Approches métrospection pour la détection de dérives de procédés

T. Alcaire

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ GRENOBLE ALPES

École doctorale : EEATS - Electronique, Electrotechnique, Automatique, Traitement du Signal (EEATS)
Spécialité : Nano électronique et Nano technologies
Unité de recherche : Laboratoire des Technologies de la Microélectronique

Approches métrospection pour la détection de dérives de procédés

Metrospection approaches for process drift detection

Présentée par :

Thomas ALCAIRE

Direction de thèse :

Jean-Hervé TORTAI CHARGE DE RECHERCHE, Université Grenoble Alpes	Directeur de thèse
Sébastien SOULAN Maître de conférences, Université Grenoble Alpes	Co-encadrant de thèse
Delphine Le Cunff STMicroelectronics	Co-directrice de thèse

Rapporteurs :

Tatiana NOVIKOVA
INGENIEUR HDR, Ecole Polytechnique
Yves JOURLIN
PROFESSEUR DES UNIVERSITES, Université Jean Monnet

Thèse soutenue publiquement le **10 janvier 2023**, devant le jury composé de :

Jean-Hervé TORTAI CHARGE DE RECHERCHE, CNRS	Directeur de thèse
Tatiana NOVIKOVA INGENIEUR HDR, Ecole Polytechnique	Rapporteure
Yves JOURLIN PROFESSEUR DES UNIVERSITES, Université Jean Monnet	Rapporteur
Etienne GHEERAERT PROFESSEUR DES UNIVERSITES, Université Grenoble Alpes	Examinateur
Enrique GARCIA-CAUREL DOCTEUR EN SCIENCES, Ecole Polytechnique	Examinateur
Patrick SCHIAVONE DIRECTEUR DE RECHERCHE, CNRS	Examinateur

Invités :

Delphine Le Cunff
DOCTEUR EN SCIENCES, STMicroelectronics
Sébastien Soulan
Maitre de Conférence, Université Grenoble Alpes

Remerciements

Liste des abréviations

AdaGrad	Adaptative Gradient
ADC	Automatic Defect Classification
ANR	Agence Nationale de la Recherche
BAB	Bande A Bande
BEOL	Back End Of Line
BF	Bright Field
CCD	Charge Coupled Device
CD	Critical Dimension
CEA	Commissariat à l'Energie Atomique et aux Energies Alternatives
CMOS	Complementary Metal-Oxide-Semiconductor
CMP	Chemical Mechanical Polishing
CNN	Convolutional Neural Network
DF	Dark Field
DNN	Deep Neural Network
EM	Energy Matrix
EWS	Electrical Wafer Sorting
FCNN	Fully Connected Neural Network
FD-SOI	Fully Depleted Silicon On Insulator
FEOL	Front End Of Line
FLOPS	Floating Point Operations Per Second
GPS	Global Positioning System
GPU	Graphical Processing Unit
HVM	High Volume Manufacturing
IA	Intelligence Artificielle
BIPM	Bureau International des Poids et Mesures

IFTA	Iterative Fourier Transform Analysis
IGBT	Insulated Gate Bipolar Transistor
IRT	Institut de Recherche Technologique
ITRS	International Technology Roadmap for Semiconductors
LED	Light-Emitting Diode
LETI	Laboratoire d'Electronique et de Technologie de l'Information
LR	Learning Rate
LTM	Laboratoire des Technologies de la Microélectronique
MM	Matrice de Mueller
MPL	MacroPhotoLuminescence
MSE	Mean Square Error
NN	Neural Network
OCD	Optical Critical Dimension
ODIFF	Optical DIFFusers
PECVD	Plasma Enhanced Chemical Vapor Deposition
PL	Photoluminescence
PM	Photomultiplicateur
PSI	Phase-Shifting Interferometer
PWG	Pattern Wafer Geometry
RAM	Random Access Memory
RCNN	Region-based Convolutionnal Neural Network
RCWA	Rigorous Coupled-Wave Analysis
RGB	Red Green Blue
RMSProp	Root Mean Square Propagation
RPN	Region Proposal Network
SAM	Scanning Acoustic Microscopy
SE	Spectroscopic Ellipsometry
SEM	Scanning Electron Microscopy
SVM	Support Vector Machine

SWOT	Strength Weakness Opportunity Threat
TEM	Transmission Electron Miscroscopy
UV	UltraViolet
VRAM	Video Random Access Memory
VSI	Vertical Scanning Interferometry

Table des matières

Introduction générale

L'évolution des technologies dans la microélectronique et la réduction des dimensions des transistors, composants élémentaires des circuits intégrés, entraine l'apparition de nouveaux challenges, que ce soit dans le design des produits, l'intégration, ou le choix des matériaux. Les techniques de suivi de procédés, contrôlant les propriétés des composants, leurs dimensions pour assurer le fonctionnement des produits, doivent alors être capable de répondre aux attentes de ces innovations.

Dans le cadre de la 4ème révolution industrielle, l'utilisation de traitement avancé de données permet d'accélérer l'introduction de nouvelles technologies dans l'industrie des semi-conducteurs. Afin que la métrologie puisse suivre le rythme de ces innovations, il est nécessaire de développer des approches ne reposant pas sur des modèles physiques rigoureux, couteux en temps et en argent, et donc peu réactives aux évolutions régulières imposées par la R&D. En effet, la mise en place d'une étape de contrôle par un équipement de métrologie requiert la création d'une recette spécifique, qui peut dans les cas les plus complexes, nécessiter plusieurs mois pour être opérationnelle. De plus, cette complexification du processus de fabrication induit une augmentation significative du nombre de contrôles et de mesures pour assurer une production fiable, et donc du volume des données générées.

Dans ce contexte, il est alors intéressant d'évaluer l'apport d'approches reposant sur l'utilisation des données brutes obtenues par les équipements de suivi de procédé (métrologie ou inspection/défectivité) via des traitements intelligents, notamment des réseaux de neurones, afin de prendre des décisions process. On réduirait donc grandement le temps nécessaire à la mise en place de solutions de contrôle de procédé, en s'appuyant sur la large quantité de données générées par l'industrie des semi-conducteurs pour entrainer ces algorithmes.

Cette méthode, à la frontière entre la métrologie (utilisation de technique hautement sensible aux dérives) et l'inspection (acquisition à large échelle et haut volume) a alors été nommé « métrospection ». Cette thèse consiste alors à évaluer l'intérêt de cette approche via application à des cas d'intérêts industriels fournis par STMicroelectronics, mais aussi en développant une technique de mesure innovante (imagerie ellipsométrique) avec le Laboratoire des Technologies de la Microélectronique (LTM) en collaboration avec des équipementiers industriels (Horiba/Semilab).

Le 1er chapitre décrit alors le contexte industriel dans lequel s'est déroulé cette thèse, et plus précisément celui de la métrologie industrielle, dont certaines limitations amènent aux objectifs de cette thèse. Les différentes techniques utilisées seront présentées, comme les équipements industriels servant à la récolte des données seront alors décrits, mais aussi les algorithmes d'intelligence artificielle utilisés pour traiter ces dernières.

Le 2ème chapitre présente le traitement de données brutes de spectres pour la détermination de dimensions critiques de structures périodiques. La technique d'ellipsométrie, puis de scatterométrie, utilisées pour obtenir ces spectres, ainsi que le processus de modélisation nécessaire à l'utilisation conventionnelle de cette dernière sont décrits dans une seconde partie. Une troisième partie consiste en l'évaluation de cette approche sur deux cas industriels de complexité croissante.

Le 3ème chapitre décrit les approches de traitement d'images avancés, utilisées pour réaliser des tâches de complexité grandissante, passant de la détection de lignes, à la classification d'images, pour finir par la classification d'objets présents dans ces images. Chaque algorithme utilisé est alors présenté en début de partie, dont l'apport sur la qualité de suivi de process est évalué sur des cas d'intérêts industriels de dérives de procédés.

Le 4ème et dernier chapitre présente la mise en place de la technique d'imagerie ellipsométrie sur un prototype de recherche. L'équipement utilisé, et les protocoles d'acquisitions et de traitements

des données générées par ces dernières sont décrits. Cette technique de mesure innovante est alors évaluée sur de multiples cas de dérives de procédé industriels, liés aux propriétés physiques de matériaux, à des variations d'épaisseurs de couches à large échelle, et à des variations de dimensions de structures gravées.

Chapitre I –
Métrologie Industrielle

Afin de répondre aux défis actuels et produire des architectures et circuits innovants, il est primordial pour une entreprise de tout premier plan comme STMicroelectronics de réaliser une veille technique non seulement sur les avancées technologiques dans le secteur de la microélectronique mais aussi sur les équipements de salle blanche les plus adaptés pour répondre aux besoins de la production et de la recherche, le milieu des semi-conducteurs étant en constante évolution et les avancées technologiques très rapides.

L'objectif de ce premier chapitre est d'introduire le contexte industriel dans lequel s'est déroulé ma thèse ainsi que la place de la métrologie dans l'industrie de la microélectronique et des semi-conducteurs, puis de présenter les diverses techniques et équipements utilisés par ce département pour réaliser le suivi de dérives durant tout le procédé de fabrication des produits. Les données obtenues par ces équipements ont, dans les travaux présentés dans cette thèse, permis d'évaluer l'intérêt d'approches innovantes afin d'améliorer le contrôle de dérives de procédés.

I. Contexte industriel

1. Contexte industriel global

a. Part de marché et croissance

De nos jours, que ce soit dans notre vie courante, ou dans de nombreux domaines d'applications tels que la production d'énergie, l'agriculture, la sécurité et la santé, les composants électroniques sont omniprésents. Ils sont essentiels au fonctionnement des voitures, des ordinateurs, des automates industriels, et sont principalement issus de l'industrie des semi-conducteurs. Sachant qu'un smartphone est composé d'entre 500 et 600 composants, on comprend alors la forte hausse de demande à laquelle doit répondre cette industrie. De plus, la crise du Covid-19 a brutalement fait évoluer les usages, notamment avec l'essor du télétravail. De ce fait, le marché mondial, pesant actuellement entre 500 et 600 milliards de dollar (Figure 1), est appelé à doubler d'ici 2030 pour répondre à un marché du numérique bien plus important. L'électrification des véhicules, la prolifération des datacenters, et la multiplication des objets connectés présent dans notre quotidien participent aussi à cette augmentation.

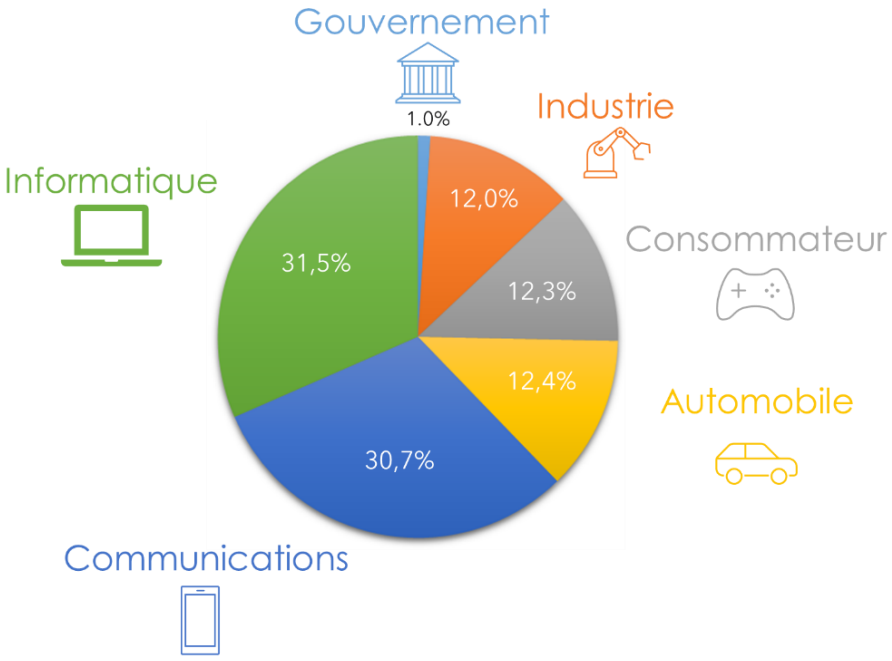

Figure 1 - Répartition des parts de marché de l'industrie des semi-conducteurs en 2021 (en pourcentage de 556 milliards de dollars) – ITRS

Dans ce contexte, la course technologique pour les semi-conducteurs devient un enjeu politique majeur, comme le confirme le « EU Chips Act », visant à renforcer la compétitivité des entreprises de microélectronique européennes, qui en 2020, n'était responsable que de 10% des puces fabriquées dans le monde.

b. 4ème révolution industrielle et collaborations

L'Industrie 4.0 révolutionne la façon dont les entreprises fabriquent, améliorent et distribuent leurs produits. Les fabricants intègrent à leurs sites de production et à l'ensemble de leurs activités des technologies porteuses d'autonomie, notamment l'Internet des objets (IoT), le *cloud computing* et l'analyse, ainsi que l'IA et l'apprentissage automatique. Ces usines intelligentes sont équipées de capteurs avancés, de logiciels embarqués et de solutions robotiques qui collectent et analysent les données et permettent une meilleure prise de décision. La rentabilisation est encore plus importante lorsque les données des opérations de production sont combinées à des données opérationnelles provenant des différents systèmes de l'entreprise, offrant une visibilité sans précédent à partir d'informations autrefois cloisonnées en silos. Cette technologie conduit à une automatisation accrue, à la maintenance prédictive, à l'auto-optimisation des améliorations des processus, et surtout à une efficacité et une réactivité inédites vis- à-vis des clients.

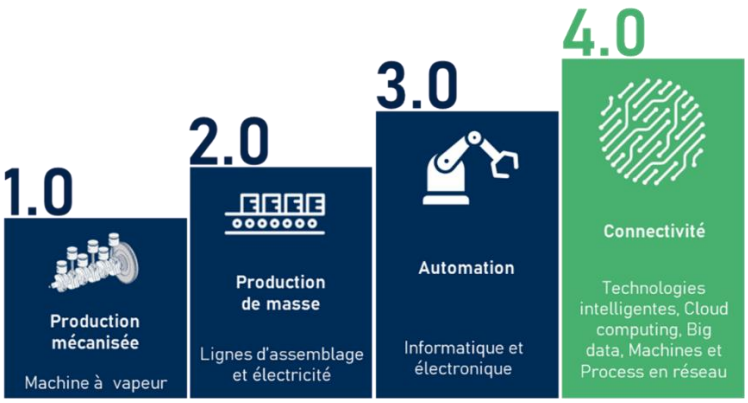

Figure 2 - Stages des différentes révolutions industrielles, avec les innovations les ayant rendues possibles.

Cette thèse s'inscrit de plus dans le programme européen MADEin4 qui regroupe un consortium de 47 partenaires provenant de 10 pays différents, connectant tous les acteurs de la chaine d'approvisionnement : Des fabricants d'équipements des semi-conducteurs et sociétés de systèmes intégrés de métrologie aux IRTs (Instituts de Recherche Technique) et aux applications clés comme l'industrie automobile. Ce projet développe la prochaine génération d'outils de métrologie, de méthodes d'intelligence artificielles et d'applications supportant la fabrication à large volume (*High Volume* Manufacturing – HVM) de l'industrie 4.0 des semi-conducteurs.

Figure 3 - Organigramme des collaborateurs et pays impliqués dans le projet européenMADEin4

c. *Protocole de fabrication en microélectronique*

Avant de s'intéresser à la métrologie en microélectronique, il est nécessaire de comprendre comment sont fabriqués les produits contrôlés par cette dernière.

La grande majorité des semi-conducteurs sont fabriqués à partir de Silicium monocristallin, obtenus selon une méthode similaire à celle du procédé de Czochralski, où un germe monocristallin est utilisé pour faire croitre un cristal de taille centimétrique. Ce dernier est alors découpé en tranches qui sont polies pour formées les plaques (*wafers*), sur lesquelles seront réalisés les produits. Pour ce faire, une succession d'étape, notamment de lithographie, où une résine photosensible est déposée sur la surface entière du wafer, puis illuminée à travers un masque pour définir la structure qui sera ensuite gravée chimiquement ou physiquement. Divers dépôts (d'oxydes, de nitrures, de métaux, etc…) et dopages (modification des propriétés électroniques du silicium par ajout d'impureté) sont alors réalisés entre ces étapes de lithographie afin de créer le circuit électronique souhaité (Figure 4).

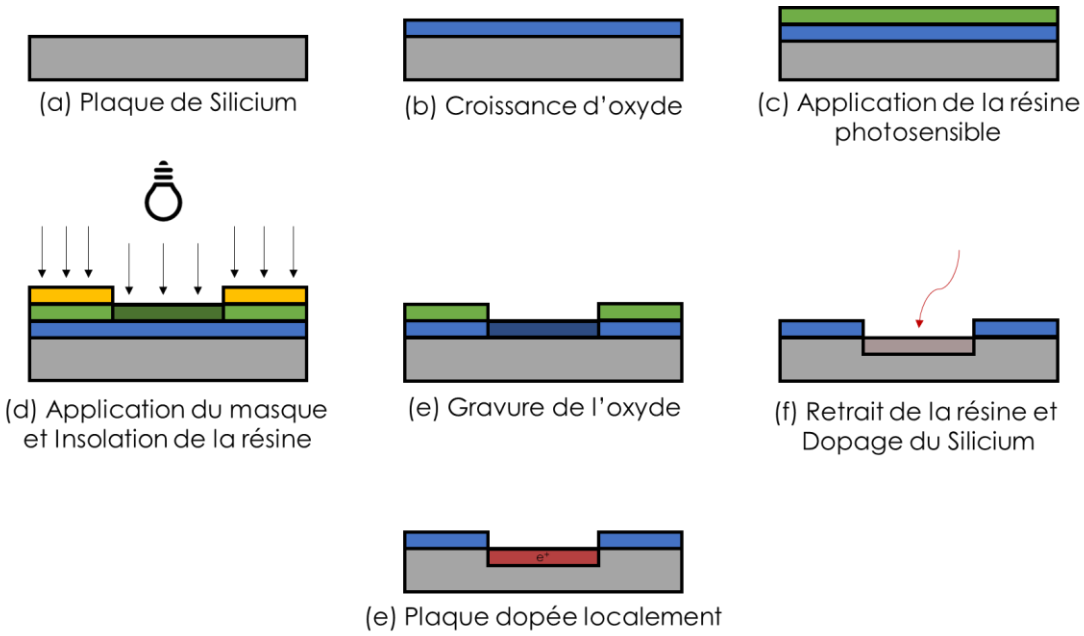

Figure 4 - Exemple de protocole de fabrication en microélectronique permettant le dopage sélectif de régions du wafer.

2. Métrologie

Dans l'industrie des semiconducteurs, la miniaturisation omniprésente, couplée à l'exigence de progrès technologiques mènent à une diversification ainsi qu'à une complexification des techniques de métrologie, de défectivité et de caractérisation physique utilisées, qui se doivent de plus d'être capables d'être sensibles à des dimensions de plus en plus réduites, pouvant être nanométriques. Ces besoins sont à l'origine d'avancées technologies majeures, portées par l'ITRS (*International Technology Roadmap for Semiconductors*), dont l'objectif est d'assurer des avancées robustes et économiques des performances des circuits intégrés, des produits et des applications utilisant ces puces, afin d'assurer la pérennité de l'industrie de la microélectronique.

a. *Historique de la métrologie*

La métrologie est la science de la mesure, et de ses applications. Ses progrès sont intimement liés aux progrès technologiques et scientifiques depuis de 4 millénaires. Mais, la mesure seule est inutile, en l'absence de standardisation, permettant de la faire correspondre à celle réalisée par d'autres individus. C'est en Égypte, 2900 ans avant J-C, que la première trace de volonté de

standardisation fait son apparition, avec la « Coudée Royale » mesurant 52,5 cm (longueur du bras du Pharaon, ajouté à la largeur de sa main), et ayant permis d'assurer le succès de la fabrication des pyramides, indiqué par une différence de longueur des bases inférieurs à 0,05%. Cette coutume d'évaluer les grandeurs physiques à partir de l'anatomie humaine persiste jusqu'au XVIIIème siècle, avec l'apparition des « pouces » et « pieds » comme référence de distance. Les incertitudes obtenues avec des mesures étaient donc importantes, les références étant extrêmement variables pour chaque individu. De plus, elles étaient pour la plupart très locales, et empêchaient alors un développement des échanges commerciaux et la diffusion d'informations à large échelle.

La métrologie moderne trouve son origine au siècle des lumières, quand la volonté politique d'uniformiser les unités à l'échelle de la France a entrainé la création d'un standard de longueur basé sur une source « naturelle ». Ainsi, en mars 1791, le mètre fut défini comme dix-millionième partie du quart du méridien, induisant la création du système métrique en 1795, permettant à son tour l'établissement des standards d'autres mesures (poids, temps, etc.). Ce système se repend alors dans d'autres pays et afin d'assurer une conformité internationale, le Bureau International des Poids et Mesures (IBPM) est fondé en 1875. La définition du mètre a alors été en perpétuelle évolution afin de suivre les développements technologiques permettant d'augmenter la précision des étalons choisis :

- En 1889 : La référence du mètre est un barreau gradué de platine iridée nommé « mètre étalon ».
- En 1960 : Le mètre se dématérialise et est alors décrit comme égal à 1 650 763,73 longueurs d'onde dans le vide de la transition radiative entre les niveaux 2p10 et 5d5 de l'atome de Krypton-86.
- En 1983 : Le mètre est défini comme la longueur du trajet parcouru dans le vide par la lumière pendant une durée de $\frac{1}{299\,792\,458}$ secondes (Ces dernières étant définies par la fréquence de transition hyperfine de l'atome de Césium-133).

L'augmentation de précision de ces définitions d'étalons sont nécessaires au développement de nouvelles technologies, car les incertitudes de mesures résultantes des premières définitions entraineraient des graves dérives très rapidement. Par exemple, une erreur de mesure de la seconde d'un facteur 5.10^{-10} (soit 38 µs/jour) entrainerait une dérive de 11 km par jour pour le positionnement par GPS.

On définit trois grandes catégories de métrologies :

- La métrologie scientifique : C'est la métrologie décrite plus tôt, soit celle dont l'objectif est la définition et l'établissement des unités de mesure, du développement de nouvelles méthodes de mesure, de la réalisation d'étalons et du transfert de la traçabilité de ces étalons aux utilisateurs industriels.
- La métrologie légale : Elle concerne les activités résultant d'exigences légales (e.g. transactions commerciales) et établit la validité des mesures dans ces cadres spécifiques.
- La métrologie industrielle : Elle est associée à un organisme, un laboratoire ou une industrie. Celle-ci permet de garantir la fiabilité des mesures, par exemple lors d'un processus de fabrication, en veillant à la conformité des instruments de mesure, à leur étalonnage et au contrôle de leur qualité dans le temps.

Ayant réalisé ma thèse dans le département de métrologie de STMicroelectronics, c'est la dernière catégorie sur laquelle nous allons principalement porter notre attention.

b. *Métrologie industrielle*

L'objectif de la métrologie industrielle (que l'on abrégera désormais « métrologie ») est de contrôler la qualité et la conformité des produits tout au long du processus de fabrication, et ce afin de réduire les pertes dues aux produits défectueux, et donc d'améliorer les rendements de l'entreprise tout en répondant aux attentes des clients. Le département de métrologie est alors organisé en deux types de tâches : La métrologie opérationnelle – qui vérifie les procédés de fabrication (*process control*) – et la métrologie conventionnelle – où le suivi de la conformité des moyens de mesure est régulièrement contrôlé à l'aide de plaques étalons, afin de détecter des dérives temporelles des équipements.

Pour chaque produit livré à un client final, l'industriel a l'obligation de délivrer sa « déclaration de conformité » : Elle décrit, entre autres, toutes les incertitudes des mesures réalisées durant la fabrication du produit, et donc l'indicateur de fiabilité de cette technique. En effet, des nombreux paramètres peuvent, à diverses étapes de la fabrication, entrainer des dégradations de l'exactitude des mesures, et notamment les fameuses « 5M » du diagramme d'Ishikawa présenté en Figure 5.

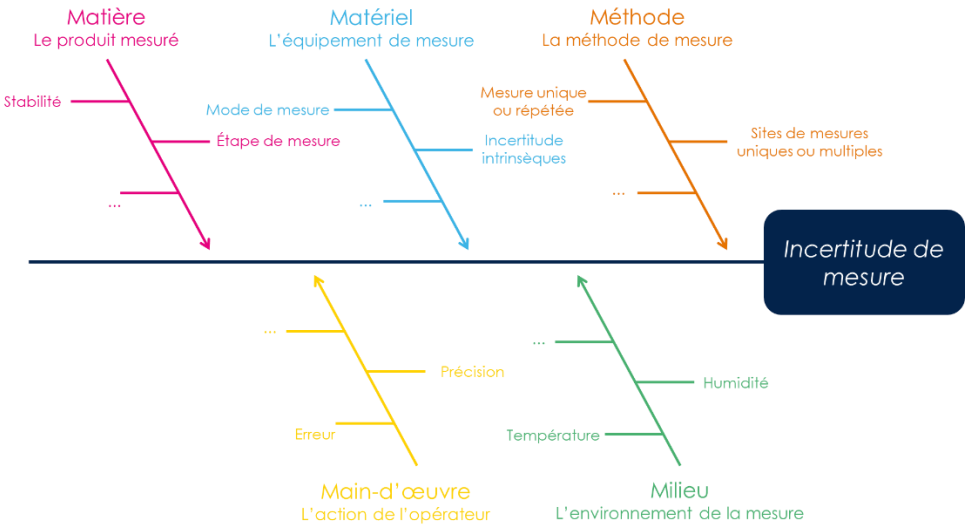

Figure 5 - Diagramme d'Ishikawa 5M des facteurs impactant les incertitudes de mesure.

Bien que l'incertitude liée à l'action de l'opérateur puisse être minimisée via l'automatisation de la mesure, celles liées aux équipements de mesure sont généralement intrinsèques à la technique. Des étalons de référence maintenus conformes par des laboratoires d'essais et d'étalonnage accrédités, sont utilisés pour évaluer ces erreurs, qu'elles soient systématiques ou aléatoires.

c. *La métrologie dans l'industrie de la microélectronique*

L'industrie microélectronique est en constante évolution, dans le but d'optimiser ses procédés de fabrication, tout en poursuivant la course à la miniaturisation, suivant la loi de Moore, mais aussi en diversifiant ses technologies. Cette première est intimement liée aux avancées des équipements de lithographie et de gravure, permettant la création de motifs aux dimensions de plus en plus réduites. Cette réduction entraine aussi une complexification des empilements de couches minces utilisées lors de la fabrication des puces, et la métrologie industrielle doit accompagner ces nouvelles approches et techniques émergentes afin d'être capable de suivre ces procédés innovateurs. En effet, on peut notamment illustrer cette progression en observant la tendance à la réduction des longueurs

d'ondes utilisées par les techniques de métrologies introduites au fur et à mesure des nœuds technologiques (définie par la densité de transistor possible), passant de la réflectométrie infrarouge, à la diffraction des rayons X (Figure 6).

La réalisation de puces aux nœuds les plus avancés (< 10 nm) nécessite l'utilisation de technologies extrêmement couteuses, entrainant certaines entreprises de la microélectronique à privilégier la diversification et la conception de nouveaux produits à des nœuds technologiques moins pointus, mais en s'appuyant sur des matériaux et procédés novateurs tels que les nano-films ou les boites quantiques. On parle alors d'une approche « More than Moore », où les avancées technologies sont atteintes autrement que par la constante miniaturisation des composants.

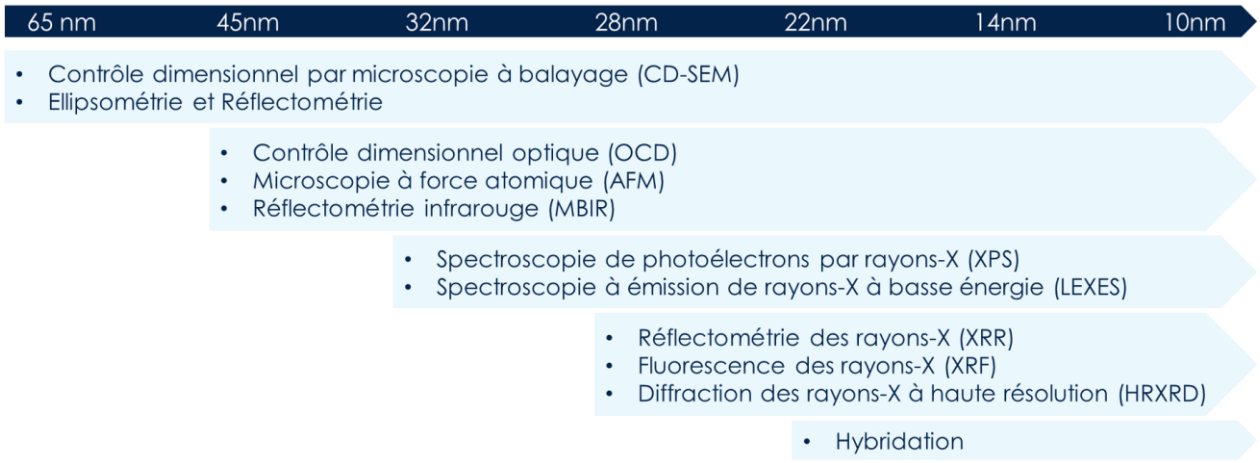

Figure 6 - Évolution non exhaustive des techniques de métrologie en fonction des nœuds technologiques.

i. La métrologie in-line, la défectivité et la caractérisation physique

Avant de décrire plus en détail la mission de la métrologie chez STMicroelectronics, il est important de comprendre les distinctions entre cette dernière et deux autres départements de support à la production que sont la défectivité et la caractérisation physique. Nous avons défini plus tôt la métrologie industrielle comme ayant pour objectif de « contrôler la qualité et la conformité des produits tout au long du processus de fabrication, et ce afin de réduire les pertes dues aux produits défectueux », et on pourrait alors penser que ces trois organisations répondent à cette définition. Néanmoins, des distinctions émergent pour chacun de ces départements, permettant de définir précisément leurs objectifs et responsabilité respectifs.

