Construction of sparse dictionary for visual scene recognition

Claire Béranger[1], Nicolas Cuperlier, Sylvain Colomer

CY Université, ENSEA, ETIS UMR 8051, Cergy-Pontoise, France
claire.beranger@ensea.fr

***Abstract*— Les systèmes de localisation robotiques actuels demandent un coût computationnel assez grand, limitant leur utilisation à des espaces de taille réduite. L'utilisation de Sparse coding pourrait permettre de résoudre ces problématiques en améliorant l'encodage des informations visuelles. Des expériences ont montré qu'il est possible de diviser le coût computationnel de modèle de Visual Place Recognition via des mécanismes de Sparse coding, imitant le fonctionnement du cortex visuel. Dans ce papier, nous étudions d'autres architectures de sparsification des informations visuelles, afin d'améliorer les résultats obtenus. De premières expériences avec l'algorithme Sparse Hmax ont été réalisées, notamment sur des données telles que KITTI, et des points d'intérêts particuliers de ces images (POIs). De premiers résultats ont pu être observés, notamment les activités résultantes de l'application de la classification de Sparse Hmax sur des POIs. Un premier constat, est que l'algorithme Sparse Hmax semble permettre par sa classification, de reconnaître des types de POIs. En effet, pour un même type d'objet, on obtient le même type d'activité. Tout en ayant une activité unique à chaque POIs. L'objectif qui suit ces premières expériences, est de continuer à observer plus en détails ces résultats, puis d'observer en terme de coûts computationnels et mémoires, l'intérêt de Sparse Hmax. Enfin, de proposer des améliorations possibles pour obtenir un algorithme des plus efficaces pour répondre à notre problématique.**

Mots clés : Sparse coding, visual place recognition, autonomous navigation, bio-inspired robotic

I. Introduction

La réalisation de tâche de localisation visuelle sur des véhicules autonomes nécessite de mettre en place des architectures complexes de traitement des informations. Une solution couramment utilisée est l'approche SLAM pour Simultaneous Localization And Mapping. Le SLAM est un algorithme d'intelligence artificielle permettant aux robots de cartographier leur environnement et de s'y localiser sans avoir une connaissance préalable des obstacles. Ces méthodes souffrent cependant de contraintes computationnelles, et il devient difficile de pouvoir couvrir de grands espaces.

L'utilisation de modèles bio-inspirés semble apporter un réel avantage, car cette notion intègre des choses moins fixées, donc plus de souplesses algorithmiquement. Plusieurs modèles basés sur le fonctionnement hippocampique ont été proposés et étudiés, notamment en s'appuyant sur le fonctionnement des aires V1 et V2 [2] du cerveau de mammifères. Ces aires ont des caractéristiques très importantes dans le fonctionnement de la vision. Leur étude permet la compréhension des mécanismes établis par le cerveau, pour transformer des informations visuelles en informations à réutiliser plus tard dans n'importe quel contexte, tout ceci, en ne consommant que très peu en terme de coût computationnel et mémoire. Ces études ont montré que le cerveau fonctionne notamment par l'usage de neurones. Chacun de ces neurones étant stimulés pour des actions différentes et sont activés de manière plus ou moins complexes. Ces approches sont très efficaces pour réaliser des tâches de localisation dans des environnements maîtrisés. Cependant, cette méthode n'est pas toujours efficace, et applicable dans un contexte réel de navigation, car elle est très coûteuse et n'est pas temps-réel. Enfin, les tests ont été réalisés dans des conditions restreintes qui ne nous permettent pas de connaître la réaction de ce système dans des conditions météorologiques complexes [4].

Dans le cadre de ce projet, le fonctionnement de la vision dans le cerveau humain au niveau du cortex cérébral[17] va être la base de nos études. Notamment, le fonctionnement de l'aire V1 fortement impliqué dans ce domaine[2]. Rolls et Tovee ont réalisé des études sur des Macaques dans le but de comprendre comment les informations visuelles telles que des objets ou des images de visage sont représentés par l'activité des neurones corticaux temporels [18]. L'expérience a montré qu'à partir de 68 stimuli, seul un neurone est utilisé. Les diverses observations effectuées, en plus de cette expérience, montrent qu'une représentation sparse des données est réalisée. Ces recherches nous intéressent, car cela signifierait que nous pourrions, à l'aide des techniques d'intelligence artificielle, trouver un moyen de s'inspirer de ce comportement Sparse. Cela permettrait d'une part, de rendre la mise en oeuvre de la navigation de manière moins coûteuse qu'avec un algorithme classique, mais également de calculer plus rapidement les activités à mettre en place pour naviguer convenablement dans un espace, aussi bien connu ou inconnu pour le véhicule.

Dans ce papier, nous étudions comment les informations visuelles peuvent être encodées, via des mécanismes de Sparse coding, dans le but d'être utilisées dans des systèmes de localisation. Des travaux sur l'encodage d'informations

[1]Étudiante à l'Université de Cergy-Pontoise, ENSEA, Projet au laboratoire ETIS UMR 8051, France. Encadrée par : Nicolas CUPERLIER, Sylvain COLOMER

visuelles, via des mécanismes Sparse dans l'objectif d'être intégré dans des systèmes de localisation ont déjà été réalisés. La problématique essentielle de notre projet est donc plus précisément, dans ce contexte, d'étudier une autre architecture, le Sparse Hmax, pour identifier s'il peut être utilisé pour récupérer les informations essentielles d'une scène visuelle de manière Sparse. Puis, d'en déduire ces avantages et inconvénients, ainsi que les améliorations qu'il peut apporter par rapport au modèle en cours d'étude. Tout d'abord, le principal objectif va être de trouver des solutions pour diminuer les coûts mémoires et computationnels, que les algorithmes envisagés à mettre en oeuvre engendrent. Un autre axe est d'arriver à comprendre un modèle d'algorithme applicable dans ce contexte et de définir son efficacité vis-à-vis des travaux actuellement menés. A savoir le taux de compression à y appliquer si besoin, ou encore si le résultat en navigation obtenu est correct. L'objectif final sera de déterminer ce qui est à récupérer de l'algorithme Sparse Hmax pour améliorer le modèle en cours d'étude parallèlement à ces travaux.

Afin de répondre au mieux à ces différentes problématiques que compose notre sujet, nous proposerons une explication du Sparse coding dans sa globalité à partir de nos recherches bibliographiques. Ensuite, nous nous attarderons sur les différents algorithmes sparse applicables dans notre contexte, puis sur la solution Sparse Hmax qui nous intéresse. La partie Matériels et Méthodes présentera les outils utilisés pour nos premières expériences mises en oeuvre. Ensuite, la partie Résultats exposera nos premiers résultats intéressants pour la continuité de ce projet. Nous verrons vers quoi ces résultats nous ouvrent pour la suite dans la partie Discussion. Enfin, nous ferons l'analyse globale de l'état actuel du projet.

