INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

PFE

N° d'ordre : XXX

Projet de fin d'études
présenté par
Fernando Lucas ARAUJO AMARAL
Elève Ingénieur de l'INSA Rennes
Spécialité EII
Année scolaire 2019-2020

ETUDE, ADAPTATION ET IMPLEMENTATION DE METHODES DE SEPARATION DE SOURCES POUR LE CODAGE DE VIDEOS HOLOGRAPHIQUES

Lieu du Projet de Fin d'Études
IRT b<>com
Tuteur du Projet de Fin d'Études
Patrick GIOIA
Correspondant pédagogique INSA
Luce MORIN

PFE soutenu le 24/08/2020

b<>com

Résumé

Ce rapport présente le stage de fin d'études que j'ai effectué durant ma dernière année au département d'électronique et informatique industrielle de l'INSA Rennes. Ce stage s'est déroulé à l'IRT b<>com et était encadré par Patrick GIOIA, mon tuteur du projet de fin d´études, et Luce MORIN, mon correspondant pédagogique de l'INSA, au sein du laboratoire *Advanced Media Content.*

L'holographie est une technique désirable pour la visualisation 3D, une fois que les techniques plus répandues actuellement, comme la stéréoscopie, ont des limitations et des problèmes. L'implémentation de l'holographie pour visualisation 3D est encore un travail récent, donc il y a toujours des études en cours de développement, où il s'agit d'une tâche pas facile et avec beaucoup des contraintes.

Le sujet de mon stage était le « étude, adaptation et implémentation de méthodes de séparation de sources pour le codage de vidéos holographiques ». Au sein d'une équipe, j'ai participé au projet HARDY, qui vise développer un prototype de casque holographique miniaturisé et alimenté en temps réel.

Avec le déroulement du stage, on a observé des difficultés avec le développement de la mission initialement proposée, de cette façon d'autres techniques et approches ont été implémentés et étudiés, évidemment encore sur le domaine des hologrammes numériques.

Les autres approches développées sont l'implémentation et l'étude des techniques de localisation des sources via une segmentation de la scène 3D et l'implémentation des réseaux de neurones pour résoudre les problèmes de classification et régression, que visent à identifier le nombre de sources ponctuelles des hologrammes et leurs positions dans la scène, respectivement.

Parallèlement à ces études, la distribution de Wigner, une représentation espace-fréquence, a été étudié et testé pour vérifier sa pertinence comme outil de représentation des hologrammes numériques.

Mots-clés : algorithme-de-régression ; distribution-de-Wigner ; holographie-numérique ; localisation-des-sources ; réseaux-de-neurones ; séparation-aveugle-des-sources.

Abstract

This report presents the internship that I did during my last academic year in the department of electronics and computer engineering at INSA Rennes. This internship took place at the IRT b<>com and was supervised by Patrick GIOIA, my end-of-studies project tutor, and Luce MORIN, my INSA pedagogical correspondent, in the Advanced Media Content laboratory.

Holography is a desirable technique for 3D visualization, once the more widespread techniques currently used, such as stereoscopy, have limitations and problems. The implementation of holography for 3D visualization is still a recent work, so there are still studies under development, where it is not an easy task with many constraints.

The subject of my internship was the "study, adaptation and implementation of source separation methods for holographic video coding". As part of a team, I participated in the HARDY project, which aims to develop a prototype of a miniaturized real-time powered holographic headset.

With the course of the internship, we observed difficulties with the development of the initially proposed mission, in this way, other techniques and approaches were implemented and studied, obviously still in the field of digital holograms.

The other approaches developed are the implementation and study of source localization techniques via 3D scene segmentation and the implementation of neural networks to solve classification and regression problems, which aim at identifying the number of point sources of holograms and their positions in the scene, respectively.

In parallel to these studies, the Wigner distribution, a space-frequency representation, was studied and tested to verify its relevance as a tool for the representation of digital holograms.

Keywords: blind-source-separation; computer-generated-holography; neural-networks; regression-algorithm; source-localization; Wigner-distribution.

Remerciements

En premier lieu, je tiens à remercier Patrick GIOIA, en tant que mon maître de stage, pour sa confiance, sa patience et ses connaissances qu'il a su partager avec moi.

Je remercie ensuite tout l'équipe HARDY avec qui j'ai le plaisir de travailler, tout le personnel de b<>com pour son accueil et en particulièrement le personnel du laboratoire de « Nouveaux Contenu Média ».

Je désire aussi remercier les professeurs de l'INSA Rennes, qui m'ont fourni les outils nécessaires au bon déroulement du stage et je remercie particulièrement Luce MORIN, mon encadrant de stage.

Table des matières

Liste des figures

Liste de tableaux

Liste d'équations

Glossaire

Terme	Définition
API	*Application Programming Interface*
BSS	*Blind Source Separation*
CNN	*Convolution Neural Network*
CUDA	*Compute Unified Device Architecture*
DW	Distribution de Wigner
ICA	*Independent Component Analysis*
IRT	Institut de Recherche Technologique
HARDY	*Holographic Augmented Reality DisplaY*
HD	*High Definition*
MAD	Mise à Disposition
MLP	*Multilayer Perceptron*
NMF	*Non-negative Matrix Factorization*
PME	Petites et Moyennes Entreprises
RNN	*Recurrent Neural Network*
SAS	Séparation Aveugle des Sources
TWD	Transformation de Wigner-Ville
WD	*Wigner Distribution*
2D	Deux Dimensions
3D	Trois Dimensions

I – Introduction

1. Présentation

1.1. L'IRT b<>com

L'IRT b<>com est un centre d'innovation privé fournisseur de technologies pour les entreprises, où IRT est l'acronyme de l'Institut de Recherche Technologique et il s'agit, depuis sa définition, d'un institut de recherche thématique interdisciplinaire qui associe des établissements d'enseignement supérieur et de recherche, des grands groupes et des P.M.E. (petit et moyenne entreprise) autour d'un programme commun de recherche technologique.

Figure 1 : Logo de b<>com

B<>com a été créé en fin de 2012 et il fait partie des huit IRT français. Il se divise en un campus à Rennes, le site principal, et en trois sites secondaires, les sites de Paris, Brest et Lannion. Il s'illustre dans trois domaines d'activité : hypermédia, réseaux & sécurité et e-santé, et il dispose de 6 laboratoires parmi ces trois domaines.

Les plusieurs laboratoires mènent des projets innovants basés sur les nouvelles technologies. Chaque équipe est constituée de salariés b<>com, de personnes mises à disposition par des entreprises extérieures (MAD) et de doctorants accompagnés par leurs encadrants issus du monde académique.

Cet IRT compte environ 300 collaborateurs et le campus de b<>com à Cesson-Sévigné (Rennes) offre une très bonne infrastructure et un agréable environnement de travail (https://b-com.com).

1.2. Le projet HARDY

Le projet HARDY est l'acronyme de *Holographic Augmented Reality DisplaY*, il s'agit d'un projet innovant du laboratoire de « Nouveaux Contenu Média » et a la proposition de développer un prototype de casque holographique miniaturisé et alimenté en temps réel pour susciter des transferts de technologie.

Initialisé en 2019, le projet a une durée prévisionnelle de 2 à 5 ans et il est résultat d'une riche et longue expérience avec l'holographie chez b<>com, plus précisément 7 ans avec la création du laboratoire d'holographie.

L'expertise acquis ayant permis avoir une reconnaissance scientifique dans le domaine de la génération numérique d'hologrammes.

1.3. Méthodologie et objectif du stage

La méthode *Scrum*

Le cadre méthodologique *Scrum* est la méthode Agile largement utilisée dans le monde. Durant ce stage, j'ai rejoint une équipe *scrum* composé d'une dizaine de personnes. L'équipe se réunit généralement une fois par semaine lors d'une réunion de synchronisation afin de suivre l'avancement du projet et de chaque individu.

Mission du stage

L'objectif de ce stage était d'étudier l'application et l'extension d'algorithmes de séparation de sources au problème de la superposition d'ondes monochromatiques provenant d'objets animés. On considère aussi bien le cas linéaire, où le mélange se fait dans le plan de l'hologramme par convolution, que le cas non-linéaire, par exemple en analysant la distribution de Wigner du signal.

En autres termes, le sujet du stage était faire d'étude, l'adaptation de méthodes de séparation de sources pour le codage de vidéos holographiques. Cependant, il faut dire qu'au cours du stage, les objectives ont changé en raison de quelques motifs, parmi lesquelles on peut citer la difficulté des sujets à étudier et à adapter et quelques conditions extérieures qui ont conduit à un développement plus lent du projet. De cette façon, d'autres approches ont été adoptés et étudiés sur le domaine des hologrammes générés numériquement. Plus d'informations seront données au cours de l'avancement de ce rapport.

2. Connaissances de base

2.1. Holographie

Holographie conventionnelle

L'holographie a été inventée en 1948 par le physicien hongrois Dennis Gabor alors qu'il effectuait des recherches en microscopie électronique. L'holographie s'agit d'une technique qui permet d'enregistrer l'amplitude et la phase de l'onde lumineuse provenant d'un objet éclairé à la lumière d'un laser, de cette manière il s'agit d'une technique différente de la photographie conventionnelle qui capture que l'énergie lumineuse, qui est résultat de l'intensité de l'onde lumineuse provenant de la scène.

L'holographie est une technique de plus en plus étudiée car elle fournit une illusion de relief naturelle et réaliste, donc une technique pour la visualisation 3D. Contrairement aux autres techniques existant pour la visualisation 3D, comme la stéréoscopie, technique qui cherche reproduire une perception du relief à partir de deux images planes, l'holographie n'a pas des problèmes avec la disparité convergence/accommodation, limitation qui peut gérer la fatigue visuelle.

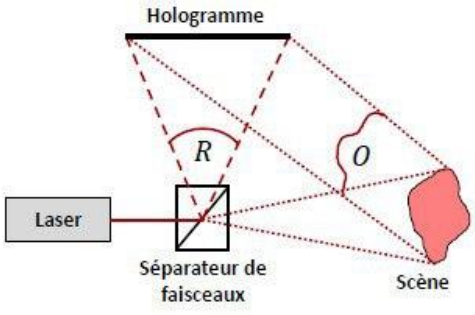

Figure 2 : Processus d'acquisition optique d'un hologramme

Holographie numérique

Le processus d'acquisition optique d'un hologramme offre plusieurs contraintes, une fois que l'acquisition s'effectue à l'aide d'interférence entre deux faisceaux lumineux issus d'un laser. Quelques contraints sont : la nécessité que le système optique soit extrêmement stable, une fois que des vibrations minimales peuvent modifier radicalement les franges d'interférence ; la nécessité d'utiliser une source laser cohérente ainsi qu'en raison du fait que la scène doit être éclairée par un laser, c'est interdit l'acquisition directe de scènes illuminées par une éclairage naturel.

De cette façon, plusieurs techniques ont été proposées pour remplacer le processus physique d'interférence entre deux ondes lumineuses et éviter les limitations présentées. Grâces à ces méthodes, il est possible de générer l'hologramme d'une scène synthétique ou réelle permettant à l'holographie d'être utilisée en extérieur et ouvrant son champ d'application à la vidéo 3D.

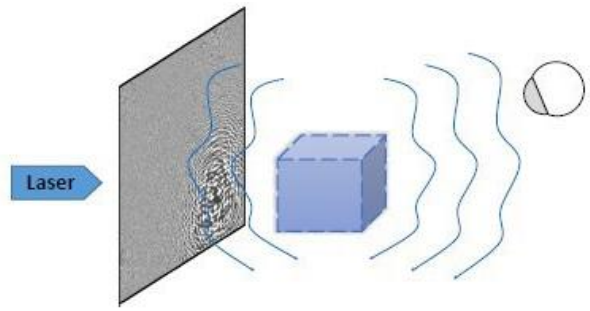

Figure 3 : Principe de la visualisation 3D holographique : en illuminant l'hologramme avec le même laser utilisé lors de l'acquisition, la scène est perçue comme si elle était physiquement présente devant le spectateur

Propriétés des hologrammes gérés numériquement

En raison de l'interférence dans le processus de génération, les hologrammes numériques ont des caractéristiques différentes des images photographiques. Ils n'ont pas des caractéristiques géométriques qui permettent d'être reconnu par une interprétation visuelle.

Ils agissent des signaux non-stationnaires avec des propriétés statistiques (spatiales et spectrales) irrégulières, ce qui rend compliqué de formuler un modèle approprié pour décrire leurs caractéristiques.

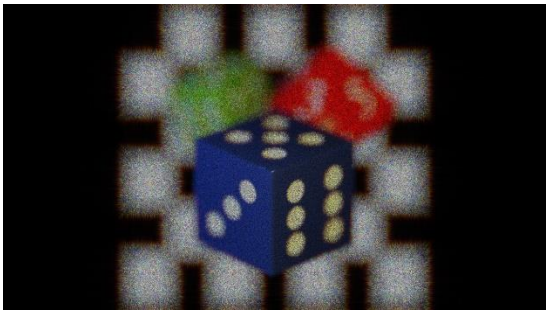

Figure 4 : Amplitude d'un hologramme synthétisé par l'IRT b<>com. Disponible sur le site https://hologram-repository.labs.b-com.com. Accès le 01/04/2020.

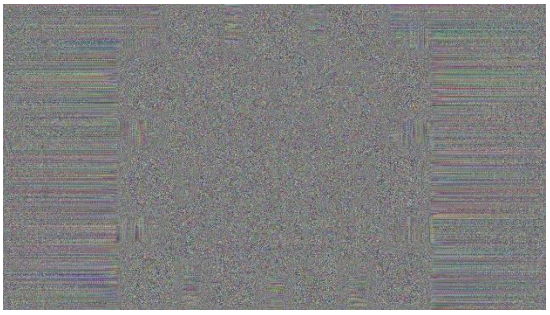

Figure 5 : Phase de l'hologramme synthétisé ci-dessus. Disponible sur le site https://hologram-repository.labs.b-com.com. Accès le 01/04/2020.