- Métrologie : Les mesures sont réalisées dans une unité de production, principalement pour le contrôle du processus de fabrication. Elle est généralement planifiée et automatisée, afin que les étapes critiques de fabrication soient contrôlées dès leurs réalisations. Les principales quantités contrôlées sont les dimensions critiques (CD) des dispositifs, les épaisseurs et alignements des différentes couches présentes, leurs compositions ainsi que diverses propriétés physiques des matériaux les composants. Ces mesures sont réalisées dans des structures spécifiquement crées, souvent localisées en périphérie des champs lithographiques, et de très petite taille. L'intégralité des plaques produites n'est pas mesurée à chaque étape de process, un *sampling* basé sur des études statistiques poussées permettent d'évaluer le nombre optimal de lots mesurés pour chaque étape, ainsi que le nombre optimum de plaques par lot qui devra être étudié. De plus, ces études ont aussi déterminé le nombre de mesures à réaliser par wafer, souvent limité à une dizaine de point pour l'intégralité de la plaque de 300mm. Elle intervient aussi pour assister au développement de procédés de fabrication innovants pour la R&D, ainsi que pour l'optimisation de recettes et le

choix des matériaux les plus adaptés aux produits réalisés. Les équipements utilisés ne nécessitent pas d'intervention humaine pour leurs mesures de routine, et comme elles sont connectées à un système d'automation, l'extraction, le stockage et le traitement des données obtenues lors de la mesure sont elles aussi automatisées. L'automatisation de ces dernières requiert une forte charge de travail lors de la phase de R&D pour assurer une justesse et une répétabilité satisfaisante.

- Défectivité : Elle a pour but la détection et le contrôle des excursions (accidents) afin d'en limiter l'impact sur le processus de fabrication, mais aussi la réduction du niveau de base de défauts afin d'améliorer les rendements des plaques produites. Contrairement à la métrologie, une large majorité des puces sont étudiées par plaque par la défectivité. Ces inspections sont souvent basées sur l'étude à large échelle de plaque, où les puces imagées sont comparées à une puce dite « de référence », ne possédant aucun défaut. Dans le cas d'une détection de défaut à l'inspection, une revue de la plaque est réalisée avec une technique hautement sensible et résolue permettant l'identification et la classification (parfois automatique à l'aide de réseaux de neurones) des défauts détectés.
- Caractérisation physique : Elle consiste en la mesure, généralement manuelle, destructive et hors ligne de production, de structures, afin d'en caractériser les dimensions et la composition. Elle est souvent liée à des besoins R&D, à la compréhension de problème, mais aussi au soutien de développement de procédés de fabrication ou de contrôle. Elle peut de plus aider lors de la validation de nouveaux équipements de métrologie. Par exemple, l'utilisation de microscopie électronique à transmission sur des coupes par le département de caractérisation physique permet de valider les modélisations utilisées dans les programmes de métrologies (dimensions, épaisseurs, compositions, etc.).

ii. Contrôle des procédés

La métrologie en ligne contribue au contrôle du bon déroulement du processus de fabrication en salle blanche (*process control*). Il est alors primordial de comprendre que le processus de mesure lui-même ne doit que très peu impacter le flux des lots de production. On exige des équipements qu'ils répondent aux mêmes standards de propreté que les équipements de procédés, mais aussi que les mesures réalisées soient, au possible, rapides, non destructives et non-invasives.

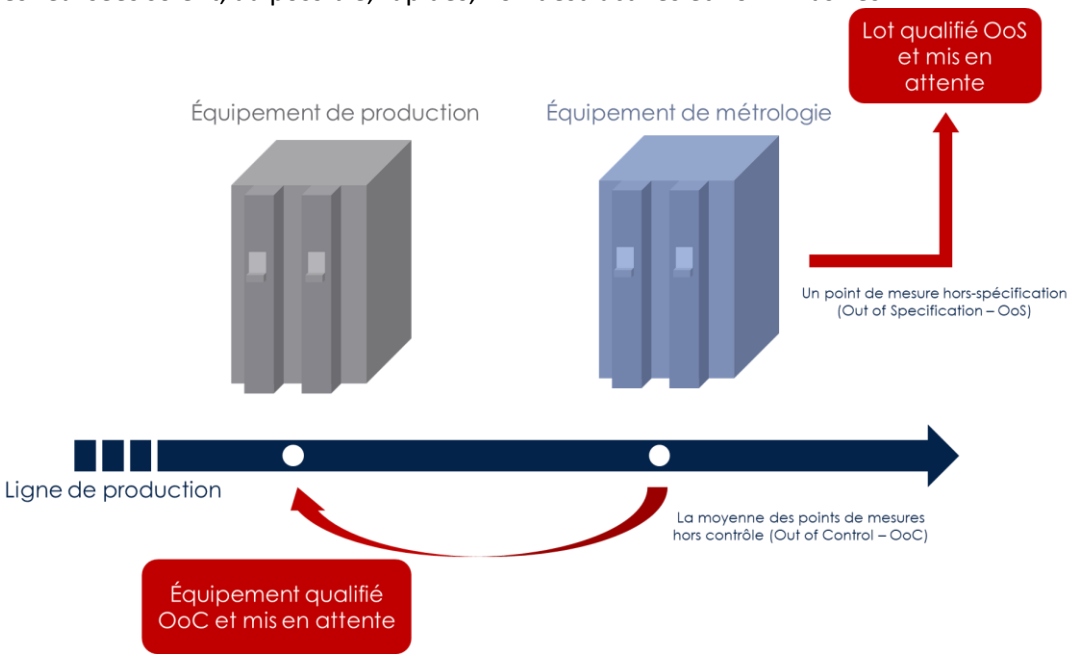

Figure 7 - Illustration de contrôle de procédé réalisé par une équipement de métrologie à la sortie d'une étape de production.

On observe sur la Figure 7 que les mesures de métrologie permettent de contrôler, à la fois le produit en lui-même, par la comparaison aux spécifications produit attendues, mais aussi les équipements de procédé, à l'aide d'un suivi statistique des tendances de ces derniers. En effet, si un hors-contrôle d'équipement est avéré, il est alors mis en attente pour qu'une analyse soit lancée, afin de corriger la dérive, qu'elle soit ponctuelle ou temporelle. Les lots hors spécifications quant à eux sont aussi mis en attente afin de déterminer l'impact et sa séverité, pouvant entrainer un « *rework* », soit l'annulation de l'étape jugée défectueuse et sa nouvelle réalisation, quand possible, ou si un « *scrap* », soit le rejet du lot pour minimiser les pertes. À travers ses mesures, la métrologie est aussi capable d'optimiser le procédé de fabrication des plaques en temps réel, par le biais de boucle de rétroactions mis en place afin d'ajuster les étapes précédents (feed-backward) ou suivants (feed-forward) la mesure. Par exemple, une mesure d'épaisseur dans la moyenne basse suivant un dépôt peut induire une plus faible durée de polissage à l'étape suivante.

iii. Propriétés mesurées en métrologie

Durant toute la durée de la ligne de production, la métrologie intervient pour en contrôler le bon déroulé, donc aussi bien en début de production, avant le dépôt des interconnexions métalliques (Front End Of Line – FEOL), qu'après ces dernières (Back End Of Line – BEOL). Ce suivi est réalisé de diverses façon, variant par la mesurande d'importance à l'étape étudiée, et les propriétés suivantes sont particulièrement suivies :

- Les dimensions critiques : Dimensions des structures (lignes, via, etc.) crées en lithographies et en gravure.
- Les propriétés électriques : Durée de vie et mobilités des porteurs dans le Silicium, dopage des zones implantées, etc.
- Propriétés des couches après dépôt : Épaisseur, composition, structure cristalline, etc.
- Topographie : Courbure, tension, rugosité de surface, etc.

Afin d'être capable de mesurer toutes ces propriétés, la métrologie bénéficie alors de l'utilisation d'un très large nombre de techniques, comme illustré en Figure 8.

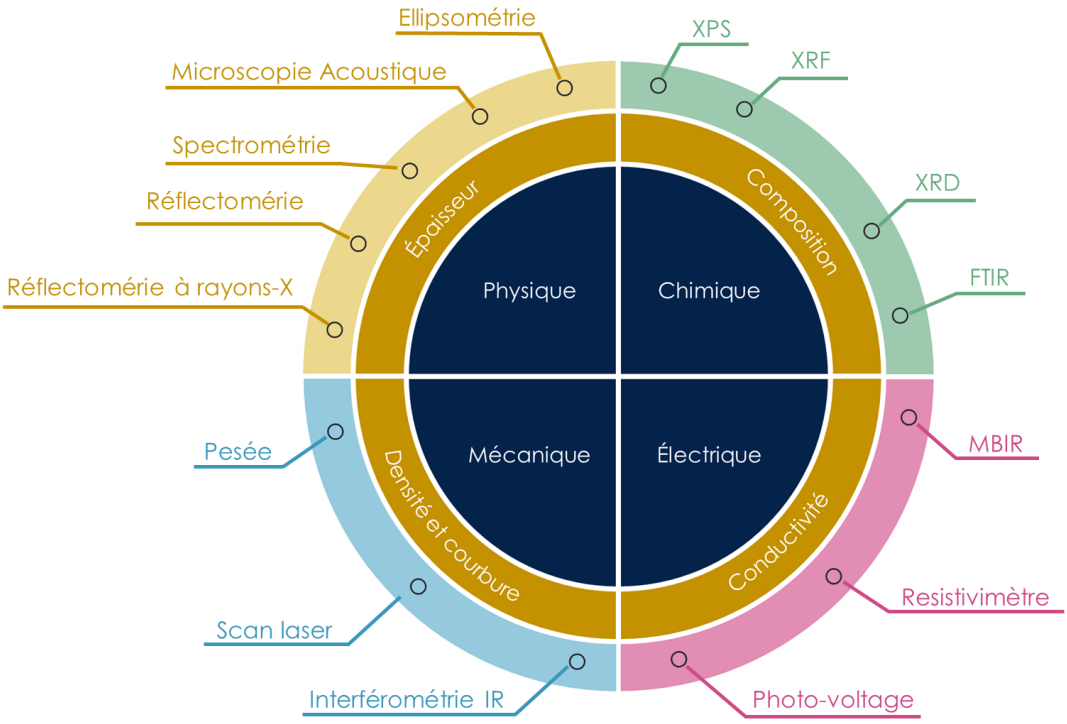

Figure 8 - Liste (non-exhaustive) des techniques utilisables en métrologie pour le contrôle et la mesure de diverses propriétés.

iv. Structures de métrologie

Le suivi des lots de production en ligne repose sur l'utilisation de structures spécifiquement dédiées à la métrologie, plutôt que sur la mesure des dispositifs eux-mêmes, du fait de leur complexité, mais aussi parce que bien que les techniques de métrologies soient « non-destructives » par essence, cela n'empêche pas certaines d'entre elles (laser à haute puissance ou source à électrons) d'avoir une incidence sur le dispositif. Ces structures sont intégrées sur les zones sacrificielles du wafer, dans les chemins de découpe, afin d'avoir une empreinte « nulle » sur les puces finales (Figure 9). Cela induit néanmoins une contrainte forte sur les équipements de métrologie, qui doivent être capable de réaliser leur mesure dans des structures les plus petites possibles (< 100µm x 100 µm), afin d'augmenter la surface du wafer dédiée au produit. Elles ont alors pour objectif d'être représentative du produit, mais avec une structure simplifiée, permettant une modélisation rigoureuse de l'empilement et ainsi obtenir une haute précision des mesures réalisées. Avec les avancées en matière de découpe, ces surfaces ont tendances à diminuer, induisant l'introduction de taille de faisceau de mesure de plus en plus petites

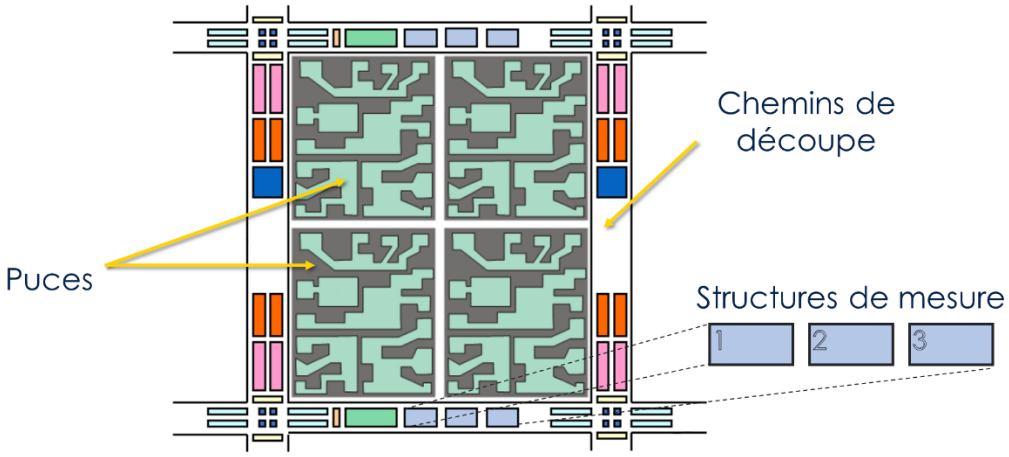

Figure 9 - Schéma illustratif d'un champ lithographique, contenant 4 puces, entouré par des structures de métrologie, localisées dans les chemins de découpe primaires.

v. Limitations

La métrologie étant dépendante des mesures réalisées dans des structures spécifiques induit de multiples limitations en plus de celle imposée aux équipements de métrologie cité précédemment. En effet, de par leurs tailles réduites, une déviation du procédé de fabrication à large échelle et à faible fréquence spatiale ne pourra pas être détecté via l'étude de ces structures, car la variation sur la surface de ces dernières serait négligeable. Leurs localisations couplées au nombre limité de structures mesuré par wafer sont aussi limitantes pour l'étude de déviation en bord de wafer, où ces mesures sont particulièrement rares. De surcroît, la grande majorité des mesures de métrologies reposent sur le développement très couteux en temps et en argent de modèles physiques rigoureux de ces structures, afin d'obtenir des mesures précises et répétables. Cette dépendance est particulièrement handicapante dans le cadre des développements de technologies innovantes par la R&D, où le produit, et donc ces structures, sont vouées à évoluer rapidement, à un rythme insoutenable pour la création de modèles physiques.

Afin de palier à ces limitations, il est alors d'intérêt d'évaluer des approches innovantes, basées sur l'étude à large échelle du wafer par des techniques de métrologie, pour ne plus être limité par l'utilisation de structures dédiées, ainsi que le traitement des données obtenues sans avoir à recourir à des modèles physiques, de sorte à être réactif aux évolutions constantes de la R&D. Cette

méthode à la frontière entre la métrologie (utilisation de techniques hautement sensibles) et l'inspection (acquisition à large échelle et à haut volume) a alors été nommé « MetroSpection ».

II. Techniques utilisées

Dans cette partie, les techniques utilisées lors de la thèse, qu'elles soient industrielles ou algorithmiques, seront décrites. Premièrement, les différents équipements industriels et les techniques sur lesquelles elles reposent seront présentées. Puis, le fonctionnement des réseaux de neurones, principaux outils de traitement de données basées sur l'intelligence artificielles auquel j'ai eu recourt pendant ma thèse, sera explicité.

1. Équipements industriels chez STMicroelectronics

Une grande variété d'équipements de contrôle de procédé industriels a été utilisée lors de ma thèse afin de récolter divers types de données à traiter avec des approches innovantes afin d'évaluer leur intérêt pour l'amélioration du suivi de fabrication dans l'industrie des semi-conducteurs. Un résumé des techniques utilisées, classées selon le format de sortie des données obtenues après acquisition est présenté dans le Tableau 1.

Format des données en sortie	Technique utilisée	Objectif	Modèle	Fournisseur
Données plates (Valeurs/spectres)	Microscopie Electronique	Mesure de dimensions	S9360	Hitachi
	Ellipsométrie Spectroscopique	Mesure de dimensions et indices optiques	ATLAS II	Onto Innovation
Image (Wafer entier)	Réflectométrie	Détection de défauts de surface	ALTAIR (8930)	KLA
			Surfscan SP3	KLA
	Interférométrie	Mesure de la topographie de surface	PWG	KLA
			ContourGT-X	Bruker
	Photoluminescence	Mesure des propriétés électroniques du Silicium	En-Vision	Semilab
	Microscopie acoustique	Détection des défauts de collage	SAM	PVA TePla

Tableau 1 - Résumé des techniques utilisées lors de la thèse

a. Microscopie électronique à balayage - SEM

La miniaturisation des composants de microélectronique continuant à un rythme élevé suivant la loi de Moore, cela entraine un besoin d'imagerie et d'analyse avancée pour le développement, la caractérisation et l'évaluation des structures à l'échelle nanométrique. Dans ce contexte, la microscopie électronique est régulièrement utilisée, aussi bien en défectivité pour identifier les défauts détectés par d'autres techniques [1], qu'en métrologie, pour mesurer les dimensions critiques de structures [2], ou pour mesurer l'overlay après traitement d'image [3]. La microscopie électronique à balayage repose sur l'utilisation d'un faisceau accéléré d'électrons focalisé sur une surface et la détection des électrons secondaires générés par le faisceau électronique d'électrons primaires. Le schéma de fonctionnement d'un microscope électronique à balayage

(Scanning Electron Microscope – SEM) est présenté en Figure 10. La détection des électrons secondaires synchronisée à la position du faisceau électronique lors du balayage de la surface, permet d'imager la topographie de la surface étudiée, et ainsi de générer des images représentatives de l'état de surface de l'échantillon.

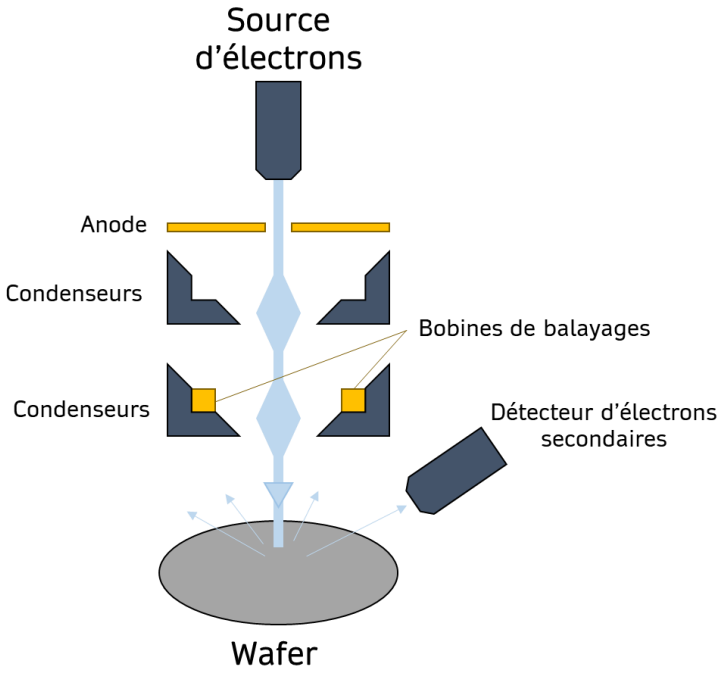

Figure 10 - Schéma fonctionnel d'un SEM

Comme les électrons secondaires ont une faible énergie (~50eV), l'épaisseur sondée est limitée à leur libre parcours moyen, soit quelques dizaines de nanomètres. Cette émission est de plus extrêmement localisée au niveau du point d'impact du faisceau, ce qui permet une résolution nanométrique. D'autres rayonnements sont générés par le faisceau (électrons rétrodiffusés ou d'Auger, rayons X etc.) et sont mesurables par le SEM pour permettre différentes caractérisations, mais seule l'étude des électrons secondaires pour la caractérisation de surface fut utilisée durant ma thèse.

En traitant les images obtenues par cet équipement, on peut ainsi obtenir les données métrologiques souhaitées. La Figure 11 montre comment les profils de niveaux de gris des images sont utilisés pour déterminer ces grandeurs. L'équipement utilisé durant ma thèse pour obtenir ces images et en extraire les dimensions d'intérêts est alors un Hitachi S9360, dont les principales caractéristiques sont présentées dans le Tableau 2.

Tableau 2 - Principales caractéristique du SEM S9360 de Hitachi

Equipement	Résolution	Répétabilité	Temps de mesure (s par point)	Vitesse (Wafer/ heure)
S9360	2 – 3 nm	0.6 – 2 nm	4 – 5.8	110

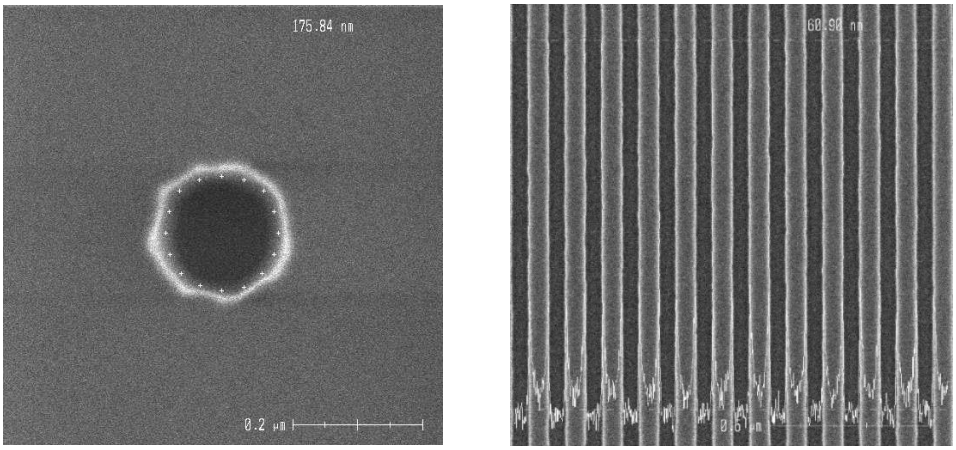

Figure 11 - Exemples d'images obtenues par le SEM et automatiquement traitées pour obtenir différents dimensions critiques.

b. *Ellipsométrie spectroscopique*

L'utilisation croissante des traitements de surface (optique, semi-conducteurs, métallurgie) a contribué au développement de techniques optiques d'analyse de couches minces, et notamment de l'ellipsométrie. Cette technique repose sur la mesure du changement de l'état de polarisation de la lumière après réflexion, et permet d'obtenir de nombreuses informations sur les caractéristiques physiques et morphologiques d'un échantillon plan. Les points forts de l'ellipsométrie sont : le caractère non-destructif, la large gamme de mesure (mesure d'épaisseur depuis une fraction de couche monoatomique jusqu'à quelques micromètres), sa possibilité de contrôle in situ permettant la mesure d'épaisseur de couches pendant leur croissance en temps réel. Un ellipsomètre mesure alors deux paramètres : l'état de polarisation de la lumière et l'intensité du rayonnement réfléchi. Une description théorique de ce type d'équipement est présentée en début de chapitre 2. Il faut distinguer l'ellipsométrie à une seule longueur d'onde – qui est l'outil basique, mais qui ne permet l'étude que de systèmes simples – de l'ellipsométrie spectroscopique, qui effectue des mesures répétables sur tout un spectre et permet d'interpréter des structures complexes : multicouche, rugosité d'interface, homogénéité, etc.

Durant ma thèse, deux types d'ellipsomètres ont été utilisés. Premièrement, des ellipsomètres de métrologie industrielle, afin d'obtenir précisément les indices optiques et épaisseurs de couches de matériaux, mais aussi les dimensions de structures périodiques par scattérométrie. La scattérométrie consistant en l'application de l'ellipsométrie à des motifs périodiques. Pour ce faire, l'ATLAS II de l'équipementier Nanometrics fut utilisé. Il possède une source couvrant une gamme de longueur d'onde de 210 à 1700 nm. Le second est un ellipsomètre académique à modulation de phase installé en salle blanche du CEA-LETI de Grenoble (et géré par le LTM), et financé par l'ANR au travers de l'EQUIPEX IMPACT. Cet outil est un UVISEL deuxième génération d'Horiba (UVISEL 2) permettant la réalisation de mesures sur des wafers de 300 mm de diamètre. Il sera décrit en plus grand détail dans le chapitre 4 consacré à l'imagerie ellipsométrique.

Dans le cas de l'ellipsométrie industrielle, les spectres obtenus doivent être comparées à des spectres modélisés pour déterminer les indices optiques et les épaisseurs des couches étudiées. Pour ce faire, une modélisation de l'empilement est nécessaire au préalable afin de pouvoir générer sa réponse optique. Les indices optiques des matériaux sont modélisés à l'aide de loi de dispersions (par exemple Fourouhi-Bloomer ou Tauc-Lorenz) et un spectre simulé est généré à l'aide des lois de Fresnel. De façon itérative il est alors possible d'ajuster le spectre obtenu au spectre expérimental par résolution du problème inverse. Le protocole résumé de cette approche est présenté en Figure 12.

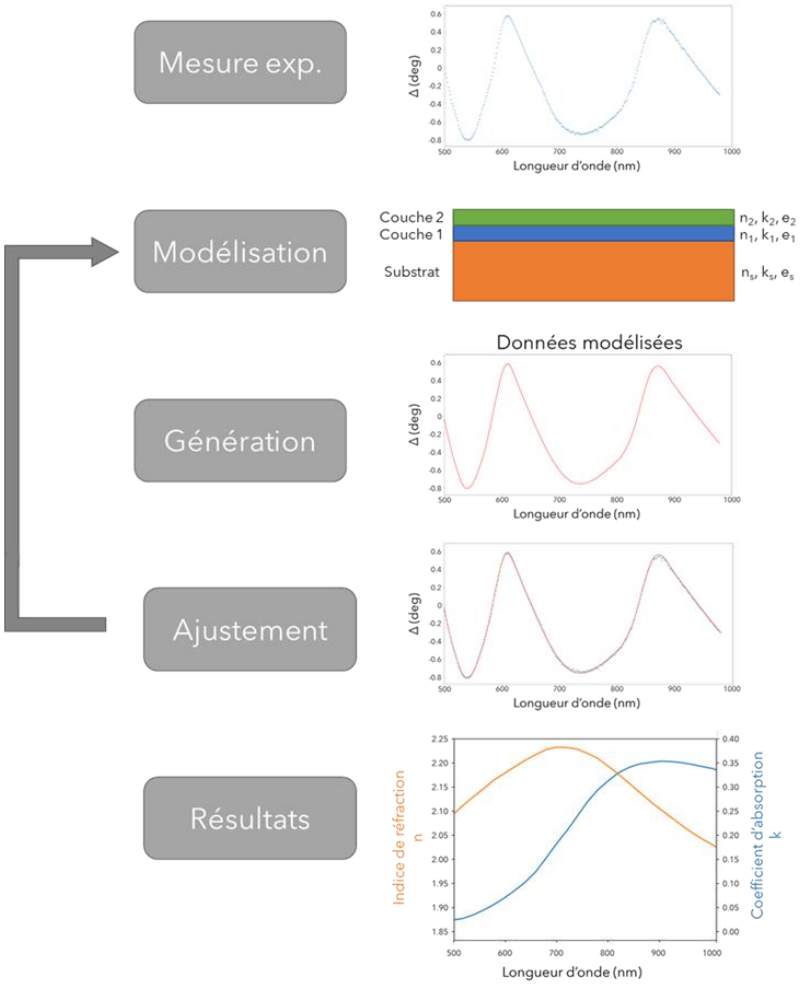

Figure 12 – Résumé du protocole de détermination des indices optiques et épaisseurs par ellipsométrie spectroscopique

Le protocole de mesure de structures par scattérométrie est très similaire à ce dernier, mais la simulation doit maintenant prendre en compte la géométrie de la structure étudiée. Une description complète de cette modélisation est présentée en détail dans le chapitre 2. Le développement de ces modèles est alors d'autant plus complexe et cela allonge le temps nécessaire à la création d'une nouvelle mesure (~ mois), et à sa validation à l'aide de techniques de caractérisations physiques, typiquement des coupes TEM (*Transmission Electron Microscopy*), où des tranches minces de l'échantillon étudié sont imagées à l'aide d'électrons transmis à travers ce dernier. On peut suite à cette validation, obtenir, par scatterométrie, des mesures extrêmement précises et répétables des propriétés de ces structures (Figure 13).

