file format. *GEXF Working Group,[online] c2009 [cit. 2013-05-07]. Available at WWW :< http ://gexf. net/format*, 2007.

[HBKM15] Lars Hertel, Erhardt Barth, Thomas Käster, and Thomas Martinetz. Deep convolutional neural networks as generic feature extractors. In *International Joint Conference on Neural Networks*, pages 1–4, 2015.

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8) :1735–1780, 1997.

[HSSC08] Aric Hagberg, Pieter Swart, and Daniel S Chult. Exploring network structure, dynamics, and function using networkx. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008.

[HV16] João F Henriques and Andrea Vedaldi. Warped convolutions : Efficient invariance to spatial transformations. *arXiv preprint arXiv :1609.04382*, 2016.

[JAP13] Bastien Jacquet, Roland Angst, and Marc Pollefeys. Articulated and restricted motion subspaces and their signatures. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1506–1513, 2013.

[JCD15] Hyung Jin Chang and Yiannis Demiris. Unsupervised learning of complex articulated kinematic structures combining motion and skeleton information. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3138–3146, 2015.

[JSD+14] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe : Convolutional architecture for fast feature embedding. *arXiv preprint arXiv :1408.5093*, 2014.

[JWB09] Dominik Jain, Stefan Waldherr, and Michael Beetz. Bayesian logic networks. *IAS Group, Fakultät für Informatik, Technische Universität München, Tech. Rep*, 2009.

[KFF15] Andrej Karpathy and Li Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3128–3137, 2015.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[KVD91] Jan J Koenderink and Andrea J Van Doorn. Affine structure from motion. *JOSA A*, 8(2) :377–385, 1991.

[LABG15] Spyridon Leonardos, Christine Allen-Blanchette, and Jean Gallier. The exponential map for the group of similarity transformations and applications to motion interpolation. In *IEEE International Conference on Robotics and Automation*, pages 377–382, 2015.

[Len95] Douglas B Lenat. Cyc : A large-scale investment in knowledge infrastructure. *Communications of the ACM*, 38(11) :33–38, 1995.

[LFHP14] Dagmar Lang, Susanne Friedmann, Marcel Häselich, and Dietrich Paulus. Definition of semantic maps for outdoor robotic tasks. In *IEEE International Conference on Robotics and Biomimetics*, pages 2547–2552, 2014.

[LFHP15] Dagmar Lang, Susanne Friedmann, Jens Hedrich, and Dietrich Paulus. Semantic mapping for mobile outdoor robots. In *14th IAPR International Conference on Machine Vision Applications*, pages 325–328, 2015.

[LLH+18] Changzhi Luo, Zhetao Li, Kaizhu Huang, Jiashi Feng, and Meng Wang. Zero-shot learning via attribute regression and class prototype rectification. *IEEE Transactions on Image Processing*, 27(2) :637–648, 2018.

[LNH09] Christoph H Lampert, Hannes Nickisch, and Stefan Harmeling. Learning to detect unseen object classes by between-class attribute transfer. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 951–958, 2009.

[Low99] David G Lowe. Object recognition from local scale-invariant features. In *Proceedings of the seventh IEEE international conference on Computer vision*, volume 2, pages 1150–1157, 1999.

[LP14] Dagmar Lang and Dietrich Paulus. Semantic maps for robotics. *Proc. of the Workshop "Workshop on AI Robotics" at ICRA*, 2014.

[LS14] H. Lynn Loomis and Shlomo Sternberg. *Advanced Calculus*. World Scientific, 2014.

[LSD15] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[LT08] Shuangzhe Liu and Gõtz Trenkler. Hadamard, khatri-rao, kronecker and other matrix products. *Int. J. Inf. Syst. Sci*, 4(1) :160–177, 2008.

[LV15] Karel Lenc and Andrea Vedaldi. Understanding image representations by measuring their equivariance and equivalence. In *IEEE conference on Computer Vision and Pattern Recognition*, 2015.

[LW14] Liu and Wichert. A generalizable knowledge framework for semantic indoor mapping based on markov logic networks and data driven mcmc. *Future Generation Computer Systems*, (36) :42–56, 2014.

[MAT17] Raul Mur-Artal and Juan D Tardós. Orb-slam2 : An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 33(5) :1255–1262, 2017.

[MC09] Manuel Marques and João Costeira. Estimating 3d shape from degenerate sequences with missing data. *Computer Vision and Image Understanding*, 113(2) :261–272, 2009.