II. État de l'art

A. Généralités sur le Sparse Coding

Le Sparse Coding est une méthodologie inspirée du comportement Sparse observé dans le cerveau de mammifères[18]. Notamment impacté par l'aire visuelle V1[2]. Les neurones situés dans l'aire visuelle V1 sont sélectifs pour un certain nombre d'attributs, comprenant entre autre l'orientation, la direction du mouvement, la fréquence spatiale et temporelle. Le comportement de cette aire qui est intéressant pour ce projet et qui retient notre attention est sa gestion de division de l'image pour pouvoir la réinterpréter à tout moment et y effectuer un apprentissage. L'aire V1 possède un comportement Sparse pouvant être représenté tel que dans la figure 1.

On y voit nettement la notion de décomposition de l'image en différents mots composant ainsi un dictionnaire. Et ensuite, pour pouvoir en traduire une activité, l'ensemble de ces mots sont sommés en utilisant les poids d'apprentissage des neurones. Notre objectif étant de s'inspirer de ce phénomène pour pouvoir utiliser un dictionnaire d'images, pour qu'un

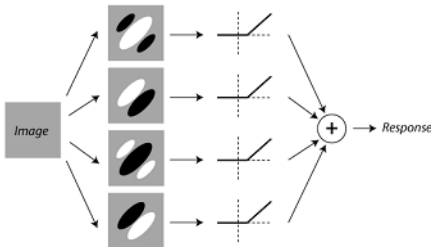

Fig. 1: Complex cells descriptive model

véhicule autonome puisse apprendre à se repérer. Cela montre l'intérêt de comprendre comment encoder ces informations visuelles d'un point de vue algorithmique. Il n'y a pas de définition unique de ce processus, mais il se base sur un principe unique, le Sparse coding est un modèle génératif linéaire. En effet, le Sparse coding est souvent formulé comme un problème d'optimisation[14], [9], [6]. D'après Olshausen [13], il est même construit par une fonction de coût. On a une image I définie par l'équation 1.

$$I(x, y) = \sum_i a_i \theta_i \tag{1}$$

Pour l'apprentissage, la norme est essentielle, pour cela le minimum de 0 dans les données possédées est recherché. Plus simplement, le Sparse coding rassemble deux types de codes [22] : local et dense. En effet, tout comme le code local, le Sparse coding stocke une information sur un neurone. Mais, pour un gain de temps, tout comme le code dense, les données sont liées grâce à la notion de cellules grands-mères. D'où le fait qu'il est possible de résumer le Sparse coding par une simple équation. La recherche d'obtention de moins d'éléments égaux à zéros parmi un maximum de données de base peut se traduire par l'équation 2.

$$\min \|a_i\|_0 \text{ subject to } s = \sum_i a_i \theta_i(x) \tag{2}$$

La norme l_0 ainsi définie permet d'obtenir les éléments de base essentiels pour l'apprentissage de type Sparse. L'équation obtenue permet de se rapprocher de la description du phénomène Sparse observée. La définition du problème ainsi donnée permet de s'appuyer sur ce résultat comme d'autres recherches ont pu le faire également[3], [16].

B. Méthodes de Sparse coding

De nombreuses recherches ont été réalisées sur l'applicabilité du Sparse coding dans de nombreux domaines[6], [10]. Des types de Sparse coding peuvent ainsi être constitués, et il est possible de les classer par familles de codes. Parmi ces différents algorithmes, on trouve trois types de familles. Les 4 algorithmes qui semblent intéressants dans notre contexte sont les suivants :

1) K-Singular Value Decomposition (K-SVD) algorithm
2) Sparse Hmax algorithm
3) K-Sparse Autoencoders algorithm
4) Local Competition Algorithm (LCA)

Algorithme	Avantages	Inconvénients
K-SVD	Permet de généraliser le problème du point de vue K-clustering	Limitation au niveau des données d'entrées
K-Sparse Autoencoder	Algorithme très rapide et capable de traiter des données de grandes tailles	Algorithme très efficace si pré-entraîné
Sparse Hmax	Décompose la problématique en 4 étapes distinctes, reconnaissance de visages	Manque de tests sur d'autres types d'images/objets
LCA	Très pratique pour les problèmes utilisant des neurones en parallèle, notamment pour appliquer à des vidéos	Manque de détails biophysiques pour des tests expérimentaux complets

TABLE I: Tableau récapitulatif de quelques algorithmes Sparse

L'algorithme K-SVD [1] est une méthode permettant de généraliser le processus de clustering K-means[7]. Cette méthode consiste à être itérative en alternant entre :

1) Sparse Coding d'exemples basés sur le dictionnaire courant
2) Mise à jour des atomes du dictionnaire pour mieux s'adapter aux données

Cet algorithme est flexible et peut travailler avec n'importe quelle méthode de poursuite. L'algorithme K-SVD possède une résolution Sparse définissable par :

$$(P_0) \min_X \|X\|_0 \text{ subject to } y = DX \tag{3}$$

$$(P_{0,\epsilon}) \min_X \|X\|_0 \text{ subject to } \|y - DX\|_2 \leq \epsilon \tag{4}$$

D étant le dictionnaire utilisé et $y = y_i{}_{i=1}^{N}$. Cet algorithme se caractérise par sa flexibilité, sa simplicité d'implémentation et son efficience. Il permet le remplissage des images et la compression, et semble avoir des avantages vis-à-vis d'autres méthodes classiques tel que Haar. Cependant, cet algorithme peut très vite être limité par les images d'entrée pouvant être prises en compte. En effet, les images doivent être synthétiques et réelles.

Une technique plus classique, mais reprenant les mêmes principes de calcul Sparse que la problématique du K-SVD est le K-Sparse Autoencoder [11]. Ce type d'algorithme a été réalisé dans l'objectif que l'étape d'encodage soit rapide d'exécution. Le k-Sparse Autoencoder est un auto-encodeur utilisant une fonction d'activation linéaire. Dans les couches cachées, seules les k activités les plus élévées sont conservées. Ce type de résolution Sparse a été testé sur des données de type MNIST, et il a été constaté que les résultats étaient meilleurs que ceux d'une classification d'auto-encodeurs de débruitage. Les k-Sparse auto-encoders sont simples à entraîner et très rapide, ce qui les rend bien adaptés à des problèmes utilisant des données de plus grandes tailles. Toutefois, les résultats de l'article [11] montrent que la base d'entraînement est 6 fois supérieure à la base de test et que l'algorithme est plus efficace s'il effectue un pré-entraînement.

