1. Introduction

A QUI S'ADRESSE CE LIVRE ?

Avec ce livre, nous souhaitons vous proposer une promenade enchantée, mais pragmatique, dans le monde merveilleux du traitement d'images :

— *Enchantée*, car le traitement d'images est un univers vaste et captivant, reposant sur des bases théoriques diversifiées mais solides, bases qui servent aussi bien à modéliser les problèmes approchés qu'à proposer des algorithmes efficaces pour les résoudre. Le nombre d'applications amusantes et/ou pratiques, que l'on peut imaginer et mettre en œuvre en traitement d'images, est potentiellement infini.
— *Pragmatique*, car nous ne souhaitons pas réaliser un simple survol des différents formalismes du traitement d'images. Nous aurons à cœur de rendre tangible chacune des techniques abordées, en « traduisant » systématiquement les points théoriques présentés sous la forme de programmes *C++*, fonctionnels et utilisables.

Cette intrication entre théorie et implémentation constitue l'essence même de ce livre, et son contenu se destine donc à des profils variés de lecteurs :

— Les programmeurs en *C++*, débutants ou initiés, amateurs de mathématiques appliquées, souhaitant étudier la discipline du traitement d'images, pour à terme développer des applications concrètes dans ce domaine.
— Les mathématiciens, traiteurs d'images ou de signaux, souhaitant se confronter au problème pratique de l'implémentation logicielle des différentes méthodes du domaine en *C++*, un langage reconnu universellement comme étant générique, puissant et rapide.
— Les enseignants ou étudiants en informatique et mathématiques appliquées, qui retrouveront dans les ateliers proposés dans ce livre des briques de base pour élaborer ou effectuer leur travaux pratiques.
— Les développeurs ou chercheurs confirmés, pour qui la bibliothèque *CImg* utilisée comme support dans ce livre, constituera un outil idéal de prototypage rapide et efficace de nouveaux algorithmes de traitement d'images.

Il est important de souligner que nous n'utiliserons que des concepts simples du langage *C++* et que les programmes proposés seront donc assez lisibles pour être retranscrits facilement dans d'autres langages si nécessaire. La bibliothèque *CImg*, sur laquelle nous nous reposons, est développée depuis plusieurs années par des chercheurs en informatique et en traitement d'images (du CNRS, de l'INRIA, et de l'Université), principalement dans le but de réaliser du prototypage rapide de nouveaux algorithmes. Elle est également utilisée comme outil de développement dans les travaux pratiques de plusieurs cours dispensés au niveau licence, master ou en école d'ingénieurs. Son utilisation est donc tout à fait adaptée à l'approche pédagogique que nous souhaitons développer ici.

Nous avons structuré ce livre pour permettre d'une part, une appropriation rapide des concepts de la bibliothèque *CImg* (ce qui motive la première partie cet ouvrage), et d'autre part, son utilisation pratique dans de nombreux domaines du traitement d'images, par l'intermédiaire d'ateliers (constituant la seconde partie du livre). L'ensemble des exemples proposés, allant de l'application la plus simple jusqu'à des algorithmes plus évolués, vous permettra de développer un savoir-faire conjoint en théorie, algorithmique et implémentation dans le domaine du traitement d'images. L'ensemble des codes sources publiés dans cet ouvrage sont également disponibles en format numérique, sur le dépôt
`https://github.com/CImg-Image-Processing-Book`.

Pourquoi faire encore du traitement d'images aujourd'hui ?

L'image est partout. Elle est le support de nombreuses informations et est utilisée dans des applications variées comme le diagnostic médical par *IRM*, scanner ou imagerie scintigraphique, l'étude de la déforestation par imagerie satellitaire, la détection de comportements anormaux dans des foules à partir d'acquisitions vidéos, la retouche photographique et les effets spéciaux, ou encore la reconnaissance d'écriture, pour ne citer que quelques exemples. Aujourd'hui, même sans s'en rendre compte, on utilise quotidiennement des algorithmes de traitement d'images : amélioration de contraste automatique de nos photos de vacances préférées, détection de plaques minéralogiques à l'entrée du parking de notre supermarché favori, détection automatique des visages ou des lieux dans les photographies que l'on poste sur les réseaux sociaux, etc.
Le traitement d'images fonde ses bases sur un socle théorique solide, issu du traitement du signal et de la théorie de l'information de Shannon [38]. Traiter une image, c'est extraire de celle-ci une ou plusieurs information(s) pertinente(s) pour un problème posé. Cette information peut être : la taille d'un objet, sa localisation, son mouvement, sa couleur, voire son identification. Extraire cette information peut nécessiter des