II – Développement

3. Génération des hologrammes numériques

Dans ce chapitre on va expliquer avec plus de détails la génération des hologrammes numériques qui seront utilisés pendant le stage pour faire des analyses. Les algorithmes développés étaient basés sur certaines références trouvées dans la bibliographie.

Pour faire la génération des hologrammes numériques on a utilisé [1] comme référence de mise en œuvre. Les codes développés pour cette référence étaient écrits en langages destinés aux logiciels MATLAB et Octave, où ce dernier s'agit d'un logiciel libre de calcul comparable à MATLAB.

Les algorithmes adaptés pour faire la génération des hologrammes numériques ont été aussi écrits en MATLAB une fois qu'il n'avait pas de nécessité de consacrer plus de temps et effort pour faire une adaptation. En revanche, pour certains chapitres suivants, on va utiliser Python comme outil de développement, car il s'agit d'un langage plus adapté aux objectives souhaités.

Il y a quelques techniques pour générer les hologrammes numériques, ces techniques sont : méthode de nuages de points ; méthode des polygones ; méthodes RGB + Profondeur et méthodes basées sur les rayons [2]. De manière simplifiée, le premier type de méthode représente les objets comme une collection de points lumineux discrets ; le deuxième type utilise des éléments de surface tels que les triangles et exploite le fait que la diffraction entre les plans peut être calculée en utilisant une convolution ; la troisième méthode utilise une carte de profondeur et encode des hologrammes avec les informations de profondeur et finalement, la dernière méthode approche l'hologramme par un champ lumineux discrétisé.

Chaque méthode a ses avantages et est la meilleure dans une situation, mais c'est encore possible de trouver des approches hybrides qui adaptent l'algorithme en fonction du contexte [3]. La figure suivante compare les méthodes présentées :

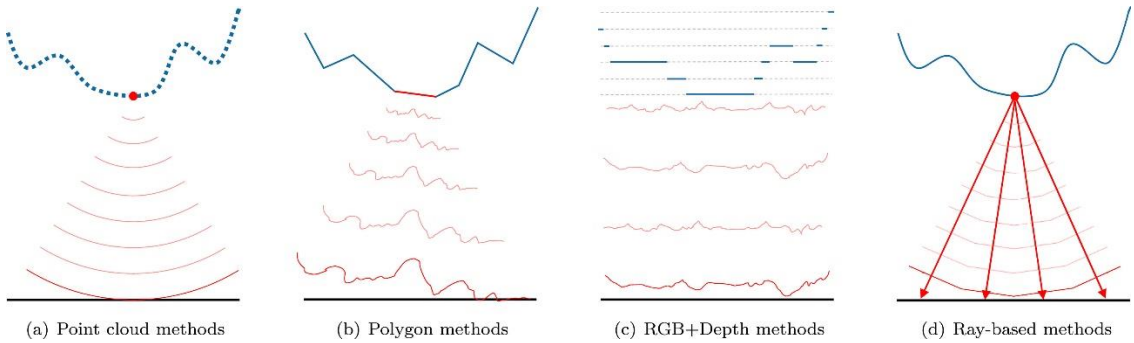

Figure 6 : Comparaison de différents types d'algorithmes. L'objet (bleu) émet des fronts d'onde (rouge) qui se propagent vers l'hologramme virtuel (noir) situé en bas.

La première étape des méthodes de génération numérique d'hologrammes est de simuler la propagation de la lumière provenant de la scène jusqu'au plan hologramme pour former l'onde objet. Pour faire cela on utilise un modèle 3D de la scène qui l'on décompose en un nuage de points. Cette

approche s'agit d'un des algorithmes plus simples pour la génération des hologrammes numériques et c'est donc elle qui a été choisie.

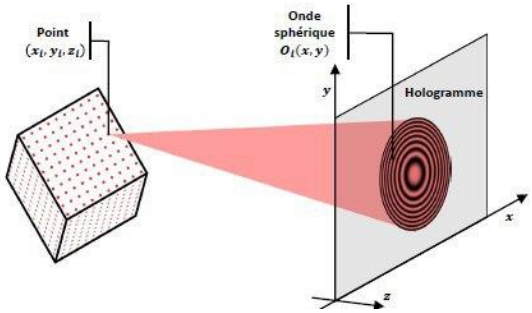

Figure 7 : Calcul de l'onde objet à partir d'un nuage de points : l'hologramme est calculé comme la somme des ondes sphériques issues des points arrivant sur le plan hologramme

Pour les scènes sous la forme d'un nuage de points, chaque point correspond à une source lumineuse ponctuelle sphérique, où chaque point *i* de coordonnées (x_i, y_i, z_i) et échantillonnée dans le plan de l'hologramme aux coordonnés $(x, y, 0)$ est donnée par l'équation suivant [4] :

Équation 1 : Onde sphérique émise par un pont i

$$O_i(x,y) = \frac{a_i}{r_i}\exp[j(kr_i + \phi_i)]\, h_i(x,y)$$

Pour un point i, a_i est son amplitude initiale, où pour l'application on a établi une valeur unitaire un, et ϕ_i est sa phase initiale, où on a établi une valeur égale à zéro. Le nombre d'onde k est donné par l'équation $k = \frac{2\pi}{\lambda}$, avec λ est la longueur d'onde de la lumière. Pour le spectre visible, la gamme des longueurs d'onde s'étend de 380 nm pour le violet à 780 nm pour le rouge le plus extrême. On a établi une valeur de longueur d'onde égale à 500 nm (vert) pour l'application.

La distance oblique r_i entre le point de la scène (x_i, y_i, z_i) est l'échantillon $(x, y, 0)$ dans le plan hologramme est donné par l'équation suivant [4] :

Équation 2 : Distance oblique entre point de la scène et plan hologramme

$$r_i = \sqrt{(x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2}$$

Finalement dans l'équation 1, h_i est une fonction fenêtre utilisé pour limiter la région de contribution du point lumineux dans le plan de l'hologramme. Cette fonction limite les fréquences spatiales pour éviter le repliement de spectre (*aliasing effect*). [1] A été utilisé comme base de mise en œuvre, cependant cette référence n'utilise pas cette fenêtre, donc au début on avait des problèmes de repliement de spectre, où il était nécessaire de l'ajouter dans le calcul de l'onde objet.

Il faut rappeler que pour l'objective du stage, on n'utilise pas une méthode que prendre en compte les occultations lors du calcul de l'onde objet au début. Cette approche effectivement simple réduit le réalisme de la scène, cependant, pour l'objective du stage et afin d'éviter compliquer

l'implémentation en ajoutant encore plus des contraints, les méthodes pour prendre en compte les occultations n'ont pas été prises en compte.

Génération numérique

L'image suivante affiche les étapes dans la génération d'un hologramme numérique d'une scène 3D synthétique.

Figure 8 : Étapes de la génération numérique de l'hologramme

La première étape est la définition des paramètres de calcul, qui peuvent se classifier en trois catégories : les paramètres de la scène (la résolution de la scène, la taille de la scène, la distance entre la scène et l'hologramme) ; les paramètres de rendu (le nombre de couleurs, la parallaxe fournie par l'hologramme et la prise en compte des occultations) ; les paramètres de l'hologramme (la résolution de l'hologramme et la taille des pixels de l'hologramme) [4]. Le tableau ci-dessous affiche les paramètres utilisés :

Tableau 1 : Paramètres prises en compte pour la génération numérique des hologrammes

Longueur de l'onde monochromatique	500 nm
Hauteur d'hologramme	2 mm
Taille d'hologramme	2 mm
Pas d'échantillonnage dans les axes x et y	10 e-6
Résolution	200 x 200 pixels
Localisation du plan de l'hologramme dans l'axe z	Z = 0
Profondeur maximale dans l'axe z	0.5 m

Comme c'est possible de vérifier dans le tableau ci-dessus certains paramètres ont des valeurs relativement faibles et petites, comme les dimensions et la résolution des hologrammes à créer. Comme l'objectif du projet HARDY est de développer un casque de réalité virtuelle, augmentée, les dimensions de l'ordre du centimètre sont attendues ainsi qu'une résolution HD, 4K, 8K ; cependant, comme les objectifs du stage sont plutôt l'étude et l'adaptation, on a adopté de valeurs plus petites pour éviter de faire beaucoup de calculs et avoir besoin de grandes ressources computationnelles.

Comme indiqué dans la figure 8, après définir la scène à enregistrer, c'est-à-dire, la quantité de sources ponctuelles et leurs positions x_i, y_i, z_i on fait le calcul de l'onde objet (*object wave*) en utilisant l'équation 1.

Un hologramme est le résultat de l'interférence entre les rayons lumineux provenant des objets de la scène avec les rayons lumineux d'une référence, donc ensuite, en simulant ce processus,

on fait le calcul de l'onde de référence *(reference wave)* où les vecteurs sont perpendiculaires au plan de l'hologramme. Pour le calcul de l'onde de référence on a utilisé aussi une équation exponentielle [1].

Finalement, la construction de l'hologramme numérique est faite en calculant leur intensité par l'équation ci-dessous, une modification de l'équation de [4] pour conserver aussi les valeurs complexes, où O et R représentent, respectivement, l'onde objet et l'onde de référence dans le plan de l'hologramme.

Équation 3 : Intensité de l'hologramme

$$I_{total} = 2OR^*$$

L'image suivant s'agit d'un exemple d'un hologramme numérique produit pour l'algorithme développé, où c'est pertinent noter que pour faire cet affichage avec le logiciel MATLAB la matrice bidimensionnelle à passer comme paramètre à la fonction *imagesc* doit contenir que des valeurs réelles. Une fois qu'on conserve la partie complexe de l'hologramme puisqu'il s'agit d'information pertinente, autres problèmes avec la manipulation de la partie imaginaire des hologrammes générés seront présentés au cours de ce rapport.

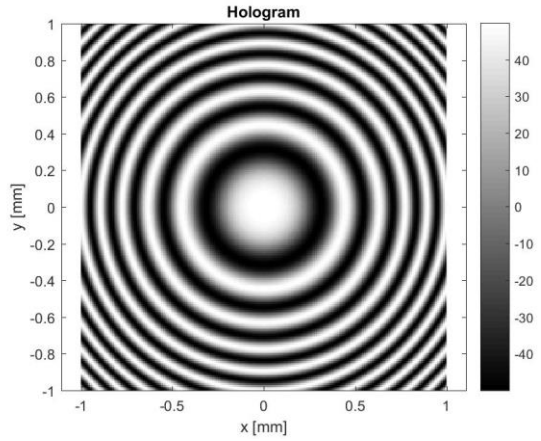

Figure 9 : Hologramme généré numériquement à partir d'une source ponctuelle située dans la position (x,y,z) = (0,0,-0.2m)

Reconstruction numérique

Pour reconstruire une scène enregistrée dans le plan de l'hologramme, il faut éclaire l'hologramme avec la même source lumineuse utilisé dans l'enregistrement, l'onde de référence, et donc les interférences constructives construiront l'image. Cependant, comme on est dans le domaine numérique, au simuler le processus, il faut choisir la position de l'image à être reconstruit dans l'axe z, donc si l'on a plusieurs sources ponctuelles dans la scène à différentes profondeurs, on doit se diriger vers une source spécifique à chaque fois.

Il faut rappeler que l'intérêt n'est pas les images restituées, cependant faire une reconstruction numérique de l'hologramme permettre dans un premier moment de vérifier si l'algorithme fonctionne correctement.

Le principe de la reconstruction est simple, en revanche la reconstruction numérique nécessite de plus d'informations sur les calculs à faire afin de simuler le processus physique. Comme il n'agit pas de l'objective, dans [1] c'est possible de trouver plus d'informations. La figure suivante est la reconstruction de l'hologramme numérique de la figure précédente pour une distance spécifique.

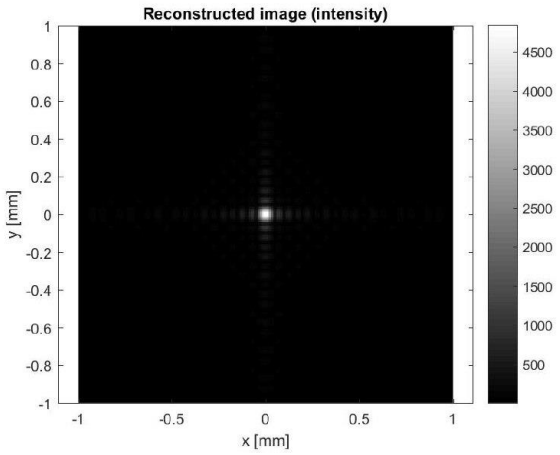

Figure 10 : Reconstruction de la scène pour une distance égale à 0.2m de l'hologramme de la figure 9

Conclusion

Ce qui peut être conclu avec ce chapitre est que les algorithmes pour la génération numérique d'hologrammes à partir d'un petit nombre de sources ponctuelles est relativement simples à mettre en œuvre, où la difficulté se rencontre plutôt sur la compréhension des connaissances d'optique pour bien comprendre le processus de la génération et de la reconstruction des hologrammes numériques.

En revanche, il faut remarquer que si l'on souhaite créer des scènes synthétiques plus complexes et avec plusieurs scènes il faut améliorer la mise en œuvre en utilisant des méthodes plus optimales, car la réalisation actuelle utilise une méthode simple et donc pour les scènes plus complexes et nombreux on va prendre beaucoup de temps pour faire le calcul et les fichiers résultats MATLAB seront larges.