L'algorithme Sparse Hmax suit le principe de l'algorithme Hmax[21]. L'algorithme Hmax est notamment appliqué pour la reconnaissance d'objet et de texture. Cette méthode décompose la problématique de décomposition d'information des images à l'aide de 4 étapes distinctes. En effet, le Sparse Hmax [8] utilise les 4 méthodes de l'algorithme Hmax, à savoir : S1 => C1 => S2 => C2. Où S1, correspond à l'application d'un filtre de Gabor, C1 récupère les maxima locaux. S2 prend en compte la distance euclidienne contenue entre les images. Enfin, C2 prend les maxima locaux. Pour que l'algorithme devienne Sparse, une des solutions est de modifier l'étape S2 en Sparse coding, permettant l'application du Sparse Hmax. Une solution s'appuyant sur ces résultats a été testée montrant qu'il est possible d'appliquer le Sparse Hmax pour encoder des images efficacement à la manière Sparse[8]. Par contre, nous ne connaissons pas l'ensemble de ces limites exactes. Et notamment, dans quel cadre les résultats obtenus peuvent être réutilisés. Les résultats obtenus [8] étaient uniquement dans le but de reconnaître des visages, mais nous ne savons pas ce qui se passe si on appliquait cet algorithme à d'autres types d'éléments.

Enfin, l'algorithme LCA [20], [5], [15], [19] qui consiste à résoudre une famille de problèmes d'approximation Sparse. Cet algorithme est appliqué dans des systèmes dynamiques composés de nombreux éléments de type neurone fonctionnant en parallèle. Cette solution s'est avérée avoir du potentiel notamment pour l'analyse d'image de vidéos. Cet algorithme est économe en énergie et semble très intéressant à appliquer dans notre contexte. Cependant, cette méthode ressort des séquences de coefficients erratiques difficiles à interpréter. De plus, les tests et capacités sont limités. En effet, cet algorithme semble fonctionner convenablement sur des architectures neuronales connues, mais elles manquent de détails biophysiques nécessaires pour pouvoir être testé expérimentalement. Ces résultats peuvent par contre être utilisés dans le cadre de reconnaissance de formes ou encore dans le suivi d'objets[20], [19]. Ces applications sont très importantes et pourraient répondre à nos attentes d'encoder des images dans le but de se localiser dans l'espace de navigation d'un véhicule.

En résumé, l'ensemble de ces algorithmes peuvent être regroupés et classés selon différents critères. En effet, d'une part l'ensemble de ces algorithmes travaillent tous sur la thématique du Sparse coding. Toutefois, la méthodologie est différente. D'une part, l'algorithme du K-SVD et le K-Sparse autoencoder font tous deux partie d'une même famille, à savoir du traitement Sparse dans le domaine des K-means. D'autre part, le LCA et le Sparse Hmax traite tous deux la problématique du suivi d'objets et reconnaissance d'objets à la manière Sparse. Cependant, leurs méthodes et hypothèses sont trop distinctes pour dire qu'ils forment une famille du Sparse coding. Chacun de ces algorithmes apporte autant d'avantage que d'inconvénient, bien qu'ils se ressemblent fortement pour le K-SVD et le k-autoencoder, ou au contraire qu'ils soient différents pour le Sparse Hmax et le LCA. Toutefois, l'un de ces algorithmes semble mieux répondre à notre contexte qui est de tester une méthode nouvelle pour encoder des images pour interprétation pour un véhicule en navigation. En effet, l'algorithme Sparse Hmax retient notre attention de par sa méthode de décomposition de calcul du problème et de par sa proposition de résultats.

C. Code Sparse Hmax

L'algorithme qui retient notre attention est le Sparse Hmax [8]. Xiaolin Hu a effectué des recherches sur ce sujet et a réalisé des tests dans le cadre de reconnaissance de visages à l'aide d'un code Sparse. L'algorithme Hmax avait été étudié [21] dans le cadre de reconnaissance d'objets en suivant le mécanisme établi par le cortex visuel. Ces deux caractéristiques qui composent le Sparse Hmax, concernant la reconnaissance d'objets et un algorithme basé sur le fonctionnement du cortex visuel intéresse particulièrement.

Le mécanisme suivi par le Sparse Hmax consiste en 4 étapes importantes. La combinaison de ces étapes, correspondante chacune à une couche de neurones, permet d'obtenir à la fin un dictionnaire d'images. Il se compose de deux couches S et de deux couches C et une couche d'unité ajustées à la manière de la hiérarchie de cellules simples à complexes que l'on peut retrouver dans l'aire V1. Entre chacune des couches, deux paramètres entrent en ligne de compte : l'inférence et le max pooling, essentiels pour un apprentissage cohérent. Et où S1, correspond à l'application d'un filtre de Gabor, C1 récupère les maxima locaux sur des positions et échelles sur les résultats de S1. S2 applique un filtre de type L2 RBF avec N patchs dépendant du résultat de C1. Enfin, C2 prend les maxima locaux du résultat sortant de S2 en les associant au patch donné.

Le plus simple pour représenter le rôle de chacune de ces couches est de représenter schématiquement la problématique. Un algorithme Hmax classique [21] peut donc être représenté à la manière de la figure 2.

Tandis que le travail réalisé par Xiaolin Hu, consiste en une structure légèrement plus complexe à savoir l'utilisation de trois couches distinctes au lieu de deux. Son travail peut être représenté tel que la figure 3. Cette figure montre nettement les étapes suivies par Xiaolin Hu [8]. Où il a été

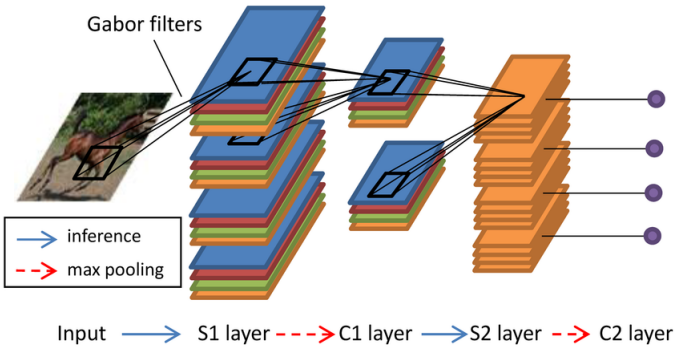

Fig. 2: Représentation de Hmax

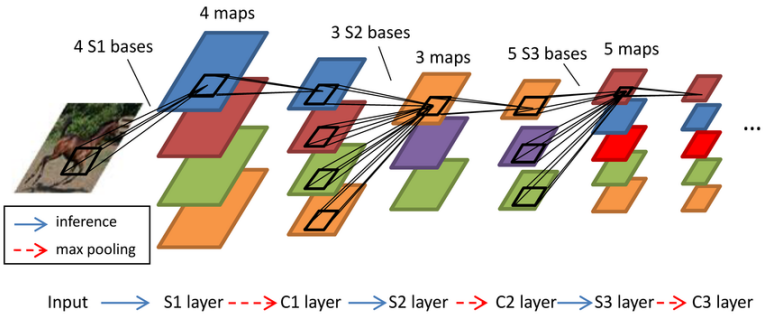

Fig. 3: Représentation de Sparse Hmax

décidé que les bases S1, S2 et S3 soient apprises par filtres SSC ou ICA. L'avantage de SSC étant qu'il peut apprendre sur des bases de grandes tailles. ICA a pour avantage de déduire des cartes de caractéristiques comme pour le Hmax classique.