pré-traitements préalables de l'image, si celle-ci est trop bruitée ou mal contrastée.
Le traitement d'images prend son essor dans les années 1960, avec l'avènement des ordinateurs et le développement (ou la redécouverte) de techniques de traitement du signal (la transformée de Fourier par exemple). Dès lors, dans tous les domaines que cet ouvrage se propose d'aborder dans sa seconde partie, de nombreux algorithmes ont été développés, toujours plus performants et précis, pouvant traiter des images de taille de plus en plus importante et en nombre conséquent.

Parallèlement à ce développement, depuis les années 2000, l'apprentissage automatique, et plus particulièrement l'apprentissage profond (*Deep Learning*), obtient des performances inégalées en vision par ordinateur, surpassant même dans certains domaines les capacités humaines. Ainsi, un réseau de neurones profond est aujourd'hui capable d'annoter une scène en identifiant l'ensemble des objets présents, de coloriser de manière réaliste une image initialement en niveaux de gris, ou encore de restaurer des images fortement bruitées.

Pourquoi alors s'intéresser encore au traitement d'images « classique » ? Le passage du traitement d'images à l'apprentissage profond s'accompagne d'un changement de paradigme dans le traitement des données : les techniques classiques calculent d'abord des caractéristiques sur les images originales (chapitre 6) et les utilisent ensuite pour le traitement à proprement parler (segmentation : chapitre 7, suivi : chapitre 8, ...). Le calcul de ces caractéristiques est éventuellement précédé de prétraitements (chapitres 3, 4 et 5) facilitant leur extraction.
A l'inverse, l'apprentissage profond *apprend* ces caractéristiques, le plus souvent par l'intermédiaire de couches de convolution dans les réseaux profonds, et utilise ces caractéristiques apprises pour effectuer les traitements.

Et c'est là que la différence majeure intervient : pour effectuer la tâche qui lui incombe, le réseau profond doit *apprendre*... Et pour cela il doit disposer d'une base d'apprentissage constituée de plusieurs milliers (voire millions) d'exemples, lui indiquant ce qu'il doit accomplir. Par exemple, pour être capable de dissocier des images de chats et de chiens, le réseau doit apprendre sur des milliers de couples (x, y), où x est une image de chat ou de chien et y est le label associé, avant de pouvoir se prononcer sur une image inconnue qui lui est présentée (Fig. 1.1).

Or, obtenir ces données est loin d'être chose aisée. Si, dans certains domaines (comme la reconnaissance d'objets), on dispose déjà de bases de données labelisées bien établies (par exemple ImageNet [1], composée de plus de 14 millions d'images différentes réparties dans 21800 catégories), il est le plus souvent très difficile, voire

1. http ://www.image-net.org

impossible, de constituer une base d'apprentissage suffisamment bien fournie et constituée, pour entraîner un réseau de neurones à résoudre un problème donné en traitement d'images.