[MCCD13] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv :1301.3781*, 2013.

[MCH+15] T. Mitchell, W. Cohen, E. Hruschka, P. Talukdar, J. Betteridge, A. Carlson, B. Dalvi, M. Gardner, B. Kisiel, J. Krishnamurthy, N. Lao, K. Mazaitis, T. Mohamed, N. Nakashole, E. Platanios, A. Ritter, M. Samadi, B. Settles, R. Wang, D. Wijaya, A. Gupta, X. Chen, A. Saparov, M. Greaves, and J. Welling. Never-ending learning. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[MCWD06] Cynthia Matuszek, John Cabral, Michael J Witbrock, and John DeOliveira. An introduction to the syntax and content of cyc. In *AAAI Spring Symposium : Formalizing and Compiling Background Knowledge and Its Applications to Knowledge Representation and Question Answering*, pages 44–49, 2006.

[MEP+15] Venkat S Malladi, Drew T Erickson, Nikhil R Podduturi, Laurence D Rowe, Esther T Chan, Jean M Davidson, Benjamin C Hitz, Marcus Ho, Brian T Lee, Stuart Miyasato, et al. Ontology application and use at the encode dcc. *Database*, 2015, 2015.

[MHYY15] Chao Ma, Jia-Bin Huang, Xiaokang Yang, and Ming-Hsuan Yang. Hierarchical convolutional features for visual tracking. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3074–3082, 2015.

[Mil95] George A Miller. Wordnet : a lexical database for english. *Communications of the ACM*, 38(11) :39–41, 1995.

[MMM+04] Frank Manola, Eric Miller, Brian McBride, et al. Rdf primer. *W3C recommendation*, 10(1-107) :6, 2004.

[MPSP+09] Boris Motik, Peter F Patel-Schneider, Bijan Parsia, Conrad Bock, Achille Fokoue, Peter Haase, Rinke Hoekstra, Ian Horrocks, Alan Ruttenberg, Uli Sattler, et al. Owl 2 web ontology language : Structural specification and functional-style syntax. *W3C recommendation*, 27(65) :159, 2009.

[MRF15] Mateusz Malinowski, Marcus Rohrbach, and Mario Fritz. Ask your neurons : A neural-based approach to answering questions about images. In *Proceedings of the IEEE international conference on computer vision*, pages 1–9, 2015.

[MV15] Aravindh Mahendran and Andrea Vedaldi. Understanding deep image representations by inverting them. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5188–5196, 2015.

[NBBB14] Daniel Nyga, Ferenc Balint-Benczedi, and Michael Beetz. Pr2 looking at things-ensemble learning for unstructured information processing with markov logic networks. In *ICRA*, pages 3916–3923. Citeseer, 2014.

[NDMG+16] Joakim Nivre, Marie-Catherine De Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D Manning, Ryan T McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, et al. Universal dependencies v1 : A multilingual treebank collection. In *LREC*, 2016.

[Nov02] Adrian Novischi. Accurate semantic annotations via pattern matching. In *FLAIRS Conference*, pages 375–379, 2002.

[PASW13] Jeremie Papon, Andrey Abramov, Markus Schoeler, and Florentin Worgotter. Voxel cloud connectivity segmentation-supervoxels for point clouds. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2027–2034, 2013.

[PBKE12] Kari Pulli, Anatoly Baksheev, Kirill Kornyakov, and Victor Eruhimov. Real-time computer vision with opencv. *Communications of the ACM*, 55(6) :61–69, 2012.

[PDM11] Slav Petrov, Dipanjan Das, and Ryan McDonald. A universal part-of-speech tagset. *arXiv preprint arXiv :1104.2086*, 2011.

[PJ11] Andrzej Pronobis and Patric Jensfelt. Hierarchical multi-modal place categorization. In *ECMR*, pages 159–164, 2011.

[PJ12] Andrzej Pronobis and Patric Jensfelt. Large-scale semantic mapping and reasoning with heterogeneous modalities. In *IEEE International Conference on Robotics and Automation*, pages 3515–3522, 2012.

[PLR+16] Ben Poole, Subhaneil Lahiri, Maithra Raghu, Jascha Sohl-Dickstein, and Surya Ganguli. Exponential expressivity in deep neural networks through transient chaos. In *Advances in neural information processing systems*, pages 3360–3368, 2016.

[PM18] Judea Pearl and Dana Mackenzie. *The Book of Why : The New Science of Cause and Effect*. Basic Books, 2018.