L'approche adoptée pour la classification des images est différente de l'algorithme Hmax. En effet, au lieu que chaque base aléatoire S conduise à une seule caractérisitque C pour une image, la mise en place du "spatial pyramid max pooling" est utilisé. De plus, contrairement au Hmax original, les codes S obtenus par le codage clairsemé ainsi mis en place peuvent être positifs ou négatifs.

III. Matériels et Méthodes

Nous avons utilisé le Sparse Hmax de Xiaolin Hu [8] pour voir si l'exécution des programmes de ses recherches donnent des résultats satisfaisants. Notamment, pour la bonne compréhension et prise en main, en vue de l'appliquer dans notre problématique d'encoder des images de manière Sparse. Tout ceci en gardant l'objectif d'un résultat satisfaisant d'un point de vue temps de calcul et méthode de classification du dictionnaire.

A. Matériels utilisés pour les expériences

Xiaolin Hu propose deux types d'implémentation du Sparse Hmax [8] respectivement nommés code V1 et code V2. Dans son premier algorithme, les images d'entrée prises en compte sont basées sur des images naturelles de référence. Dans son second algorithme, les mêmes étapes de Sparse Hmax sont présentes et implémentées sur le même principe

mais cette fois-ci l'algorithme ne travaille que sur la base où on cherche à réaliser une classification. Un choix de paramétrage supplémentaire est possible, à savoir soit on suit la méthode ICA pour classification soit SC pour Sparse Coding. Nous utiliserons pour le moment le paramètre ICA.

La machine utilisée durant l'ensemble de ce premier processus est : DELL Latitude 5590, Intel(R) Core(TM) i5-82500 CPU (1.80 GHz), Mémoire RAM 4.0 Go.

B. Données d'entrée utilisées pour les codes V1 et V2

Plusieurs types de données en entrée de l'algorithme Sparse-Hmax sont utilisées. Au total, 3 types d'entrée sont utilisées pour les deux types d'écriture de l'algorithme proposé par Xiaolin Hu.

Tout d'abord, les séquences d'images de kyoto et de caltech101 sont récupérées pour vérification que l'algorithme V1 fonctionne et tourne correctement sur la machine utilisée pour les tests. Dans un second temps, une séquence Ktest contenant les 10 premières images de KITTI00 est utilisée. Ceci dans le but d'observer la possible adaptation du code pour faire tourner le code V1 sur KITTI00. Enfin, les séquences complètes de KITTI00, afin d'observer les résultats pouvant être obtenus avec cet algorithme. Les datasets ainsi utilisés pour tester l'algorithme V1 de Xiaolin Hu, sur différents types de données, sont résumés dans le tableau II.

Type de test	**Natural Data**	**category**
Séquences originales	Kyoto	caltech101
Adaptation pour KITTI	Kyoto	Ktest
Essais sur KITTI	Kyoto	KITTI00
Data pour tentative d'amélioration du résultat	Ktest	KITTI00

TABLE II: Données utilisées pour le code V1

De la même façon, plusieurs types de données en entrée de l'algorithme Hmax sont utilisés pour le code V2. Au total, 3 types d'entrée sont utilisés sur le code V2 de l'algorithme proposé par Xiaolin Hu.

Tout d'abord, les séquences d'images de caltech101 sont récupérées pour vérification que l'algorithme V2 fonctionne et tourne correctement sur la machine utilisée pour les tests.

Dans un second temps une séquence Ktest contenant les 10 premières images de KITTI00 est utilisée. Ceci dans le but d'observer la possible adaptation du code pour faire tourner le code V2 sur KITTI00. Enfin, les séquences complètes de KITTI00, afin d'observer les résultats pouvant être obtenus avec cet algorithme.

Les datasets ainsi utilisés pour tester l'algorithme V2 de Xiaolin Hu sur différents types de données sont résumés dans le tableau III.

Type de test	**category**
Séquences originales	caltech101
Adaptation pour KITTI	Ktest
Essais sur KITTI	KITTI00

TABLE III: Données utilisées pour le code V2

C. Méthodes suivies pour les codes V1 et V2

La méthode pour ces deux algorithmes est d'abord de choisir les images données en entrée. Puis, d'observer ce que chaque étape S ressort. Enfin, de regarder le résultat final obtenu. Le processus de tests se décompose finalement selon les étapes suivantes :

1) Tests des deux programmes Matlab de Xiaolin Hu
2) Faire fonctionner Sparse Hmax sur les données fournies avec le code
3) Utiliser le code avec KITTI
4) Appliquer les résultats obtenus sur les POIs : obtention d'activités

Les paramètres utilisés pour les différentes phases d'apprentissage S1, S2, S3 sont laissés pour le moment tel que proposé par Xiaolin Hu. Cependant pour le code V2, de manière à observer l'activité des neurones au moins au niveau de la première couche de l'algorithme, on modifiera patchsz1 = 10 à patchsz1 = 54. Ceci dans le but de travailler sur des tailles d'images plus cohérentes avec nos recherches.

IV. RÉSULTATS

A. Résultats obtenus avec le code V1

L'ensemble des expériences réalisées ci-après sont des expériences effectuées sur le code Matlab V1 récupéré des travaux de Xiaolin Hu [8] sur l'algorithme Sparse Hmax (cf www.xlhu.cn/software/shmax.html)

L'objectif actuel est de prendre en main la version proposée dans le cadre des travaux de l'article [8], puis de voir dans quelles mesures les résultats sont utilisables pour l'objectif de ce projet.

Résultats avec les données originales Faces_easy

On procède à de légères modifications du code fourni pour pouvoir faire tourner le programme sur machine. L'objectif étant de faire fonctionner le code sur l'expérience initiale pour visualiser le fonctionnement du code et les résultats obtenus.