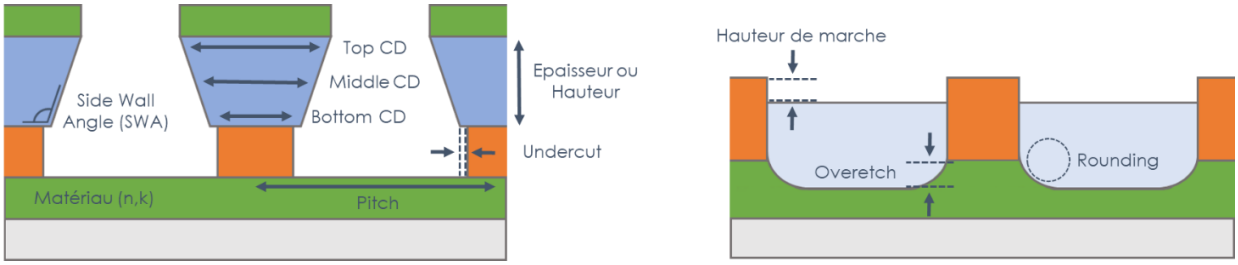

Figure 13 - Exemples de paramètres géométriques et physiques contrôlables par scatterométrie.

Ce modèle étant figé, de par sa dépendance à la génération de large librairies, la scattérométrie est peu versatile pour s'adapter à des évolutions introduites par la R&D, et n'est alors mise en place à disposition que sur des procédés de fabrication matures. Un résumé du protocole de mesure en scatterométrie est présenté en Figure 14.

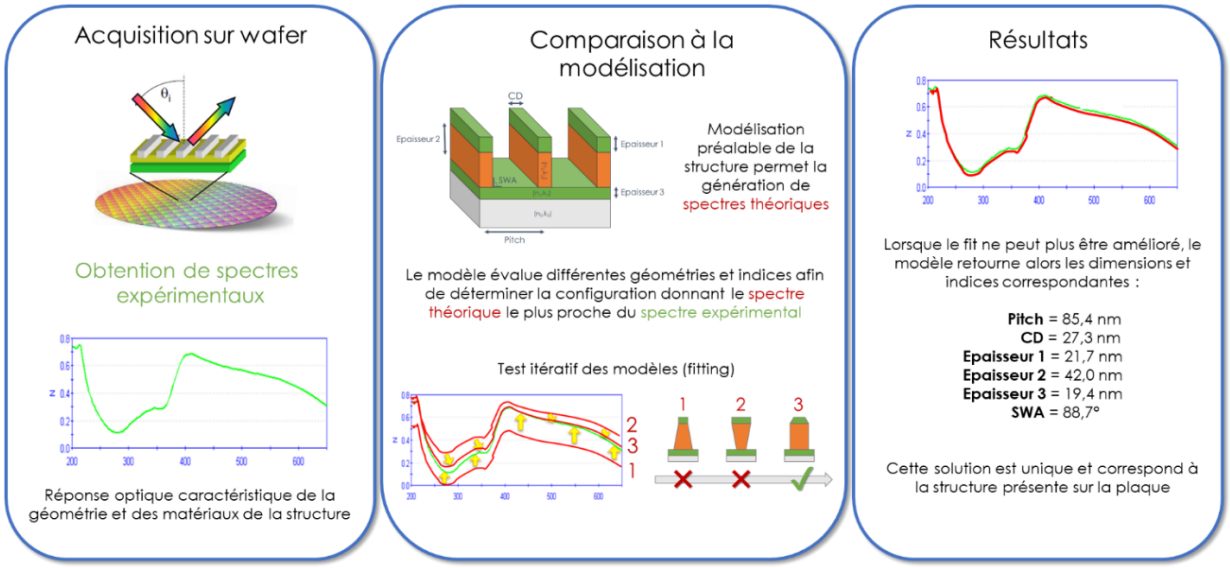

Figure 14 - Résumé du protocole de mesure de structures périodiques par scatterométrie.

c. *Réflectométrie*

La réflectométrie est une technique industrielle optique reposant sur la mesure de l'intensité lumineuse réfléchie en incidence normale par une surface pour en mesurer les caractéristiques. Elle peut être utilisée, soit via modélisation pour obtenir les épaisseurs et indices optiques des couches présentes [4], soit en scannant la surface pour détecter des défauts physiques. Dans le cadre de cette thèse, seule la seconde approche sera explicitée.

i. *KLA 8930 (Altair)*

Le premier réflectomètre de défectivité utilisé durant mes travaux est le KLA 8930 (Altair), dont le schéma de fonctionnement est présenté en Figure 15.

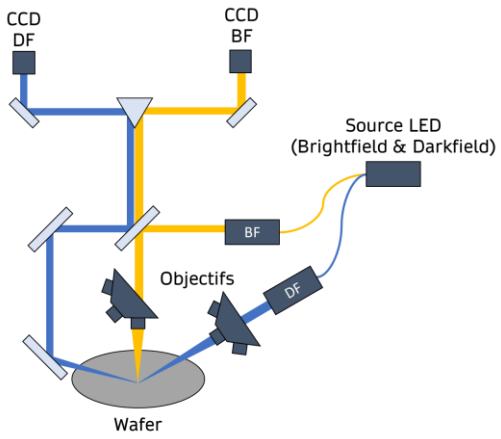

Figure 15 - Schéma de fonctionnement d'un réflectomètre de défectivité (KLA 8930 – Altair).

Cet équipement est capable de scanner la surface entière d'un wafer de 300mm et possède deux modes d'illumination : Le « brightfield » (BF) permettant avec un éclairage normal à la surface de détecter des défauts plats (tâches, résidus…), et le « darkfield » (DF) permettant avec un éclairage rasant de détecter des défauts présents sur la surface (particules, rayures…). Ces deux approches sont complémentaires, en effet, le BF est adapté pour les niveaux « plats » (CMP, dépôt) tandis que le DF est priorisé pour les niveaux avec du relief (gravure, photolithographie). Ces illuminations peuvent être réalisées à différentes longueurs d'ondes pour augmenter la sensibilité de l'équipement à certains défauts. Soit avec des LEDs mono-longueur d'onde (Bleu - 450 nm, Vert - 520nm ou Rouge - 625 nm), soit en les combinant pour obtenir une source blanche. Ces sources peuvent être simultanément utilisées dans les deux configurations d'illumination et la lumière réfléchie est alors collectée par deux capteur CCD. Différents objectifs peuvent être utilisés afin d'obtenir une meilleure résolution, au prix d'un temps d'acquisition plus long (Tableau 3).

Tableau 3 - Caractéristiques de l'Altair montrant l'évolution du temps de mesure avec la résolution

Grossissement de l'objectif	Sensibilité		Vitesse (Wafer/ heure)
	BF	DF	
1X	6µm	2µm	110
2X	4µm	1.2µm	55
5X	3µm	0.5µm	14

Dans le processus industriel, chaque puce du wafer est alors imagée par ce réflectomètre et est comparée à une puce de référence afin de permettre la détection de défauts. Lorsqu'un défaut est détecté, il est localisé dans le champ et sera, après la fin de l'acquisition du wafer, inspecté avec des objectifs à fort grossissement (20X ou 35X) pour pouvoir être automatiquement identifié. L'ADC (Automatic Defect Classification) [5] permet de déterminer la gravité de la défectivité présente et de pouvoir prendre des décisions process (nettoyage, rework, scrap…).

Pour mes travaux de recherche, cet équipement a été utilisé afin d'obtenir des images full wafer et de tester des approches innovantes de gestion de l'information (Figure 16). L'objectif est alors de traiter ces images via une approche IA afin d'automatiser la détection des déviations. La méthode a été poussée jusqu'à la prédiction de résultats de rendements électriques finaux, et est présentée dans [6].

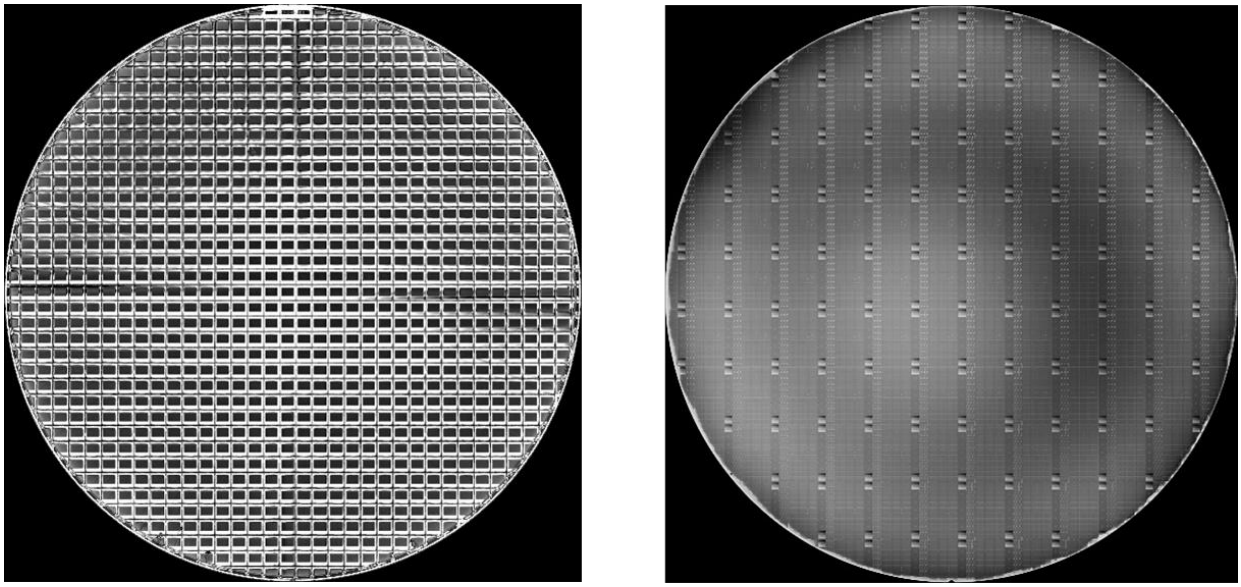

Figure 16 - Exemples d'images obtenues par acquisition full wafer avec l'Altair.

ii. Surfscan SP3

Le second réflectomètre de défectivité utilisé pour mes travaux de recherche est le Surfscan SP3 de KLA. Son principe de fonctionnement est très proche de celui de l'Altair présenté plus tôt, à la différence qu'il utilise des photomultiplicateurs (PM) plutôt qu'un capteur CCD pour mesurer la lumière réfléchie sur le wafer, qui est cette fois issue d'une source laser ajustable de longueur d'onde (λ = 266 nm). La surface totale du wafer est scannée en quelques secondes et la présence de défauts est révélée par une forte variation de l'intensité du signal mesurée par le PM du réflectomètre. De plus, le Surfscan SP3 mesure la totalité de la lumière diffusée par l'illumination DF à l'aide de deux miroirs elliptiques, ce qui permet d'évaluer l'état de surface et obtenir une image entière dite « haze » (Figure 17 et Figure 18). Cet équipement n'est alors utilisable que pour des plaques sans motifs, mais possède des capabilités de détection bien supérieur au KLA 8930. En effet, selon les conditions de la plaque, le SP3 peut détecter des défauts jusqu'à 25nm.

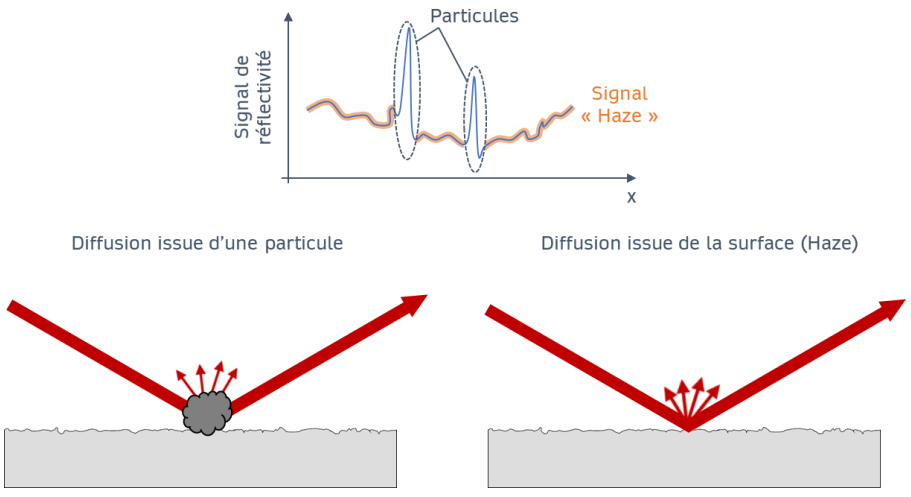

Figure 17 - Evolution du signal de réflectivité du SP3 avec les particules et la rugosité de surface

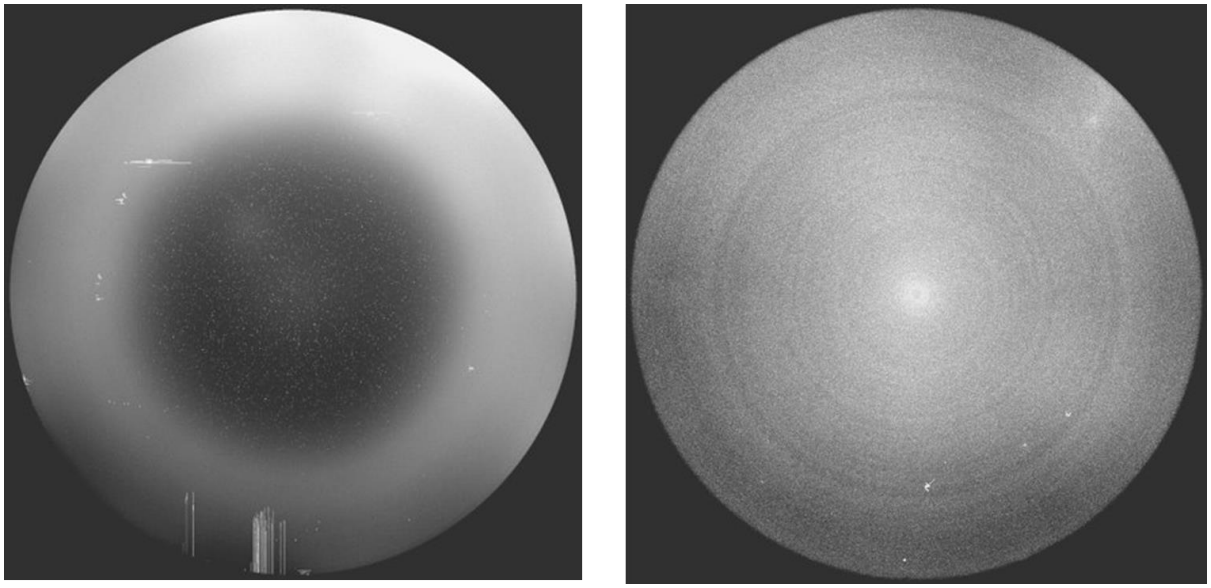

Figure 18 - Exemple d'images obtenues par le SP3 (signal « haze »), avec détection de ruptures du réseau cristallin (gauche), et de particules (droite)

d. Interférométrie

L'interférométrie est une technique optique couramment utilisée dans l'industrie des semi-conducteurs pour la caractérisation de la topographie de surfaces. L'utilisation de cette technique permet une très haute sensibilité, combinée aussi à une rapidité d'acquisition sur le wafer entier.

i. PWG

Le premier équipement de métrologie industrielle reposant sur l'interférométrie utilisé durant mes travaux de thèse est le PWG (Patterned Wafer Geometry) de KLA. Il a été utilisé pour l'étude de problème de focus et d'overlay induit par différentes étapes du procédé de fabrication [7] ou du stress induit par une dérive de dépôt d'une couche de nitrure [8]. Cet équipement utilise la technique d'interférométrie Fizeau à décalage de phase à l'aide d'un double interféromètre optique de type PSI (Phase Shift Interferometer – Interféromètre à décalage de phase) dont le montage optique est présenté en Figure 19. Lors de la mesure, un miroir étalon partiellement réfléchissant et un wafer sont placés face à face. Une lumière cohérente monochromatique à la longueur d'onde à 635nm est émise par un laser vers ces surfaces en illuminant toute la surface (300mm) en incidence normale et la lumière réfléchie par la face arrière de la pièce étalon combinée à la lumière réfléchie par le wafer va former des franges d'interférences observées à l'aide d'une caméra. Ces franges d'interférences permettent de mesurer localement et avec une grande précision la topographie du wafer et plus globalement, la forme et la courbure des plaques, puisque l'acquisition est faite sur la surface totale du wafer. La Figure 19 ci-dessous illustre le montage optique présent dans l'équipement : le wafer est maintenu verticalement au centre de l'enceinte, de ce fait, la déformation du wafer n'est pas impactée par l'effet de gravité. Il est placé entre deux interféromètres optiques et l'on peut ainsi obtenir simultanément des informations sur la face avant ainsi que la face arrière du wafer.

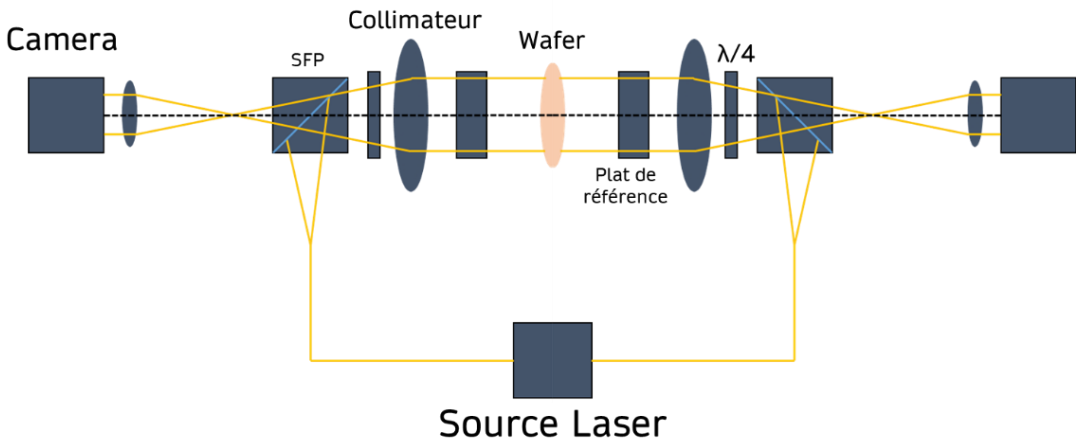

Figure 19 - Schéma fonctionnel de l'interféromètre de métrologie (PWG).

La pièce étalon (plat de référence) est séparée du wafer par une épaisseur d'air contrôlée. Les franges d'interférences, obtenues par la superposition des faisceaux de lumière réfléchis par la face arrière de la pièce étalon et par ceux réfléchis par la surface à contrôler, caractérisent donc l'épaisseur d'air entre la surface à mesurer et la pièce étalon. Afin d'éviter que la lumière réfléchie par la face avant de la pièce étalon n'interfèrent avec les autres faisceaux et ne vienne fausser la mesure, l'étalon présente un léger biseau.

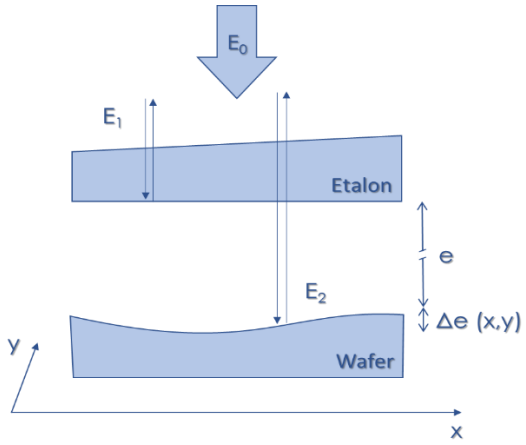

Figure 20 - Schéma de la différence de chemin optique induisant les interférences

Soit E_0 l'éclairement de l'onde incidente monochromatique. On note E_1 l'onde réfléchie par la face arrière de l'étalon et E_2 l'onde réfléchie par la surface à contrôler. Alors, l'éclairement E qui correspond à la figure d'interférence est le suivant :

$$E(x,y) = E_1 + E_2 + 2\sqrt{E_1 \times E_2}\cos\left(2\pi\frac{\Delta(x,y)}{\lambda}\right)$$

Avec λ la longueur d'onde de la lumière incidente et $\Delta(x,y) = 2e + 2\Delta e(x,y)$ la différence de marche entre les deux faisceaux réfléchis.

Sur l'équipement, l'obtention de la figure d'interférence est précédée par une mesure capacitive permettant de mesurer très précisément l'épaisseur en un point du wafer. Cette mesure sert de référence et permet ensuite après l'analyse des franges d'interférence de déterminer l'épaisseur en tout point du wafer. L'analyse de la figure de frange est faite en plusieurs étapes :

- Évaluation de la phase optique
- Déroulement de la phase

- Ré-échelonnage

Évaluation de la phase optique (cf Figure 21)

L'obtention de la distribution spatiale de la phase Δφ représente l'étape critique de la détection. Comme montré précédemment, l'interférogramme est défini selon l'expression suivante :

$$E_n = a + bcos(\Delta\varphi + (n-1)\phi)$$

Avec *a* la composante continue du signal, *b* l'amplitude des oscillations des franges, *Φ* le décalage de phase et *n* un entier positif allant de 1 à N.

En supposant le décalage de phase connu, l'expression comporte trois inconnues : *a*, *b* et *Δφ.* En faisant varier ces inconnues selon N valeurs, on obtiendra autant d'équations à trois inconnues. Il est donc nécessaire d'obtenir au minimum trois interférogrammes afin de pouvoir résoudre le système d'équation et en déterminer les trois inconnues. Afin de générer le décalage de phase, le PWG est doté d'un laser à décalage de longueur d'onde : en faisant varier la longueur d'onde *λ*, on induit un décalage de la phase et on peut ainsi obtenir plusieurs interférogrammes. La phase optique est ensuite extraite mathématiquement, et on obtient ainsi une variation de phase discontinue avec des sauts de ±π à chaque discontinuité.

Déroulement de la phase (*cf* Figure 21)

Le déroulement de la phase, ou phase unwrapping, permet d'obtenir la continuité de la phase. Jusqu'ici, nous avons été uniquement capables d'obtenir la phase optique modulo 2π. L'opération suivante consiste alors à ajouter ou retrancher 2π rad à chaque discontinuité détectée au niveau de la phase optique. Ce calcul est géré par un algorithme du software de l'équipement dont le rôle est de définir un critère pour l'identification des discontinuités.

Ré-échelonnage (*cf* Figure 21)

Enfin, la dernière étape consiste à ré échelonner la mesure, c'est-à-dire faire correspondre la phase mesurée à l'amplitude de la mesurande recherchée. Dans notre cas, la relation est la suivante :

$$h(x,y) = \frac{\lambda}{4\pi cos\theta}\Delta\varphi(x,y)$$

Avec *h(x, y)* l'altitude, *λ* la longueur d'onde du laser et θ l'angle d'incidence.

Ainsi, en calculant la répartition de la phase, on peut obtenir la mesure d'altitude h(x,y) en chaque pixel. La figure 11 ci-dessous résume les étapes de l'analyse. L'évaluation de la phase est possible grâce au décalage de phase réalisé par un décalage de longueur d'onde : on obtient plusieurs interférogrammes et on peut calculer la phase en résolvant un système d'équations. Le déroulement de la phase est ensuite réalisé par un calcul mathématique consistant à ajouter ou retrancher 2 π aux discontinuités détectées dans le but d'obtenir la continuité de la phase.

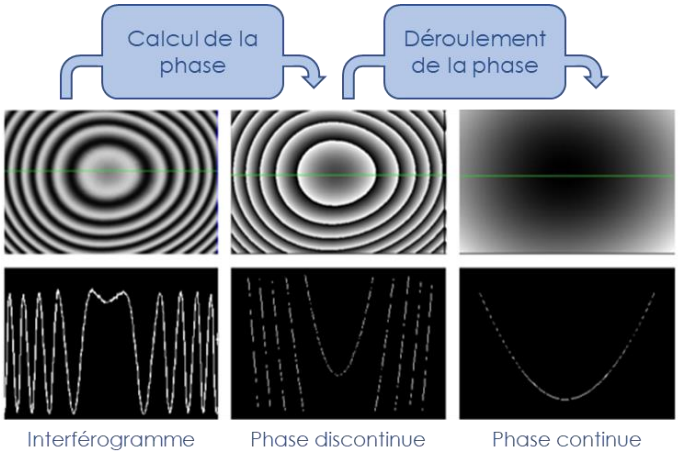

Figure 21 – Protocole d'analyse des figures d'interférence.

Le décalage de phase présente plusieurs avantages :

- Détermination du signe de la phase : la phase extraite à partir d'un seul interférogramme est définie au signe près. On l'obtient par inversion de la forme analytique de la figure de franges, c'est-à-dire en inversant la fonction cosinus. Or, la fonction cosinus n'est pas bijective mais est paire et périodique et on n'obtient ainsi pas l'information du signe. Le décalage de phase permet alors de retirer cette ambiguïté.
- Basse sensibilité au bruit stationnaire dans le domaine dans lequel ϕ varie : on est insensible aux non-uniformités de *a*, à la composante continue du signal, et à *b*, l'amplitude des oscillations.
- Automatisation de la mesure : le processus d'analyse des franges peut alors être complétement automatisé, la phase optique est déterminée en chaque pixel, le temps de calcul est très faible.

De façon synthétique, les caractéristiques du PWG sont reportées dans le tableau ci-dessous :

Tableau 4 - Caractéristiques de l'interféromètre de métrologie (PWG)

Technique	Résolution en z	Résolution latérale	Longueur d'onde	Vitesse de mesure
Interféromètre à décalage de phase	0.5 nm	100 µm	635 nm	100 wafers par heure

L'analyse des figures d'interférence permet alors d'obtenir l'information de la nanotopographie sur l'entière surface du wafer. Un exemple typique de cartographie de nanotopographie obtenue avec le PWG est présenté en Figure 22.

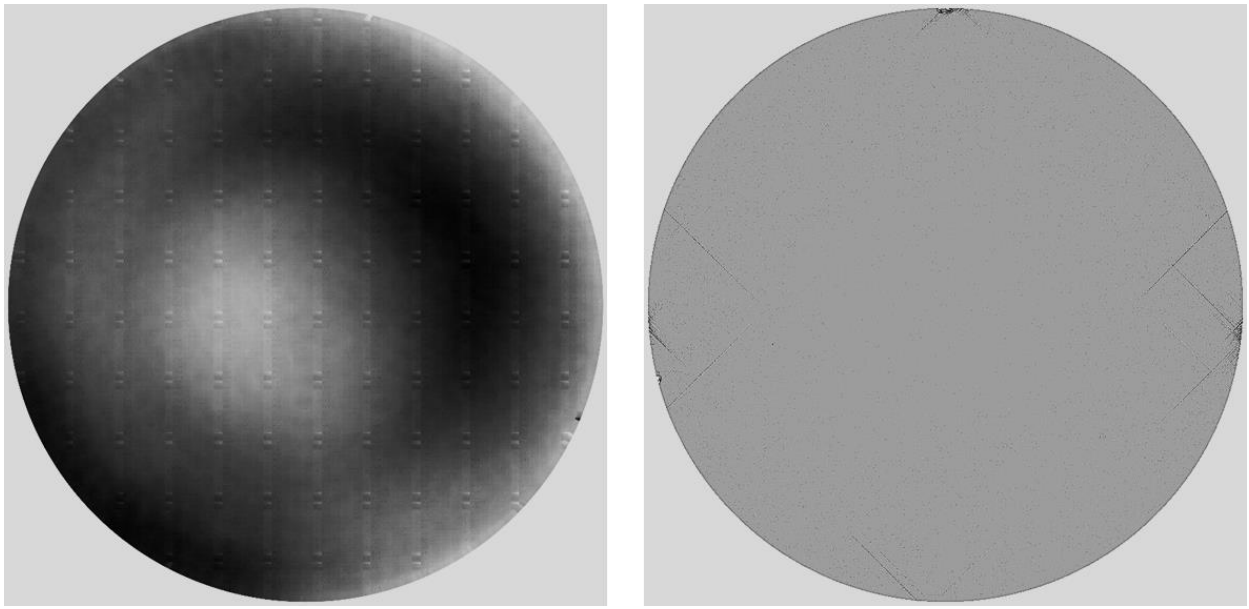

Figure 22 - Exemples de cartographies de nanotopographie obtenues avec l'interféromètre de métrologie, montrant une sensibilité à la nanotopographie à large échelle (gauche) et locale (droite)

Cet équipement n'est en revanche adapté qu'à l'étude de plaques dont la surface est opaque, afin de s'assurer que les interférences soient générées uniquement par des réflexions sur la surface, et non par des couches sous-jacentes. En effet, l'utilisation du PWG sur des couches transparentes (oxydes, résines…) entraine l'apparition d'erreurs de reconstruction. Ces erreurs sont dues au fait que l'équipement est adapté à des variations locales de topographie très faibles. La transparence des matériaux constituant l'empilement amène alors à des interférences parasites ce qui rend les interférogrammes obtenus ininterprétables par l'équipement. Un exemple d'erreur de reconstruction obtenu lors de mesures sur des plaques imageurs à résine colorée est présenté en Figure 23.