Enfin, après une étude forte sur la théorie des hologrammes et sur l'optique, c'était possible de développer et adapter des algorithmes pour créer les hologrammes à utiliser pendant le stage.

4. Séparation des sources

Dans ce chapitre on va présenter l'essai en faire l'implémentation des méthodes de séparation de sources avec la finalité de séparer les motifs d'interférence de chaque source génératrice dans le plan de l'hologramme. En autres mots, on cherche identifier dans le plan de l'hologramme les régions d'interférence qui ont été générées par chaque source ponctuelle dans la scène.

De cette façon, si l'on sait identifier les régions de contribution de chaque source ponctuelle, on peut augmenter la connaissance sur la scène « enregistré » et développer des différentes approches et techniques, comme par exemple, estimer la localisation de chaque source et de chaque objet d'une scène 3D.

Donc, on a décidé de travailler, étudier et adapter les méthodes de la séparation aveugle de sources, car il s'agit du type de séparation qui correspond mieux au contexte dans lequel on était et avec les données disponibles (champ complexe du plan de l'hologramme), car l'objectif était de partir du plan de l'hologramme et non s'appuyer sur plus d'informations

Cette approche était l'objectif principal et initial du stage, étudier la séparation de sources au problème de la superposition d'ondes monochromatiques provenant d'objets animés. Cependant, comme sera présenté au cours de ce chapitre, les résultats n'étaient pas encourageants et donc des différentes approches ont été prises pour le dérouler du stage.

Séparation aveugle de sources

La séparation des sources est l'art d'estimer des signaux « sources », supposés indépendants, à partir de l'observation d'un ou plusieurs « mélanges » de ces sources [5]. Et la séparation aveugle des sources (SAS) est le cas qu'on n'a pas des informations (or peu d'information) sur les signaux sources ou le processus de mélange [6] .

Le but est d'estimer un jeu de N sources inconnues à partir d'un jeu de P observations. Ces observations sont des mélanges de ces sources et la mélange, qui s'effectue pendant leur propagation entre ces sources est inconnue.

Équation 4 : Représentation mathématique d'un mélange de sources

$$X = AS$$

Pour l'équation ci-dessous, A est la matrice de mélange, où pour le cas de séparation aveugle est inconnue, X est la matrice avec des observations et S est la matrice avec des sources inconnues.

Plusieurs approches et méthodes ont été proposées pour résoudre la SAS, un problème évidemment plus difficile à résoudre à cause de manque de l'information. La SAS est étudiée en nombreuses applications dans de nombreux domaines tels l'acoustique, l'audio, les télécommunications et l'astrophysique.

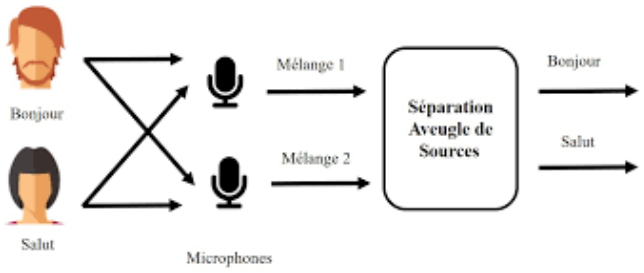

Figure 11 : Schéma de la séparation des sources dans le domaine de l'audio

Le choix de la méthode SAS dépend fortement de la nature des mélanges et de la connaissance des données. Dans la littérature les types de mélange sont découpés en quelques catégories, c'est possible de diviser en *mélanges linéaires*, où les observations s'écrivent comme des combinassions linéaires et *mélanges non-linéaires*, où ils s'agissent des combinassions non-linéaires.

On peut aussi spécifier le problème par rapport à la différence entre le nombre d'observations et le nombre de sources en trois situations : *mélange surdéterminé*, *mélange déterminé* et *mélange sous-déterminé*. Pour le premier type de mélange, on a plus d'observations que de sources ; pour le deuxième on a autant d'observations que de sources ; et finalement, pour le dernier on a moins d'observations que de sources.

C'est possible encore découpé les types de mélanges en *mélanges instantanés*, où on dit « instantanée » car un échantillon de l'observation dépend uniquement des sources à ce même échantillon, cette « moment » ; et *mélanges convolutifs*, où les observations s'écrivent comme la somme des sources convoluées par un masque de convolution.

Le mélange plus simple et plus répandu dans la littérature est le mélange linéaire instantané, cependant, il s'agit d'un mélange inadapté aux cas réels. Dans l'application sur les hologrammes générés numériquement, on va considérer le mélange comme *convolutif* et *sous-déterminé*. [7] A aussi considéré l'hologramme numérique comme un mélange convolutif.

Afin d'effectuer la séparation aveugle de sources, le cas où il faut estimer la matrice de mélange et les sources, il faut ajouter des hypothèses pour surmonter ce problème. Les hypothèses habituelles reposent sur la parcimonie, l'indépendance, la non-négativité et les statistiques de second ordre [8]. Après [8], la SAS peut être découpée en plusieurs familles :

- L'analyse en composantes indépendantes (ICA – *Independent Component Analysis*),
- L'analyse en composantes parcimonieuses,
- La factorisation en matrices non-négatives. (NMF – *Non-negative Matrix Factorization*).

Mise en œuvre

Dans la littérature on trouve beaucoup d'études et des méthodes sur des mélanges plus simples (linéaires et déterminés) et souvent pour des applications liées à l'audio, à la télédétection, à l'astrophysique e autres. Malheureusement, aucune étude n'a été trouvée avec l'objectif qu'on souhaite, donc, on peut dire qu'on est encore dans un domaine des techniques à explorer et développer.

Une fois que le mélange à être séparer est classifié et des hypothèses sont ajoutés, on cherche des méthodes. Cependant, le point de départ présenté est déjà un type de mélange peu étudié et complexe, donc ajouter des hypothèses ne serait pas d'une grande aide, une fois que ce domaine statique sur les données commence à demander des connaissances en dehors du niveau d'étudiant.

Pour simplifier, on a décidé de chercher des applications de séparation de sources dans le domaine de l'image. C'est pertinent à dire que pour faire les implémentations des algorithmes utilisés dans les méthodes de séparation aveugle, on a choisi travailler avec Python, une fois que la bibliothèque scikit-learn a déjà des algorithmes destinés à la décomposition matricielle [9].

On a adapté l'application de [10] qui cherche faire la séparation des sources en utilisant les méthodes ICA et NMF. Les mélanges sont linéaires et surdéterminés et ils s'agissent des images (1000 au totale) des combinassions de quatre chiffres manuscrits 0, 1, 4 et 7 (4 sources). L'application cherche récupérer les images sources à partir des images mélangées et la figure suivante affiche le résultat pour la méthode NFM :

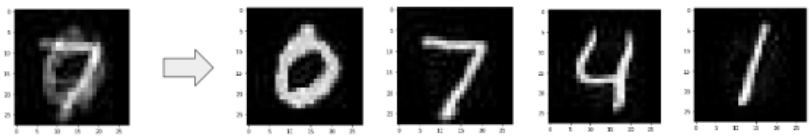

Figure 12 : Séparation d'un mélange linéaire et surdéterminé en utilisant la méthode NFM

Pour s'adapter aux méthodes de décomposition matricielle *FastICA* et *NMF* de la bibliothèque scikit-learn [9], on a généré 10 hologrammes par l'interférence entre deux sources ponctuelles en positions différentes, le but était de simplifier le contexte et avoir un mélange surdéterminé, une fois qu'on avait 10 observations et 2 sources. C'est pertinent dire qu'il faut passer par paramètre pour les algorithmes utilisés l'information préalable du numéro de sources composantes qui est égal à deux.

La figure suivante affiche les hologrammes utilisés, les paramètres prises en compte sont ceux indiqués dans le tableau 1 et les particules varient leurs positions sur les axes x et y dans le plan de l'hologramme comme il est possible de vérifier. La profondeur des particules est égale à 0.2 mètres sauf pour le dernier hologramme où leur profondeur était égale á 0.1 mètre.

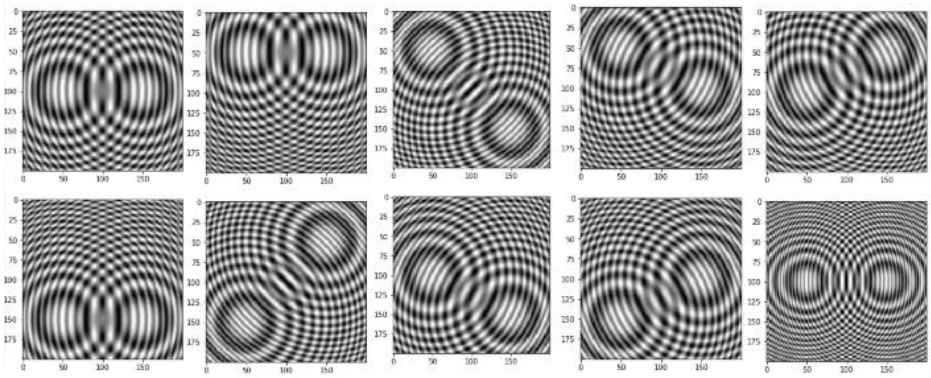

Figure 13 : Hologrammes numériques créés par l'interférence entre deux sources ponctuelles

Après avoir établi une tolérance et un nombre maximum d'interactions, les algorithmes ont été exécutés, mais les résultats obtenus n'ont été pas satisfaisants une fois qu'on n'a pas réussi à trouver les sources. Pour la méthode ICA, les résultats obtenus sont les suivantes, où il est important dire que le système n'a pas convergé.

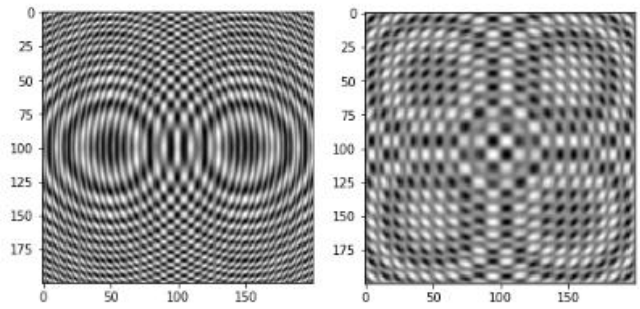

Figure 14 : Résultats obtenus après l'implémentation de la méthode FastICA

Il faut encore faire deux remarques par rapport aux adaptations à faire pour appliquer ces méthodes. Premièrement, pour appliquer l'ICA, les signaux mélangés doivent être une combinaison linéaire de signaux de sources, de plus cette approche suppose que les signaux sources soient indépendants et non gaussiens, une supposition qu'on ne peut pas énoncer pour le contexte des hologrammes numériques.

Ensuite, comme l'ICA, le NMF vise également à décomposer les données mélangées X en un produit des matrices $WH = X$, mais on a la contrainte supplémentaire que chaque matrice X, W et H sont non négatives. De cette façon, il y avait la nécessité d'ajouter une étape supplémentaire pour mettre à zéro les valeurs négatives des hologrammes numériques.

Conclusion

Ce qui on conclut avec ce chapitre est que cet objectif de séparation des sources a été largement étudié pendant les premiers mois du stage, avec la finalité de trouver des références et agrandir les connaissances dans le domaine. Et donc une fois capable d'appliquer et d'adapter des méthodes et techniques pour le problème de séparation dans le contexte des hologrammes numériques avec des valeurs complexes.

Cependant, cet objectif était plus difficile que prévu et en raison encore du peu d'études et de références pour ce type de mélange cible, on n'a pas obtenu des considérables évolutions dans la thématique.

Pour ces raisons qu'on a décidé de pivoter les approches à suivre pour le déroulement du stage et faire d'autres études et des autres implémentations encore sur les hologrammes numériques, mais en essayant de répondre d'autres questions.

5. Localisation des sources

Dans ce chapitre on va présenter les résultats d'une approche pour localiser les sources ponctuelles génératrices d'un hologramme, c'est-à-dire qu'on cherche trouver les positions (x, y, z) des particules à partir du champ complexe 2D de l'hologramme. Pour faire cela, on a pris comme base une méthode qui se consiste à faire une segmentation de la scène 3D qui contient les sources ponctuelles [11]. La méthode est plus détaillée ensuite ainsi que les résultats et les conclusions arrivés.

Méthode

L'approche adoptée est simple et consiste à analyser des différents plans restitués. On s'oriente vers les sources à partir du champ complexe de l'hologramme et on fait la restitution de l'hologramme en une image pour des différentes profondeurs. En d'autres termes, en partant de l'hologramme et avec un pas déterminé on va jusqu'à une profondeur maximale, la profondeur de la scène, et pour chaque profondeur de cet intervalle on fait la reconstruction de l'hologramme. La scène est donc segmentée en différentes profondeurs, comme c'est possible visualiser avec la figure suivante :

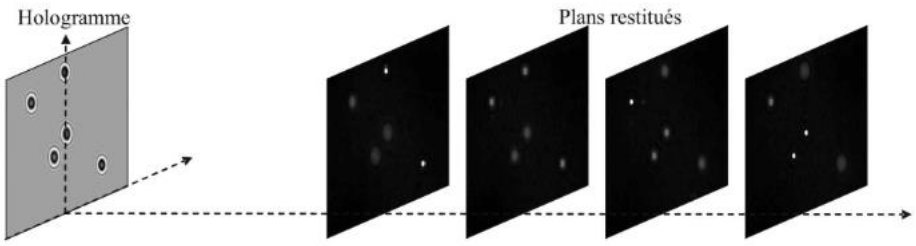

Figure 15 : Restitution d'un hologramme numérique en différentes profondeurs

Pour chaque image restituée de l'hologramme on fait l'analyse de leur intensité, c'est-à-dire qu'on vérifie les valeurs des pixels de l'image et si cette intensité dépasse un seuil de décision on conclut qu'il s'agit d'une source. Donc, une fois que la profondeur z a été trouvé, les positions x et y sont déterminés en vérifiant les positions du pixel plus fort dans l'image restitué.