Les paramètres en entrées sont les images naturelles du dossier "kyoto", qui sont des images diverses de paysages et des images pour l'apprentissage contenues dans le dossier caltech101/Faces_easy, ces images correspondant à des photographies de différents visages avec des expressions et luminosités différentes.

Après avoir fait tourner le programme on obtient trois images de sortie représentant respectivement chacune les sorties en couche S1, S2 et S3 en figure 4..

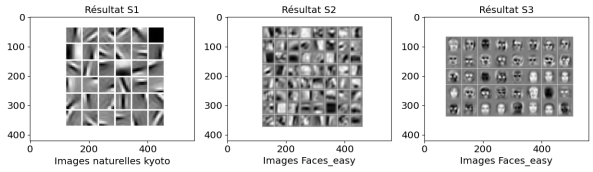

Fig. 4: Résultat avec Faces easy

Le résultat montre que le code fonctionne et ressort les résultats présentés par Xiaolin Hu. On observe sur la figure

4 en phase S1, le premier Gabor appliqué sur les images naturelles en entrée, en phase S2 le Gabor complexe et enfin en phase S3 on reconnaît des visages. Il faut tester sur les données de KITTI pour observer les résultats obtenus suite à l'apprentissage.

Résultats avec KITTI00

De nouvelles modifications sont réalisées pour pouvoir faire tourner le programme fourni sur des données KITTI00. Ces données correspondantes aux images de caméra placée sur un véhicule naviguant dans un quartier. Le nouvel objectif est de réussir à faire fonctionner le code sur les images de KITTI00 pour visualiser le fonctionnement et les résultats obtenus. Ceci, dans le but de savoir si le code est applicable sur KITTI00.

Les entrées utilisées sont, en image naturelle "kyoto" et en image d'entrée pour l'apprentissage KITTI00.

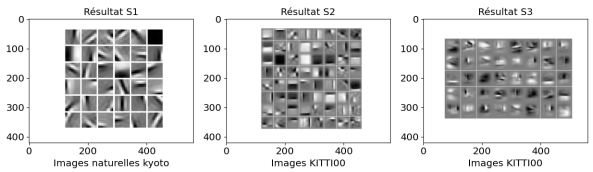

Fig. 5: Résultat avec Kitti00

Des résultats sont ressortis, le code semble applicable à KITTI00. En effet, avec ce premier algorithme, les résultats 5 en phase S3 sont très flous, mais on peut commencer à reconnaître des types de lieux. En modifiant les images d'apprentissage, il faudrait tester par exemple sur les premières données de KITTI, pour observer les résultats obtenus.

Résultats avec images naturelles et d'apprentissage Ktest

Pour ce test, dans la fonction sampleimages on laisse les rand de manière à observer le comportement avec un apprentissage sur des images choisies aléatoirement.

Les entrées sont, d'une part en image naturelle "Ktest" et d'autre part en image d'entrée pour l'apprentissage : Ktest (dossier contenant les 10 premières images de KITTI00).

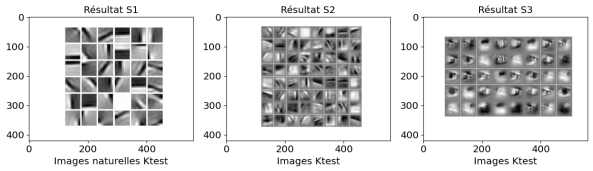

Fig. 6: Résultat avec images naturelles et d'apprentissage Ktest

En prenant des images aléatoires qui se basent sur des images naturelles Ktest, on observe un résultat beaucoup plus net. En effet, on peut constater en figure 6 que les axes routiers sont plus visibles, on sur-apprend un modèle de paysage. On peut tester avec KITTI00 en remettant le paramètre de random dans la fonction sampleimages pour observer les résultats obtenus suite à l'apprentissage avec Ktest en images naturelles.

Résultats avec KITTI00 avec images naturelles Ktest

Le but de ce nouveau test est de réaliser le code sur les 10 premières images de KITTI00 pour savoir dans quel cadre le code est applicable sur KITTI00. Cette fois-ci, dans la fonction sampleimages on laisse les rand de manière à observer le comportement avec un apprentissage sur des images aléatoires.

Les entrées sont maintenant : en image naturelle, "Ktest" et en image d'entrée pour l'apprentissage : KITTI00

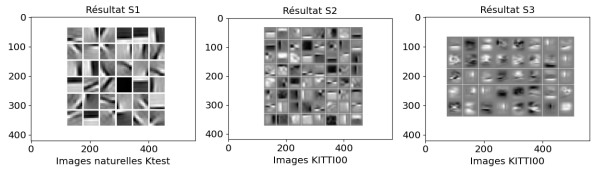

Fig. 7: Résultat avec KITTI00 avec images naturelles Ktest

En prenant pour images naturelles les 10 premières images de KITTI00, on obtient un résultat légèrement différent que celui obtenu avec kyoto. En effet, l'image résultante met en avant des éléments tels que des arbres ou autre. On a une apparition de scènes visuelles reconnaissables.

B. Résultats obtenus avec le code V2

L'ensemble des expériences réalisées ci-après sont des expériences effectuées sur le code Matlab V2 récupéré des travaux de Xiaolin Hu sur l'algorithme Sparse Hmax (cf www.xlhu.cn/software/shmax.html) L'objectif actuel est de prendre en main la seconde version proposée dans le cadre des travaux de l'article [8], puis de voir dans quelles mesures les résultats sont utilisables pour l'objectif de ce projet.

Pour l'ensemble des tests de cette seconde version de l'algorithme, le paramètre du type d'algorithme utilisé est ICA. [12]

Résultats avec Faces_easy

On procède à de légères modifications du code fourni pour pouvoir faire tourner cette seconde version sur machine. L'objectif étant de faire fonctionner le code sur l'expérience initiale pour visualiser le fonctionnement du code et les résultats obtenus.

Les paramètres en entrées sont les images pour l'apprentissage contenues dans le dossier caltech101/Faces_easy, ces images correspondant à des photographies de visages différents avec des expressions et luminosités différentes.

Après avoir fait tourner le programme on obtient trois images de sortie représentant respectivement chacune les sorties en couche S1, S2 et S3.