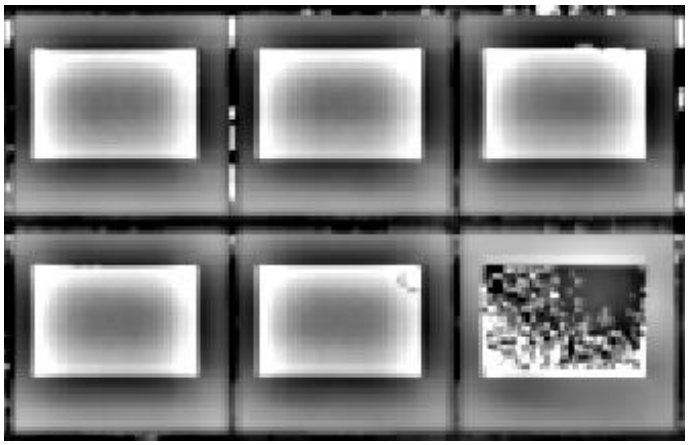

Figure 23 - Exemple d'erreur de reconstruction induit par la présence de matériau non-opaques en surface.

Cet équipement est aussi capable d'obtenir des informations sur la géométrie globale du wafer (forme, courbure) et d'en déduire des mesures de stress. Cette méthode de calcul de stress à partir de ces mesures est décrite dans [9]. Comme pour l'équipement précédant, le PWG a été utilisé durant ma thèse afin d'obtenir des images à l'échelle du wafer permettant la détection, via des algorithmes intelligents, de dérives de procédé.

ii. ContourGT-X

Le second interféromètre utilisé durant ma thèse est le ContourGT-X de Bruker, dont le schéma fonctionnel est présenté en Figure 24. Il a été utilisé pour caractériser des matrices de microlentilles en verre [10] ou pour évaluer des structures nanométriques visant à inhiber et prévenir la formation de biofilms sur des substrats biomédicaux [11]. Il utilise cette fois un interféromètre de Mirau et peut fonctionner selon deux modes : Vertical Scanning Interférométrie (VSI) ou Phase Shifting Interferometry (PSI). Pour le VSI, une source LED blanche est utilisée pour scanner verticalement la surface du wafer à l'aide de la tête motorisée de l'équipement. Ce mode d'acquisition donne des informations sur la topographie à large échelle (500nm à 10 mm) à partir des interférences générées. Pour le mode PSI, une LED verte est combinée à un système piézo-électrique afin d'en moduler le signal et compenser toute vibration du wafer. Ce mode permet d'obtenir la nanotopographie locale avec une grande précision (mesure d'1 angström à 150nm, précise à 0.1 nm près). En combinant ces deux méthodes, on peut alors obtenir, au prix d'un temps d'acquisition plus long et d'une réduction de la résolution latérale du VSI, la topographie large d'un échantillon, ainsi que la nanotopographie fine (Figure 25).

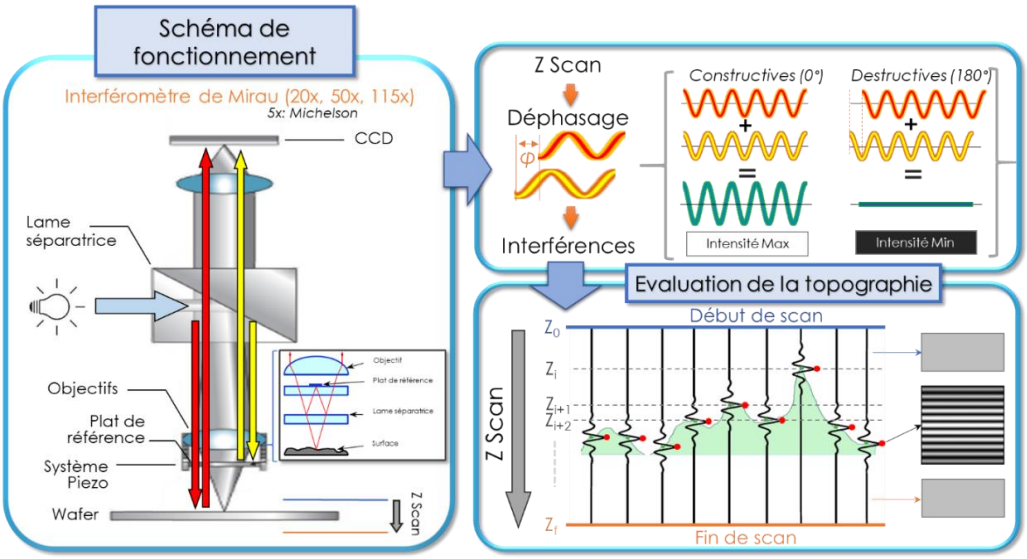

Figure 24 - Schéma fonctionnel et principe de fonctionnement du ContourGT-X.

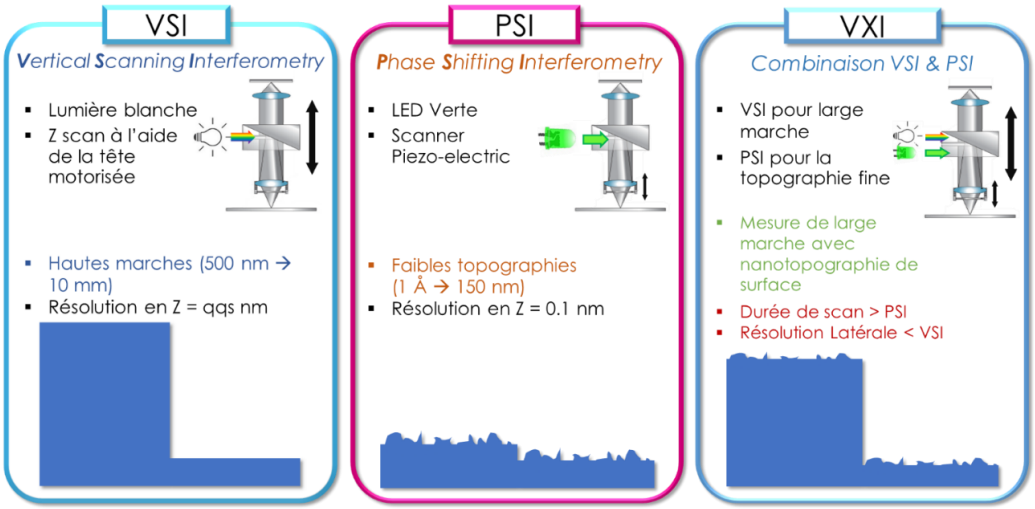

Figure 25 - Principe de fonctionnement des deux modes du ContourGT-X, et de leur combinaison.

A l'aide d'un algorithme de raccordement intelligent, ces scans (plus de 1000 possible) peuvent être combinés pour obtenir des larges cartographies de nanotopographie. On a alors utilisé cet équipement durant ma thèse afin d'obtenir ce type de cartographie (Figure 26) afin de caractériser des régions d'intérêts avec grande précision.

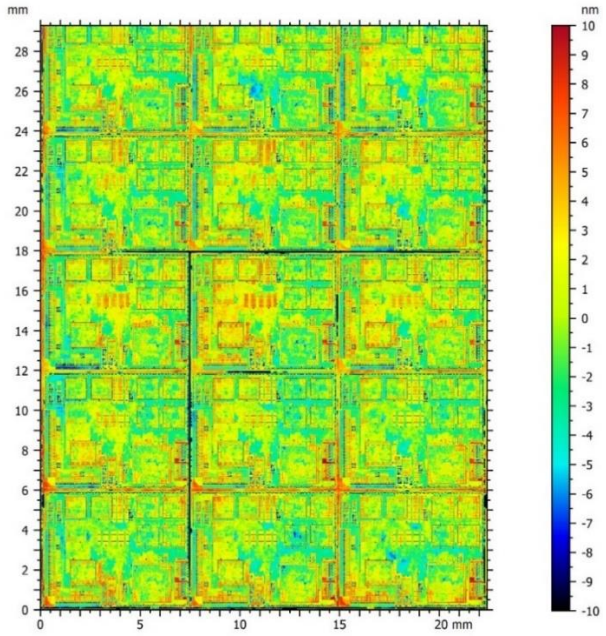

Figure 26 - Exemple de cartographie de nanotopographie obtenu avec le ContourGT-X.

e. Photoluminescence

La photoluminescence (PL) est utilisée dans l'industrie des semi-conducteurs afin de contrôler la qualité du Silicium ou en caractériser les propriétés [12], notamment pour la fabrication de cellules photovoltaïques. L'équipement industriel utilisé est l'En-Vision de Semilab, dont la source laser (λ = 808nm) permet la génération de charges libres dans le Silicium qui, lorsqu'elles se recombinent et retournent à leur état fondamental, émettent une lumière dont les propriétés (longueurs d'onde, intensité, etc.) peuvent être utilisées afin d'obtenir des informations sur la pureté, la présence de défauts cristallins, ou le niveau de dopage ainsi que d'autre propriétés. Le schéma de fonctionnement de cet équipement est présenté en Figure 27. Plusieurs méthodes d'acquisitions sont possibles avec cet équipement. La première, nommée microPL (µPL) permet de capturer le signal de photoluminescence à l'échelle micrométrique, en utilisant les différents objectifs permettant de pouvoir ainsi acquérir des régions d'intérêts aussi petites que 0,025 mm².

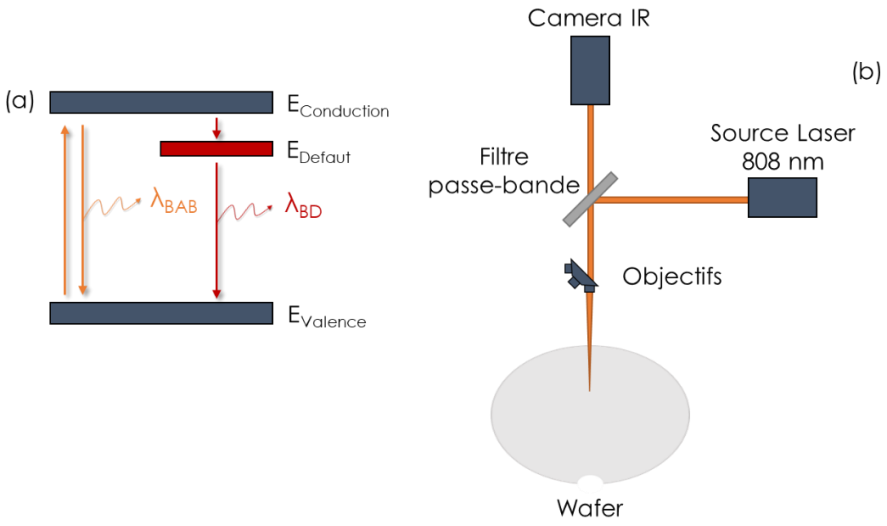

Figure 27 - (a) Diagramme d'énergie de photoluminescence montrant comment des défauts dans le Silicium induisent une perte d'intensité dans le signal bande-à-bande (BAB) et (b) schéma de fonctionnement de l'équipement de photoluminescence.

Dans le cadre de cette thèse, cet équipement fut utilisé avec la seconde approche d'imagerie macroscopique de photoluminescence (MPL) afin d'obtenir des images full wafer permettant le contrôle du process à large échelle (Figure 28). Cette fois, le laser d'excitation est transformé par différentes optiques en faisceau rectangulaire (80mm × 200µm) qui balaye la surface et le signal de photoluminescence est ainsi mesuré sur l'intégralité d'une plaque de 300mm en 4 passages.

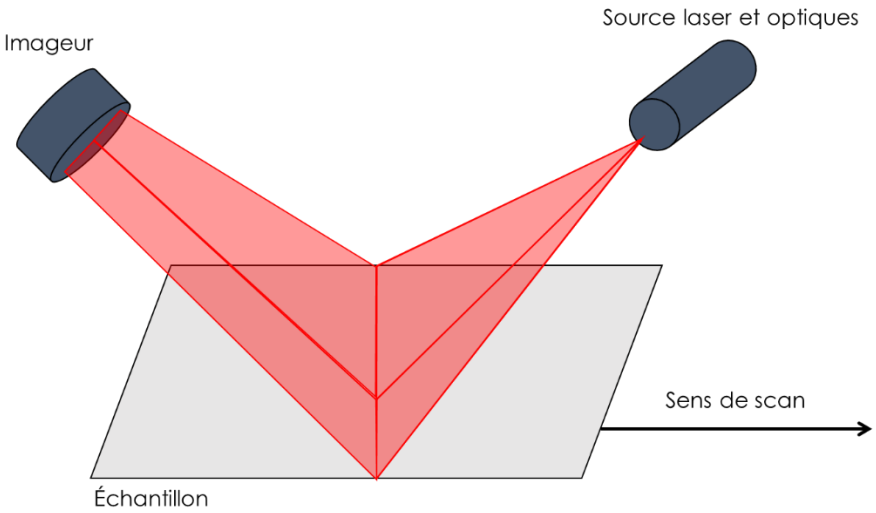

Figure 28 - Schéma de fonctionnement de l'acquisition de MacroPhotoLuminescence (MPL).

Une étape spécifique d'une technologie imageur fut sélectionnée pour évaluer la capacité de cet outil pour contrôler des dérives de procédés entrainant notamment des pertes de rendement des puces. L'étape choisie est un recuit du wafer, permettant aux défauts de diffuser dans le substrat est ainsi d'être plus facilement détectable par la photoluminescence de par leur empreinte plus large. Cet

équipement est équipé d'un filtre passe-bande permettant d'étudier uniquement les transitions bande-à-bande du Silicium, ce qui permet alors de détecter des anomalies caractérisées par une baisse de l'intensité de ce signal issue de la contamination du wafer. La présence des contaminants est visible par la présence de 'taches sombres' sur les images de wafer (Figure 27a). Cette perte de signal peut être due à de nombreuses causes [13] [14], provenant de l'intégralité de l'épaisseur du substrat et est induite par la variation de la mobilité des charges libres lors de la diffusion des défauts pendant le recuit. On peut observer ce phénomène sur la Figure 29, où des marques caractéristiques des systèmes de support (holders) lors des mesures ont induits de la défectivité en face arrière, qui ont ensuite diffusé lors du recuit, et sont ainsi détectée par l'acquisition de MPL. Ce type de tache sombres est présent sur la plupart des images de MPL obtenues mais leur intensité est bien moins forte que celles induites par des contaminants. Dans la très grande majorité des cas, la contamination sur la face arrière issue des holders n'entrainent pas de perte de performances des puces, les contaminants ne diffusant pas jusqu'à la surface. Par conséquent, il est nécessaire de différencier ces taches communes des taches atypiques pouvant entrainer des pertes de rendements des imageurs lors du suivi de procédé.

Figure 29 - Cartographie de MPL montrant des marques caractéristiques de supports de mesure induisant de la défectivité en face arrière détectée par la chute du signal BAB.

f. Microscopie acoustique

Avec la démocratisation de l'intégration 3D dans l'industrie des semi-conducteurs, la nécessité d'assurer la qualité de la liaison et du collage entre deux plaques de Silicium est primordial. Dans ce contexte, la microscopie acoustique connait un essor considérable. En effet, via l'utilisation d'un transducteur, permettant à la fois d'émettre des ondes sonores, et de mesurer les ondes réfléchies par l'échantillon étudié, on peut obtenir des informations sur la présence de nombreux défauts (délaminations, particules incluses, fissures…) [15], et notamment de « voids », soit des poches de vide/air entre les deux plaques. En effet, contrairement aux interfaces planes crées par le process, qui renvoient une onde identique à celle émise par le transducteur, mais à amplitude réduite, les voids entrainent une variation du signal sonore permettant de différencier ces dernières (Figure 30). L'équipement industriel utilisé durant mes travaux de thèse est le SAM 300 de PVA TePla, dont le schéma de fonctionnement est présenté en Figure 30.

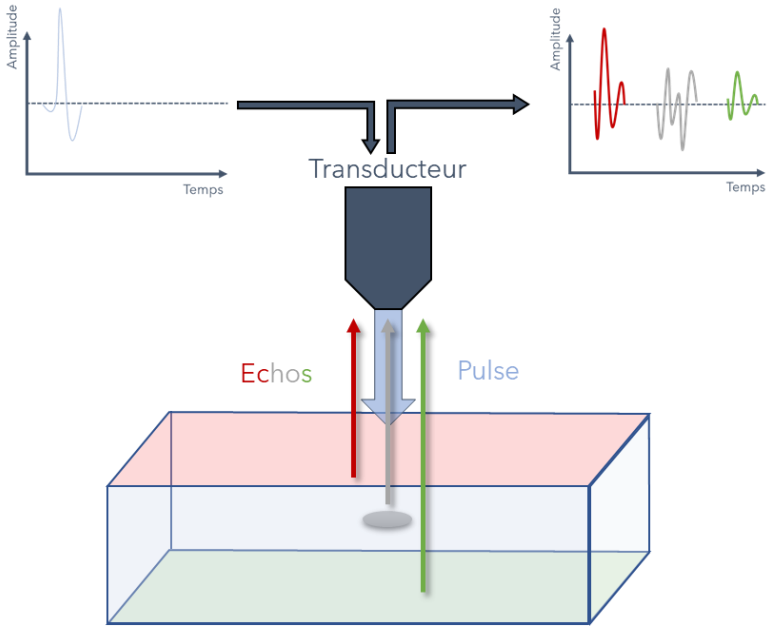

Figure 30 - Schéma de fonctionnement du microscope acoustique montrant un A-scan d'un défaut, permettant sa détection par la variation du signal qu'il induit.

Afin d'optimiser le transfert des ondes sonores du transducteur à la plaque, cette dernière est plongée dans de l'eau. La fréquence de ces ondes est typiquement comprise entre 15 et 300 MHz, jusqu'à 1GHz (pour des hautes résolutions mais limité en profondeur de pénétration). En effet, la résolution ainsi que la profondeur d'inspection dépendent non seulement de la fréquence de l'onde choisie, mais aussi des matériaux et de la structure étudiée. Plusieurs modes d'acquisitions sont alors possibles permettant chacun d'obtenir des informations spécifiques sur l'état de la structure étudiée. Celui que nous avons utilisé durant ma thèse est le « A-scan », où les amplitudes des ondes sonores réfléchies (échos) sont étudiées en fonction du temps pour chaque point du wafer. L'entière surface du wafer est alors scannée et le niveau de gris du pixel résultant sur l'image full wafer est déterminé par l'étude de ce signal. Un exemple de cartographie générée par ce type de scan est présenté en Figure 31 où les défauts sont visualisés par des régions blanches.

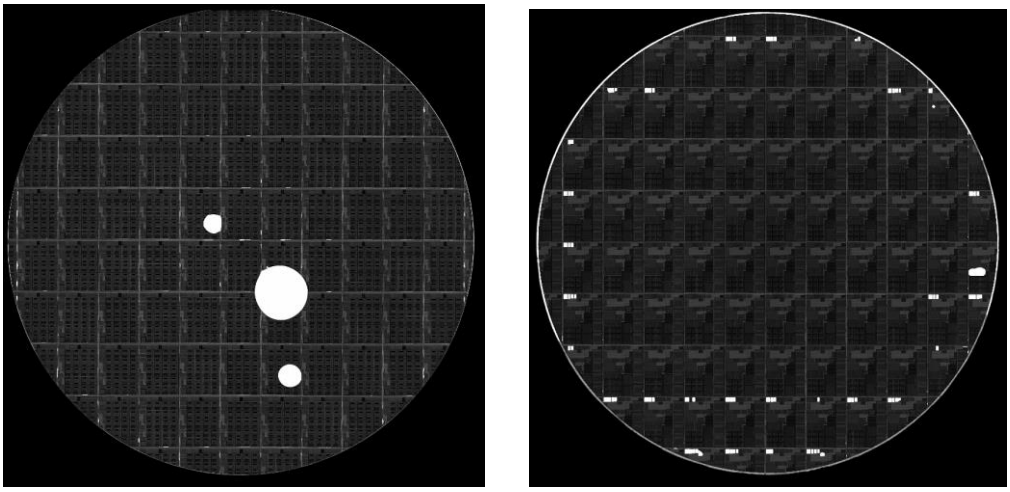

Figure 31 - Exemple de cartographies obtenues par le PVA TePla 300, permettant la détection des voids.

2. Réseaux neuronaux

L'intelligence artificielle et spécifiquement les réseaux neuronaux ont longtemps fait l'objet d'une certaine retenue de la part de la métrologie dans l'industrie de la microélectronique, du fait de l'aspect « boite noire » de ces approches, mais aussi du manque de critères statistiques (justesse, répétabilité, etc.) fournies par ces approches. Mais au vu des résultats obtenus grâce à ces outils, notamment en *Automatic Defect Classification* (ADC), on observe une tendance actuelle à l'utilisation de ces algorithmes pour traiter le large volume de données générées par les sites de productions de semi-conducteurs.

a. Machine learning

Pour comprendre le fonctionnement des réseaux neuronaux, il faut d'abord s'intéresser au cas plus large du *machine learning*. Ces algorithmes opèrent une forme d'apprentissage sans être explicitement programmés, utilisant pour cela des larges volumes de données. Ici, l'aboutissement est la génération de modèles prédictifs (Figure 32).

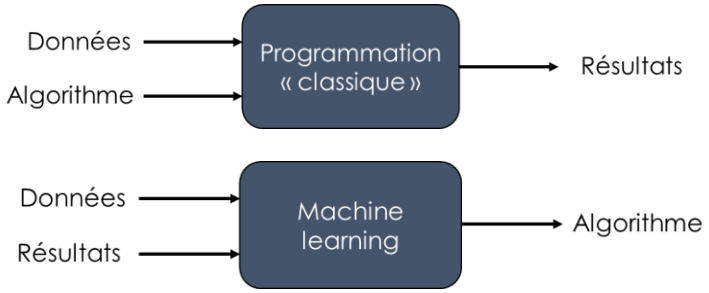

Figure 32 - Différence entre programmation classique et machine learning

Afin de pouvoir créer ces modèles, il est souvent nécessaire de traiter les données d'entrée afin d'en extraire les caractéristiques discriminantes, permettant ensuite à l'algorithme de *machine learning* d'être capable de classifier les données d'entrée. Ces fonctions d'extraction, couramment appelés descripteurs, doivent être manuellement sélectionnées et optimisées pour obtenir les meilleurs résultats après traitement par l'algorithme IA. Cela nécessite alors une expertise ainsi que du temps, mais permet d'obtenir des algorithmes légers et rapides à entrainer et à exécuter (Figure 33).

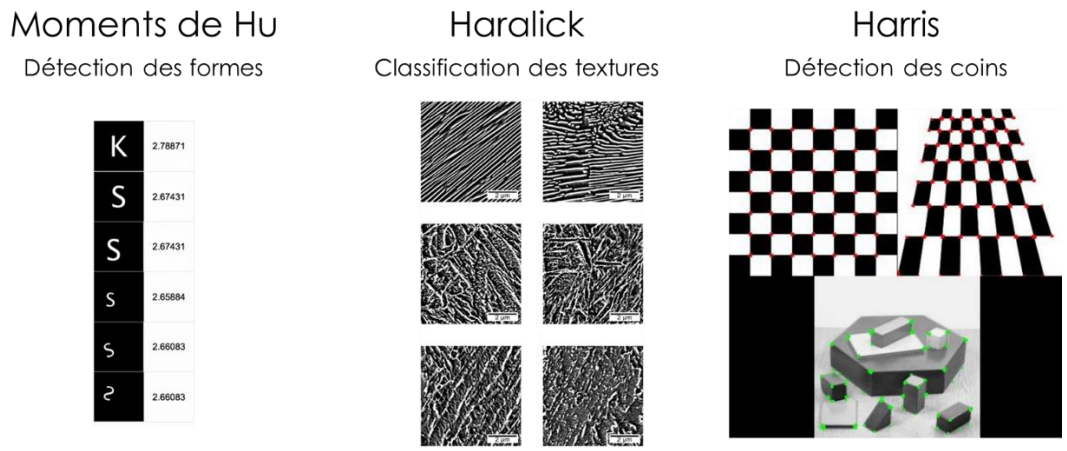

Figure 33 - Exemples de descripteurs, et de leurs fonctions. Ils reposent tous sur des traitements mathématiques à optimiser manuellement pour chaque cas d'application.

Cependant, il n'est pas toujours évident de déterminer ces fonctions, et encore moins de les optimiser, notamment lorsque les données d'entrée sont complexes, et quand leurs relations aux résultats ne sont pas facilement déductibles. Dans ce cas, les réseaux de neurones sont privilégiés car ils implémentent effectivement ces fonctions d'extraction dans leur structure ; cette structure étant optimisée automatiquement durant l'entraînement à l'aide des données d'apprentissage. Cela induit naturellement un besoin accru en données et en temps de calcul, mais permet dans certain cas d'obtenir des meilleures performances prédictives que des algorithmes de machine learning classiques (SVM, *Random Forest*…).

b. *Neurones artificiels*

Les réseaux de neurones visent à imiter le fonctionnement du cerveau humain, et plus spécifiquement la reconnaissance de motifs et la transmission d'informations entre les différentes couches de connexions neuronales. Les neurones artificiels les composants sont des fonctions d'activation mathématiques simples (Sigmoid, ReLU, etc. - Figure 34), permettant le traitement des données fournies par le neurone précèdent. En effet, chaque neurone récupère les données générées par les neurones des couches précédentes, et produit une sortie unique, distribués aux neurones des couches suivantes. Les neurones de la couche de sortie doivent, à partir des poids de la dernière couche cachée, permettre la détermination de la sortie finale attendue par le réseau. La sortie de chaque neurone est donc déterminée par la somme pondérée des valeurs de sortie des neurones précèdent, auquel on ajoute un biais, permettant de mieux ajuster les prédictions aux données d'entrée. L'entrainement d'un réseau de neurone repose ainsi sur l'optimisation du poids et du biais de chacun de ces neurones afin de créer un modèle prédictif.

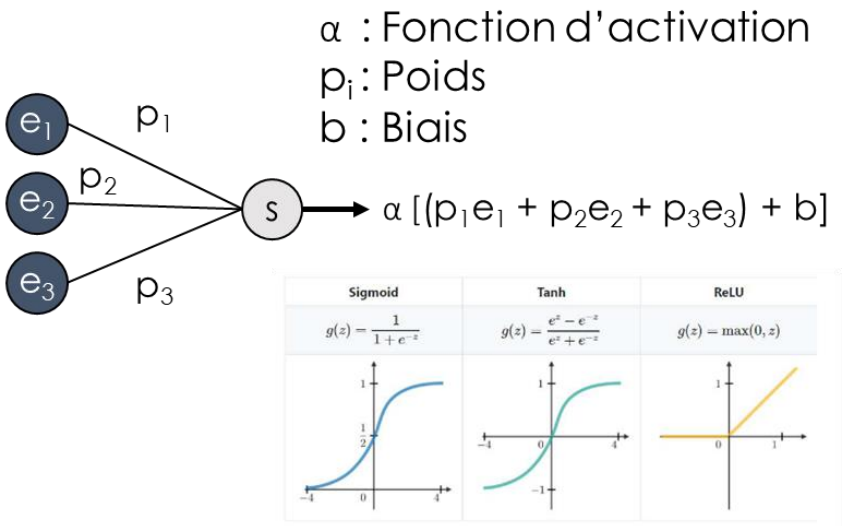

Figure 34 - Schéma du fonctionnement des neurones et exemples de fonctions d'activation couramment utilisées. Le neurone « S » combine alors les 3 entrées (e_1 - e_2 - e_3) pour déterminer, à l'aide de la fonction d'activation, s'il doit s'activer ou non.

c. *Réseaux de neurones*

Les réseaux de neurones sont des associations de ces éléments en couches. Ils sont alors, au minimum, composé d'une couche d'entrée et de sortie, et de couches « cachées » les connectant. Lorsque ces réseaux sont composés de multiples couches cachées, on parle alors de réseaux de neurones profonds (*Deep Neural Networks* – DNN), dont un exemple est présenté en Figure 35. Les dimensions de la couche d'entrée et de sortie sont donc liées à celles des données utilisées pour

l'entrainement. En effet, chaque neurone d'entrée est représentatif d'un point de donnée (e.g. valeur d'un spectre à une longueur d'onde spécifique) et les neurones de sortie représentent les valeurs inférées par les données d'entrée fournies, qui peuvent être discrètes (e.g. décision good/bad ou classification) ou continues pour un modèle de régression (e.g. mesures de métrologie).