Résultats

L'hologramme de la figure ci-dessous a été créé avec trois sources ponctuelles en positions différents. La figure 17 affiche quelques paramètres de la scène, de l'hologramme et les positions retrouvées. On remarque que le seuil de décision était égal à 20000 et un pas était égal à 0.1.

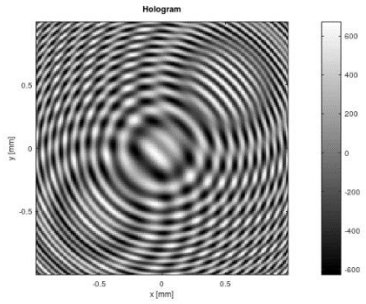

Figure 16 : Motif d'interférence générée (hologramme) par trois sources ponctuelles

```
----- Particles localization -----
Dimensions of the hologram: 0.002 m vs 0.002 m
Resolution of the hologram: 200 pixels vs 200 pixels
Limit of the reconstruction (z): -1.00 m
Number of segmentations calculated: 10 planes
Value of the threshold: 20000.00
Distance of the step: 0.10 m

Point light source 1 of 3: [0, 0, -0.1]
Point light source 2 of 3: [-0.0005, -0.0005, -0.2]
Point light source 3 of 3: [0.0005, 0.0005, -0.3]

Detected particle in (x,y,z) = [0.000, 0.000, -0.100]
Detected particle in (x,y,z) = [-0.001, -0.001, -0.200]
Detected particle in (x,y,z) = [0.001, 0.001, -0.300]

Elapsed time is 3.97559 seconds.
```

Figure 17 : Paramètres de l'algorithme utilisé pour localiser les sources ponctuelles dans la scène et les résultats obtenues

Finalement, les images ci-dessous sont les particules dans la scène 3D. À gauche, on a les positions originales et à droite on a les positions retrouvées. Comme on peut bien noter, pour ce cas-là, on a réussi à trouver les trois particules.

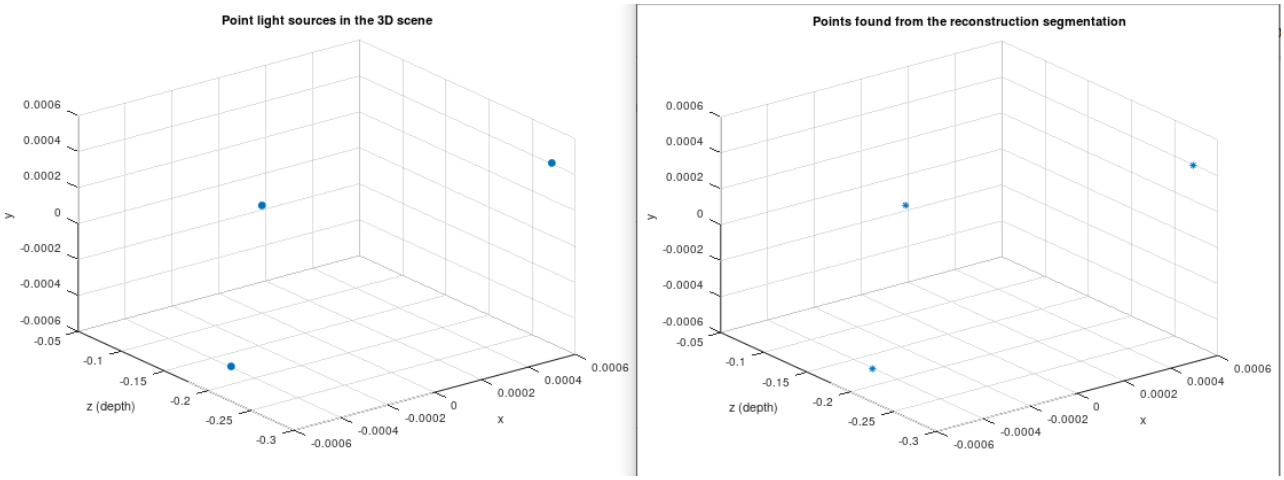

Figure 18 : Disposition des particules dans la scène 3D, à gauche on a les particules dans leurs positions d'origine et à droite on a les positions déterminées par l'algorithme développé

Limitations

Même si l'on a obtenu quelques résultats positifs, l'approche suivit présente de nombreuses contraints et limitations :

- L'algorithme indique l'existence des sources ponctuelles en analysant que l'intensité des pixels de l'image restitué. De cette façon, s'il y a plus qu'une source ponctuelle dans l'image restitué, c'est-à-dire, s'il y plus qu'une source dans une même profondeur l'algorithme n'arrive pas à les détecter, il réussit à détecter que des particules qui se trouvent en profondeurs différentes.
- Dans les tests réalisés, il s'agissait d'une scène assez petite, mais si l'on est dans une situation où la scène est plus profonde et large ou si l'on utilise un pas plus petit, par exemple, le temps de calcul de l'algorithme peut être augmenté facilement.

- L’algorithme développé est très dépendant de la valeur du pas, de la distance entre les plans restitués et de la valeur du seuil de décision. Dans les tests réalisés, ces valeurs ont été déterminées empiriquement,

Conclusion

Après ces premiers tests, on a vérifié que cette méthode de localisation de sources était très restrictive et il fallait d’ajouter des autres caractéristiques pour mieux identifier la présence des sources dans une image restitué, comme par exemple, analyser aussi la forme des particules dans les images restituées ou appliquer des filtres.

Finalement, comme il s’agissait d’une méthode assez coûteuse et pas beaucoup intelligent par rapport à la manière d’identifier les sources, au cours du stage, cette approche n’été pas poursuivie.

6. Distribution de Wigner

Dans ce chapitre on va présenter une représentation espace-fréquence qui sera utilisé pour représenter les hologrammes numériques générés. L'objectif avec cette représentation est de vérifier si elle offert des avantages, c'est-à-dire, si avec représentation on peut mieux comprendre l'hologramme une fois que ses informations sont mieux organisées. On cherche savoir en quel domaine c'est mieux de travailler.

Les réponses pour cette question sont destinées au prochain chapitre, ce chapitre est plutôt destiné à présenter cette représentation, sa définition, ses limitations et son implémentation.

Représentation des données

Analyser une information n'est pas toujours facile à faire, soit un audio ou une image, on utilise des transformées comme outils pour obtenir des nouvelles représentions des signaux. Plus faciles à lire et interpréter, ces représentations sont capables de décrire le contenu du signal à être décomposé.

On peut prendre comme exemple les représentations temps-fréquences qui sont de plus en plus utilisés, car elles ont le grand avantage de décrire conjointement en temps et en fréquence un signal. Cette interprétation est utile pour décrire des signaux non stationnaires dont la fréquence varie au cours du temps.

Cependant, dans la situation présentée on doit travailler avec des signaux bidimensionnels, des hologrammes, donc on cherche plutôt des représentations espace-fréquence, où la distribution de Wigner a était choisie comme représentation à être étudie.

Définition

La distribution de Wigner (DW), aussi connue comme distribution de Wigner-Ville, des noms de Eugene Wigner et Jean Ville, a été introduite par Wigner en 1932 initialement dans le cadre de la physique quantique pour introduire des corrections quantiques à la physique statistique, puis a été introduite par Ville dans la théorie du signal [12]. La distribution de Wigner peut être interprétée comme une représentation de fréquence spatiale locale ou régionale d'une image [13].

Une fois que les hologrammes ont des propriétés irrégulières, l'objectif est d'exploiter la transformé de Wigner comme la méthode spectrale pour faire l'analyse. On peut distinguer trois grandes classes de méthodes d'analyse d'image, où analyser une image c'est essayer d'isoler des composantes pertinentes dans l'image [14].

Les classes sont : méthodes spatiales, méthodes spectrales et méthodes spectrales locales. Les méthodes spatiales sont utilisées pour analyser des images plus simples, une fois que les méthodes spectrales sont utilisées pour les images plus complexes, par exemple, la transformation de Fourier.

La distribution de Wigner d'un signal ϕ est donnée par l´équation suivante :

Équation 5 : Distribution de Wigner d'un signal ϕ

$$W(x,f) = \int \phi^*\left(x - \frac{t}{2}\right)\phi\left(x + \frac{t}{2}\right)e^{-2i\pi tf}\,dt$$

La transformation de Wigner-Ville bidimensionnelle (TWV 2D) d'une image $f(x,y)$ (réelle ou complexe) est définie par [14] :

Équation 6 : Transformation de Wigner-Ville d'une image f(x,y)

$$W_f(x,y,u,v) = \int f\left(x+\frac{\alpha}{2}, y+\frac{\beta}{2}\right) f^*\left(x-\frac{\alpha}{2}, y-\frac{\beta}{2}\right) e^{-2j\pi(u\alpha+v\beta)} d\alpha\, d\beta$$

La TWV d'une image 2D est donc une fonction 4D de quatre variables x, y, u, v, où les x, y sont des variables spatiales et u, v sont des variables fréquentielles. Donc, on a une représentation conjointe espace/fréquence spatiale de cette image, où pour chaque point de l'image on a un spectre local bidimensionnel.

La distribution de Wigner d'une image 2D est une fonction 4D qui implique une transformation de Fourier pour chaque point de l'image. Ce fait conduit à envisager différentes alternatives pour faire le calcul de la DW pour des images discrètes, une fois que le calcul est couteux et prend un temps de calcul considérable, ce point sera mieux présenté au cours de ce chapitre.

Limitations

La distribution de Wigner présente quelques limitations à discuter. Parmi ces limitations on remarque le fait que la distribution de Wigner n'est pas linéaire, c'est-à-dire que la somme des distributions de Wigner de deux signaux fait apparaître des termes croisés. On remarque aussi que la distribution de Wigner n'est pas toujours positive.

D'autres représentations espace-fréquence sont linéaires et n'ont pas ce problème. Un exemple de représentation linéaire est les ondelettes de Gabor qui permettent d'optimiser la localisation conjointe espace / fréquence.

Distribution de Wigner Discrète

Bien que la DW a été initialement proposée pour des fonctions continues, mais des définitions pour des fonctions discrètes ont été aussi proposées. Cependant, l'un des principaux inconvénients de cette définition discrète est qu'il y a des propriétés de la distribution continue qui ne sont pas préservées par la discrétisation.

Quelques définitions ont été proposés, mais on suit [15], où l'approximation de la DW pour les signaux discrets est connue comme *pseudo-Wigner distribution* et est mathématiquement définie comme :

Équation 7 : Distribution de Wigner discrète 1D

$$W(n,k) = 2\sum_{m=-\frac{N}{2}}^{\frac{N}{2}-1} z(n+k)z^*(n-k)e^{-2j\pi k\left(\frac{2m}{N}\right)}$$

Dans l'équation ci-dessus, la variable $z(n)$ représente la valeur de gris du pixel n dans une image donnée z, les variables n et k sont respectivement les variables d'espace et de fréquence et m est un paramètre de décalage.

Il faut remarquer que cette équation s'agit d'une DW 1D locale et directionnelle, donc dans [15], c'est possible choisir l'inclination de la fenêtre coulissant de 1D, pixel par pixel, conforme à une chaîne de valeurs $z = z(-\frac{N}{2}), \dots, z(0), \dots, z(\frac{N}{2})$. La position de la fenêtre centrale correspond exactement au pixel n de l'image. L'équation peut être interprété comme la transformée de Fourier discrète du produit $r(n,m) = z(n,m)z*(n-m)$, où $z*$ indique le complexe-conjugué du signal z.

De cette façon, l'analyse des images numériques peut être effectuée grâce aux informations de fréquence locale qu'elles contiennent, où la fonction discrète pour la DW, qu'il s'agit d'une des représentations espace/fréquence plus populaire, est responsable pour faire la collecte de ces informations de fréquence spatiale locale.

Il y a aussi des fonctions discrètes 2D de la distribution de Wigner, où depuis [16], l'approximation 2D de la DW est définie comme :

Équation 8 : Distribution de Wigner discrète 2D

$$W_{f_w}(m,n,u_p,v_p) = 4\sum_{r=-L}^{L}\sum_{s=-L}^{L} K(m,n,r,s)W_4^{rp+sq}$$

Équation 9 : Notation

$$K(m,n,r,s) = w(r,s)w^*(-r,-s)f(m+r,n+s)f^*(m-r,n-s)$$

Dans l'équation de distribution discrète, $N = (2L+2)$, $W_4 = \exp\left(-\frac{j4\pi}{N}\right)$, et la valeur normalisée la paire de fréquences spatiales normalisés est $(u_p, v_p) = (\frac{p}{N}, \frac{q}{N})$.

Implémentation

Pour faire l'implémentation de cette distribution dans les hologrammes générés, il faut dire qu'on a choisi d'utiliser la distribution discrète 1D pour simplifier l'étude. Une fois qu'on obtient des résultats positifs pour une dimension on peut élargir pour deux dimensions.

On a aussi décidé d'utiliser une fenêtre sans inclination et de taille égal à 9, et on a utilisé l'implémentation de [15] qui demande que la taille de la fenêtre doive être une valeur impaire, où 9 est la valeur plus utilisée.

Ensuite, on a choisi un vecteur de fréquence spatial relativement petite, avec un ensemble des fréquences entières de -4 jusqu'à 3, donc 8 valeurs au total. De cette façon, pour un hologramme numérique générée de 200 x 200 pixels qui l'on l'a redimensionné cette matrice en un vecteur 1D de taille 40000 avant d'appliquer la distribution de Wigner, on a obtenu comme réponse après appliquer la transformation, comme indiqué l'équation 7, une matrice $W(n,k)$ de dimension égale à $(40000, 8)$.