[PP+08] Kaare Brandt Petersen, Michael Syskind Pedersen, et al. The matrix cookbook. *Technical University of Denmark*, 7 :15, 2008.

[PSM14] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove : Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–43, 2014.

[RASC14] Ali S Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf : an astounding baseline for recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 802–813, 2014.

[RBB09] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms (fpfh) for 3d registration. *IEEE International Conference on Robotics and Automation*, pages 3212–3217, 2009.

[RC11] Radu Bogdan Rusu and Steve Cousins. 3D is here : Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, 2011.

[RD06] Richardson and Domingos. Markov logic networks. *Machine Learning*, 62 :107–136, 2006.

[RHGS15] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn : Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[RKB04] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. Grabcut : Interactive foreground extraction using iterated graph cuts. In *ACM transactions on graphics (TOG)*, volume 23, pages 309–314. ACM, 2004.

[RPSM17] Santhosh K Ramakrishnan, Ambar Pal, Gaurav Sharma, and Anurag Mittal. An empirical evaluation of visual question answering for novel objects. *arXiv preprint arXiv :1704.02516*, 2017.

[RPT15] Bernardino Romera-Paredes and Philip Torr. An embarrassingly simple approach to zero-shot learning. In *International Conference on Machine Learning*, pages 2152–2161, 2015.

[RTDM+15] Luis Riazuelo, Moritz Tenorth, Daniel Di Marco, Marta Salas, Dorian Gálvez-López, Lorenz Mösenlechner, Lars Kunze, Michael Beetz, Juan D Tardós, Luis Montano, et al. Roboearth semantic mapping : A cloud enabled knowledge-based approach. *IEEE Transactions on Automation Science and Engineering*, 12(2) :432–443, 2015.

[RTZ10] David A Ross, Daniel Tarlow, and Richard S Zemel. Learning articulated structure and motion. *International Journal of Computer Vision*, 88(2) :214–237, 2010.

[San90] Beatrice Santorini. Part-of-speech tagging guidelines for the penn treebank project (3rd revision). *Technical Reports (CIS)*, page 570, 1990.

[SBWK10] Agnes Swadzba, Niklas Beuter, Sven Wachsmuth, and Franz Kummert. Dynamic 3d scene analysis for acquiring articulated scene models. *IEEE International Conference on Robotics and Automation*, pages 134–141, 2010.

[SDM+16] Niko Sünderhauf, Feras Dayoub, Sean McMahon, Ben Talbot, Ruth Schulz, Peter Corke, Gordon Wyeth, Ben Upcroft, and Michael Milford. Place categorization and semantic mapping on a mobile robot. In *IEEE International Conference on Robotics and Automation*, pages 5729–5736, 2016.

[SEZ+13] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. Overfeat : Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv :1312.6229*, 2013.

[SFL17] Carina Silberer, Vittorio Ferrari, and Mirella Lapata. Visually grounded meaning representations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[SH12] Robert Speer and Catherine Havasi. Representing general relational knowledge in conceptnet 5. In *LREC*, pages 3679–3686, 2012.

[SLJ+15] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich, et al. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[SM00] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(8) :888–905, 2000.

[SP97] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11) :2673–2681, 1997.

[SPL+16] Niko Sünderhauf, Trung-Thanh Pham, Yasir Latif, Michael Milford, and Ian D. Reid. Meaningful maps - object-oriented semantic mapping. *CoRR*, 2016.

[SPW15] Markus Schoeler, Jeremie Papon, and Florentin Wörgötter. Constrained planar cuts-object partitioning for point clouds. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5207–5215, 2015.

[SSG+17] Iulian V Serban, Chinnadhurai Sankar, Mathieu Germain, Saizheng Zhang, Zhouhan Lin, Sandeep Subramanian, Taesup Kim, Michael Pieper, Sarath Chandar, Nan Rosemary Ke, et al. A deep reinforcement learning chatbot. *arXiv preprint arXiv :1709.02349*, 2017.

[SSPW14] Simon Christoph Stein, Markus Schoeler, Jeremie Papon, and Florentin Worgotter. Object partitioning using local convexity. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 304–311, 2014.

[SVZ13] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks : Visualising image classification models and saliency maps. *arXiv preprint arXiv :1312.6034*, 2013.

[SZ14] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv :1409.1556*, 2014.

[SZS+13] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv :1312.6199*, 2013.