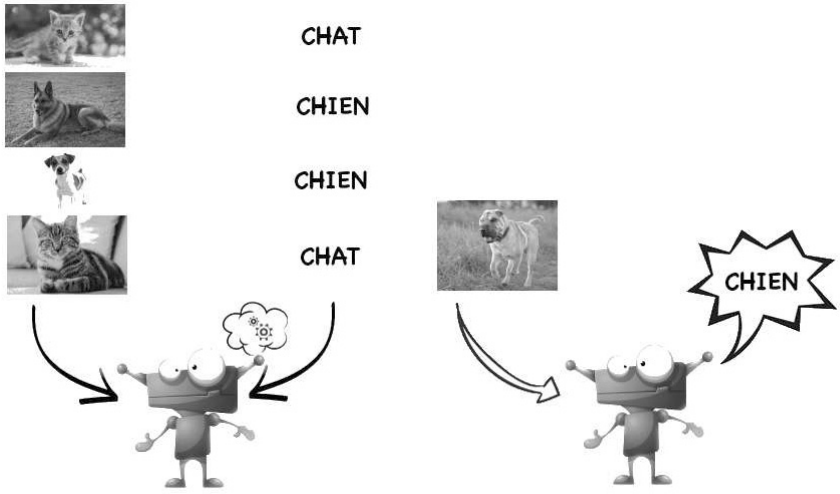

FIGURE 1.1 – Principe de l'apprentissage en traitement d'images.

De plus, par delà le problème de disponibilité des données d'entrainement, les réseaux de neurones profonds nécessitent le plus souvent, dans leur phase d'apprentissage, une puissance de calcul et des ressources matérielles importantes (via l'utilisation de GPU - *Graphics processing unit* et de TPU - *Tensor Processing Unit*). Puissance que l'étudiant, ou l'ingénieur en quête d'un résultat rapide, n'aura pas nécessairement à sa disposition.

Aussi, même si le domaine de l'apprentissage des réseaux de neurones profonds est en pleine expansion depuis une vingtaine d'années, et fournit des résultats impressionnants, le traitement d'images classique n'a certainement pas dit son dernier mot !

POURQUOI FAIRE DU TRAITEMENT D'IMAGES EN *C++* ?

Parmi la pléthore de langages de programmation existants, le langage *C++* possède les avantages suivants :

— C'est un langage *multi-paradigmes*, *bien établi*, et *populaire*. Il est généralement enseigné dans les universités et écoles d'ingénieurs proposant des formations

liées à l'informatique. Il touche donc un large public, qui saura l'utiliser pour écrire des programmes abordant un spectre large de problématiques, afin de résoudre des tâches variées, aussi bien « bas-niveau » que « haut-niveau ».
— Le *C++* est un langage *compilé*, qui produit des binaires très *optimisés*. En traitement d'images, les données à traiter sont souvent volumineuses : une image de résolution standard possède plusieurs millions de valeurs à analyser et il est donc important d'avoir des programmes suffisamment rapides pour itérer sur ces valeurs en un temps raisonnable, ce que ne permettent pas toujours les langages interprétés. En Python par exemple, la plupart des modules existants pour le traitement d'images sont implémentés en *C*/*C++*, pour des questions de rapidité (si vous avez déjà testé le parcours des pixels d'une image avec une boucle écrite en Python « pur », vous devinez pourquoi !).
— L'utilisation des *templates C++* facilite la manipulation de données images *génériques*, par exemple lorsque les pixels des images que l'on veut traiter peuvent-être de types différents (booléens, entiers, nombres flottants, etc.).

Pourquoi recourir à une bibliothèque externe ?

Classiquement, la programmation d'algorithmes de traitement d'images requiert la possibilité de pouvoir importer/exporter des images sous la forme de tableaux de valeurs, ces images étant stockées sur le disque dans des formats normalisés (*JPEG*, *PNG*, *TIFF*, ...) et de pouvoir les afficher à l'écran. Or, aucune fonctionnalité de ce genre n'est présente dans la bibliothèque standard du *C++*, ni pour le chargement ou la sauvegarde de fichiers images, ni pour l'analyse et le traitement d'images, ni pour leur visualisation.

Il faut prendre conscience du fait qu'écrire de telles fonctionnalités de prise en charge de formats d'images à partir de zéro est une tâche fastidieuse. Aujourd'hui, les formats de fichier classiques pour stocker des images ont une structure très complexe : les images sont le plus souvent stockées sur le disque sous forme compressée, chaque format utilisant sa propre méthode de compression, destructive ou non. En pratique, à chaque format d'image est déjà associée une bibliothèque tierce évoluée (par exemple `libjpeg`, `libpng`, `libtiff`, ...), focalisée sur le chargement et la sauvegarde des données images dans chacun de ces formats. De même, l'action d'afficher une image dans une fenêtre est une tâche plus complexe qu'il n'y paraît, et qui se réalise toujours via l'utilisation de bibliothèques tierces, soit spécialisées dans l'affichage « brut » (`libX11`, `Wayland` sous Unix, ou `gdi32` sous Windows), soit dans l'affichage d'interfaces graphiques plus évoluées, à l'aide de *widgets* (`GTK`, `Qt`, ...).