Conclusion

Les résultats et les conclusions par rapport à l'utilisation de cette représentation espace / fréquence seront présentés dans le prochaine chapitre, où on va faire une comparaison en utilisant des réseaux de neurones.

On conclut que même la distribution de Wigner 1D prend un certain temps pour être calculé, une fois que le calcul est fait point par point. Par exemple, pour faire le calcul de 500 hologrammes, l'algorithme a pris presque une heure.

Il faut dire qu'on a essayé des méthodes de parallélisassions en python pour augmenter la vitesse de calcul et on a même envisagé d'utiliser la carte graphique disponible à partir de CUDA (*Compute Unified Device Architecture*), mais qui n'ont pas été adoptés par manque de temps et priorité pour d'autres activités.

Finalement, on conclut que même que la distribution de Wigner soit une représentation espace-fréquence très intéressante, elle est plutôt utilisée dans le domaine temps-fréquence. On fait cette déclaration car des références sont plutôt trouvées pour le domaine temps-fréquence que pour le domaine espace-fréquence.

7. Implémentation des réseaux des neurones

Dans ce chapitre on va présenter les tests qui ont l'objectif de découvrir en quelle domaine de représentation c'est mieux en travailler, c'est-à-dire les hologrammes numériques générés (pixels) ou leurs distributions de Wigner.

L'approche choisie était en utilisant les algorithmes de réseau de neurones pour vérifier en quel domaine le réseau apprendre mieux, on va faire cette analyse par rapport à la précision. Pour faire l'implémentation on va utiliser le Keras API [17]. Il s'agit d'une bibliothèque *open source* écrite en python et souvent utilisé dans le domaine du *deep learning.*

Considérations initiales

Avant de continuer c'est pertinent clarifier la différence entre quelques termes : l'intelligence artificielle *(artificial intelligence),* de l'apprentissage automatique *(machine learning)* et de l'apprentissage profonde *(deep learning*). Encore que les trois terminologies soient généralement utilisées de manière interchangeable, elles ne font pas tout à fait référence aux mêmes choses, l'image suivant donné quelques pistes :

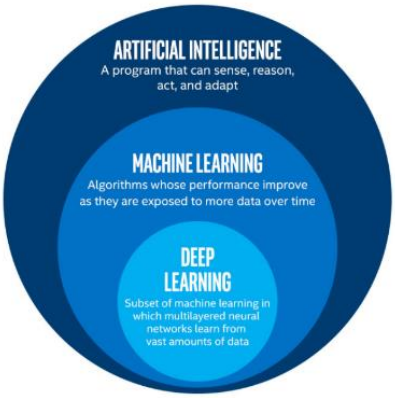

Figure 19 : Relations entre les trois domaines : artificial intelligence, machine learning et deep learning

L'intelligence artificielle englobe les concepts d'apprentissage automatique et d'apprentissage profond, il s'agit plutôt de « l'ensemble des théories et des techniques mises en œuvre en vue de réaliser des machines capables de simuler l'intelligence ». L'apprentissage profond est un sous-ensemble de l'apprentissage automatique, dans le cas on va distinguer ces deux types d'apprentissage en trois points.

Le premier c'est par rapport à l'extraction des caractéristiques *(feature extraction)*, dans les problèmes d'apprentissage automatique on doit choisir les caractéristiques avec lesquelles on va analyser les données, faire l'entraînement, donc ils s'agissent des éléments qui vont être prédicteurs ; une fois que dans les problèmes d'apprentissage profonde on ne les choisi pas. Ce point c'est le plus souvent utilisé pour faire la distinction entre ces apprentissages.

Le deuxième point c'est par rapport à la complexité des donnés traités, dans l'apprentissage automatique on traite souvent des données quantitatives et structurés (des valeurs numériques), une fois que l'apprentissage profonde utilisé des donnés non-structurés, comme le son, le texte, l'image.

Le troisième point c'est par rapport à la puissance de calcul, les algorithmes d'apprentissage profonde ont besoin de plus de ressources de calcul, une fois que les bases des donnés et les temps d'exécution sont souvent plus grandes.

De cette manière, en analysant le type de donné qu'on va travailler avec, les hologrammes complexes, et les ressources disponibles on a décidé d'utiliser les algorithmes d'apprentissage profonde.

Réseau de neurones

Définition

Un réseau de neurones artificielles, ou *Artificial Neural Networks* en anglais, est un système informatique dont la conception est à l'origine schématiquement inspirée du fonctionnement des neurones biologiques du cerveau humaine. En résumé, il s'agit d'un ensemble des neurones organisés en plusieurs couchés et reliés entre eux dans une disposition d'un réseau, où chaque neurone reçoit une information, un signal et sort une information, un signal si ce neurone a été activé.

Les neurones du réseau sont disposés en différents couches, la première couche est désignée la couche d'entrée (*input layer)*, la dernière couche est désignée la couche de sortie (*output layer)* et les couches intermédiaires sont les couches cachées (*hidden layers).* Il n'y a pas un consensus par rapport à la manière de compteur de nombre de couches d'un réseau, mais on va suivre la convention de ne pas compter la couche d'entrée *(input layer).*

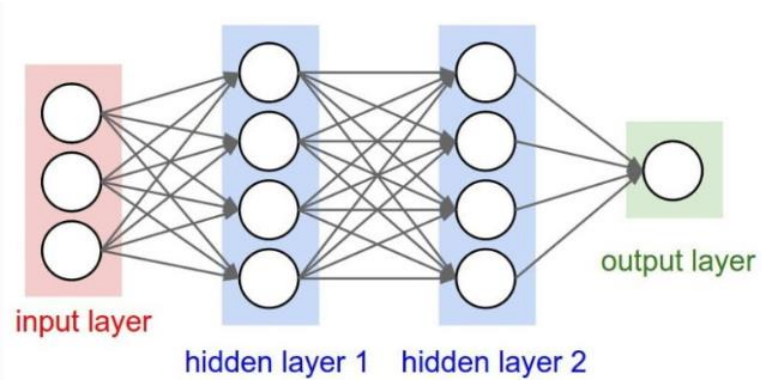

Figure 20 : Réseau de neurones 4/4/1

Types de réseaux de neurones

Actuellement il y a plusieurs types de réseaux de neurones avec des applications et cas d'usages différents, les réseaux plus populaires sont :

- Réseau de neurones de convolution - *Convolution Neural Network* (CNN) en anglais : Ces réseaux reposent sur des filtres de convolution et sont utilisés en applications avec des images.

Ces réseaux capturent les caractéristiques spatiales d'une image. Les caractéristiques spatiales se réfèrent à la disposition des pixels et à la relation entre eux dans une image.

- Réseau de neurones récurrent - *Recurrent Neural Network* (RNN) en anglais : Ces réseaux utilisent le contexte des entrées lors du calcul de la sortie, c'est-à-dire que la sortie dépend des entrées et des sorties calculées précédemment. Les RNN sont utilisés en application où les informations historiques sont importantes, par exemple dans les applications de type *chatbot* et traduction automatique.

Le réseau de neurones choisi

L'objective n'est pas forcément la précision du réseau de neurones qu'on va choisir, donc ainsi que le fait qu'on a peu d'expérience avec les réseaux des neurones, on a choisi d'utiliser un réseau de neurones plus simple dans un premier moment.

On a choisi d'utiliser le Perceptron Multicouche, ou *Multilayer Perceptron* (MLP) en anglais, ce cas est le cas particulier plus ancien. Le MLP est un type de réseau organisé en plusieurs couches au sein desquelles une information circule de la couche d'entrée vers la couche de sorties uniquement, il s'agit donc d'un réseau à propagation directe *(feedforward).*

Actuellement il y a plusieurs techniques et réseaux plus complexes que le Perceptron Multicouche, où ce réseau a un caractère plus pédagogique et exploratoire, cependant, comme dit avant, on a choisi d'utiliser ce réseau même avec ces limitations dans un premier moment.

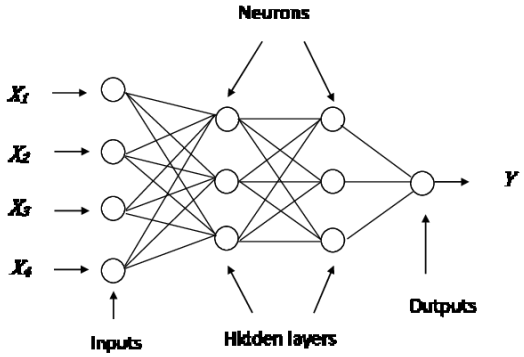

Figure 21 : Exemple d'un Perceptron Multicouche

Ce réseau de neurones sera implémenté en deux problèmes différents, dans un cas de classification et dans un cas de régression. Pour le problème de classification, l'objective est de classifier les entrées par rapport à la quantité de sources ponctuelles qui ont créé l'hologramme. On va avoir comme entrée une base des données des hologrammes avec d'une 1 à 5 sources ponctuelles, de cette façon, l'objective du réseau est de classifier l'hologramme par rapport à leur classe correspond.

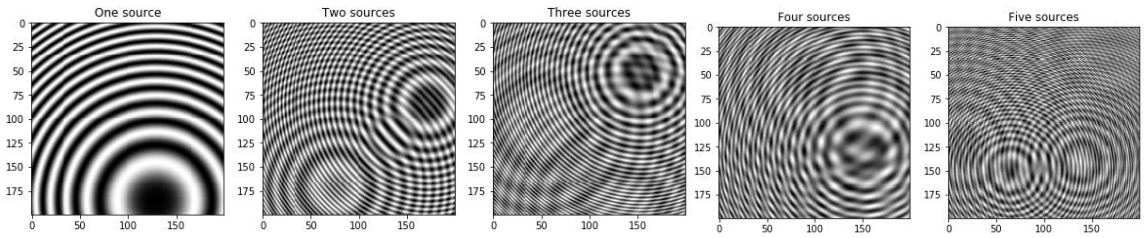

Figure 22 : Exemples d'hologrammes numériques de classes 1, 2, 3, 4 et 5, respectivement

Par rapport au problème de régression, l'objective est de retrouver les positions des sources ponctuelles dans la scène 3D, c'est-à-dire, les valeurs des leurs positions x, y et z. Il convient de noter qu'en cas de régression, les hologrammes numériques ont qu'une source ponctuelle génératrice.

Figure 23 : Objectifs de problèmes de classification et régression, respectivement

Framework

Dans le monde du *deep learning,* on remarque l'existence de plusieurs *frameworks* implémentés en Python, en fait le Python est actuellement le langage de programmation plus utilisé pour les applications de *Data Science, Machine Learning* et *Deep Learning* en raison principalement de ses librairies et de sa communauté. On remarque les librairies TensorFlow, Keras et PyTorch.

Ces outils sont écrits en python où le TensorFlow était par Google une fois que le PyTorch par Facebook. On a décidé d'utiliser le Keras comme outil principale d'implémentation, car c'est le plus facile d'utilisation et a une syntaxe simple, facilitant un développement rapide.

Keras est une API *(Application Programming Interface)* de haut niveau capable d'utiliser le TensorFlow comme *backend* et même que sa performance est inferieur comparativement aux autres son architecture est plus simple, lisible et concise étaient des facteurs importants dans la décision.

Figure 24 : Logos des bibliothèques Keras, Tensorflow et PyTorch, respectivement

7.1. Problème de classification

L'objective de ce problème est de classifier dans un moment un hologramme et dans une autre moment sa distribution de Wigner par rapport à quantité de sources ponctuelles qui l'ont générée et à la fin faire une comparaison entre les résultats. Pour l'étude on a choisi travailler avec 5 classes, où la première classe est des hologrammes avec 1 source ponctuelle et la cinquième classe sont des hologrammes avec 5 sources ponctuelles, ayant ainsi un nombre croissant de sources ponctuelles.

Pour chaque classe on a généré 2000 hologrammes, cette quantité a été choisie en prenant comme références des bases des donnés connues dans les applications d'apprentissage automatique comme la base des données MNIST et CIFAR-10, où cette dernière a 6000 exemples pour chaque classe.

Dans la base des données, les positions des particules ont été générés de manière aléatoire, pour les dimensions x et y, les particules sont à l'intérieur du plan d'hologramme et par rapport à la dimension z, la particule sera positionnée à partir d'une distance du plan d'hologramme jusqu'à une profondeur maximale de la scène 3D, il est pertinent d'informer que pour l'axe z la distance minimale a été choisie de manière expérimentale, car pour les valeurs plus proches d'hologramme on a des problèmes de repliement de spectre *(aliasing effect)*.

Les propriétés des hologrammes ont été déjà présentes dans le tableau 1 au chapitre 3, mais on ajoute quelques informations dans le tableau suivant :

Tableau 2 : Propriétés de la génération des hologrammes numériques de la base des données

Dimensions d'hologramme	2 mm x 2 mm
Résolution d'hologramme (nombre d'échantillons)	200 x 200 pixels
Intervalle des valeurs possibles pour l'axes x et y	[-1, +1] mm
Intervalle des valeurs possibles pour l'axe z	[-0.5, -0.05] m

Prétraitement des données

Avant d'utiliser les méthodes d'apprentissage automatique il faut préparer la base des données pour mettre les données dans le bon format e aussi pour faciliter le processus d'entraînement des algorithmes. Comme on aura deux types d'entrée, l'hologramme et la distribution de Wigner, quelques étapes peuvent être différents, mais le principe et le même.