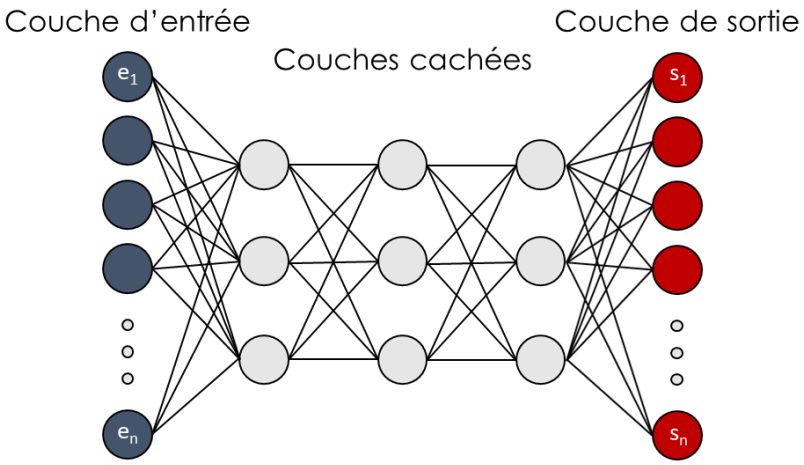

Figure 35 - Structure typique d'un réseau de neurones profond (3 couches cachées), les dimensions des couches d'entrées et de sorties étant alors identiques à celle des données fournies pour son entrainement.

d. Apprentissage et optimisation

La façon dont on fait apprendre aux réseaux de neurones consiste donc à réduire l'erreur obtenue lors de la prédiction. Cette erreur est définie par la différence entre la valeur prédite et celle fournie en tant que donnée d'entrainement. On parle alors d'entrainement supervisé, puisque le réseau de neurone connait la sortie attendue de toutes ses données d'entrée. L'entrainement consiste donc à modifier le poids des neurones et observer la variation de cette erreur, afin de trouver les poids optimaux, générant la plus faible erreur possible. Cette optimisation est gérée par le taux d'apprentissage (Learning Rate – LR), qui régule l'intensité de variation des poids du réseau à chaque itération et qui permet, lorsque sa valeur est adaptée au problème étudié, de faire progresser le réseau, tout en étant capable de ne pas rester encré dans un minimum local (Figure 36).

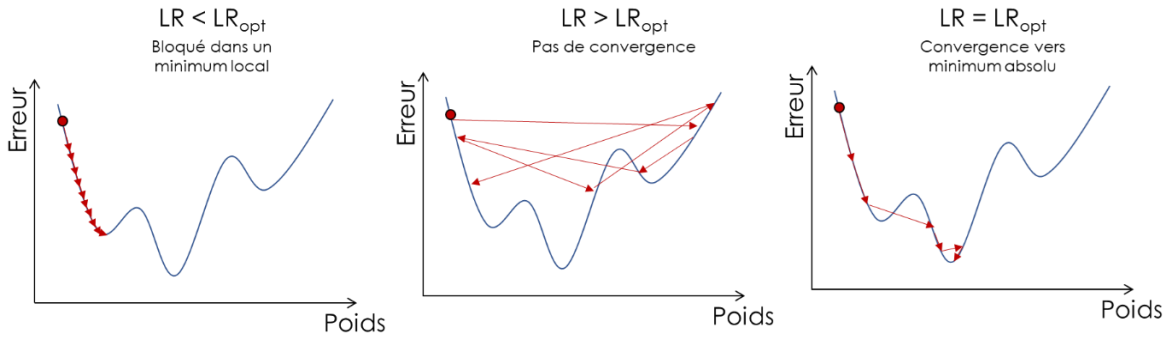

Figure 36 - Représentation de l'effet du taux d'apprentissage sur la qualité de l'entrainement d'un réseau de neurones.

Cette valeur est l'une des principales optimisations à réaliser lors de la création et l'entrainement d'un réseau de neurones, et de multiples itérations d'entrainement sont la plupart du temps nécessaire pour la déterminer.

Le taux d'apprentissage peut soit être fixée sur une valeur en début d'entrainement (descente stochastique du gradient), où évoluer au fur et à mesure, à l'aide d'un « optimiseur ». Ces algorithmes permettent de déterminer le LR optimal en calculant les gradients locaux du problème, et ainsi permettre de plus rapidement converger vers les poids optimaux, en utilisant des forts LR loin de la solution, puis en réduisant sa valeur pour affiner le modèle. L'un des optimiseurs les plus répandu est « ADAM », qui combine les avantages de deux autres approches typiques de descente de gradient :

- Algorithme de gradient adaptatif (AdaGrad) qui maintient un LR fixe par paramètre, adapté aux problèmes à gradient épars (e.g. langage naturel et vision par ordinateur).
- Propagation de la moyenne quadratique (RMSProp) où les LR sont adaptés par paramètre à la moyenne des gradients (e.g. à quelle vitesse les poids changent). Ils sont alors performants pour les problèmes non-stationnaires (e.g. bruités).

En effet, plutôt que d'adapter les LR de chaque paramètre à la moyenne (1er moment) comme dans RMSProp, cette variation est basée sur la moyenne du second moment (la variance). ADAM permet alors de converger plus rapidement que ces deux approches, et cela pour la plupart des problèmes, quel que soit leur nature.

La détermination du LR optimal à tout moment de l'entrainement n'est pas suffisant pour garantir un modèle performant en fin d'entrainement. En effet, la quantité d'entrainement a une grande influence sur la qualité prédictive du réseau neuronal, et principalement sur sa qualité de généralisation. En effet, le réseau neuronal n'ayant à disposition qu'une quantité limité de donnée d'entrainement (aussi grande soit-elle), il aura tendance à créer un modèle extrêmement spécifique à ce jeu de donnée si l'entrainement est prolongé (Figure 37).

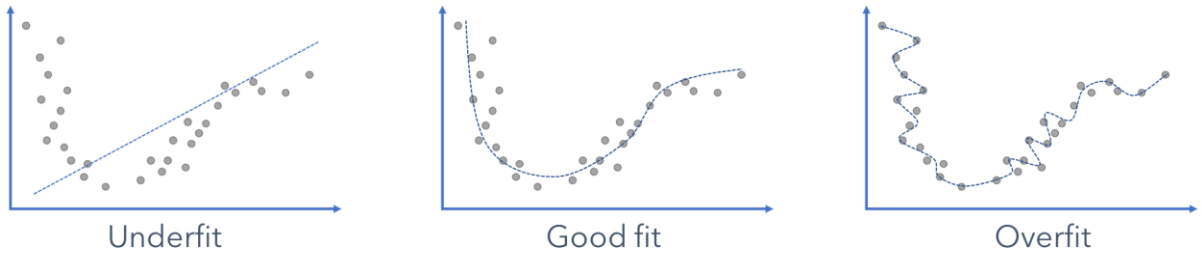

Figure 37 - Visualisation de l'influence du temps d'entrainement sur la qualité du modèle généré, représenté par les 3 régimes typiques.

e. Validation de l'entrainement

Afin de déterminer le point où le réseau de neurones est entrainé, une des approches régulièrement utilisée est l'usage d'un second jeu de donnée, nommé « dataset de validation ». Ce dernier ne sera pas utilisé directement pour l'entrainement du réseau, mais sera régulièrement évalué durant l'entrainement par ce dernier pour observer si, comme pour le dataset d'entrainement, l'erreur décroit bien au fur et à mesure des itérations d'optimisation. On peut ainsi fixer l'arrêt de l'entrainement lorsque l'erreur diminue sur le dataset d'entrainement, mais stagne, ou augmente, sur le dataset de validation. Ainsi, on assure que le réseau génère un modèle capable de généraliser, et non pas uniquement adapté aux données d'entrainement fournies. La Figure 38 présente un cas typique d'entrainement de réseau de neurones, où sont présentées les courbes d'erreurs et de précision (opposées l'une à l'autre) pour les deux datasets, et on observe bien qu'à partir d'un certain

point (~22 itérations ou *Epochs*), seule la courbe d'erreur de l'entrainement diminue, tandis que celle de la validation augmente.

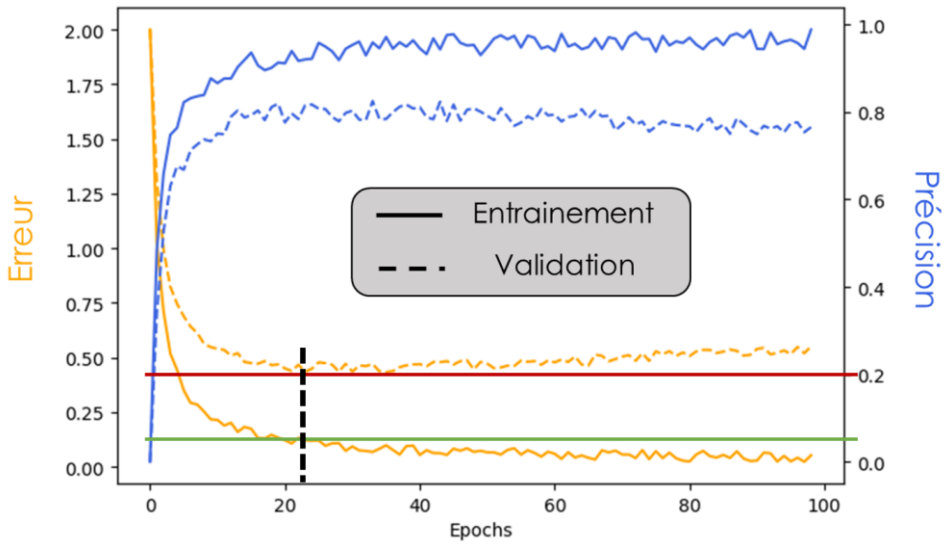

Figure 38 - Allure typique des courbes de précision et d'erreur pour les datasets d'entrainement et de validation lors de l'entraînement d'un réseau neuronal.

L'entrainement de ces réseaux de neurones induit donc un nombre conséquent de calculs, chaque itération (epoch) nécessitant le traitement de toutes les données d'entrainement par le réseau pour l'évaluation de sa progression. De plus, afin d'être capable de créer des modèles complexes, le nombre de couches, et le nombre de neurones les composants doivent être augmentés. Des réseaux de neurones complexes peuvent donc être constitués de centaines de couches, composées de dizaine de milliers de neurones, nécessitant quelques milliards de calculs (FLOPs - floating point operations) pour chaque passage de données dans le réseau [16]. On comprend alors pourquoi malgré leur invention dans les années 1940, l'essor de ces réseaux de neurones a tardé. En effet, c'est grâce à l'avènement dans les années 2010 de puces de calcul à l'architecture massivement parallèle, les GPUs (Graphical Processing Units), qui permettent de réaliser ce nombre de calcul dans un temps raisonnable, que les réseaux de neurones ont pu révolutionner le monde de l'intelligence artificielle.

Afin de réaliser ces entrainements, des frameworks (bibliothèques et outils logiciels) sont disponibles. Ces derniers sont dans leur très grande majorité sous licence libres ou très permissives (MIT, Apache, etc.) permettant leur utilisation gratuite sous la condition de citer le copyright original. Les deux principaux sont PyTorch [17], issu des équipes de recherche de Facebook, et Tensorflow [18], développé par Google. Ils sont très simples d'utilisation, et extrêmement bien documentés, ce qui permet d'optimiser aisément la structure et la phase d''entrainement des réseaux de neurones sur lesquels on travaille.

Durant ma thèse, j'avais à ma disposition au LTM, une plateforme de calcul nommée "Carri", équipé de 2 processeurs Intel i9-9820X (soit 20 cœurs à 3.30 GHz), 125 Go de mémoire vive, et surtout de 2 GPUs Nvidia Titan X (GPU/RAM : 1350/7000 MHz, 24 Go VRAM, 130 TFLOPS), qui m'ont permis d'entrainer et d'évaluer de nombreux réseaux de neurones sur des cas d'intérêt industriels.

III. Conclusion

Le monde de la métrologie industrielle est en pleine évolution, poussée par la 4ème révolution industrielle, et ses limitations, induites par sa dépendance aux mesures basées sur des modèles rigoureux, dans des cibles aux dimensions de plus en plus réduites, sont alors un axe privilégié de progression. Dans ce contexte, des approches de métrospection sans modélisation et basée sur le traitement intelligent de données brutes obtenus par les nombreuses techniques variées de suivi de procédés pourrait ainsi apporter des solutions et ainsi améliorer globalement le procédé de fabrication des semi-conducteurs.

Références

[1] B. Ho and M. Inokuchi, "SEM ADC (Auto Defect Classification): How it improves the Cost of Ownership without Risk of Yield Loss," *IEEE/SEMI Advanced Semiconductor Manufacturing Conference,* pp. pp. 293-298, 2007.

[2] B. Bunday, A. Cepler, A. Cordes and A. Arceo, "CD-SEM metrology for sub-10nm width features," *SPIE Advanced Lithography,* 2014.

[3] F. Dettoni, T. Shapoval, R. Bouyssou, T. Itzkovich, R. Haupt and C. Dezauzier, "Image based overlay measurement improvements of 28nm FD-SOI CMOS front-end critical steps," *Metrology, Inspection, and Process Control for Microlithography XXXI, International Society for Optics and Photonics.,* p. 101450C, 2017.

[4] J. Fanton, J. Opsal, D. L. Willenborg, S. M. Kelso and A. Rosencwaig, "Multiparameter measurements of thin films using beam-profile reflectometry," *Journal of Applied Physics 73,* pp. 7035-7040, 1993.

[5] P. Chou, A. Rao, M. Sturzenbecker, F. Wu and V. H. Brecher, "Automatic defect classification for semiconductor manufacturing," *Machine vision and applications,* pp. 9(4), 201-214, 1997.

[6] A. Kovalenko, P. Lenhard and R. Lenhard, "Deep learning techniques for integrated circuit die performance prediction.," *MRS Advances,* 2022.

[7] T. A. Brunner, Y. Zhou, C. Wong, B. Morgenfeld, G. Leino and S. Mahajan, "Patterned wafer geometry (PWG) metrology for improving process-induced overlay and focus problems," *Proc. SPIE 9780, Optical Microlithography XXIX,* vol. 97800W, 2016.

[8] V. Brouzet, V. Gredy, F. Chenevas-Paule, K. Le-Chao, D. Guiheux, A. Laurent and D. Le-Cunff, "Full Wafer Stress Metrology for Dielectric Film Characterization: Use Case," *30th SEMI ASMC,* 2019 .

[9] J. Gong, P. Vukkadala and K. Turner, "Determining local residual stresses from high resolution wafer geometry measurements," *J. Vac. Sci. Technol. B31,* p. 051205, 2013.

[10] K. Li, X. Huang, Q. Chen, G. Xu, Z. Xie, Y. Wan and F. Gong, "Flexible fabrication of optical glass micro-lens array by using contactless hot embossing process," *Journal of Manufacturing Processes,* vol. 57, pp. 469-476, 2020.

[11] H. Alalwan, C. J. Nile, R. Rajendran, R. McKerlie, P. Reynolds, N. Gadegaard and G. Ramage, "Nanoimprinting of biomedical polymers reduces candidal physical adhesion," *Nanomedicine: Nanotechnology, Biology and Medicine,* vol. 14(3), pp. 1045-1049, 2018.

[12] Z. Y. Xu, M. Gal and M. Gross, “Photoluminescence studies on porous silicon,” *Applied physics letters,* vol. 60(11, pp. 1375-1377, 1992.

[13] R. Duru, D. Le Cunff, M. Canna, I. Mica, J. Baruchel, T. Tran-thi and G. Brémond, “Photoluminescence imaging for buried defects detection in silicon: Assessment and use-cases,” *IEEE Transactions on Semiconductor Manufacturing,* vol. 32(1), pp. 23-30, 2018.

[14] T. Nassiet, R. Duru, D. Le Cunff, G. Brémond and J. Bluet, “The impact of surface volatge on photoluminesence response for the detection of copper and iron contamination in silicon,” *Physica status solidi,* vol. 2100410, 2021.

[15] F. Bertocci, A. Grandoni and T. Djuric-Rissner, “Scanning Acoustic Microscopy (SAM): A Robust Method for Defect Detection during the Manufacturing Process of Ultrasound Probes for Medical Imaging,” *Sensors,* Vols. 19, 4868, 2019.

[16] M. Kohler and S. Langer, “On the rate of convergence of fully connected deep neural network regression estimates,” *The Annals of Statistics,* no. 49(4), pp. 2231-2249, 2021.

[17] A. Paszke, S. Gross and F. Massa, “PyTorch: An Imperative Style, High-Performance Deep Learning Library,” in *Curran Associates, Inc.*, http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf, H. Wallach and H. Larochelle and A. Beygelzimer and A. d'Alché-Buc and E. Fox and R. Garnett, 2019, pp. 8024-8035.

[18] M. Abadi, A. Agarwal and P. Barham, “TensorFlow, Large-scale machine learning on heterogeneous systems,” Zenodo, 2015.

Chapitre II –
Traitement model-less de spectres

I. Introduction

Afin d'accélérer la courbe d'apprentissage de nouveaux développements technologiques, des stratégies dites *model-less*, c'est-à-dire sans développement de modèle, ont été introduites dans les départements de métrologie industrielle [1] [2]. Les techniques utilisées en métrologie sont, par essence, extrêmement sensibles aux variations des propriétés des matériaux, qu'elles soient géométriques, physico-chimiques ou encore physiques. De plus, avec l'essor actuel de l'intelligence artificielle, couplé à la 4ème révolution industrielle que nous vivons actuellement, ces approches *model-less* peuvent être favorisées par un traitement *machine learning*, permettant de générer, uniquement à partir des données brutes récoltées par les équipements, des algorithmes prédictifs fiables. Ces approches tendent souvent à associer des techniques de référence très précises, mais dont le développement de modèles est complexe et très couteux en temps et en argent, à des techniques plus simples et plus rapides, afin de s'affranchir de l'étape de modélisation.

L'approche évaluée durant ma thèse est celle de l'association de la scatterométrie et de la microscopie électronique (SEM-CD). En effet, dans le cadre de la mesure de dimensions critiques de structures périodiques, ces deux techniques sont souvent complémentaires : la scatterométrie est plus précise et répétable mais nécessite une modélisation complexe. De plus, la reconstruction de la réponse optique de structure à 3 dimensions est d'autant plus lourde en calcul [3], augmentant ainsi l'intérêt d'une approche model-less. Le SEM-CD est quant à lui rapide mais ne fournit de l'information que sur la surface de l'échantillon étudié.

Plusieurs approches IA sont possibles pour traiter ces données brutes, chacune ayant leurs propres spécificités. Les algorithmes classiques de *machine learning* ne nécessitent que peu de données pour être entraînés et sont relativement légers et rapides, mais requièrent une expertise en traitement avancé de données pour trouver les fonctions permettant de discriminer les différents cas étudiés. Ces approches sont privilégiées pour les cas simples où ces fonctions sont relativement évidentes. Lorsque les dimensions et la complexité des données augmentent, il est alors préférable de s'orienter vers des réseaux de neurones. En effet, ils possèdent l'avantage d'être capable d'automatiquement s'optimiser, mais nécessitent une plus grande quantité de données d'entraînement, ainsi qu'un temps de calcul allongé. Afin de traiter les données brutes obtenues par scatterométrie et SEM-CD, nous avons alors créé une architecture spécifique de réseaux de neurones qui sera présenté dans la partie 4.

Cette approche fut évaluée à l'aide de deux cas de complexité croissante : 1) Réseaux périodiques de lignes gravées dans le silicium (2D) et 2) réseaux de trous périodiques gravés (3D). Afin de valider cette approche, il est tout de même nécessaire de modéliser rigoureusement la réponse optique de ces structures par RCWA (*Rigorous Coupled-Wave Analysis),* qui sera présentée dans la partie 3. Et cela afin de s'assurer que l'algorithme d'IA est bien capable de généraliser son apprentissage, et pas simplement d'apprendre « par cœur » le jeu de données d'entraînement. De plus, cela nous permettra de définir des critères communs permettant de définir les avantages et inconvénients de notre approche, comparée à celle basées sur la modélisation rigoureuse.

Avant de présenter la modélisation du signal optique de ce type de structure périodique, nous allons introduire la technique sur laquelle repose cette approche : l'ellipsométrie.

II. Ellipsométrie

Pour rappel, l'ellipsométrie repose sur la mesure du changement de l'état de polarisation de la lumière après réflexion, et permet d'obtenir de nombreuses informations sur les caractéristiques physiques et morphologiques d'un échantillon plan.

1. Précisions théoriques

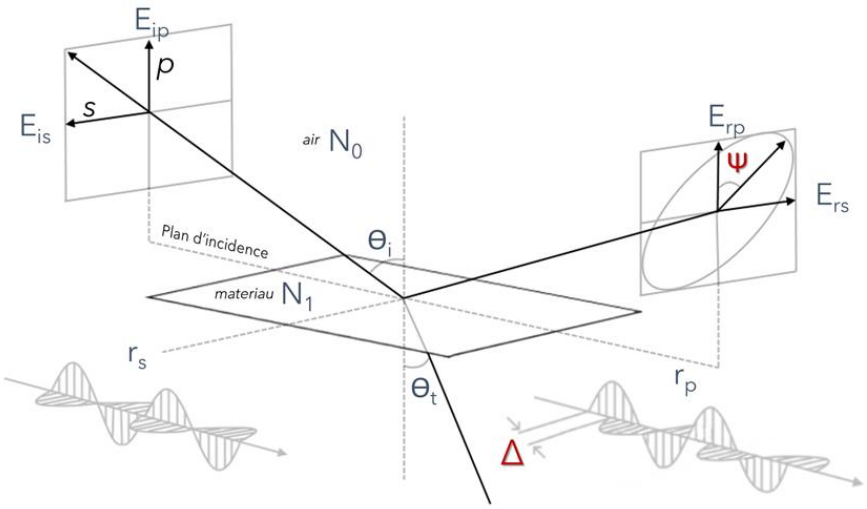

Figure 39 - Schéma représentatif des réflexions des axes de polarisation sur une surface

Considérons le cas de la Figure 39 représentant une onde plane rencontrant une surface avec un angle d'incidence Φ_0. Une partie de l'onde est transmise ou absorbée et une partie est réfléchie par la surface. Le champ électrique $\vec{E}^{\,i}$ peut être décomposé suivant un axe $\vec{E}^{\,i}_{p}$ parallèle au plan d'incidence et un axe $\vec{E}^{\,i}_{s}$ perpendiculaire au plan d'incidence. Le champ électrique après réflexion $\vec{E}^{\,r}$ peut être représenté par le coefficient de réflexion de l'échantillon pour une polarisation parallèle r_p et une polarisation perpendiculaire r_s :

$$r_p = \frac{E_p^r}{E_p^i} = |r_p|\, e^{-j\delta_p} \text{ et } r_s = \frac{E_s^r}{E_s^i} = |r_s|\, e^{-j\delta_s}$$

Les coefficients de réflexion sont des grandeurs complexes. Leur module $|r|$ représente la modification de l'amplitude de la composante du champ électrique et leur phase δ le retard introduit par la réflexion. En pratique, la quantité mesurée est le rapport entre le coefficient parallèle et le coefficient perpendiculaire que l'on écrit en ellipsométrie de la manière suivante :

$$\rho = \frac{r_p}{r_s} = \frac{|r_p|}{|r_s|}\, e^{-j(\delta_p - \delta_s)} = \tan\Psi e^{j\Delta}$$

On appelle donc « angles ellipsométriques » les paramètres Ψ et Δ qui représentent respectivement, l'amplitude réfléchie du champ, et la différence de phase après réflexion. Après réflexion d'une onde plane sur une surface, l'extrémité du vecteur du champ électrique décrit généralement une ellipse dont l'ellipticité est décrite par Ψ et l'angle de rotation par Δ. Les deux paramètres peuvent être mesurés de manière indépendante et absolue, aucune référence n'est

nécessaire. Ces deux angles, une fois mesurés, permettent par la suite de remonter aux épaisseurs et aux indices optiques des couches étudiées à l'aide d'un modèle mathématique. Pour rappel, l'objectif de ce stage est de s'affranchir de ces modèles et d'étudier les valeurs brutes des angles obtenus. Plusieurs techniques de mesure de polarisation par réflexion existent et elles utilisent toutes le montage optique suivant : une source, un polariseur, un analyseur et un détecteur. Ce sont les constituants de base auxquels peuvent être ajoutés différents éléments comme des modulateurs, un compensateur, etc.

Méthode de modulation par élément tournant

Les méthodes de mesure par élément tournant se prêtent bien à l'automatisation de la mesure ainsi qu'à son utilisation sur un large domaine spectral. Le faisceau peut être modulé en polarisation par la rotation du polariseur, de l'analyseur ou d'un compensateur. Dans cette partie, nous décrirons le fonctionnement des montages à polariseurs tournant.

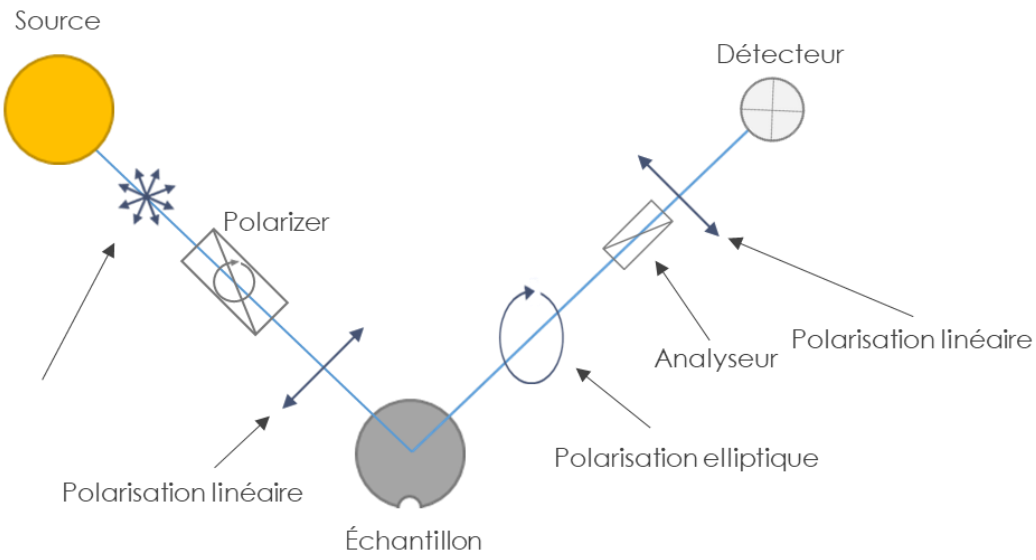

Figure 40 - Schéma fonctionnel d'un ellipsomètre à modulation par polariseur tournant

L'amplitude du champ est décomposée suivant les axes $\vec{S}$ et $\vec{P}$ (Figure 40). L'action de chaque élément sur la polarisation du faisceau peut être représentée par leur matrices de Jones respectives, et on peut alors déterminer l'intensité du champ que l'on obtiendra au niveau du détecteur :

$$\mathrm{I} = \left(|r_p|^2 \cos^2 A \cos^2 P + |r_s|^2 \sin^2 A \sin^2 P + \left(r_p r_s^* + r_p^* r_s\right) \cos A \sin A \cos P \sin P\right) \overrightarrow{|E_0|}^2$$

Où A et P sont les angles d'orientation des éléments polarisants, en notant r_p^* et r_s^* respectivement les complexes conjugués des coefficients de réflexion r_p et de r_s et sachant que :

$$\tan \Psi e^{\mathrm{j}\Delta} = \frac{r_p}{r_s}$$

où $\tan \Psi = \frac{|r_p|}{|r_s|}$ et $\Delta = \delta_p - \delta_s$.

Après linéarisation des sinus et cosinus, l'intensité s'écrit :

$$I = I_0(\alpha \cos 2P + \beta \sin 2P + 1)$$

Avec :

$$\begin{cases} \alpha = \dfrac{\tan^2\Psi - \tan^2 A}{\tan^2\Psi + \tan^2 A} \\ \beta = 2\cos\Delta \dfrac{\tan\Psi \tan A}{\tan^2\Psi + \tan^2 A} \\ I_0 = \dfrac{|r_s|^2 |E_0|^2}{2} \cos^2 A \,(\tan^2\Psi + \tan^2 A) \end{cases}$$

Dans la dernière expression de I, les coefficients α et β sont mis en évidence, la modulation étant réalisée par polariseur tournant (d'angle P = ωt). Pour un montage à analyseur tournant, les coefficients de cos 2A et sin 2A joueraient un rôle symétrique. Il est à noter que les coefficients α et β, soit les valeurs brutes mesurées par l'ellipsomètre, ne dépendent pas de l'intensité de la lampe $|E_0|^2$, ce qui permet de s'affranchir de toute mesure de référence de l'intensité du faisceau. À partir de α, β et A, on exprime aisément tan Ψ et cos Δ :

$$\tan\Psi = \sqrt{\frac{1+\alpha}{1-\alpha}}\tan A \quad \text{et} \quad \cos\Delta = \frac{\beta}{\sqrt{1-\alpha^2}}$$

Cette technique, relativement facile à mettre en œuvre, nécessite toutefois quelques précautions quant à la collimation du faisceau et à l'alignement des composants optiques pour atteindre un bon niveau de précision. Il est alors celui le plus souvent utilisé dans l'industrie [4], et notamment chez STMicroelectronics, pour la mesure d'épaisseur et d'indices optique.