Par ailleurs, les algorithmes basiques de traitement eux-mêmes ne sont pas toujours triviaux à implémenter, surtout lorsqu'on en veut des versions optimisées. Pour

toutes ces raisons, on a le plus souvent recours à une bibliothèque tierce de haut-niveau spécialisée dans le traitement d'images, pour travailler confortablement dans ce domaine en *C++*.

Quelles bibliothèques *C++* pour le traitement d'images ?

Une bibliothèque de traitement d'images pertinente doit permettre à minima la lecture/écriture de données images dans les formats de fichiers les plus courants, l'affichage de ces images et doit proposer quelques algorithmes parmi les plus classiques pour leur traitement. Parmi les dizaines de choix existants, nous proposons cette liste épurée de bibliothèques, vérifiant ces conditions minimales :

CImg, *ITK*, *libvips*, *Magick++*, *OpenCV*, et *VTK*.

Pourquoi seulement ces six bibliothèques ? Parce que ce sont des bibliothèques bien établies (existantes toutes depuis plus de 15 ans), amplement utilisées dans la communauté du traitement d'images et qui ont donc fait leurs preuves en terme de performances et de robustesse. Ce sont aussi des bibliothèques au développement toujours actif, libres d'utilisation, multi-plateformes, et suffisamment fournies pour permettre l'élaboration de programmes complexes et diversifiés en traitement d'images. Nous avons volontairement mis de côté les bibliothèques qui sont soit distribuées sous licence propriétaire, soit trop jeunes, soit non maintenues activement, ou dont le domaine d'application est trop restrictif (par exemple des bibliothèques juste capables de lire/écrire des images dans quelques formats de fichiers, ou qui proposent un panel trop limité d'algorithmes de traitement). Cette diversité de choix reflète les domaines applicatifs variés qui ont été visés initialement par les auteurs de ces différentes bibliothèques.

Notre sélection peut par exemple se résumer de la façon suivante :

— *CImg* (*Cool Image*) a été créée au début des années 2000 par l'Institut National de Recherche en Sciences et Technologies du Numérique (*Inria*), institut public français de recherche. C'est une bibliothèque libre qui a été conçue dans un cadre de recherche en algorithmique du traitement d'images, afin de permettre à ses utilisateurs (à l'origine, principalement des chercheurs, des enseignants et des doctorants) d'imaginer, d'implémenter facilement et tester de nouveaux algorithmes originaux de traitement d'images, même en partant de zéro. *CImg* est téléchargeable à l'adresse `http://cimg.eu`.

— *ITK* (*Insight Segmentation and Registration Toolkit*) est une bibliothèque rendue disponible dès 2001, créée initialement dans un but d'analyse et de traitement d'images médicales (le projet a été initié par la Bibliothèque Américaine Nationale de Médecine). Cette spécialisation médicale est toujours d'actualité, et *ITK*

est utilisée aujourd'hui principalement pour visualiser, segmenter et recaler des images médicales. *ITK* est téléchargeable à l'adresse `https://itk.org`.

— *libvips* est une bibliothèque écrite principalement en *C*, dans les années 1990, spécialisée à l'origine dans le traitement de très grandes images. Elle fait partie d'un cadriciel plus large (nommé *VIPS*) comprenant également un logiciel proposant une interface graphique pour la visualisation et le traitement de grandes images. C'est une bibliothèque bien adaptée lorsque les images à analyser ou traiter sont très volumineuses, typiquement, lorsque l'ensemble de ces images est plus large que la mémoire disponible sur l'ordinateur. *libvips* est téléchargeable à l'adresse `https://libvips.github.io/libvips`.