Pour le processus de traitement des hologrammes numériques, la première étape est de charger les fichiers en format *mat*, une fois que la génération des hologrammes est réalisée par des scripts MATLAB. Ensuite, on fait le prétraitement des données, la première action est de remodeler la forme des bases des données, on veut que des images soient unidimensionnelles. On a décidé de travailler en une dimension (1D), car il s'agit d'un scenario plus simple et donc, si les résultats sont favorables après on peut s'orienter pour un cas de 2 dimensions, et l'entrée du réseau choisi est un vecteur d'une dimension.

La deuxième action à prendre dans le processus de prétraitement des donnés est faire sa normalisation. Ensuite on calcule un vecteur *(target)* avec la quantité de sources ponctuelles de chaque exemple.

Après on divise la base des données pour faire l'entraînement des algorithmes d'apprentissage automatique (*training dataset*) et valider l'entraînement (*validation dataset*). Dans la littérature d'apprentissage automatique la division de la base des donnés pour l'entrainement et la validation est souvent 80% pour la première et 20% pour la deuxième, on a pris la même proportion.

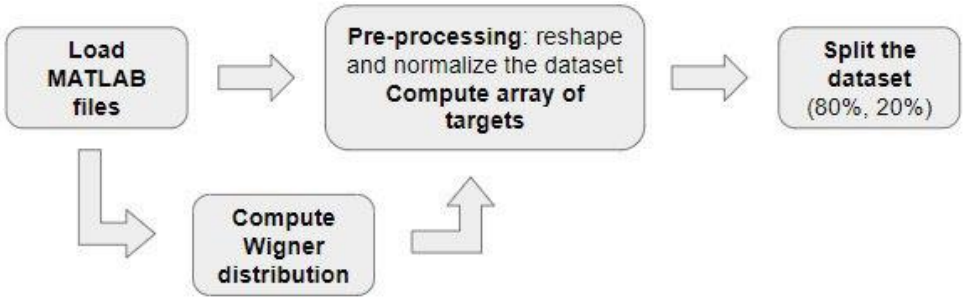

Figure 25 : Étapes de la préparation de la base des données de formation (train dataset) et validation (test dataset)

Pour les processus de traitement des distributions de Wigner il y a une petite différence après le calcul de la distribution de Wigner. Après celui-ci, on fait aussi le remodelage des données, mais dans ce cas, on est déjà dans une dimension, car on utilise la distribution de Wigner 1D, ce qui est fait, c'est sélectionner une fréquence spatiale pour créer la base des données à être entraînée, plus de détails seront données au cours de ce rapport.

Finalement, les données sont sauvegardées en format *npy (NumPy array)* pour être utilisé pour les réseaux des neurones.

Implémentation du Perceptron Multicouche

Construction du réseau de neurones

La structure centrale de Keras est le modèle (***model)***, il permet d'organiser et faire la conception des couches. Il y a deux façons de construire un modèle avec Keras, la manière séquentielle (***Sequential***) et la fonctionnelle ***(Functional)***. Le mode séquentiel est la façon plus simple de construire un modèle, où l'on peut ajouter couche par couche, en revanche il est plus limité ; le mode fonctionnel est plus complexe et avancé, avec celui c'est possible partager des couches, avoir branches et avoir plusieurs entrées et sorties. Dans le cas, on a choisi de travailler avec le modèle séquentiel en raison de sa simplicité et de la nécessité momentané.

L'API Keras offre types différents de couches, on remarque les trois groupes de couches plus pertinents : les couches de base (***Core Layer***), les couches de convolution ***(Convolutional Layer***) et les couches de *pooling* (***Pooling Layer***). Pour le premier groupe on met en évidence les couches de type ***Dense***, il s'agit des couches de réseau de neurones standards qu'on va utiliser dans l'application, sont des couches où tous les neurones sont connectés aux neurones de la couche précèdent et l'entrée et la sortie sont des vecteurs unidimensionnels.

Les couches de convolution sont des couches présentant des filtres de convolution d'une dimension jusqu'à trois dimensions et les couches de *polling* sont des couches utilisées pour réduire la taille de la donnée d'entrée et extraire les informations plus importantes. Cependant, dans l'application présente, on va utiliser que les couches de type *Dense.*

Le réseau de neurones implémenté consiste en un Perceptron Multicouche avec 2 couches cachés et une couche de sortie, dans l'implémentation les paramètres passées aux couches de type *Dense* sont : la fonction d'activation ***(activation)***, le nombre de nœuds ***(units)*** et dans le cas de la première couche, il faut passer aussi la dimension des données d'entrée ***(input_dim)***.

Keras offre plusieurs fonctions d'activation, une fonction d'activation est une fonction mathématique appliquée à un signal en sortie d'un neurone artificiel. Parmi les fonctions possibles, on souligne les fonctions : unité de rectification linéaire, ou *Rectified Linear Unit* (ReLU) en anglais ; sigmoïde *(sigmoid*) ou logistique ; softmax ou fonction exponentielle normalisée.

Dans l'application on utilise la fonction d'activation *ReLU* pour les couches cachées, il s'agit d'une fonction standard et pour la couche de sortie, on utilise la fonction *softmax* car elle est capable de générer une distribution de probabilité, souvent utilisé dans les problèmes de classification avec plusieurs classes. Le choix de la fonction d'activation est très important pour les couches de sortie car il définit le format que les prévisions prendront.

Tableau 3 : Détails du réseau MLP 1000/500/5 pour le problème de classification

	Type de couche	Nombre de nœuds	Fonction d'activation
Première couche caché *(hidden layer)*	Dense	1000	ReLU
Deuxième couche caché (*hidden layer*)	Dense	500	ReLU
Couche de sortie (*output layer*)	Dense	5	softmax

Une question souvent posée dans le monde de l'apprentissage profonde (*deep learning)* est comment choisir le nombre de couches et le nombre de nœuds en chaque couche quand on va créer un réseau de neurones, ces deux sont connues comme des hyper paramètres. Malheureusement, il n'a pas d'une réponse exacte et facile, pour trouver la meilleure configuration possible il faut faire des exhaustifs tests and expérimentations.

Par contre, on trouve quelques pistes à suivre en quelles mots, la première c'est essayer différentes configurations pour le réseau et vérifier qui fonctionne le mieux ; la deuxième c'est suivre des « règles générales » (*rule of thumb* en anglais) ; un exemple de règle concerne le nombre de neurones dans les couches cachés, où il faut être la taille de l'entrée et de la sortie, il faut signaler que ces règles ne sont pas toujours valables, il s'agissent des règles empiriques et sont plutôt des points de départ ; finalement la troisième piste est utiliser des algorithmes qui changent dynamiquement la configuration du réseau.

Pour le réseau de neurones implémentée on a pris deux couches cachées en suivant un résultat théorique que dit qu'avec deux couches cachées est suffisant pour créer des régions de classification

de n'importe quelle forme. Les nombres de nœuds a été pris de manière empirique, où ces valeurs, montrés dans le tableau antérieur ont été considéré suffisant.

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 1000)              40001000
_________________________________________________________________
dense_2 (Dense)              (None, 500)               500500
_________________________________________________________________
dense_3 (Dense)              (None, 5)                 2505
=================================================================
Total params: 40,504,005
Trainable params: 40,504,005
Non-trainable params: 0
_________________________________________________________________
```

Figure 26 : Résumé de la structure construit

Figure 27 : Étapes du processus d'entrainement

Compilation

Après la création du model séquentielle c'est l'étape de la compilation, la fonction *compile* du Keras accepte trois paramètres importants : la méthode d'optimisation (***optimizer***), la fonction de perde (***loss function***) et un système de métrique pour mesurer les résultats ***(metrics)***.

Parmi les méthodes d'optimisation de Keras plus utilisées se trouvent le **SGD** (Algorithme du gradient stochastique, ou *stochastic gradient descent* en anglais) et l'**Adam** *(Adaptative Moment Estimator* en anglais). Pour l'implémentation, on a choisi l'Adam car il s'agit d'une méthode efficace et qui nécessite peu de mémoire.

La fonction de perde représente une certaine mesure de la différence entre les valeurs observées des données et les valeurs calculés, leur objectif est de calculer la quantité qu'un modèle doit chercher à minimiser pendant la formation. Pour le cas de classification on remarque les fonctions ***binary_crossentropy*** et le ***categorical_crossentropy***, ces calculent la perte d'entropie croisée *(cross entropy)* entre les résultats attendus *(true labels)* et les résultats prévues *(predicted labels)*, le premier est utilisé quand on a deux classes et le deuxième quand on a plus que deux classes, donc on a utilisé le dernier cas.

Les métriques sont des fonctions pour évaluer la performance du modèle, on a choisi la métrique ***accuracy*** pour calculer combien les prédictions sont égales aux valeurs attendues.

Fit

Après la compilation du model séquentielle, c'est l'étape de l'entraînement. Pour faire l'entraînement il faut spécifier deux hyper paramètres, le nombre d'époques ***(epoch***) et la taille de lot (***batch size***), le premier s'agit du nombre de fois que le modèle est exposé à la base d'apprentissage *(training dataset*), c'est-à-dire le nombre de passages sur l'ensemble des exemples de la base

d'apprentissage lors de la descente de gradient ; le deuxième s'agit du nombre de cas d'entraînement passés avant d'effectuer une mise à jour des poids du réseau, c'est-à-dire le nombre d'exemples utilisé pour estimer le gradient de la fonction de coût.

Avant d'appliquer la méthode *fit* de Keras pour entraîner le modèle il faut faire une petite modification dans les tableaux avec les classes de chaque exemple *(target classes array)* car il s'agit d'un problème de classification. On applique un encodage one-hot *(one hot* encode) qui consiste à représenter des états en utilisant pour chacun une valeur dont la représentation binaire n'a qu'un seul chiffre. Pour faire cette modification on utilise la méthode ***to_categorical*** de Keras, cet outil fait la conversion d'un vecteur de classe (nombres entiers) en une matrice de classe binaire. L'image suivant montre la transformation :

$$y = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{bmatrix} \rightarrow \quad y = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Figure 28 : Principe de l''encodage one-hot

Autre question souvent posé c'est comment choisir le nombre de d'époques et la taille de lot, la taille de lot doit être supérieure ou égale à un et inférieure ou égale au nombre d'exemples dans la base d'apprentissage. Dans la littérature et des tutoriels c'est possible de trouver des exemples avec une taille de lot de 32, 64, 128 et un nombre d'époques de 10, 100, 1000 ou plus grande. Après quelques essaies on a mis 50 comme le comme le nombre d'époques et 1000 comme la taille de lot.

Dans les applications d'apprentissage automatique il faut faire attention pour éviter avoir un état de **surajustement** (en anglais *overfitting*), où il s'agit d'une analyse statistique qui correspond trop précisément à une collection particulière d'un ensemble données et un état de **sous-ajustement** (en anglais *underfitting*), il est le contraire, où le modèle ne s'approche pas assez de la fonction et donc est incapable de saisir la tendance des données.

Finalement, c'est pertinent présenter deux autres fonctionnalités exploitées d'API Keras. Le premier est le ***callback***, il s'agit d'un objet qui peut effecteur des actions dans les différents étages de l'étape d'entraînement, dans le cas on a utilisé le *EarlyStopping* et le *Tensorboard.* Ce premier arrêt la formation lorsqu'une métrique contrôlée a cessé d'améliorer, dans le cas la perte de la perte de la validation (*validation loss* en anglais) est analysée et à partir du moment qu'elle n'y a pas de progression, la formation est arrêtée. Cette métrique est souvent monitorée dans les problèmes de classification, il s'agit de l'erreur des exemples dans la base des données de validation, le *testdaset*. Le deuxième est un outil de visualisation.

L'autre fonctionnalité exploité était l'enregistrement et le chargement du modèle. Keras offre la possibilité d'enregistrer l'architecture et les valeurs de poids, de cette façon après on charge ces valeurs, faire la compilation du modèle et peut l'utiliser pour faire des prédictions.

Résultats

Il faut toujours rappeler que les réseaux des neurones sont des algorithmes stochastiques, donc un même algorithme avec une même base des données peut entrainer et gérer une model différent à chaque fois que l'algorithme est exécuté.

De cette manière, en utilisant une base des données avec 10000 hologrammes au totale, donc 2000 hologrammes per classe, le tableau suivant présente la précision moyenne pour base des données de formation et de validation, on a exécuté l'algorithme d'entraînement cinq fois.

Tableau 4 : Précisions obtenues pour le problème de classification en utilisant les hologrammes numériques

Train accuracy	98.81%
Test accuracy	77.84%

Les résultats ci-dessus semblent très prometteurs, mais lorsqu'on a utilisé le modèle entrainé pour faire des prévisions avec les mêmes bases des données, on obtient les résultats affichés dans la figure suivante :

```
Test predictions:

- Predictions for class 0, accuracy: 97.75%
[0]: 391.0, [1]: 0.0, [2]: 2.0, [3]: 6.0, [4]: 1.0

- Predictions for class 1, accuracy: 0.50%
[0]: 385.0, [1]: 2.0, [2]: 2.0, [3]: 4.0, [4]: 7.0

- Predictions for class 2, accuracy: 0.25%
[0]: 389.0, [1]: 2.0, [2]: 1.0, [3]: 6.0, [4]: 2.0

- Predictions for class 3, accuracy: 1.00%
[0]: 391.0, [1]: 3.0, [2]: 1.0, [3]: 4.0, [4]: 1.0