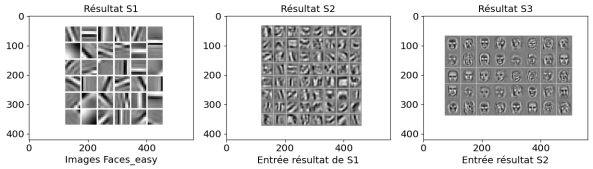

Fig. 8: Résultat avec Easy Faces

Le résultat 8 montre que le code fonctionne et ressort les résultats présentés par Xiaolin Hu. Cette seconde méthode permet un résultat plus net, les visages sont plus reconnaissables que dans la version 1 du code. Avec 10 mots pour la version 1, on avait déjà une précision forte, mais avec la version 2, on obtient un résultat nettement plus précis encore. Il est maintenant intéressant de tester sur les données de KITTI pour observer les résultats obtenus suite à l'apprentissage.

Résultats avec Ktest

Avant de réaliser un test trop gros au vu de la taille des données il est nécessaire de faire fonctionner le code sur les 10 premières images de KITTI pour voir la faisabilité et le fonctionnement du code dans notre cas d'utilisation.

Les images d'entrée pour l'apprentissage sont contenues dans Ktest/Kitti, correspondant aux 10 premières images de KITTI.

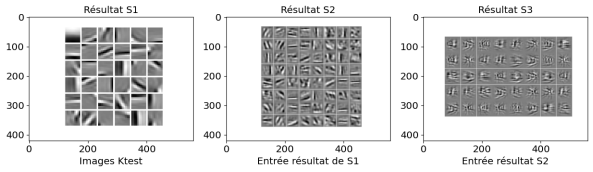

Fig. 9: Résultat avec Ktest

Un ensemble de résultats ressort, le code semble fonctionner et utilisable pour KITTI. On observe déjà une amélioration dans la reconnaissance de scènes visuelles dans le résultat en phase S3 de la figure 9.

Résultats avec KITTI00

L'objectif de ce test est de faire tourner le code sur l'ensemble des données KITTI à disposition pour visualiser le fonctionnement et les résultats obtenus.

Les images d'entrée pour l'apprentissage sont donc KITTI00/Kitti, soit l'ensemble des images KITTI00 à notre disposition.

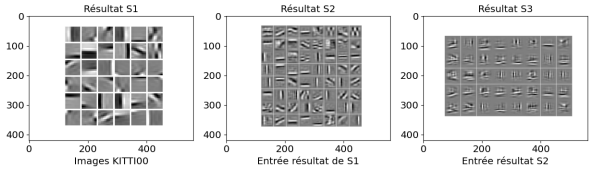

Fig. 10: Résultat avec KITTI00

Cette deuxième version est bien utilisable pour nos tests futurs sur KITTI00. On peut observer en figure 10 la nette amélioration par rapport à la version 1 où le résultat ici présent est moins flou, et les scènes visuelles nettement plus visibles.

Résultat obtenu avec un patch de 54x54

La seconde version semble plus prometteuse sur les résultats obtenus en sortie. C'est pourquoi, le paramètre patchsz1 est modifié en entrée de l'algorithme. En effet, en modifiant ce paramètre, il est possible de travailler sur des images de tailles 54x54 au lieu de 10x10 comme jusqu'à maintenant pour la couche S1.

En réalisant cette modification, cela permet d'appliquer les poids résultants de l'algorithme aux POIs, points d'intérêts pré-répertoriés pour les images KITTI00, pour ensuite pouvoir analyser les activités ainsi obtenues sur nos différentes couches.

La couche 1 obtenue avec ce paramétrage est celle présentée en figure 11.

Fig. 11: Résultat avec KITTI00 pour patchsz1 = 54

En appliquant ce poids obtenu en S1 aux POIs pour une première intuition, on obtient par exemple les activités en figure 12 pour des POIs entre 50 et 55.

Cette application de vignettes aux champs récepteurs, soit le dictionnaire préalablement obtenu, nous permet de lire des activités qui semblent différentes entre deux activités. Enfin, en regardant les graphiques de certaines vignettes en fonction du poids pour des cas particuliers, on obtient les résultats qui suivent.

Pour deux vignettes qui se ressemblent, mais qui sont fortement translater l'une par rapport à l'autre, on obtient les deux graphiques d'activité suivants.

Pour la première vignette testée, on a la figure 13.

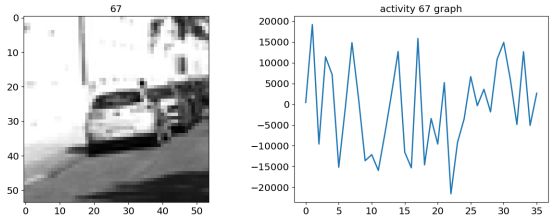

Fig. 13: Activités pour la couche S1 en fonction de la vignette 67

Pour la seconde vignette ressemblant dans les détails présents, mais avec un point de vue légèrement translaté, on obtient la figure 14.

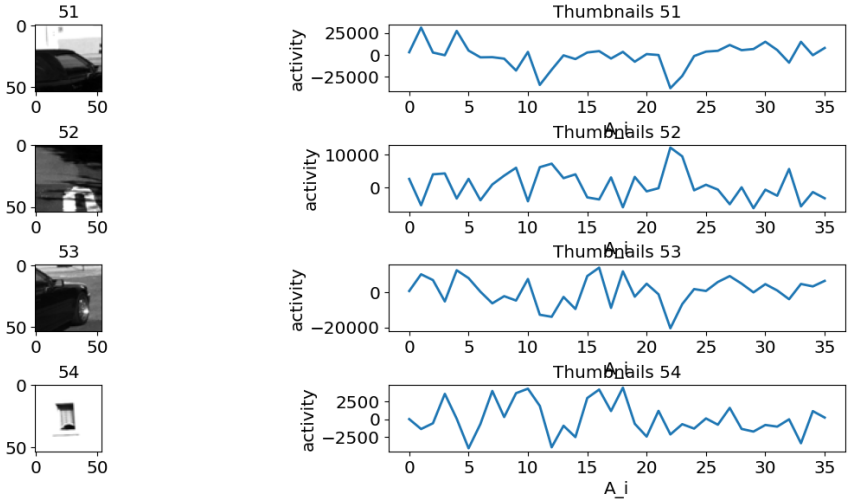

Fig. 12: Activités pour la couche S1 en fonction des vignettes de départ

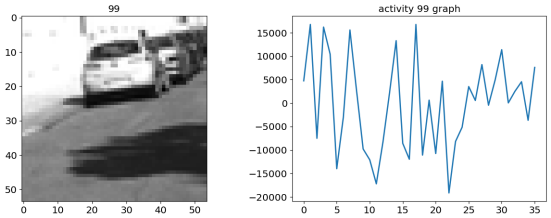

Fig. 14: Activités pour la couche S1 en fonction de la vignette 99

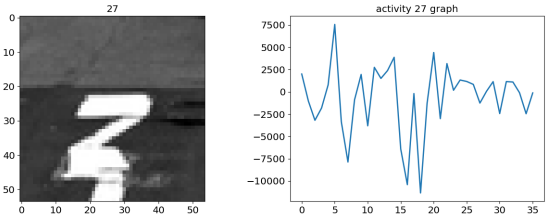

Fig. 16: Activités pour la couche S1 en fonction de la vignette 27

On constate que pour les vignettes 67 et 99, qui se ressemblent très fortement visuellement, que leurs activités respectives se ressemblent en terme de pics présents sur les graphiques. Tout en restant, cependant, différentes pour l'une et l'autre vignette.