— *Magick++* est l'une des plus anciennes bibliothèques de notre liste, puisqu'elle a été conçue à la fin des années 1980. C'est une bibliothèque dont l'objectif principal était la conversion de formats entre images $2D$ en couleur ou en niveaux de gris. Il est cependant difficile de l'utiliser pour écrire des algorithmes de traitements sur des images plus génériques (par exemple, des images volumiques $3D$ ou à nombre de canaux supérieurs à 4). *Magick++* est téléchargeable à l'adresse
`https://imagemagick.org/Magick++`.

— *OpenCV* est une bibliothèque développée au début des années 2000, qui se focalise sur la *vision par ordinateur*, domaine au croisement du traitement d'images et de l'intelligence artificielle cherchant à imiter la vision humaine. *OpenCV* est une bibliothèque très populaire, et propose un grand ensemble d'algorithmes déjà implémentés, souvent de manière très optimisée. Elle est idéale pour un utilisateur souhaitant enchaîner des briques algorithmiques élémentaires toutes faites, afin de construire un pipeline de traitement efficace. En contrepartie, l'utilisation d'*OpenCV* pour le *prototypage* de nouveaux algorithmes est moins aisée : les algorithmes déjà présents fonctionnent en effet comme des « boîtes noires », difficilement modifiables, et l'*API* relativement complexe de la bibliothèque ne facilite pas vraiment l'écriture de nouveaux algorithmes à partir de zéro. *OpenCV* est téléchargeable à l'adresse `https://opencv.org`.

— *VTK* est une bibliothèque créée au milieu des années 1990, spécialisée dans le traitement et la visualisation de maillages $3D$. Elle est donc centrée sur le traitement et la visualisation de données structurées sous forme de graphes ou de maillages, plutôt que sur le traitement d'images plus classiques, définies sur des grilles d'échantillonnages régulières. Cette bibliothèque est distribuée par la société américaine *Kitware, Inc*, qui développe également la bibliothèque *ITK*. *VTK* et *ITK* sont utilisées conjointement, principalement pour l'analyse et la visualisation d'images médicales. *VTK* est téléchargeable à l'adresse `https://vtk.org`.

a) *CImg*

b) *ITK*

c) *VTK*

d) *OpenCV*

e) *Magick++*

FIGURE 1.2 – Logos des principales bibliothèques *C++* libres pour le traitement d'images (à noter que la bibliothèque *libvips* ne possède pas de logo officiel).

POURQUOI AVONS NOUS ADOPTÉ *CImg* POUR CE LIVRE ?

CImg est une bibliothèque *C++* légère, qui a vu le jour il y a plus d'une vingtaine d'années. C'est une bibliothèque libre, dont le code source est ouvert (distribué sous licence *open-source CeCILL-C*), et qui fonctionne sur différents systèmes d'exploitation (*Windows*, *Linux*, *Mac OSX*, *FreeBSD*, etc.). *CImg* donne au programmeur accès à des classes et des méthodes pour manipuler des images ou des séquences d'images, une image étant définie ici au sens large du terme, comme un tableau volumique ayant jusqu'à trois coordonnées spatiales (x,y,z) et contenant des valeurs vectorielles de dimensions et de types quelconques. La bibliothèque permet de décharger le programmeur des tâches usuelles « bas-niveau » de manipulation des images sur un ordinateur, comme par exemple la gestion des allocations mémoire et des entrées-sorties, l'accès aux valeurs des pixels, l'affichage d'images, l'interaction de l'utilisateur avec celles-ci, etc. Elle offre également un éventail assez complet d'algorithmes usuels de traitement, dans plusieurs domaines parmi lesquels :

— les opérations arithmétiques entre images, et les applications de fonctions mathématiques usuelles : *abs(), cos(), exp(), sin(), sqrt(),...*
— le calcul statistique : extraction du minimum, du maximum, de la moyenne, de la variance, de la valeur médiane, ...
— la manipulation des espaces couleurs : RGB, HSV, YC_bC_r, $L^*a^*b^*$, ...
— les transformations géométriques : rotation, miroir, extraction de sous-parties, redimensionnement, déformation, ...
— le filtrage : convolution/corrélation, transformée de Fourier, calcul du gradient et du laplacien, filtres récursifs, filtres morphologiques, lissage anisotrope, ...