- Predictions for class 4, accuracy: 0.75%
[0]: 390.0, [1]: 0.0, [2]: 1.0, [3]: 6.0, [4]: 3.0
```

Figure 29 : Prédictions obtenues pour le problème de classification en utilisant des hologrammes numériques. Dans cette base des données, il y avait 400 exemples pour chaque classe, on a compté la quantité des prédictions correctement classées pour chaque classe

Les résultats ci-dessus ne sont évidemment pas bons, mais l'analyse sera faite dans la partie de conclusion.

Ensuite, pour la distribution de Wigner, on a fait les mêmes tests antérieurs. L'objectif était de faire calculer la DW pour tous les hologrammes de la base des données utilisé dans le test précèdent pour avoir une comparaison plus fidèle, cependant, en raison du tempo de calcul, où pour calculer la DW de 500 hologrammes on a pris 45 minutes, jusqu'au moment où ce rapport a été écrit on a fait des tests avec la DW des 500 hologrammes, donc 100 hologrammes per class.

On rappelle que les deux paramètres importants pour le calcul de la DW, comme présenté dans le chapitre 6, sont la taille de la fenêtre et les fréquences spatiales, où on avait une fenêtre de taille égal à 9 et des fréquences spatiales entières de -4 jusqu'à 3. On remarque autre fois qu'après calculer la DW d'un hologramme on a un résultat pour chaque fréquence, 8 fréquences au total. Pour le cas présent, on a utilisé des distributions de Wigner pour la fréquence spatiale égale à 0.

Donc, on a fait autre fois l'entrainement et le tableau et la figure suivante affichent les résultats de la précisions moyenne de cinq entraînements et les prédictions d'un modèle formé parmi ces entraînements, respectivement.

Tableau 5 : Précisions obtenues pour le problème de classification en utilisant les distributions de Wigner

Train accuracy	36.85%
Test accuracy	29.60%

```
Test predictions:

- Predictions for class 0, accuracy: 85.00%
[0]: 17.0, [1]: 0.0, [2]: 1.0, [3]: 2.0, [4]: 0.0

- Predictions for class 1, accuracy: 0.00%
[0]: 7.0, [1]: 0.0, [2]: 1.0, [3]: 12.0, [4]: 0.0

- Predictions for class 2, accuracy: 15.00%
[0]: 0.0, [1]: 0.0, [2]: 3.0, [3]: 17.0, [4]: 0.0

- Predictions for class 3, accuracy: 95.00%
[0]: 1.0, [1]: 0.0, [2]: 0.0, [3]: 19.0, [4]: 0.0

- Predictions for class 4, accuracy: 0.00%
[0]: 0.0, [1]: 0.0, [2]: 2.0, [3]: 18.0, [4]: 0.0
```

Figure 30 : Prédictions obtenues pour le problème de classification en utilisant des distributions de Wigner. Dans cette base des données, il y avait 20 exemples pour chaque classe, on a compté la quantité des prédictions correctement classées pour chaque classe

Conclusion

Premièrement, on va faire une analyse des résultats obtenus avec la base des données des hologrammes numériques. Même si après la formation du modèle, on a obtenu une valeur de précision élevé tant pour la base d'entrainement que pour la base de validation quand on a fait des prédictions, on a obtenu de mauvais résultats.

Après avoir fait des recherches sur ce problème, et analysé d'autres pistes comme la figure ci-dessous, on a pris que probablement au lieu de chercher un modèle dans les données le réseau de neurones a commencé à mémoriser l'entrée. Avec la figure ci-dessous, on peut observer que l'erreur de validation s'est stabilisée en fait, cependant il s'agit d'une erreur toujours élevée, donc lorsqu'on fait des prédictions comme dans la figure 29, le modèle prédit erronément, dans la figure, le modèle a prévu tous les hologrammes comme appartenant à la classe d'une source ponctuelle.

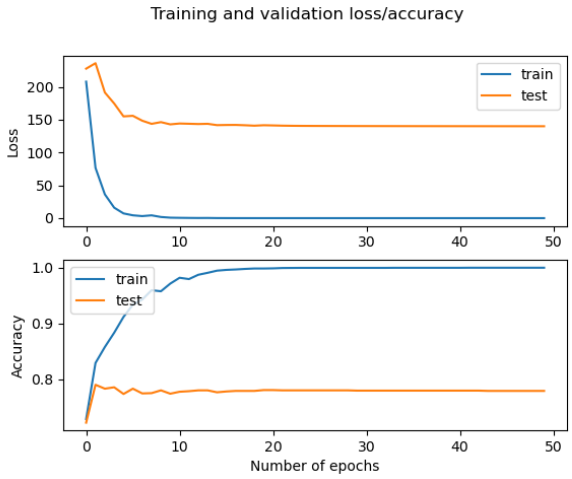

Figure 31 : Erreurs et précisions obtenues pour les bases de données de formation (train dataset) et validation (test dataset) avec des hologrammes numériques

En ce qui concerne les distributions de Wigner, il peut même sembler qu'on a des résultats plus prometteurs en ce qui concerne l'erreur et l'erreur de validation, comme on peut noter dans la figure ci-dessous, où ces valeurs sont proches de zéro. Cependant, dans un premier moment, on a utilisé peu de données pour avoir assez de résultats pour arriver à une conclusion et, de plus, le réseau de neurones utilisé, le Perceptron Multicouche, ne semble pas être le mieux ni le plus approprié et capable de résoudre le problème.

Il y a encore d'autres questions qui doivent encore être mieux analysés, telles que les fréquences spatiales du calcul de la distribution de Wigner, certaines limitations et problèmes lors de la mise en œuvre en utilisant l'API Keras, comme le fait que les valeurs complexes des hologrammes numériques sont ignorées au faire le processus d'entrainement.

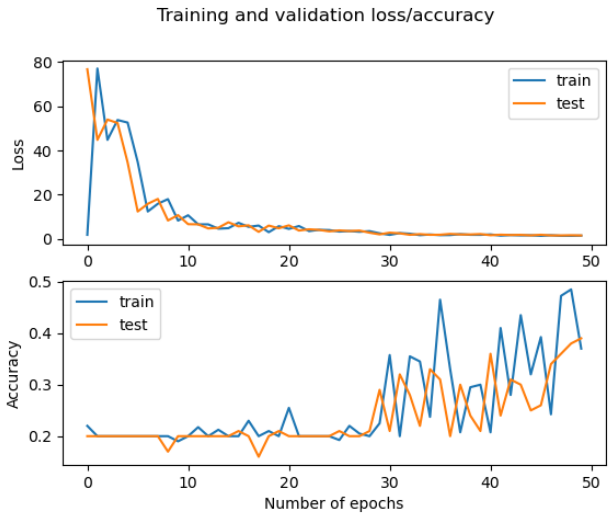

Figure 32 : Erreurs et précisions obtenues pour les bases de données de formation (train dataset) et validation (test dataset) avec les distributions de Wigner

7.2. Problème de régression

Dans cette partie, on va présenter avec plus de détails le problème de régression et même si on a quelques points en communs par rapport au problème de classification, comme par exemple, dans le processus de construction du modèle, il y a encore des points différents importants à discuter.

Implémentation du Perceptron Multicouche

Par rapport à la construction du modèle à être entrainé, la différence par rapport à celui-ci du problème de classification est le nombre de nœuds dans chaque couche et la fonction d'activation dans la dernière couche, la couche de sortie.

Pour le modèle de régression on a aussi mis 1000 nœuds pour la première couche cachée, mais pour la deuxième couche cachée on a mis 400 nœuds et finalement pour le dernier couche on a mis 3 nœuds, une fois que les sorties seront les positons 3D. Il faut spécifier que la base de la donnée utilisée pour le problème de régression est différente de la base des données du problème de classification, une fois que dans un premier moment, on a décidé de faire la régression pour des hologrammes numériques avec seulement une source ponctuelle. Finalement, la fonction d'activation choisie pour la couche de sortie est le ***linear*** au lieu du ***softmax***, une fois qu'on veut les valeurs des positions.

Par rapport à la compilation, on a aussi choisi la méthode d'optimisation ***adam***, en revanche pour la fonction de perde et la métrique d'évaluation on a mis ***mse***. Il faut remarque que différentient du problème de classification, où on sait facilement si on a obtenu des bons résultats, dans le problème de régression on fait l'évaluation des résultats en faisant trois calculs entre les résultats et les prédictions. On calcule l'écart quadratique moyen (*root mean squared error* en anglais), l'écart moyen *mean squared error* en anglais) et l'erreur absolue moyenne (*mean absolute error* en anglais).

Par rapport à l'entraînement du model, on a pris 50 comme nombre d'époques et 1000 comme la taille de lot et on a aussi mis des fonctions *callback* afin d'arrêter l'entrainement s'il n'y a pas des changements considérables.

Résultats

Pour le problème de régression, on a aussi généré une base des données, dans le cas, on avait une base des données de 2000 hologrammes avec des hologrammes générées pour seulement une source ponctuelle. Ainsi, on a fait la régression en utilisant le même réseau de neurones utilisé dans le problème de classification, le Perceptron Multicouche.

Le tableau suivant contient la moyenne de l'évaluation de cinq entrainements, c'est-à-dire, les moyennes des erreurs obtenues. Il faut rappeler que les réseaux de neurones sont des algorithmes stochastiques, c'est-à-dire qu'à chaque fois qu'on fait l'entrainement, on obtient des résultats différents :

Tableau 6 : Erreurs pour le problème de régression des positions x, y et z d'une base des hologrammes numériques générées par seulement 1 source ponctuelle

	Train	Test
Root Mean Squared Error	0.630	0.687
Mean Squared Error	0.397	0.473
Mean Absolute Error	0.397	0.473

La figure suivante affiche quelques prédictions faites, où c'est possible de noter que les résultats prévus pour les positions x et y sont extrêmement erronées, puisque le réseau prévoit des résultats de l'ordre du mètre et non du millimètre comme attendue. Les résultats pour les positions z sont aussi mauvais.

```
Test predictions:
Prediction shape: (400, 3)
Example [0]
Real position:     (x, y, z) = (0.00043, 0.00011, -0.07114)
Predicted position (x, y, z) = (-0.16366, -0.11607, -0.02185)
Example [1]
Real position:     (x, y, z) = (-0.00024, -0.00021, -0.28837)
Predicted position (x, y, z) = (-0.69372, 0.36386, -0.52931)
Example [2]
Real position:     (x, y, z) = (-0.00096, 0.00016, -0.46610)
Predicted position (x, y, z) = (-0.36461, 0.09712, -1.54474)
Example [3]
Real position:     (x, y, z) = (-0.00079, 0.00046, -0.48526)
Predicted position (x, y, z) = (0.40150, 0.33103, -0.87378)
Example [4]
Real position:     (x, y, z) = (0.00022, 0.00044, -0.15138)
Predicted position (x, y, z) = (0.27533, 0.88060, 0.36802)
```

Figure 33 : Comparaison entre les résultats prévus par le réseau de neurones et les vraies positions des sources ponctuelles en utilisant les hologrammes numériques pour entraîner le modèle

Après ces résultats on a décidé de séparer les prédictions, une fois qu'on a des différents ordres de grandeur dans les intervalles, pour les positions x et y ont d'ordre millimétrique et pour les positions z ont d'ordre centimétrique. Pour implémenter cette modification, il était nécessaire de modifier la dernière couche du réseau des neurones, la couche de sortie, pour contenir un numéro de nœuds égal aux positions souhaitées, ainsi que modifier la matrice de cibles (*target array*).

Donc, on a fait autre fois l'entrainement et le tableau suivant affiche les résultats de la moyenne de cinq entraînements.

Tableau 7 : Erreurs pour le problème de régression des positions x et y et z

	Prédictions faites par un réseau de neurones entrainé où la dernière couche de sortie a deux sorties (x et y)		Prédictions faites par un réseau de neurones entrainé où la dernière couche de sortie a une sortie (z)	
	Train	Test	Train	Test
Root Mean Squared Error	0.876	0.925	0.560	0.571
Mean Squared Error	0.770	0.858	0.322	0.334
Mean Absolute Error	0.770	0.858	0.322	0.334

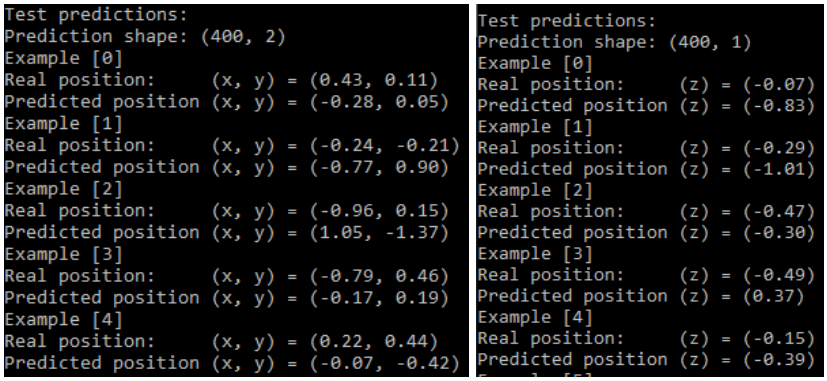

Figure 34 : Exemples de prédictions obtenus après l'entraînement du réseau de neurones, à gauche le réseau prévoit les deux sorties (positions x et y) et à droite une sortie (position z)

Ensuite, pour la distribution de Wigner, on a fait les mêmes tests antérieurs. L'objectif était de faire le calcul de la DW pour tous les hologrammes de la base des données utilisé dans le test précèdent pour avoir une comparaison plus fidèle, cependant, en raison du temps de calcul, où pour calculer la DW de 500 hologrammes on a pris 45 minutes, jusqu'au moment où ce rapport a été écrit on a fait des tests avec la DW des 500 premiers hologrammes de la base précédente de 2000 hologrammes.

On rappelle que les deux paramètres importants pour le calcul de la DW, comme présenté dans le chapitre 6, sont la taille de la fenêtre et les fréquences spatiales, où on avait une fenêtre de taille égal à 9 et des fréquences spatiales entières de -4 jusqu'à 3. On remarque autre fois qu'après calculer la DW d'un hologramme on a un résultat pour chaque fréquence, 8 fréquences au total. Pour le cas présent, on a utilisé des distributions de Wigner pour la fréquence spatiale égale à 0.

Tableau 8 : Erreurs pour le problème de régression des positions x, y et z d'une base des distributions de Wigner des hologrammes numériques générées par seulement 1 source ponctuelle

	Train	Test
Root Mean Squared Error	0.777	0.769
Mean Squared Error	0.658	0.645
Mean Absolute Error	0.658	0.645

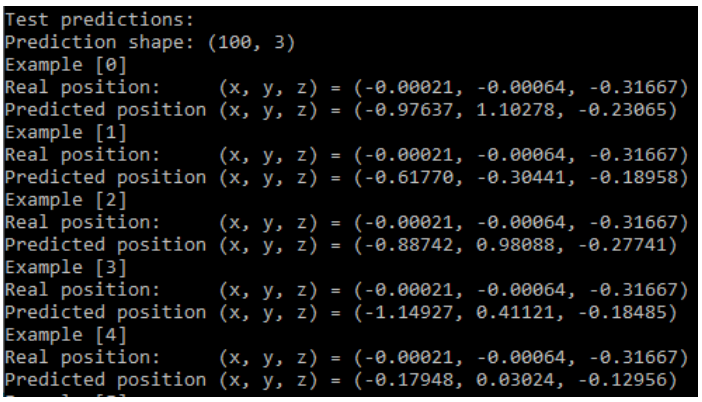

Figure 35 : Comparaison entre les résultats prévus par le réseau de neurones et les vraies positions des sources ponctuelles en utilisant les distributions de Wigner pour entraîner le modèle

On a aussi divisé le problème comme dans le cas des hologrammes numériques, les résultats sont présentés ci-dessous :

Tableau 9 : Erreurs pour le problème de régression des positions x et y et z

	Prédictions faites par un réseau de neurones entrainé où la dernière couche de sortie a deux sorties (x et y)		Prédictions faites par un réseau de neurones entrainé où la dernière couche de sortie a une sortie (z)	
	Train	Test	Train	Test
Root Mean Squared Error	0.693	0.687	0.643	0.677
Mean Squared Error	0.503	0.493	0.435	0.476
Mean Absolute Error	0.503	0.493	0.435	0.476