[TB17] Moritz Tenorth and Michael Beetz. Representations for robot knowledge in the knowrob framework. *Artificial Intelligence*, 247 :151–169, 2017.

[TBF05] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. MIT Press, Cambridge, 2005.

[Ten11] Moritz Tenorth. *Knowledge Processing for Autonomous Robots*. PhD thesis, Technische Universität München, 2011.

[Ter03] Gy Terdik. Higher order statistics and multivaxiate vector hermite polynomials for nonlinear analysis of multidimensional time series. *Theory of Probability and Mathematical Statistics*, (66) :165, 2003.

[TK92] Carlo Tomasi and Takeo Kanade. Shape and motion from image streams under orthography : a factorization method. *International Journal of Computer Vision*, 9(2) :137–154, 1992.

[TSSW15] Minija Tamosiunaite, Rahel Magdalena Sutterlütti, Simon Christoph Stein, and Florentin Wörgötter. Perceptual influence of elementary three-dimensional geometry :(2) fundamental object parts. *Frontiers in psychology*, 6 :1427, 2015.

[Tur48] Alan M Turing. Intelligent machinery, a heretical theory. *The Turing test : Verbal behavior as the hallmark of intelligence*, 105, 1948.

[Tur95] Alan M Turing. Computing machinery and intelligence. *Brian Physiology and Psychology*, page 213, 1995.

[ÜEHR14] Andre Ückermann, Christof Eibrechter, Robert Haschke, and Helge Ritter. Real-time hierarchical scene segmentation and classification. In *14th IEEE-RAS International Conference on Humanoid Robots*, pages 225–231, 2014.

[ÜHR13] Andre Ückermann, Robert Haschke, and Helge Ritter. Realtime 3d segmentation for human-robot interaction. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2136–2143, 2013.

[VBM10] Olga Veksler, Yuri Boykov, and Paria Mehrani. Superpixels and supervoxels in an energy optimization framework. *European conference on Computer vision*, pages 211–224, 2010.

[WGB12] David Weikersdorfer, David Gossow, and Michael Beetz. Depth-adaptive superpixels. *ICPR*, pages 2087–2090, 2012.

[Wil12] Anthony Williams. *C++ concurrency in action : practical multithreading*. Manning Publ., Shelter Island, NY, 2012.

[WLSM+15] Thomas Whelan, Stefan Leutenegger, Renato F Salas-Moreno, Ben Glocker, and Andrew J Davison. Elasticfusion : Dense slam without a pose graph. *Proc. Robotics : Science and Systems, Rome, Italy*, 2015.

[WLY13] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. Online object tracking : A benchmark. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[WS08] Denis F Wolf and Gaurav S Sukhatme. Semantic mapping using mobile robots. *IEEE Transactions on Robotics*, 24(2) :245–258, 2008.

[WST15] F. Worgotter, R. Sutterlutti, and M. Tamosiunaite. Perceptual influence of elementary three-dimensional geometry : (1) objectness. *Frontiers in Psychology*, 6 :1317, 2015.

[WSTL12] Jan Wielemaker, Tom Schrijvers, Markus Triska, and Torbjörn Lager. SWI-Prolog. *Theory and Practice of Logic Programming*, 12(1-2) :67–96, 2012.

[WWS+15] Peng Wang, Qi Wu, Chunhua Shen, Anton van den Hengel, and Anthony Dick. Explicit knowledge-based reasoning for visual question answering. *arXiv preprint arXiv :1511.02570*, 2015.

[XCK06] Jing Xiao, Jinxiang Chai, and Takeo Kanade. A closed-form solution to non-rigid shape and motion recovery. *International Journal of Computer Vision*, 67(2) :233–246, 2006.

[YP05] Jingyu Yan and M. Pollefeys. A factorization-based approach to articulated motion recovery. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2005.

[YP08] Jingyu Yan and Marc Pollefeys. A factorization-based approach for articulated non-rigid shape, motion and kinematic chain recovery from video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(5) :865–877, 2008.

[Zie15] Leon Ziegler. *The Attentive Robot Companion : Learning Spatial Information from Observation and Verbal Interaction*. PhD thesis, Bielefeld University, 2015.

[ZLS09] Luca Zappella, Xavier Llado, and Joaquim Salvi. New trends in motion segmentation. In *Pattern Recognition*. InTech, 2009.

[ZXG16] Li Zhang, Tao Xiang, and Shaogang Gong. Learning a deep embedding model for zero-shot learning. *arXiv preprint arXiv :1611.05088*, 2016.