Méthode à modulation de phase

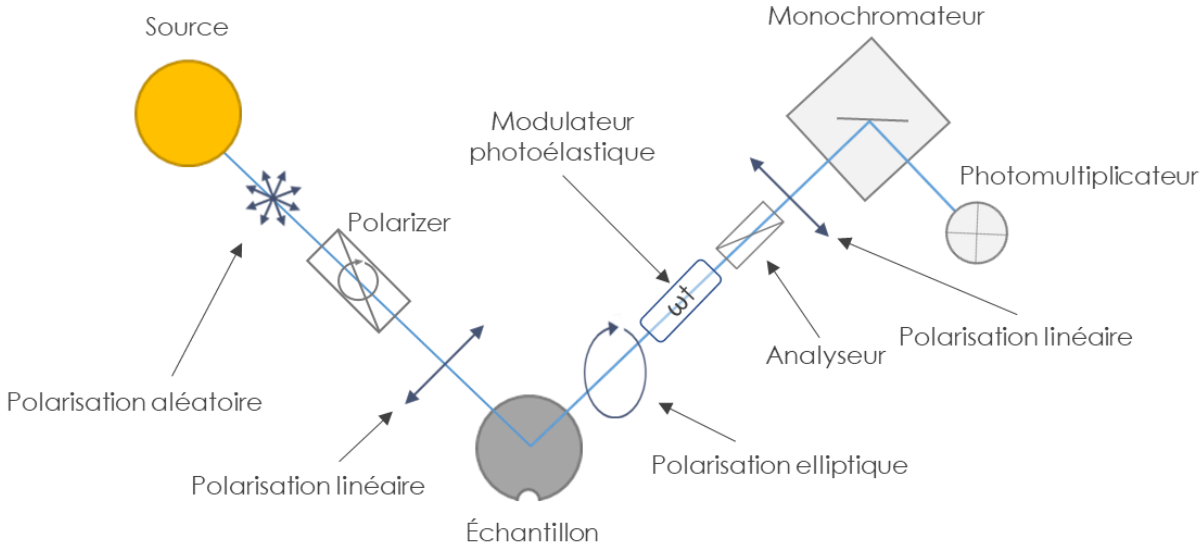

Figure 41 - Schéma fonctionnel d'un ellipsomètre à modulation de phase

Le montage optique (Figure 41) reprend les mêmes éléments que précédemment (source, polariseur, analyseur, détecteur), mais inclut un modulateur photoélastique après le polariseur. Dans cette configuration, aucune caractéristique particulière n'est requise au niveau de la polarisation pour la source et le détecteur. L'utilisation d'un modulateur sur un large domaine spectral nécessite toutefois quelques précautions d'utilisation.

L'état de polarisation du faisceau peut être décrit de la même manière que dans une configuration à polariseur tournant, auquel il faut ajouter la matrice de transfert du modulateur (ici un modulateur photoélastique) qui induit un déphasage de $\delta(t) = a \sin \omega t$. On peut alors déterminer l'intensité au niveau du détecteur :

$$I = \left|\overrightarrow{E_d}\right|^2 = I_0 + I_s \sin \delta(t) + I_c \cos \delta(t)$$

Avec :

$$\begin{cases} I_0 = B(1 - \cos 2\Psi + \cos 2A + \cos 2(P-M)\cos 2M\ (\cos 2A - \cos 2\Psi) \\ \qquad + \sin 2A \cos \Delta \cos 2(P-M) \sin 2\Psi \sin 2\Delta \\ I_s = B\ [\sin 2(P-M) \sin 2A \sin 2\Psi \sin \Delta] \\ I_c = B\ (\sin 2(P-M)\ [(\cos 2\Psi - \cos 2A)\ sin2M \\ \qquad + \sin 2A \cos 2M \sin 2\Psi \cos \Delta] \end{cases}$$

Où : $B = \frac{E_0^2}{4|r_p^2 + r_s^2|}$

Is et Ic sont alors les mesures brutes, et sont reliées à l'état de polarisation imposé par les polariseurs et le modulateur, ainsi qu'aux angles ellipsométriques. Typiquement, deux états de polarisation sont utilisés pour les mesures :

- Configuration II (P − M = ±45° ; M = 0° ; A = ±45°) : $Is = \sin 2\Psi \sin \Delta$ et $Ic = \sin 2\Psi \cos \Delta$
- Configuration III (P − M = ±45° ; M = ±45° ; A = ±45°) : $Is = \sin 2\Psi \sin \Delta$ et $Ic = \cos 2\Psi$

Les fonctions sin δ(t) et cos δ(t) sont définies par leur développement en série de Fourier :

$$\begin{cases} \sin \delta(t) = 2 \sum_{m=0}^{\infty} J_{2m+1}(a) \sin[(2m+1)\omega t] \\ \cos \delta(t) = J_0(a) + 2 \sum_{m=1}^{\infty} J_{2m}(a) \cos(2m\omega t) \end{cases}$$

Avec J_m fonction de Bessel de première espèce d'ordre m.

Dans l'exploitation du signal, seules les harmoniques d'ordre m = 0, 1 et 2 sont utilisées, les autres harmoniques n'apportant pas d'informations supplémentaires. Pour les conditions de mesure optimales, *a* doit vérifier la condition : $J_0(a) = 0$ (J_0 fonction de Bessel d'ordre zéro), or l'amplitude de modulation *a* est proportionnelle à la tension d'excitation V_m du cristal modulateur et inversement proportionnelle à la longueur d'onde λ :

$$a = \frac{QV_m}{\lambda}$$

Il conviendra donc d'ajuster la tension en fonction de la longueur d'onde. Cette méthode ne nécessite pas un alignement très précis (en théorie il n'y a pas d'éléments en rotation). Son utilisation

requiert toutefois une électronique performante, capable d'assurer la saisie du signal et son traitement à une fréquence compatible avec la fréquence de modulation de 50 kHz. Le modulateur devra être étalonné en fonction de la longueur d'onde, et la tension d'excitation asservie à celle-ci. D'avantages de précisions sur le fonctionnement de ce type d'ellipsomètre sont disponibles dans [5].

2. Scatterométrie

L'ellipsométrie peut être appliquée à l'étude de motifs périodiques, et on parle alors de scatterométrie (ou OCD pour Optical Critical Dimensions). En effet, la reconstruction de la réponse optique de ce type de structure permet, par résolution du problème inverse, d'en obtenir les dimensions caractéristiques (Figure 42). Cette technique est très couramment utilisée dans l'industrie des semi-conducteurs pour la caractérisation de structure périodiques due à sa haute précision ainsi qu'à sa rapidité de mesure.

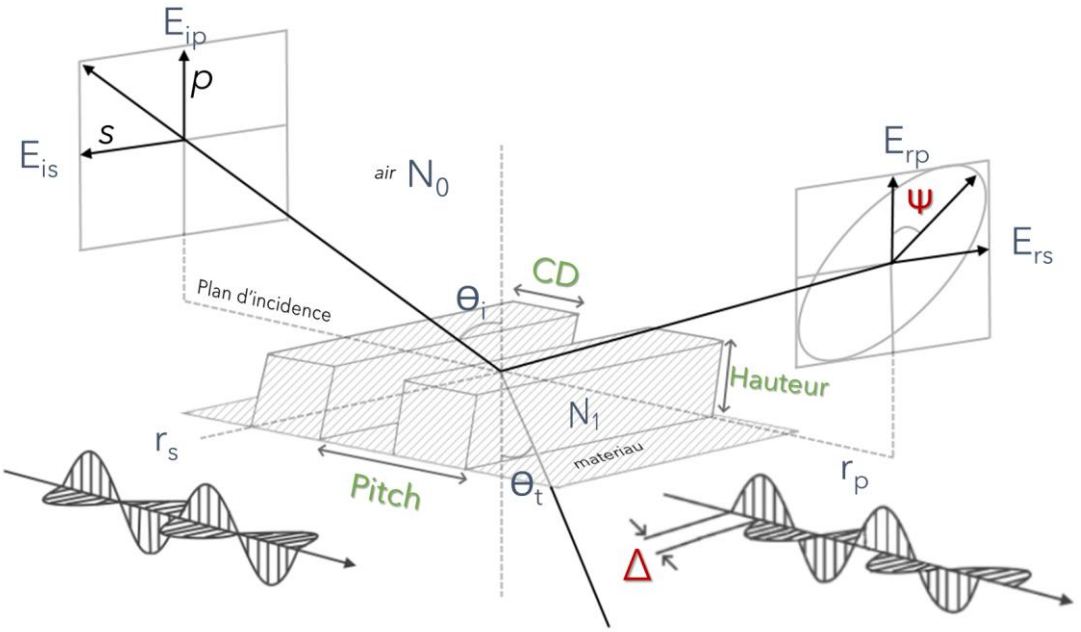

Figure 42 - Principe de la scatterométrie, soit de l'ellipsométrie appliquée à une structure périodique.

De la même façon que pour l'ellipsométrie, il faut ici définir précisément la structure afin de pouvoir modéliser sa réponse optique (par RCWA – Rigorous Coupled-Wave Analysis, qui sera présenté plus en détail plus tard). Afin d'obtenir une sensibilité maximale aux dimensions du motif, des états de polarisations supplémentaires à ceux utilisés en ellipsométrie traditionnelle peuvent être utilisés pour obtenir la signature ellipsométrique complète de la structure, appelée « Matrice de Mueller », et qui contient 16 spectres, dont seulement sept sont uniques, dues aux symétries de la structure. Un exemple de matrice de Mueller est présenté en Figure 43. On y voit que les éléments MM12 et MM21, MM14 et MM41, et MM24 et MM42 sont symétriques, tandis que les éléments MM13 et MM31, MM23 et MM32, et MM34 et MM43 sont antisymétriques, dans le cas d'un échantillon isotrope et uniforme.

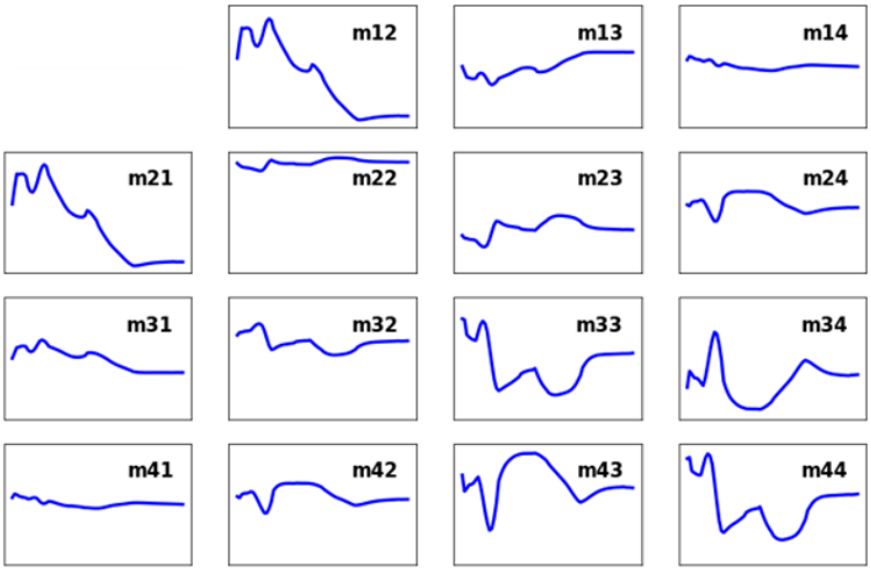

Figure 43 - Exemple de matrice de Mueller

III. Modélisation RCWA

Comme expliqué dans la partie précédente, la scatterométrie consiste en l'utilisation de l'ellipsométrie appliquée à des structures périodiques. Le principe de mesure est alors le même, l'équipement mesurant le rapport entre le coefficient parallèle et le coefficient perpendiculaire du champ électrique : $\rho = \frac{r_p}{r_s} = \tan\Psi\ e^{i\Delta}$. Seulement, dans le cas de structures non-isotropes, la réflexion entraîne des polarisations croisées (r_{sp} et r_{ps} non nuls) et peut aussi amener à de la dépolarisation. Afin d'être capable de caractériser ces matériaux, on utilise l'ellipsométrie spectroscopique à matrice de Mueller, permettant de représenter toutes ces contributions. De plus, afin d'augmenter la quantité d'information fournie à l'algorithme d'IA, il est préférable de caractériser entièrement l'état de polarisation de la lumière réfléchie par la structure périodique.

1. Formalisme

Contrairement à l'ellipsométrie conventionnelle, utilisant la représentation de Jones, il est préférable dans ce cas d'utiliser la représentation de Stokes, adaptée à l'étude de lumière partiellement polarisée, et où la polarisation d'une onde est décrite par les quatre paramètres de Stokes :

$$\vec{S} = \begin{bmatrix} S_1 \\ S_2 \\ S_3 \\ S_4 \end{bmatrix}$$

On peut relier la lumière incidente $\vec{S}_{in}$ à la lumière émergente $\vec{S}_{out}$ à l'aide de la matrice de Mueller M : $\vec{S}_{in} = \boldsymbol{M}.\vec{S}_{out}$. Pour un matériau quelconque, avec une matrice de Jones non-diagonale :

$$r = \begin{bmatrix} r_{pp} & r_{ps} \\ r_{sp} & r_{ss} \end{bmatrix}$$

La matrice de Mueller correspondante s'écrit alors :

$$M = \begin{bmatrix} M_{11} & M_{12} & M_{13} & M_{14} \\ M_{21} & M_{22} & M_{23} & M_{24} \\ M_{31} & M_{32} & M_{33} & M_{34} \\ M_{41} & M_{42} & M_{43} & M_{44} \end{bmatrix}$$

Où :

$$M_{11} = \frac{1}{2}\left(|r_{pp}|^2 + |r_{ss}|^2 + |r_{ps}|^2 + |r_{sp}|^2\right) \qquad M_{12} = \frac{1}{2}\left(|r_{pp}|^2 - |r_{ss}|^2 + |r_{ps}|^2 - |r_{sp}|^2\right)$$

$$M_{13} = \Re\left(r_{pp}r_{sp}^* + r_{ps}r_{ss}^*\right) \qquad M_{14} = \Im\left(r_{pp}r_{sp}^* + r_{ps}r_{ss}^*\right)$$

$$M_{21} = \frac{1}{2}\left(|r_{pp}|^2 - |r_{ss}|^2 - |r_{ps}|^2 + |r_{sp}|^2\right) \qquad M_{22} = \frac{1}{2}\left(|r_{pp}|^2 + |r_{ss}|^2 + |r_{ps}|^2 - |r_{sp}|^2\right)$$

$$M_{23} = \Re\left(r_{pp}r_{sp}^* - r_{ps}r_{ss}^*\right) \qquad M_{24} = \Im\left(r_{pp}r_{sp}^* - r_{ps}r_{ss}^*\right)$$

$$M_{31} = \Re\left(r_{pp}r_{ps}^* + r_{sp}r_{ss}^*\right) \qquad M_{32} = \Im\left(r_{pp}r_{ps}^* - r_{sp}r_{ss}^*\right)$$

$$M_{33} = \Re\left(r_{pp}r_{ss}^* + r_{sp}r_{ps}^*\right) \qquad M_{34} = \Im\left(r_{pp}r_{ss}^* + r_{sp}r_{ps}^*\right)$$

$$M_{41} = -\Im\left(r_{pp}r_{ps}^* + r_{sp}r_{ss}^*\right) \qquad M_{42} = -\Im\left(r_{pp}r_{ps}^* - r_{sp}r_{ss}^*\right)$$

$$M_{43} = -\Im\left(r_{pp}r_{ps}^* - r_{sp}r_{ss}^*\right) \qquad M_{44} = \Re\left(r_{pp}r_{ss}^* - r_{sp}r_{ps}^*\right)$$

Ces matrices définissent entièrement la réponse ellipsométrique d'un échantillon, et sont alors très utiles pour en caractériser les dimensions ainsi que la géométrie. L'ellipsomètre permet d'obtenir les matrices de Mueller expérimentales de l'échantillon, qu'il faudra alors comparer aux spectres modélisés. La génération de ces spectres est réalisée par RCWA, qui est un algorithme de calcul optique que nous allons maintenant décrire.

2. Définition de la géométrie

La première étape du protocole de simulation de réponse optique d'une structure par RCWA est la définition du problème : Géométrie de la structure (dimensions/angles), matériaux, et caractéristiques de la source (longueurs d'ondes, angles). Comme les motifs étudiés par scatterométrie sont périodiques, il n'est alors nécessaire de modéliser que la réponse d'une cellule élémentaire du réseau, répétée selon les différents axes de symétrie de la structure. On peut définir cet élément par une coupe selon chaque axe où la structure n'est pas homogène. Un exemple de cette étape de description du problème est présenté en Figure 44.

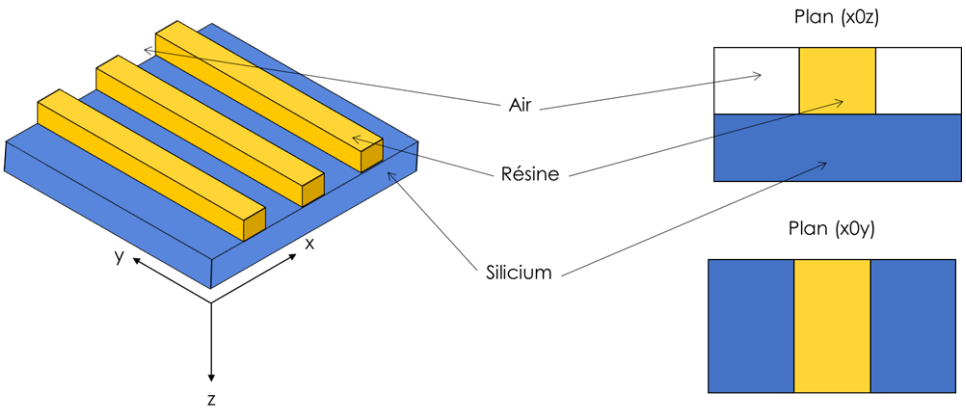

Figure 44 - Protocole de définition de l'élément périodiquement répété dans la structure, dont la réponse optique sera simulée par RCWA.

Afin que les symétries postulées dans le développement du modèle de la structure soient respectées, que toutes les interfaces soient planes, et les couches invariantes en z, tous les éléments doivent être décomposés sous forme de rectangles. Augmenter le nombre de couches améliore ainsi la reconstitution de la forme réelle de la structure, mais augmente aussi le nombre d'interface, et donc de calculs à réaliser, induisant un temps accru de traitement par l'algorithme (Figure 45).

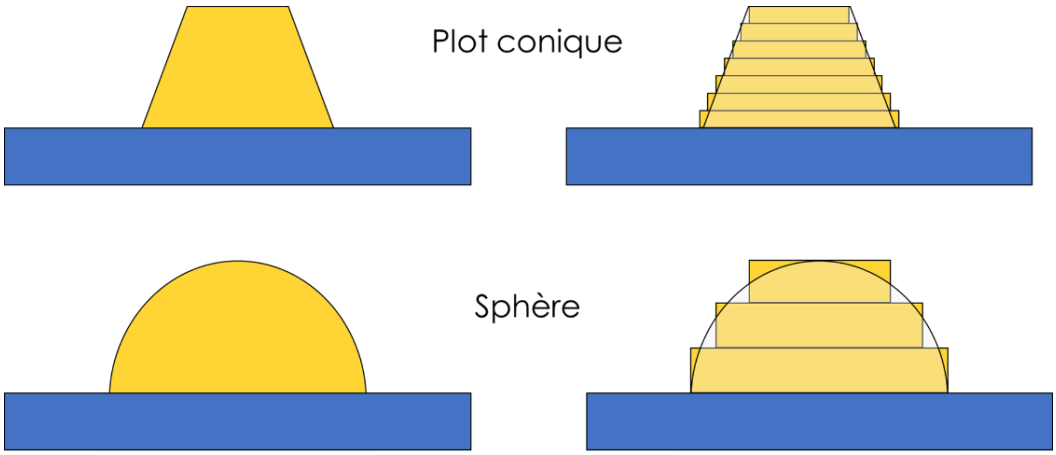

Figure 45 - Représentation de la segmentation nécessaire au traitement par l'algorithme RCWA

On définit alors les dimensions de cette structure, ainsi que les indices optiques (n et k) des matériaux la composant. Des études ellipsométriques préalables sont souvent nécessaires pour définir au mieux ces indices dont la précision impacte directement la qualité de la mesure par scatterométrie : ils sont en effet essentiels au calcul du problème direct (détermination du spectre optique à partir de la structure). La plage de longueur d'onde de la source ainsi que son échantillonnage doivent être renseignés, ces deux variables permettant d'augmenter la précision du modèle, au prix d'un coût en calcul plus élevé. De plus, la définition de l'angle d'incidence du faisceau lumineux sur la structure (azimut) est primordiale, car son choix influe très fortement sur la réponse optique.

3. Calcul de la réponse optique

À partir de ces données, l'algorithme RCWA peut alors calculer les matrices de diffractions (*scattering matrices*, ou matrices-S) pour chaque interface de la structure élémentaire, afin d'obtenir la matrice globale en les combinant. Ces matrices caractérisent les réflexions et transmissions à chaque interface, comme décrit dans la Figure 46. La matrice globale s'obtient ensuite grâce au

produit de Redheffer [6] [7] : il s'agit de la combinaison des matrices de diffusions de chaque couche de la structure avec celles des domaines de réflexion (typiquement l'air) et de transmission (typiquement le bulk de Silicium). Finalement, cela permet d'obtenir la réflectance (r_{pp} - r_{ps} - r_{sp} - r_{ss}) et la transmittance (t_{pp} - t_{ps} - t_{sp} - t_{ss}) théoriques de la structure, mais aussi les différents spectres de la matrice de Mueller à partir des équations présentées plus haut.

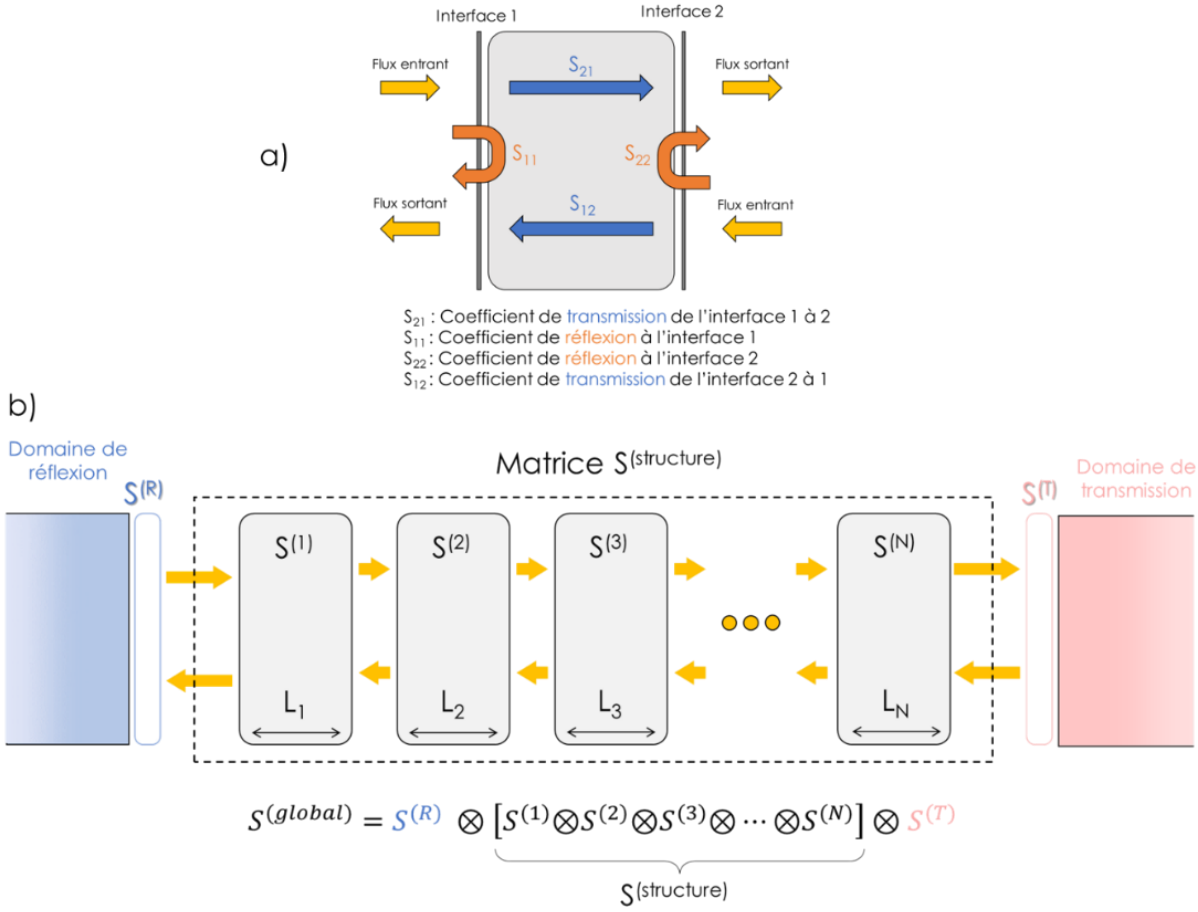

Figure 46 - a) Matrice de diffusion d'un élément de la structure et b) association en série des matrices de diffusions pour obtention de la matrice $S^{(global)}$.

Une fois le modèle crée, les paramètres géométriques de la structure peuvent être ajustés par comparaison au spectre expérimental.

4. Résolution du problème inverse

Chaque paramètre de cette structure (dimensions, indices optiques, etc.) peut alors être défini selon trois groupes :

- Fixés, qui ne varient donc pas durant l'ajustement.
- Flottants, libres de varier dans une gamme choisie pour améliorer le fit.
- Couplés, donc non-fixés, mais dont la variation est liée à d'autres paramètres (e.g. CD du haut d'une structure, liée à celle du bas).

Le nombre de paramètres flottant est libre, mais l'augmentation de ce dernier peut entraîner des instabilités du modèle, causé par des corrélations entre paramètres [8]. L'ajustement se déroule alors par des itérations successives des valeurs autorisées de paramètres flottant, avec l'objectif de réduire l'erreur résiduelle, définie par la MSE (*Mean Square Error*) :

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Valeur\ expérimentale_i - Valeur\ modélisée_i)^2$$

Afin d'accélérer ce processus, et ne pas avoir à simuler les spectres théoriques à chaque mesure, une large bibliothèque de spectre est générée en avance de phase (par résolution du problème direct), permettant de retrouver rapidement les dimensions géométriques correspondant aux spectres mesurés expérimentalement, définie comme la résolution du problème inverse.

5. Coût en calcul

Selon la géométrie de la structure et ses symétries, on peut parfois se limiter à l'étude 2D du problème (invariance selon selon un axe, comme dans le cas d'un réseau de lignes), mais lorsque ce n'est pas possible (réseau de plots par exemple), les temps de calculs sont alors plus longs (Figure 46). Par exemple, la simulation d'un réseau de lignes (Pitch : 500nm, CD : 300nm, Hauteur : 300nm) est réalisable rapidement, où un réseau de plots (Pitch : 500nm, CDx : 300nm, CDy : 600 nm, Hauteur : 300nm) nécessitera en moyenne 60 à 100 fois plus de temps de calcul (à même ordre de diffraction).

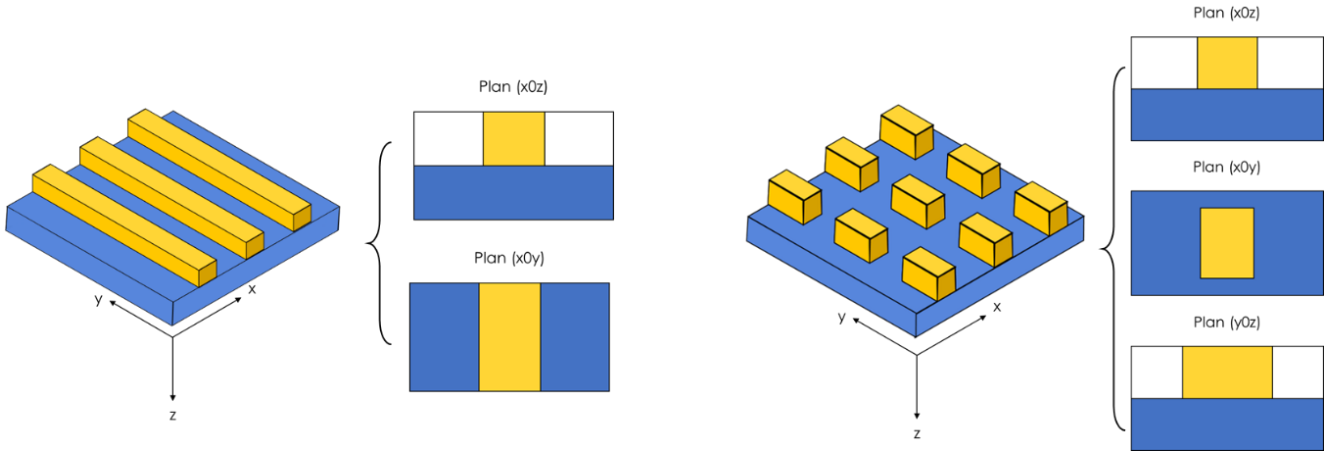

Figure 47 - Comparaison entre les structures périodiques définies pour la simulation RCWA pour une structure 2D (haut) et 3D (bas).