```
Test predictions:
Prediction shape: (100, 2)
Example [0]
Real position:     (x, y) = (-0.21, -0.64)
Predicted position (x, y) = (-0.42, 0.07)
Example [1]
Real position:     (x, y) = (-0.21, -0.64)
Predicted position (x, y) = (-0.23, -0.48)
Example [2]
Real position:     (x, y) = (-0.21, -0.64)
Predicted position (x, y) = (-0.26, 0.15)
Example [3]
Real position:     (x, y) = (-0.21, -0.64)
Predicted position (x, y) = (-0.40, -0.06)
Example [4]
Real position:     (x, y) = (-0.21, -0.64)
Predicted position (x, y) = (-0.47, -0.09)
```

```
Test predictions:
Prediction shape: (100, 1)
Example [0]
Real position:     (z) = (-0.32)
Predicted position (z) = (0.74)
Example [1]
Real position:     (z) = (-0.32)
Predicted position (z) = (-0.81)
Example [2]
Real position:     (z) = (-0.32)
Predicted position (z) = (0.59)
Example [3]
Real position:     (z) = (-0.32)
Predicted position (z) = (-0.57)
Example [4]
Real position:     (z) = (-0.32)
Predicted position (z) = (-0.58)
```

Figure 36 : Exemples de prédictions obtenus après l'entraînement du réseau de neurones, à gauche entraînement pour 2 sorties (positions x et y) et à droite pour 1 sortie (position z)

Conclusion

Le but principal de ces tests était de valider ou non le domaine dans lequel il était préférable de travailler, c'est-à-dire, avec les hologrammes numériques ou leurs distributions de Wigner. Cependant, on n'ose pas à fixer une réponse basée sur les tests effectués jusqu'au présent dans ce rapport, car lorsque on analyse les tableaux avec les erreurs, c'est difficile d'arriver à une conclusion, ainsi que cette comparaison est entre des bases de données de tailles différents. En revanche, on peut encore faire des autres conclusions et pistes.

La première conclusion concerne le problème des échelles des positions x, y et z, où c'est possible observer que quand on a entraîné le réseau de neurones avec ces 3 positions comme sortie, les résultats pour les positions x et y sont très éloignés de l'échelle. Ce problème peut être résolu en normalisant les valeurs, tout restant dans le même ordre de grandeur, cependant, il faut analyser si cette action peut influencer négativement le problème.

Une deuxième conclusion concerne l'utilisation des réseaux des neurones pour résoudre le problème de régression. Après étudier plus sur le sujet, on a pris que les réseaux de neurones ne sont pas souvent utilisés dans les problèmes de régression, il n'y a pas beaucoup de références et études, les réseaux de neurones sont plutôt étudiés et utilisés pour résoudre des problèmes de classification.

Finalement, cette étude qui cherche à trouver les positions des sources ponctuelles est encore loin d'être terminée. Il existe encore de nombreux autres tests qui peuvent être effectués, comme l'essai d'autres algorithmes de régression, la génération des hologrammes numériques à partir de scènes plus grandes et avec plus des sources, la génération des hologrammes plus grandes et de plus haute résolution.

7.3. Conclusion

Ce chapitre voudrait plutôt de répondre la question posée au début du chapitre et donc analyser la pertinence de la distribution de Wigner, où pour l'implémentation on était plus intéressés au problème de régression, parce que, sous un certain aspect, l'objectif finale est similaire à celui de la séparation des sources qui cherche à la fin aussi identifier la contribution de chaque.

Cependant, malgré les grands apprentissages et les études nécessaires faites pour réaliser ces mises en œuvre, on n'est pas encore arrivé aux conclusions souhaitées au début, où il faut remarquer autre fois les deux conclusions et problématiques présentées : la quantité insuffisante des données à être analyser, tant pour la base des hologrammes numériques que pour ces distributions de Wigner, pour entrainer le réseau de neurones et aussi l'utilisation d'une réseau de neurones, le Perceptron Multicouche, qui s'est révélé non suffisant et pas assez puissante pour résoudre le problème.

En testant différentes possibilités, on a découvert ce qui fonctionne et ce qui ne fonctionne pas à mesure qu'on a avancé sur les sujets, malgré les résultats incomplets et les questions non encore répondues, et il y avait des autres approches qui n'ont été pas adoptés. La prochaine approche adoptée à être à suivre, mais qui n'a pas été adopté en raison de la fin de stage, au moins jusqu'au moment où ce rapport a été rédigé, c'est l'étude d'un algorithme de régression sur les paramètres d'une surface de type *Spline* ou NURBS. Dans un premier temps, on a décidé d'utiliser des *B-Splines* car le nombre de paramètres est moins important que pour des NURBS ; ainsi, on n'aura que les positons des points de contrôle comme paramètres.

Ce qu'on conclut avec ce chapitre n'est pas les résultats attendus, en revanche on a commencé à étudier un sujet encore peu analysé et on a pris des approches encore peu testées.

III – Conclusion

Résumé du travail réalisé et vision critique sur celui

Tout d'abord, ce projet de fin d'étude était un stage plus orienté sur la recherche, notamment dans un domaine innovant et particulier. Ainsi, contrairement à d'autres stages où il y a un plan bien structuré à suivre, ce stage était caractérisé par le fait d'avoir des objectifs sur des sujets non encore étudiés se situant dans un domaine de connaissance aux applications très récentes, celui de l'holographie numérique. Ainsi, les objectifs ont été ajustés en fonction des progrès et des découvertes.

L'objectif initial de cette étape était d'étudier l'application ou l'extension des algorithmes de séparation des sources au problème de la superposition des ondes monochromatiques provenant d'objets animés. Où l'analyse de la distribution de Wigner serait un point clé du processus.

Cependant, au fur et à mesure de l'avancement de l'étape et de l'étude, nous avons remarqué que cette séparation n'était pas possible pour un niveau de stage et nous avons décidé de tester d'autres approches encore dans le domaine de l'holographie. Comme les méthodes pour localiser les sources, des implémentations de réseaux des neurones et les implémentations de algorithmes de régression.

Finalement, même que les résultats obtenus n'étaient pas les attendues, le stage a été caractérisé comme une première étude dans cet objectif de séparer les sources du plan de l'hologramme, de localiser les sources, ce qui a permis de tracer des pistes pour de futures études. Enfin, [18] contient les codes et les mises en œuvre développées pendant le stage.

Bilan personnel

Par rapport au bilan personnel, ce projet de fin d'études m'a permis d'acquérir de nombreuses connaissances dans plusieurs domaines, non seulement en informatique, comme dans le domaine de l'optique et de l'holographie numérique, dans le domaine de la séparation des sources et dans le domaine de l'apprentissage profond, domaine auquel je me suis actuellement très intéressé.

Il faut admis qu'Il y avait aussi certaines difficultés à surmonter qui ont causé le retard dans le développement du projet. La plus grande difficulté a été occasionné par le confinement pris avec l'objectif de combattre la propagation de la pandémie de covid-19 qui on a obligé à s'adapter au travail à distance. Et ce qui m'a apporté beaucoup de difficultés au début, puisque nous étions au début du stage, et on avait des questions théoriques complexes qui seraient beaucoup plus faciles à discuter par une communication en face à face.

Enfin, malgré les difficultés, mon expérience avec ce projet d'étude final a été très positive, j'ai été bien accueilli par l'équipe de b<>com, j'ai eu de bonnes relations avec mes encadrants et j'avais une première expérience avec le domaine de la recherche, un domaine avec lequel je n'avais pas encore eu de contact.

IV – Références

[1] P. LOBAZ, «COMPUTER GENERATED HOLOGRAPHY at University of West Bohemia,» 2018. [En ligne]. Available: http://holo.zcu.cz/. [Accès le 01 04 2020].

[2] D. BLINDER, A. AHAR, S. BETTENS, T. BIRNBAUM, A. SYMEONIDOU, H. OTTEVAERE, C. SCHRETTER et P. SCHELKENS, «Signal processing challenges for digital holographic video display systems,» vol. 70, pp. 114-130, 2019.

[3] A. GILLES, P. GIOIA, R. COZOT et L. MORIN, «Hybrid approach for fast occlusion processing in computer-generated hologram calculation,» *Applied optics, Optical Society of America,* pp. 5459-5470, 2016.

[4] A. GILLES, P. GIOIA, R. COZOT et L. MORIN, «Génération Numérique d'Hologrammes : État de l'Art,» *Revue Electronique Francophone d'Informatique Graphique, Association Française d'Informatique Graphique,* vol. 70, pp. 23-25, 2015.

[5] R. BADEAU, «Séparation de sources audio,» [En ligne]. Available: https://perso.telecom-paristech.fr/grenier/mva/Sap-Separation-Transp.pdf.

[6] Wikipedia, «Méthode de séparation avuegle de source,» [En ligne]. Available: https://fr.wikipedia.org/wiki/M%C3%A9thode_de_s%C3%A9paration_aveugle_de_source. [Accès le 01 04 2020].

[7] J. HATTAY, D. LEBRUN et S. BELAID, «APPLICATION DE LA SEPARATION AVEUGLE DESOURCES POUR LA SUPPRESSION D'IMAGESJUMELLES EN HOLOGRAPHIE NUMERIQUEDANS L'AXE,» chez *3ème rencontre francophone d'holographie numérique appliquée à la métrologie desfluides*, Ecully, 2014.

[8] F. FENG, «Séparation aveugle de source : de l'instantané au convolutif,» 2017.

[9] «scikit-learn,» [En ligne]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html. [Accès le 2020 04 01].

[10] K. YIN, «Step-by-Step Signal Processing with Machine Learning: PCA, ICA, NMF for source separation, dimensionality reduction,» 12 11 2019. [En ligne]. Available: https://towardsdatascience.com/step-by-step-signal-processing-with-machine-learning-pca-ica-nmf-8de2f375c422?gi=8d78853ca4f3. [Accès le 01 04 2020].

[11] «Holographie Numérique,» [En ligne]. Available: https://dossier.univ-st-etienne.fr/fcorinne/public/holonum/imagerie3d_holonum.html.

[12] Wikipedia, «Wigner distribution function,» [En ligne]. Available: https://en.wikipedia.org/wiki/Wigner_distribution_function. [Accès le 01 04 2020].

[13] G. CRISTOBAL, C. CONZALO et J. BESCOS, «Image filtering and analysis through the Wigner Distribution».

[14] Y. M. ZHU, F. PEYRIN, R. GOUTTE et M. AMIEL, «Analyse spectrale locale de l'image par transformation de Wigner-Ville,» *Traitement du Signal,* vol. 9.

[15] S. GABARDA, «DIGITAL IMAGE PROCESSING IN THE SPATIAL-SPATIAL/FREQUENCY DOMAIN,» 02 01 2018. [En ligne]. Available: https://notebooks.azure.com/salva/projects/Digital-Image-Processing/. [Accès le 01 06 2020].

[16] A. BEGHDADI et R. IORDACHE, «Image Quality Assessment Using the Joint Spatial/Spatial-Frequency Representation,» *EURASIP Journal on Applied Signal Processing,* vol. 2006, pp. 1-8.

[17] «Keras : the Python deep learning API,» [En ligne]. Available: https://keras.io/. [Accès le 01 06 2020].

[18] F. L. ARAUJO AMARAL, «Github repository,» 2020. [En ligne]. Available: https://github.com/fernandolucasaa/computerGeneratedHolography.

INSA Rennes
20 Avenue des Buttes de Coësmes
CS 70839
35708 Rennes Cedex 7

Tél. +33 (0) 2 23 23 82 00
Fax +33 (0) 2 23 23 83 96

www.insa-rennes.fr