Pour deux vignettes contenant deux informations distinctes, on obtient les figures 15 et 16.

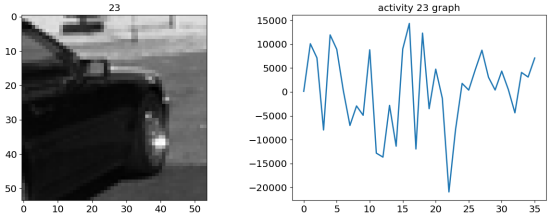

Fig. 15: Activités pour la couche S1 en fonction de la vignette 23

Contrairement aux vignettes 67 et 99, qui se ressemblaient fortement visuellement, les vignettes 23 et 27 sont très distinctes. On peut observer que de ce fait leurs graphiques d'activités respectives sont également très différents.

V. Discussion

Les résultats effectués suite aux recherches réalisées par Xiaolin Hu [8], nous permettent de développer de premières intuitions amenant à différentes analyses et ouvertures pour la suite du projet. L'analyse de ces résultats nous permettra d'établir les prochains objectifs.

A. Conclusions établies pour le code V1

Les expériences pour le code V1 nous ont montré diverses choses. Tout d'abord, en figure 4, les mêmes résultats que ceux testés dans l'article [8] sont convenablement identifiables. Il est possible de constater que la dernière couche obtenue ressort de manière nette des visages appris. De la même manière, nous avons obtenu des résultats pour KITTI. Le résultat obtenu en figure 5, montre qu'il est tout à fait envisageable de tester cet algorithme sur des images de navigation. Le résultat premièrement obtenu sans changement de paramètre met en avant que le résultat en S3 est flou, mais il est possible de distinguer des axes de route. Finalement, en reprenant la prise en compte aléatoire du programme, on observe que des points ressortent très fortement. En effet, sur la figure 6 il est possible de distinguer nettement des lieux et détails à l'oeil nu. Comme un arbre par exemple. Cela montre qu'il est tout à fait possible que ce programme nous permette d'identifier des points d'intérêts plus forts que d'autres. Et par la suite d'analyser s'il est envisageable de penser notre problématique différemment. C'est pourquoi, le test réalisé ensuite est sur l'ensemble des données KITTI00. La figure 7 montre les trois couches obtenues suite à un apprentissage basé sur les 10 premières

images de KITTI. Certains mots du dictionnaire semblent plus nets.

Le code V1 nous a permis de visualiser l'intérêt de la classification avec Sparse Hmax pour des données de navigation telles que KITTI. L'inconvénient est le fait qu'il se base sur une base de données d'images naturelles, ce qui pourrait causer des problématiques de lieu non ou mal appris. Notamment, cela impliquerait de savoir comment bien choisir une banque d'images naturelles pour travailler correctement sur notre base KITTI. Et par la suite, sur n'importe quelle image d'entrée provenant d'une navigation quelconque. Le code V2 fonctionne différemment sur cet aspect, puisqu'il prend directement en compte l'ensemble de la base de données sur laquelle nous travaillons.

B. Conclusions établies pour le code v2

Les expériences pour le code V2 apportent des améliorations visuelles par rapport au code V1. Malgré leur similitude algorithmique, le fait que le code V2 prenne en compte la base de données complète dès la première phase d'apprentissage, cela semble améliorer le résultat final de la phase S3. Tout d'abord, en figure 8, sont ressortis les mêmes résultats que ceux testés dans l'article [8] et ils sont convenablement identifiables. Il est possible de constater que la dernière couche obtenue ressort de manière nette des visages appris. La netteté obtenue est plus forte que celle obtenue avec le code V1. Ce qui est cohérent avec l'implémentation du code. En effet, cette fois-ci il n'y a pas d'images naturelles prises en compte pour la couche S1. L'algorithme ne travaille qu'avec les données d'apprentissage et ses couches d'apprentissage successives. Pour vérifier que cet algorithme peut être appliqué dans notre contexte de navigation. Le test avec les 10 premières images de KITTI est réalisé en figure 9, le résultat qui en ressort est très intéressant. Une fois de plus on obtient une image plutôt nette. On peut tout à fait y observer des rues et penser reconnaître des détails alentours sur les images de sortie.

Finalement, en testant avec l'ensemble de la base KITTI, on obtient en figure 10 une image certes un peu floue, mais où on peut parmi les mots du dictionnaire obtenu, encore distinguer des rues et imaginer des détails alentours. Le dernier test qui a pu être réalisé est la mise en application du résultat obtenu sur la couche S1 du code pour un patchsz1 = 54. 11 En mettant en application ce code sur les POIs à notre disposition sous Python, on obtient des activités graphiques 12. On constate sur ce premier graphique résultat, que pour des vignettes successives différentes les unes des autres, les activités obtenues sont uniques. On obtient donc un type d'activité par vignette.

Ensuite, les graphiques 13 et 14 sont cohérents avec nos attentes. Les champs récepteurs obtenus sont cohérents puisqu'ils semblent similaires pour des vignettes comportant des mêmes types d'objets. Tout en étant unique pour chacune des deux vignettes. Il est donc peut-être probable que cet algorithme permette de décrire des types d'objets. Et donc de catégoriser des types de lieux dans une navigation. Des tests supplémentaires devront être réalisés dans cet optique. Enfin, les graphiques 15 et 16 sont deux vignettes très distinctes l'une de l'autre. On peut de nouveau constater à l'aide de ces graphiques que les activités sorties suite à l'algorithme Sparse Hmax sont bien uniques pour chaque vignette.

Ces résultats semblent nous confirmer qu'une approche plus précise est nécessaire et nous ouvre vers une analyse plus précise de l'algorithme Sparse Hmax.