Ce temps de calcul étant essentiellement du au problèmes aux valeurs propres d'une compléxité $O(n^3)$. Or, « n » est proportionnel à M en 2D, donc $O(M^3)$, mais proportionnel à M^2 en 3D, donc $O(M^6)$. Il est donc d'intérêt d'étudier l'apport d'une approche basée sur l'intelligence artificielle, et plus précisement les réseaux de neurones, afin de pouvoir génerer des modèles predictifs capables de traiter quasi-instantanément les données issues d'une mesure de scattérometrie sur des structures complexes.

IV. Évaluation de l'approche sur des cas d'intérêt industriels

1. Protocole de création du réseau de neurones

Comme discuté plus tôt, la modélisation du signal de scatterométrie est très lourde, spécialement dans le cas de structures à trois dimensions. Notre objectif était de proposer une solution model-less de scatterométrie, reposant sur l'utilisation de réseau de neurones pour traiter les spectres bruts obtenus, et de pouvoir alors prédire les dimensions des structures étudiées. Des mesures SEM-CD ont été réalisées sur ces structures afin de fournir une référence nécessaire pour l'entraînement supervisé du NN.

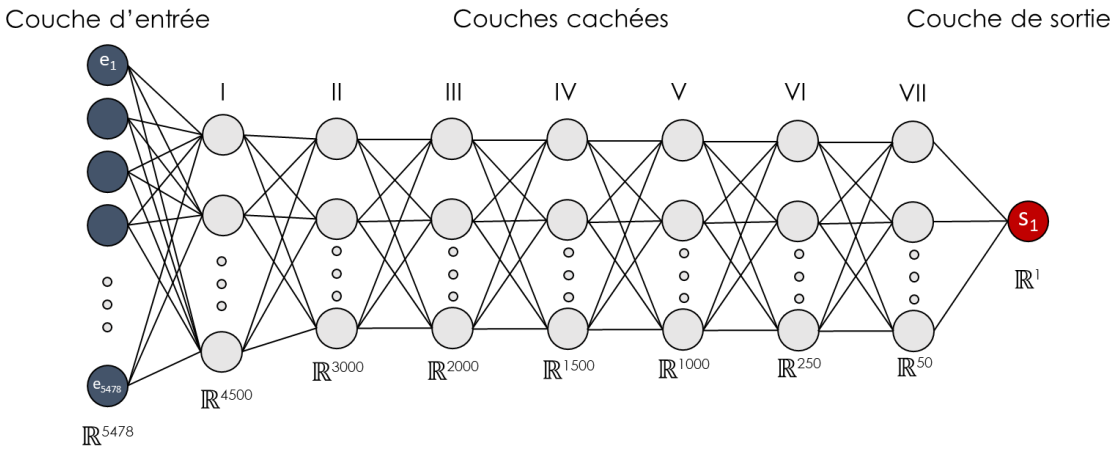

Figure 48 - Architecture du FCNN utilisé dans le traitement des spectres de scatterométrie associées aux mesures SEMCD.

Afin de traiter ces données, un FCNN (*Fully Connected Neural Network* – NN entièrement connecté), où chaque neurone d'une couche reçoit de l'information de l'entièreté des neurones de la couche précédente, a été spécifiquement créer en utilisant le framework Tensorflow, préféré à PyTorch pour sa simplicité d'utilisation et d'optimisation. Ces réseaux à l'architecture relativement simple sont extrêmement performants pour la génération de modèle de régression [9].

2. Pré-traitement des données

Ce type de réseau n'est en revanche capable que de traiter des données "plates" (à 1 dimension) en entrée et sortie. Or, nos données d'entrées sont des matrices de Mueller (11 matrices obtenues sur les 16 possibles – M11 identitaire donc sans information), soit autant de spectres de scatterométrie, et donc bidimensionnelles (longueurs d'ondes et valeurs). Comme toutes les acquisitions ont volontairement été réalisées avec des longueurs d'ondes identiques, afin d'éviter des problèmes de gammes ou d'échantillonnage, nous pouvons uniquement traiter les valeurs brutes de ces dernières pour chaque longueur d'onde et les concaténées pour n'obtenir qu'un vecteur à une dimension, traitable par le réseau de neurones. La Figure 49 explicite ce protocole de formatage des données. Les données de sortie (mesures SEM-CD de la structure) étant déjà à une dimension, elles n'ont pas eu à être traitées.

L'architecture du FCNN spécifiquement crée est présentée en Figure 48. On y observe bien que les couches d'entrée et de sortie sont de la même dimension que les données d'entrée (Spectres scatterométriques : 11 matrices x 497 longueurs d'ondes = 5478 données) et de sortie (Mesure SEM-CD : 1 donnée).

Alors, nous évaluons notre approche sur deux cas d'intérêt à complexité croissante : *Réseaux périodiques de lignes gravées dans le silicium (2D)*, puis des *Réseaux de trous périodiques gravés (3D).*

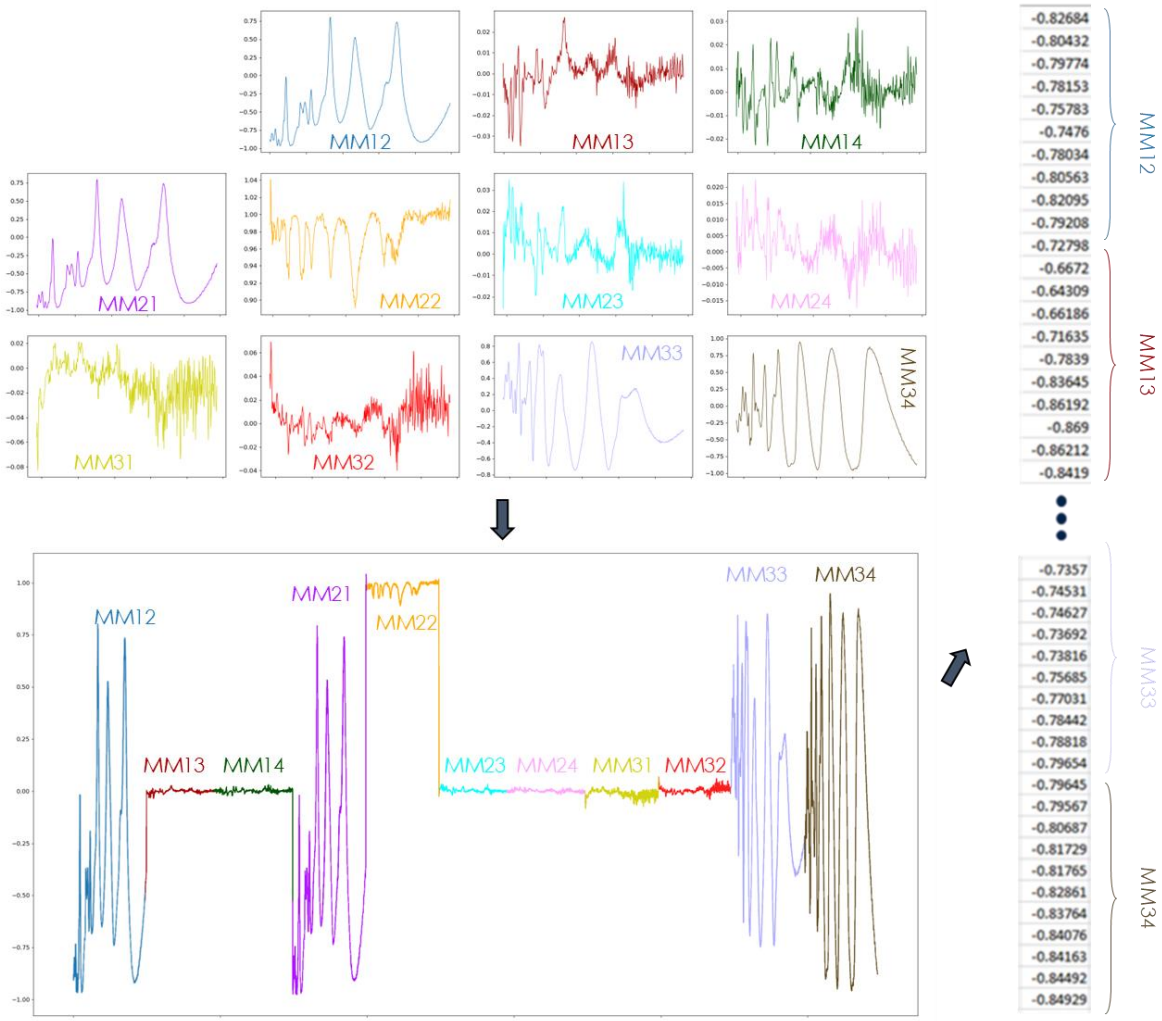

Figure 49 - Protocole de formatage des données de scatterométrie pour traitement par un FCNN, où les matrices de Mueller sont concaténées afin qu'uniquement les valeurs brutes soient conservées dans un vecteur.

3. Réseaux périodiques de lignes gravées dans le silicium (2D)

Présentation du cas d'intérêt

Les plaques possédant des lignes gravées périodiques (que l'on nommera « AGUA » - du nom du masque utilisé – pour simplification) sur lesquels nous avons évalué notre approche utilisant les réseaux de neurones sont en réalité basées sur des plaques étalons. Ces dernières sont régulièrement mesurées par les équipements de métrologie industrielle pour s'assurer de la stabilité et du *matching* (concordance des mesures entre équipements) de leurs mesures de CD et de *pitch* (période des structures). Elles ont été créées dans le cadre de l'introduction et la validation d'une nouvelle technique, et sont composées de multiples réseaux de lignes gravées dans le silicium.

Afin, d'induire une variation des CD sur la surface de la plaque, il est courant en lithographie d'avoir recourt à une matrice d'exposition (*Energy Matrix* – EM), où chaque colonne de champ lithographique est exposée différemment. Un exemple typique d'EM est présenté en Figure 50. Quatre plaques ont été produites suivant ce protocole, et on aura alors une variation horizontale des CD des lignes gravées (Figure 50), comprise entre 35 et 70 nanomètres.

Ces structures sont relativement simples, autant par leur composition (silicium pur), mais aussi par leur géométrie (ligne 2D), et sont alors aisément modélisables et traitables par une approche OCD

traditionnelle. L'objectif sur ce cas d'intérêt est de vérifier la correspondance entre les prédictions des CD du NN à partir des matrices de Mueller brutes, et celles obtenues par modélisation RCWA de la structure.

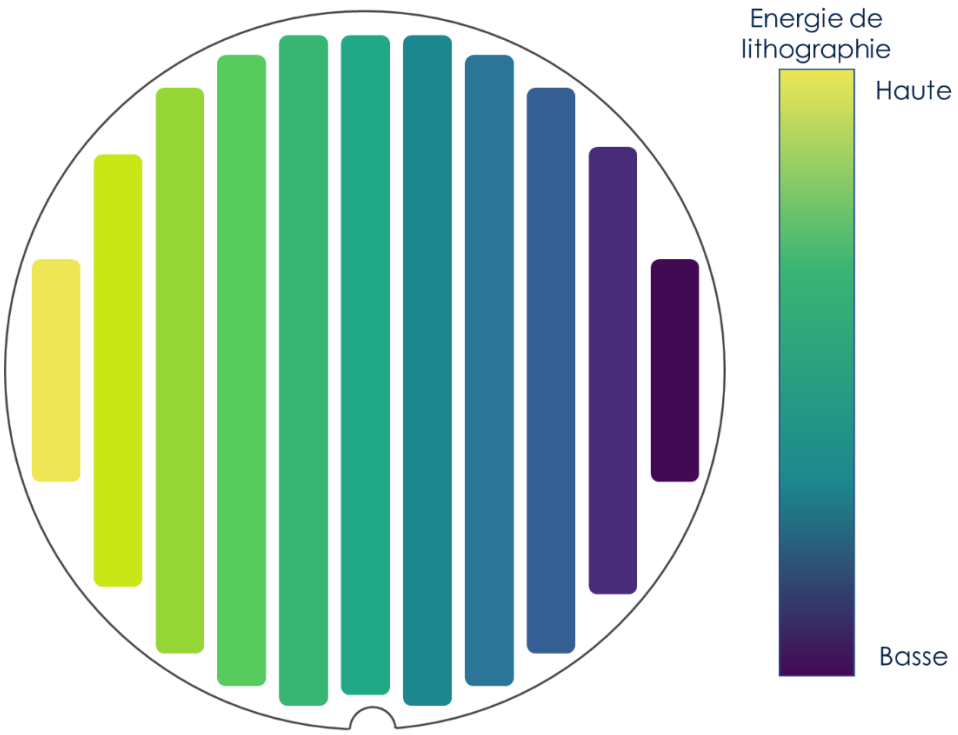

Figure 50 - Schéma représentatif du procédé d'energy matrix en lithographie, où chaque colonne de champ lithographique est exposée avec une énergie différente. Il est important de noter qu'une énergie haute entraîne des CD plus faibles.

Sur chacune de ces quatre plaques, on étudie 4 sites (Colonnes -4, -3, 0 et 5 – Ligne 0), où 25 mesures SEMCD ainsi que 25 mesures OCD seront réalisées. La répartition des mesures est représentée en Figure 51. De plus, des mesures de répétabilité, couramment utilisées par le département de métrologie afin de s'assurer de la stabilité des équipements, où le même site est mesuré plusieurs fois, la variation obtenue dans ces mesures indiquant l'erreur de mesure statistique induite par l'équipement. Pour cette étude, 30 mesures successives sur le même site furent réalisées sur les même 4 sites que précédemment, et cela pour tous les wafers crées.

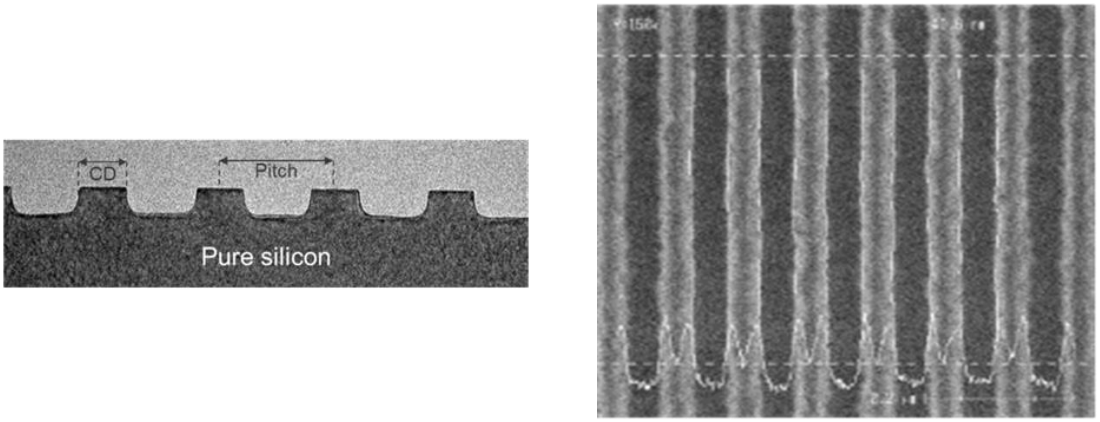

Figure 51 – Image SEM en coupe (gauche) et du dessus (droite) du réseau de lignes gravées présent sur les plaques étudiées.

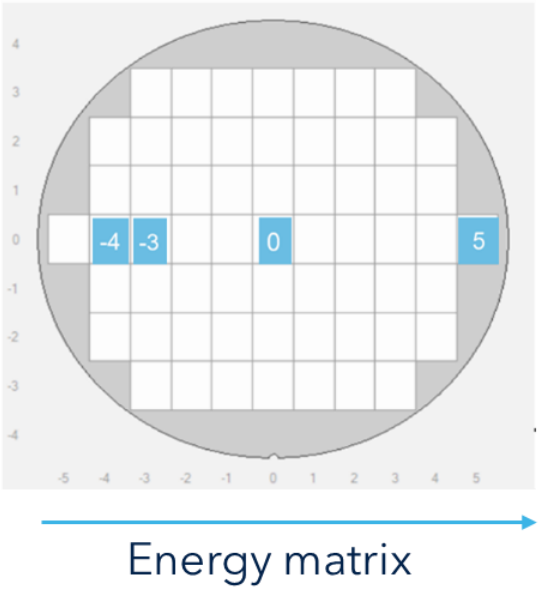

Figure 52 - Représentation des sites de mesures (25 points par site) pour les acquisitions de matrices de Mueller et ainsi que de SEMCD.

En parallèle de ces mesures, on modélise le réseau de lignes via RCWA afin d'en obtenir les dimensions, permettant la validation de l'approche NN. La structure élémentaire utilisée est présentée en Figure 53 et l'indice optique du silicium ainsi que les différentes dimensions (CD, profondeur, etc.) ont été fournies par une caractérisation préalable de STMicroelectronics. On peut alors résoudre le problème inverse pour ajuster les paramètres et obtenir la solution optimale. Ces valeurs seront à comparer à celles obtenues par l'approche IA.

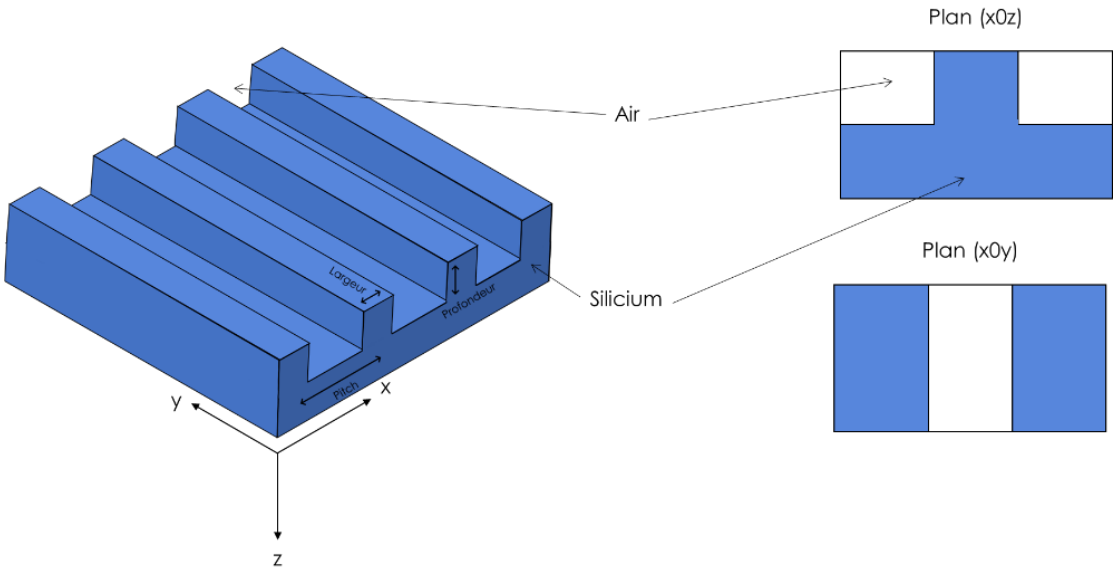

Figure 53 - Représentation de la structure modélisée par RCWA afin de valider l'approche reposant sur les réseaux de neurones.

Traitement des données par un réseau de neurone

À la suite des différentes mesures, nous avons alors à notre disposition 400 couples de données scatterométriques brutes et SEMCD. Nous les avons réparties dans les ensembles d'entraînement, de

validation et de test avec un ratio 70%/15%/15% (300/50/50). Le réseau de neurones spécifiquement crée a été entraîné en suivant le protocole d'arrêt d'entraînement lorsque le réseau ne s'améliore plus sur le *dataset* de validation, présenté dans le chapitre 1. La courbe d'erreur obtenue à la fin de cet entraînement est présentée en Figure 54. Comme nous développons un modèle de régression, seule l'erreur est utilisée pour évaluer l'état de l'entraînement du réseau, la précision n'étant pas applicable à ce type de problème. En effet, aucune prédiction n'est parfaitement juste, et la précision serait alors constamment nulle.

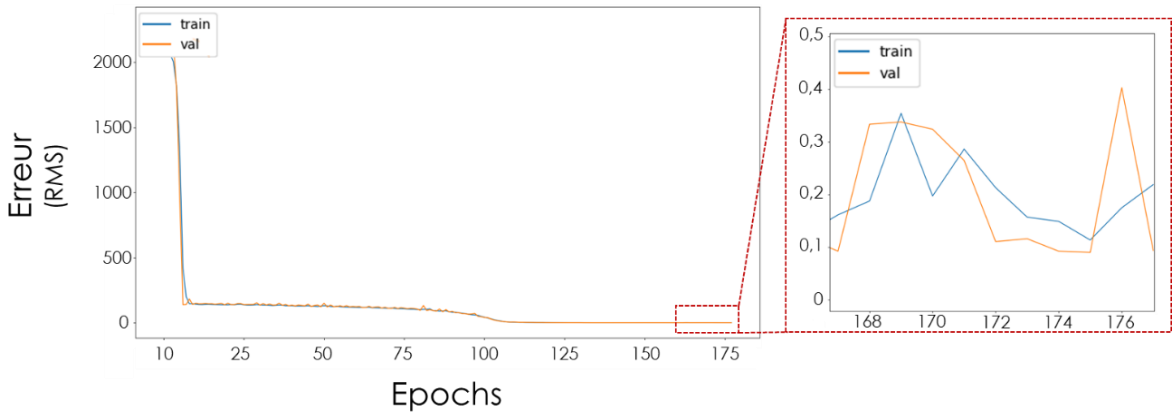

Figure 54 - Courbes d'entraînement du réseau de neurones à l'aide des données scatterométriques et SEM-CD récoltées sur les wafers AGUA, l'erreur étant représentée par l'erreur quadratique moyenne (RMS) obtenue à chaque epoch.

À la fin de l'entraînement, on a une erreur quadratique moyenne de 0,2 sur l'ensemble de données d'entraînement, et de 0,1 sur celui de validation. On utilise alors ce réseau entraîné sur le *dataset* de test, et les résultats obtenus sont présentés en Figure 55 et en Figure 56. On y remarque une très bonne corrélation entre les prédictions à partir des spectres bruts avec les mesures SEMCD, mais surtout que toutes les prédictions sont correctes à moins d'un nanomètre près par rapport aux SEMCD, et que la plupart sont contenues entre -0,2 et 0 nm.

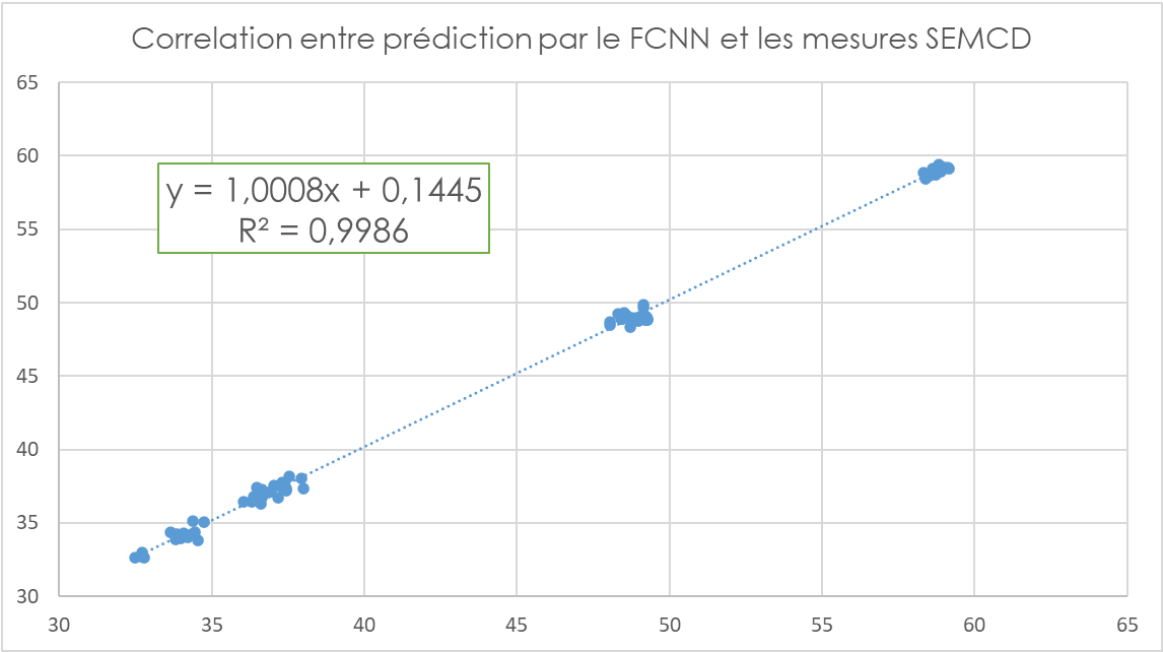

Figure 55 - Résultats des prédictions de CD à partir des spectres OCD, comparés aux CD mesurés en SEM.

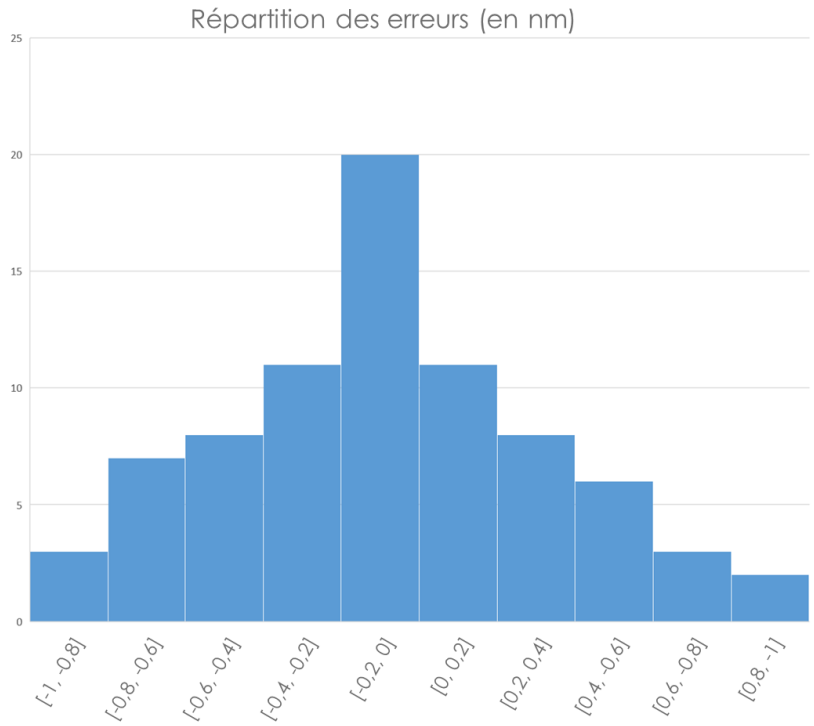

Figure 56 - Distribution des différences entre les prédictions réalisées par le réseau de neurones et les mesures des CD des lignes.

Évaluation des causes d'erreur

Afin d'évaluer la stabilité des prédictions du réseau de neurones, nous avons utilisé les mesures de répétabilité précédemment mentionnées. Les 30 matrices de Mueller obtenues pour chaque site sur tous les wafers étudiées (2 wafers supplémentaires étaient disponibles au moment de ces mesures) sont alors traitées par le réseau de neurones entraîné avec les précédentes données, et la variation des prédictions obtenues est présentée en Figure 57. On y remarque une très bonne stabilité, avec des écart-types inférieurs à 0,15 nm, même dans le cas des lignes les plus larges (~60nm).