C. Applications à venir suite aux travaux réalisés

Suite aux différents travaux réalisés et à l'ouverture que nous donne notre dernier test avec le code V2, l'objectif est d'arriver à faire tourner l'algorithme sur les phases suivantes S2 et S3 de Sparse Hmax. C'est-à-dire, plus loin que la couche 1 avec patchsz1 = 54 pour observer les résultats obtenus. Ensuite, d'en analyser les activités des différents neurones et si l'algorithme est suffisamment robuste vis-à-vis notamment de translation d'objets dans l'image. Par exemple, si un arbre est un peu plus à gauche qu'habituellement, est-ce que le véhicule pourra interpréter l'arbre comme un objet connu de son dictionnaire ou est-ce que l'apprentissage ne sera pas suffisamment robuste à ce niveau.

Un autre objectif sera également d'analyser les capacités de l'algorithme en terme de coût mémoire et de temps de calcul. Notamment, observer les possibilités d'optimisation. Ou si l'algorithme a des limites. Ou encore s'il y a des solutions pour réguler cela. C'est-à-dire, quel est le coût de l'algorithme en lui-même et comment le réduire le cas échéant dans notre contexte. Dans quelle mesure le code peut-il être compressé et dans quel cadre le résultat obtenu nous assurera de donner une navigation correcte.

La prise en main de l'algorithme sera dans tous les cas à revoir et reprendre pour trouver une solution optimale. Mais surtout qui réponde bien au problème d'encoder des informations visuelles, via des mécanismes de Sparse coding, dans le but d'être utilisé dans des systèmes de localisation. Et finalement, les apports de cet algorithme vis-à-vis de la solution existante en cours d'étude, et l'intérêt de son intégration dans cette solution.

La finalité serait de pouvoir tester l'algorithme obtenu dans un cas de navigation concret.

VI. CONCLUSION

L'ensemble des travaux réalisés, nous montre que l'étude bibliographique a permis de conclure sur une solution potentiellement viable. Notamment, pour la problématique majeure de ce projet qui consiste à étudier l'intérêt d'une nouvelle architecture de sparsification, pour encoder des informations visuelles via des mécanismes de Sparse coding, dans le but d'être utilisé dans des systèmes de localisation. L'algorithme Sparse Hmax est étudié dans cet optique et de premiers essais ont été réalisés, afin de comprendre son fonctionnement et son application dans ce contexte. Les premières analyses des activités semblent s'ouvrir vers la

suite des objectifs. Notamment, l'étude des coûts mémoire et temps de calcul. Mais également vers l'analyse de l'application de cet algorithme en contexte de navigation. Ceci, pour pouvoir ensuite envisager l'intégration potentielle de cette solution à une architecture en cours d'étude. Durant la suite de ce projet, il sera intéressant de voir les possibilités qu'offrent cette solution pour la problématique donnée.

REFERENCES

[1] Michal Aharon, Michael Elad, and Alfred Bruckstein. K-SVD: An Algorithm for Designing Overcomplete Dictionaries for Sparse Representation. *IEEE TRANSACTIONS ON SIGNAL PROCESSING*, 54(11):12, 2006.

[2] M. Carandini. Area V1. *Scholarpedia*, 7(7):12105, 2012. revision #137292.

[3] Matthew Chalk, Olivier Marre, and Gašper Tkačik. Toward a unified theory of efficient, predictive, and sparse coding. *Proceedings of the National Academy of Sciences*, 115(1):186–191, January 2018.

[4] Yoan Espada, Nicolas Cuperlier, Guillaume Bresson, and Olivier Romain. Application of a Bio-inspired Localization Model to Autonomous Vehicles. In *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, pages 7–14, Singapore, November 2018. IEEE.

[5] Kaitlin L. Fair, Daniel R. Mendat, Andreas G. Andreou, Christopher J. Rozell, Justin Romberg, and David V. Anderson. Sparse coding using the locally competitive algorithm on the truenorth neurosynaptic system. *Frontiers in Neuroscience*, 13:754, 2019.

[6] P. Foldiak and D. Endres. Sparse coding. *Scholarpedia*, 3(1):2984, 2008. revision #145589.

[7] Cyprien Gilet, Marie Deprez, Jean-Baptiste Caillau, and Michel Barlaud. Clustering avec sélection de variables par minimisation alternée et application à la biologie. page 3, 2018.

[8] Xiaolin Hu, Jianwei Zhang, Jianmin Li, and Bo Zhang. Sparsity-Regularized HMAX for Visual Recognition. *PLoS ONE*, 9(1):e81813, January 2014.

[9] Koray Kavukcuoglu, Marc'Aurelio Ranzato, and Yann LeCun. Fast Inference in Sparse Coding Algorithms with Applications to Object Recognition. *arXiv:1010.3467 [cs]*, October 2010. arXiv: 1010.3467.

[10] Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro. Online dictionary learning for sparse coding. In *Proceedings of the 26th Annual International Conference on Machine Learning - ICML '09*, pages 1–8, Montreal, Quebec, Canada, 2009. ACM Press.

[11] Alireza Makhzani and Brendan Frey. k-Sparse Autoencoders. *arXiv:1312.5663 [cs]*, March 2014. arXiv: 1312.5663.

[12] Rubén Martín-Clemente and Susana Hornillo-Mellado. Image processing using ICA: a new perspective. page 4.

[13] Bruno A. Olshausen and David J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607–609, June 1996.

[14] Bruno A. Olshausen and David J. Field. Sparse coding with an overcomplete basis set: A strategy employed by V1? *Vision Research*, 37(23):3311–3325, December 1997.

[15] Dylan M Paiton. Analysis and applications of the Locally Competitive Algorithm. page 99.

[16] Laurent U. Perrinet. Role of Homeostasis in Learning Sparse Representations. *Neural Computation*, 22(7):1812–1836, July 2010.

[17] T. Poggio and T. Serre. Models of visual cortex. *Scholarpedia*, 8(4):3516, 2013. revision #149958.

[18] E. T. Rolls and M. J. Tovee. Sparseness of the neuronal representation of stimuli in the primate temporal visual cortex. *Journal of Neurophysiology*, 73(2):713–726, February 1995.

[19] C. Rozell, D. Johnson, R. Baraniuk, and B. Olshausen. Locally competitive algorithms for sparse approximation. 4:IV – 169–IV – 172, 2007.

[20] Christopher J. Rozell, Don H. Johnson, Richard G. Baraniuk, and Bruno A. Olshausen. Sparse Coding via Thresholding and Local Competition in Neural Circuits. *Neural Computation*, 20(10):2526–2563, October 2008.

[21] Thomas Serre, Lior Wolf, Stanley Bileschi, Maximilian Riesenhuber, and Tomaso Poggio. Robust Object Recognition with Cortex-Like Mechanisms. *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, 29(3):16, 2007.

[22] Anton Spanne and Henrik Jörntell. Questioning the role of sparse coding in the brain. *Trends in Neurosciences*, 38(7):417–427, July 2015.