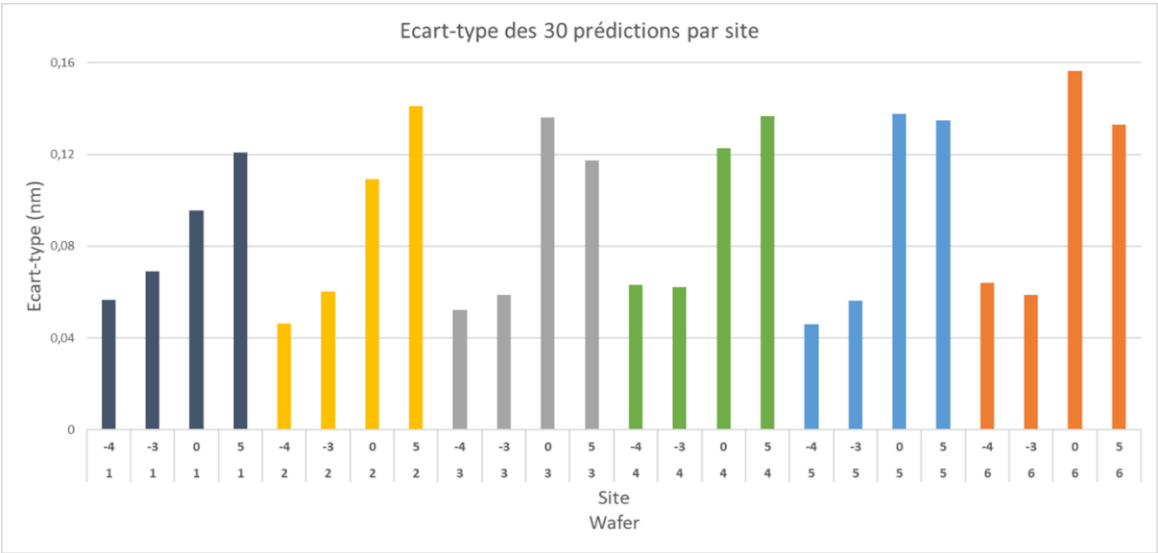

Figure 57 - Dispersion des prédictions réalisées sur les spectres mesurés sur chaque site des 6 wafers étudiés.

Comme les mesures répétabilité ont aussi été réalisé pour le SEMCD, on pourra alors évaluer la part de variabilité des prédictions due à l'erreur intrinsèque à cette mesure. En effet, on a observé que sur l'ensemble des mesures de répétabilité, le SEMCD induit en moyenne une erreur de 0.1 nm. La variabilité induite par le scatteromètre est plus complexe à évaluer. En effet, elle n'est pas directement quantifiable en nanomètres, car nous ne pouvons pas déterminer la relation directe entre une variation du spectre mesuré et une variation de CD prédit par le réseau de neurone, par nature. Mais la mesure de scatterométrie étant réputée comme très stable, on peut considérer que toute erreur induise par une variation de spectre sera négligeable, et ainsi supposer que l'erreur globale obtenue lors de la prédiction n'est que très peu dépendante de la mesure de scatterométrie. Alors, si l'on retire la participation de la mesure SEMCD de l'erreur, on peut déterminer dans ce cas d'intérêt que le réseau de neurone est responsable en moyenne d'une erreur d'environ 0,5 nm.

Conclusion

Suite à l'étude de ce premier cas d'intérêt simple, on peut voir qu'une approche IA de prédiction de CD de lignes gravées, basée sur le traitement *model-less* de spectres de scatterométrie combinés à des mesures de référence SEMCD, pourrait réaliser un suivi de procédé rapide, précis et répétable. Cependant, pour ce type de structure simple, une approche modélisée via RCWA pourrait paraitre plus adaptée, étant donné que la géométrie étudiée est relativement basique, permettant donc une simulation rapide de la réponse optique. Alors, nous souhaitions évaluer l'apport de cette approche pour des structures plus complexes, difficilement modélisables en un temps raisonnable.

4. Réseaux de trous périodiques gravés (3D)

Présentation du cas d'intérêt

Ces structures périodiques à 3 dimensions sont utilisées sur des plaques pour les technologies « ODIFF » pour ***O**ptical **DIFF**users* (diffuseurs optiques), qui ont pour but de diffuser un faisceau optique incident afin de repartir uniformément l'intensité lumineuse (Figure 58). Cette technologie a été développée dans le cadre de la reconnaissance faciale des smartphones, où une source laser infrarouge est utilisée pour imager les visages. Cette source doit donc logiquement être diffusée pour couvrir l'entièreté du visage, mais aussi pour ne pas endommager l'œil de l'utilisateur.

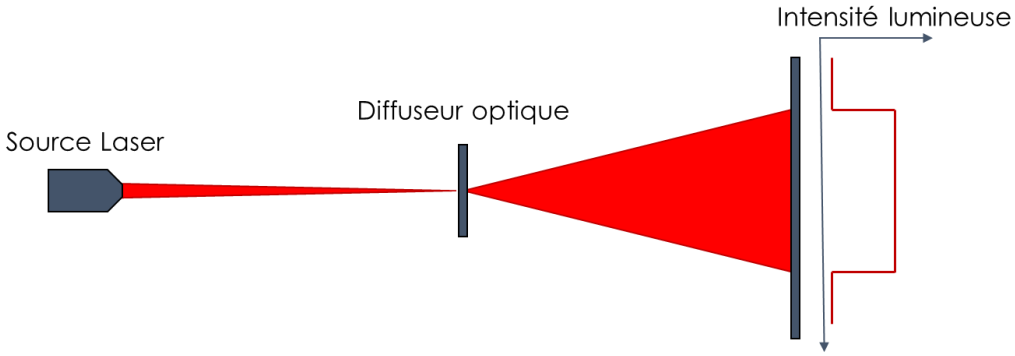

Figure 58 - Principe d'un diffuseur optique, où la lumière est uniformément répartie sur la surface cible.

Cette diffusion est ici réalisée à l'aide d'un réseau apériodique de trous. Le résumé simplifié de la production de ce type de structures est présenté en Figure 59. On y remarque que ces trous sont gravés dans un empilement de couches de nitrures et oxydes de silicium, puis remplis de Silicium amorphe qui devient, après recuit, polycristallin. Ces structures permettent alors d'induire une variation d'indice optique contrôlée, et donc de la diffraction. La diffusion basée sur la diffraction permet d'obtenir une haute transmittance, une faible déviation angulaire, ainsi qu'une faible aberration chromatique [10]. Afin d'obtenir la diffusion souhaitée, il faut alors modéliser optiquement (via analyse itérative de type IFTA - *Iterative Fourier Transform Analysis*) la diffraction de l'ensemble

de ces structures apériodiques. La propriété apériodique de ces structures permet d'en casser la symétrie, et ainsi de pouvoir obtenir une diffusion uniforme. Un exemple de ce type de structure est présenté en Figure 59. Ces structures possèdent alors des CD allant de 120 à 350 nm de diamètre, avec un pitch constant de 470nm.

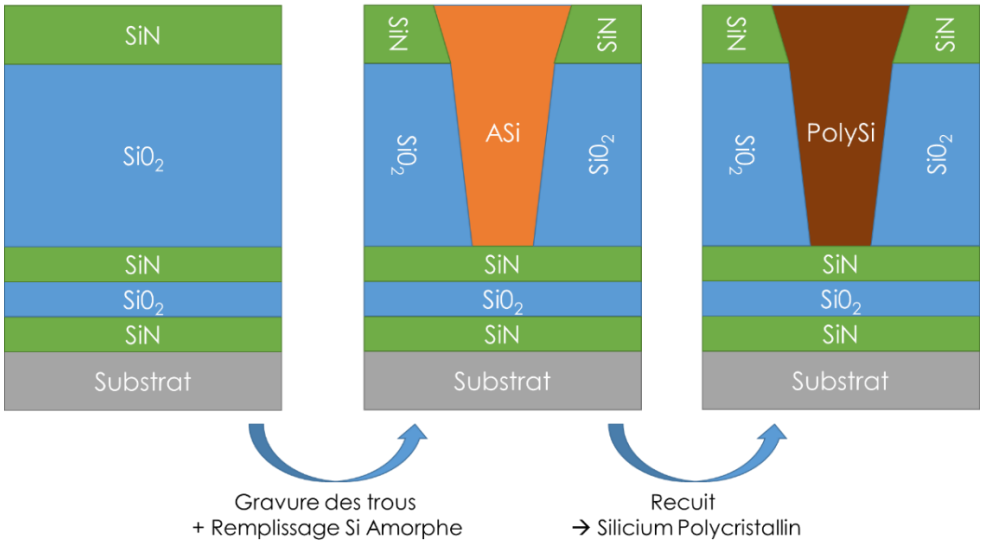

Figure 59 - Process simplifié de réalisation des structures de réseaux de trous pour les plaques ODIFF

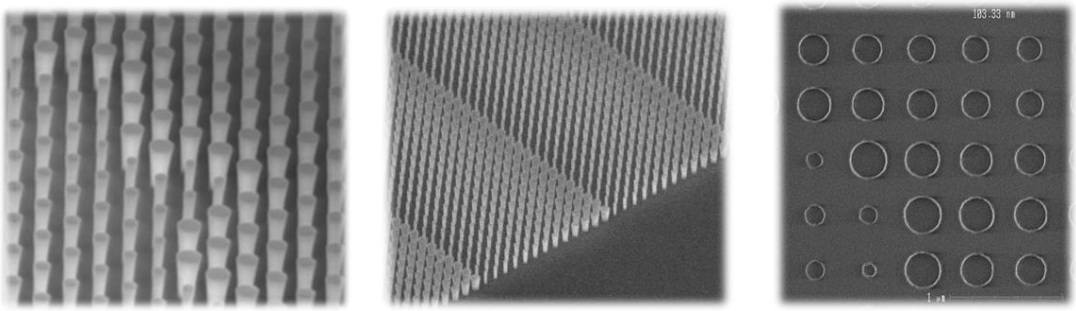

Figure 60 - Images SEM montrant les structures de trous présents sur les wafers ODIFF, permettant à terme la diffusion homogène de la lumière incidente.

Afin d'assurer le suivi de *process* de gravure de ces dispositifs, quatre structures de métrologie pour la SEM-CD ont été placées dans chaque champ lithographique. Elles ont pour but d'être représentatives du produit, et sont composées de réseaux, périodiques cette fois, de trous à CD fixes (120, 218, 290 et 350 nm – Figure 61). Ce suivi pourrait être réalisé par scatterométrie, mais comme indiqué dans l'introduction, la modélisation de structures à 3 dimensions est extrêmement lourde, et la génération d'une bibliothèque de spectres sur une aussi large gamme de CD n'est pas envisageable dans un temps raisonnable avec les algorithmes couramment utilisés chez STMicroelectronics.

Afin d'évaluer l'approche de scatterométrie *model-less* combinée à la SEMCD sur ce cas d'intérêt, quatre plaques ont été spécifiquement créées. L'étape critique à suivre pour ce produit est la gravure des trous, et ces plaques ont donc été étudiées après cette dernière. Pour valider la sensibilité de notre approche à des dérives de CD, nous avons encore une fois eu recourt à une *Energy* Matrix. De plus, afin d'obtenir le plus de données possibles pour notre réseau de neurone, toutes les mires présentes sur le wafer (4 cibles x 116 champs = 464 mesures par wafer) furent mesurées par le scatteromètre et le SEM-CD. Au vu de la complexité de la structure et afin de maximiser l'information récoltée par le scatteromètre, ces sites sont mesurés à deux azimuts (23° et 45°). On obtient alors

deux matrices de Mueller pour chaque mire, qui seront combinées pour être traitées par le réseau de neurones.

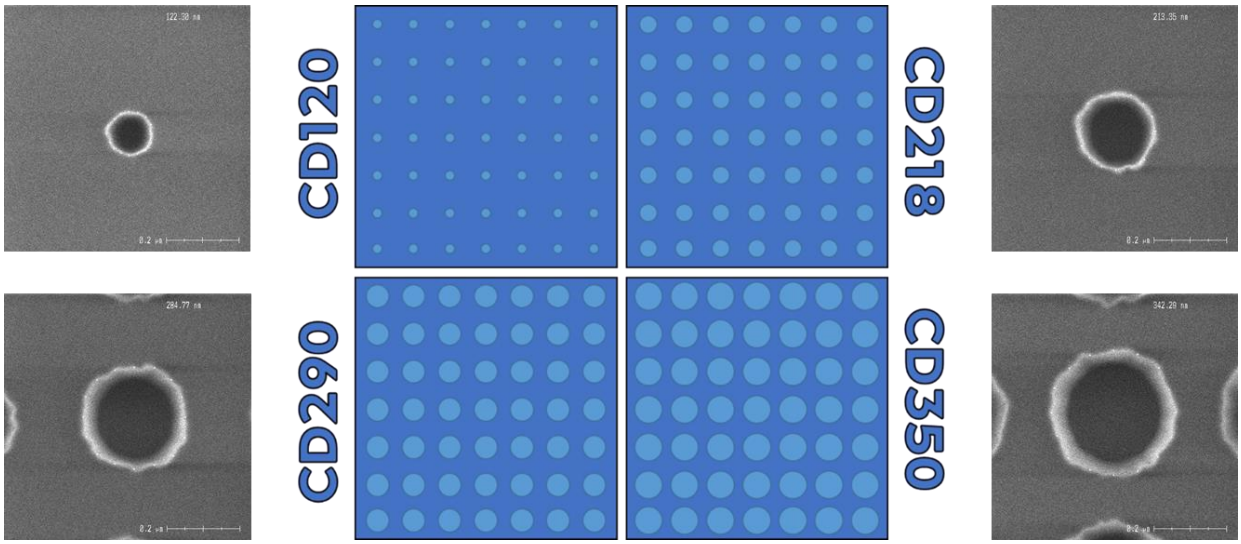

Figure 61 - Mires de métrologies dédiées aux mesures SEMCD, constituées de réseaux de trous périodiques de CD fixes (120, 218, 290 et 350 nm) et de pitch fixe (470 nm), avec exemple d'image SEM obtenues dans chacune de ces mires.

Traitement des données par un réseau de neurone

De la même façon que présenté dans la partie précédente, les spectres contenus dans les matrices de Mueller obtenues à 23° et 45° sont concaténées pour être réduits à une dimension. Une architecture de FCNN similaire à celle utilisée pour les plaques AGUA est utilisée, et entraînée avec une répartition 70%/15%/15% dans les *datasets* d'entraînement, de validation et de test. Les dimensions de toutes les couches ont dû être doublées pour pouvoir traiter les données générées dans ce cas d'intérêt. L'entraînement converge alors en un peu plus de 100 epochs (Figure 62) à une erreur quadratique inférieur à 5.

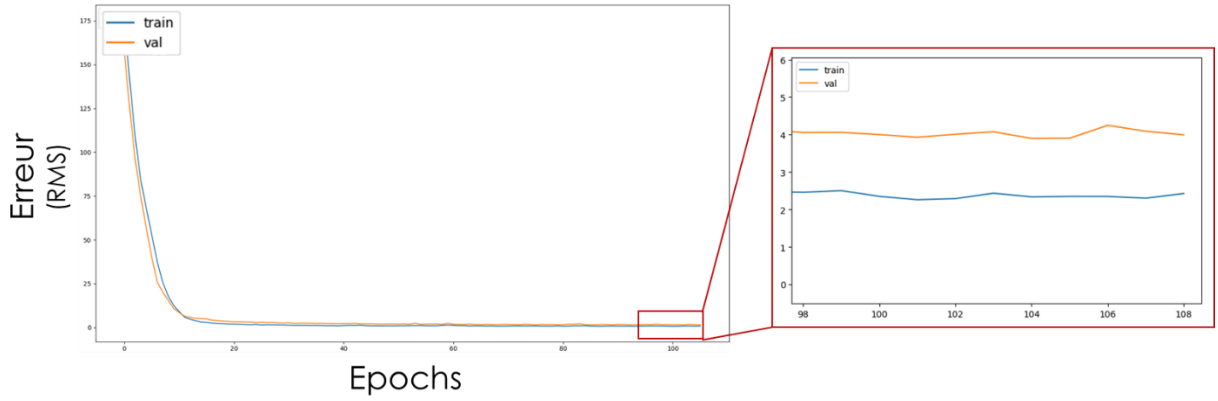

Figure 62 - Courbe d'entraînement du réseau de neurones sur les données récoltées sur les wafers ODIFF.

En utilisant le dataset de test, ce réseau entraîné nous permet de prédire les CD des structures à partir de leurs spectres de scatterométrie, et en vérifier le bon entraînement en les comparant aux CD mesurés par SEM-CD (Figures 63 et 64). On remarque alors une très bonne qualité de prédiction, autant par les valeurs de R^2 très élevées (> 0.98) pour chaque jeu de données (un par structure de métrologie), mais aussi par la répartition des erreurs, très largement distribuées entre -0.5 et 0.8 nanomètres.

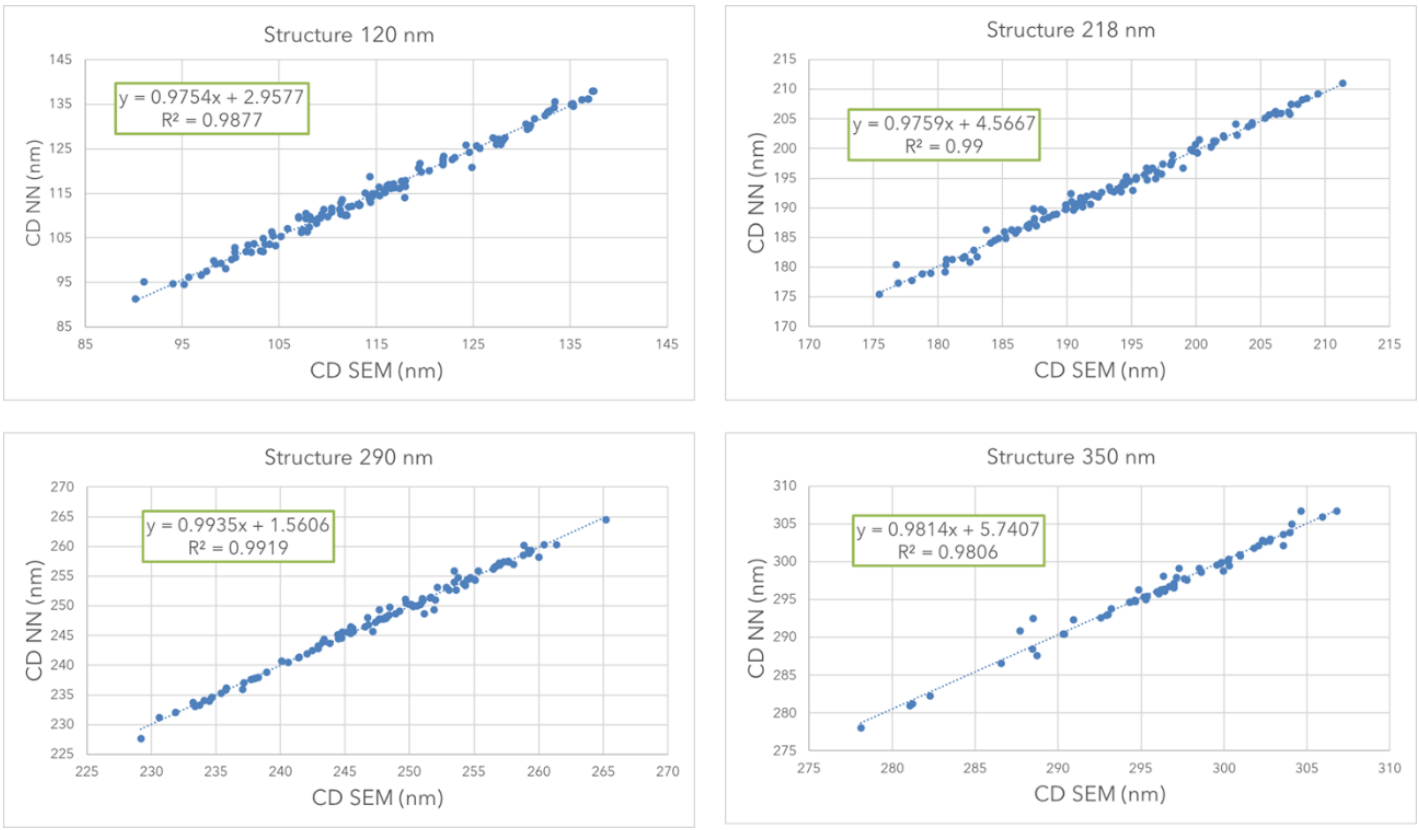

Figure 63 - Résultats des prédictions de CD à partir des spectres OCD, comparés aux CD mesurés en SEM, pour chaque mire de métrologie.

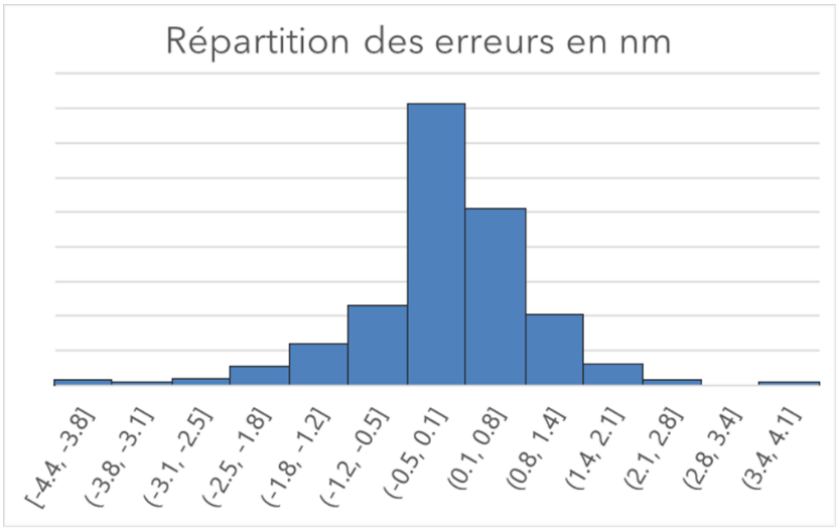

Figure 64- Histogramme des erreurs obtenus par prédiction des CD par le réseau de neurones en utilisant les spectres OCD.

Pour évaluer la robustesse de notre modèle, des mesures de répétabilités (SEM-CD et Scatterométrie) ont été réalisées sur des wafers de production (sans *energy matrix*), permettant alors de vérifier la bonne capacité de généralisation de l'algorithme, mais surtout de déterminer la propagation des erreurs induites par chaque technique (mesures de métrologie et réseau de neurones). À partir des mesures de répétabilité de SEM-CD (10 mesures par structure), on détermine qu'elle introduit une erreur moyenne de 0.37 nm. Comme pour le cas d'intérêt précèdent, on

considère que l'erreur induite par la répétabilité de mesure du scatteromètre est négligeable devant celle du SEMCD et du réseau neuronal.

Évaluation des causes d'erreur

Pour évaluer cette dernière, on utilise alors les 10 acquisitions réalisées pour chaque site pour mesurer la variance des prédictions. Comme on pouvait s'y attendre, au vu de la stabilité des spectres fournis par le scatteromètre, les prédictions réalisées sur les mêmes sites sont très peu variantes (Figure 65), avec en moyenne une erreur induite de 0.12 nm.

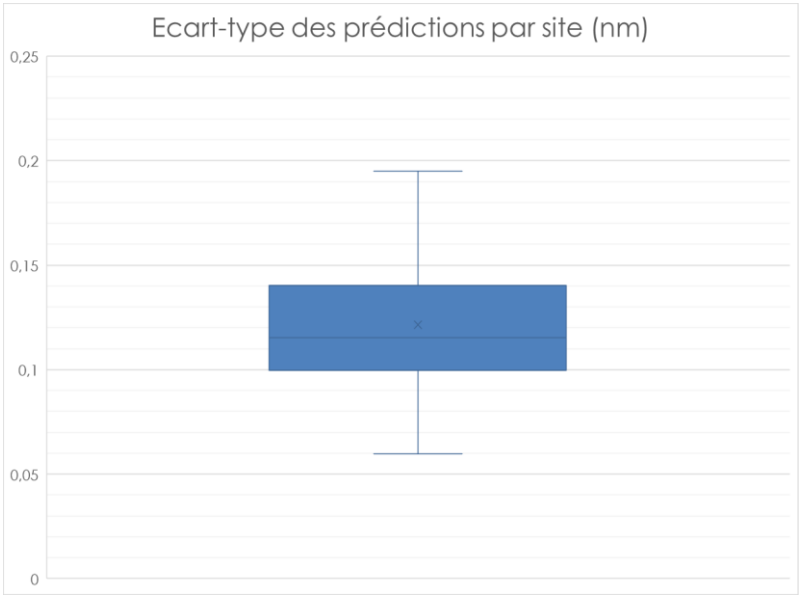

Figure 65 - Distribution des écarts-types des prédictions de CD réalisées par le réseau de neurones à partir des mesures de répetabilité de scatterométrie.

Validation de l'approche par utilisation de données synthétiques

Enfin, pour valider notre approche NN, il nous semblait important de vérifier que des données générées par RCWA, lorsqu'elles sont traitées par le réseau neuronal, convergent bien vers les dimensions définies lors de la simulation. Pour ce faire, un modèle rigoureux de la structure de réseau de trous périodiques a été généré, et 50 matrices de Mueller caractéristiques de cette dernière ont été créées, en utilisant des CD variant de 110 à 380 nanomètres. Ces dimensions nous permettent alors aussi de valider la capacité de généralisation de notre algorithme sur des plages de CD non-utilisées lors de l'entraînement.

Les résultats de cette étude sont présentés en Figure 66. On y remarque que pour la plupart des cas, l'erreur de prédiction est contenue entre -5 et 2 nanomètres. Ces résultats sont explicables par le fait que par simulation, certaines variations fines des spectres scatterométriques sont lissées (Figure 67).

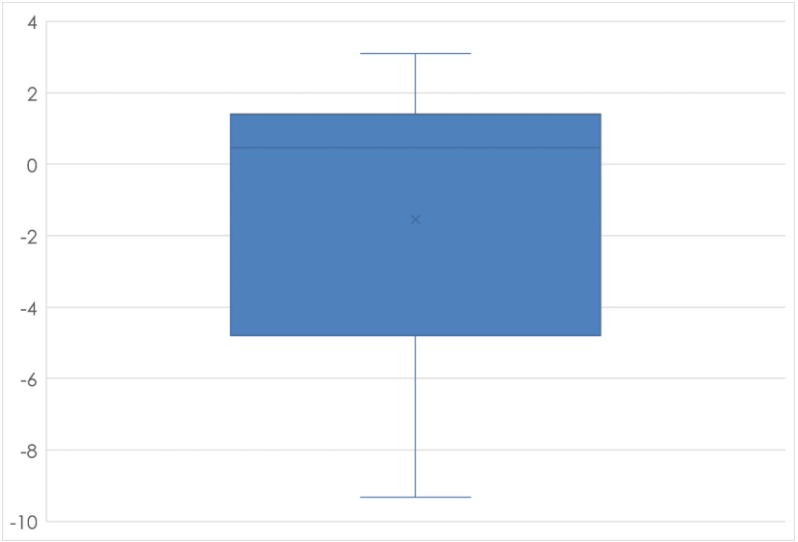

Figure 66 - Distribution des erreurs de prédictions (en nanomètres) de CD à partir des spectres modélisés par RCWA.

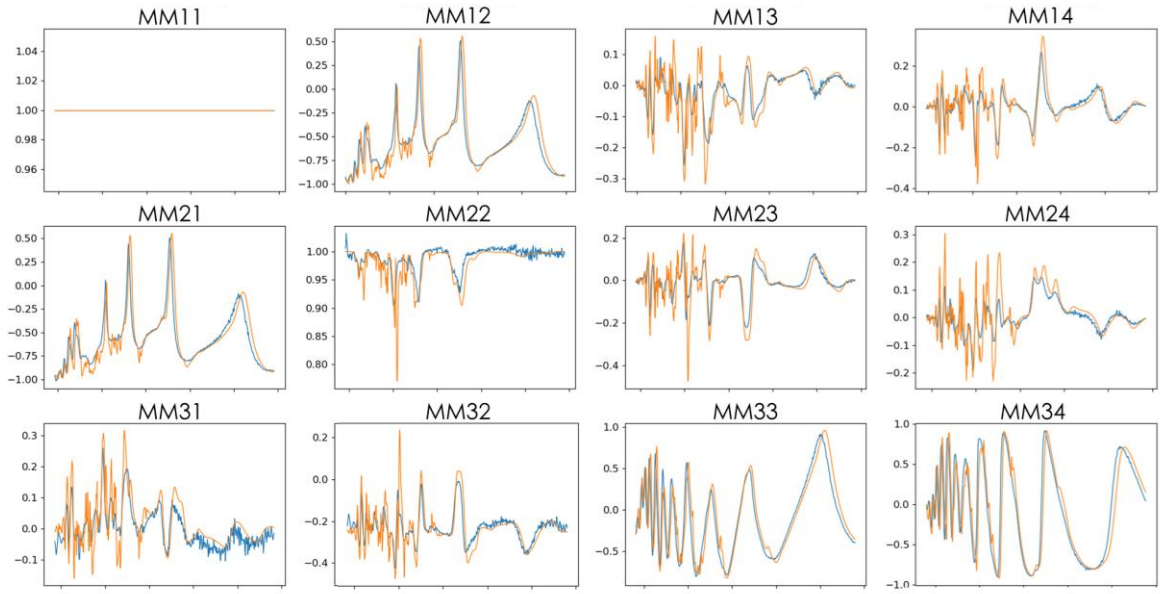

Figure 67 - Exemple de matrices de Mueller obtenues expérimentalement (bleu) et par simulation RCWA (orange) montrant une très bonne corrélation.

L'utilisation de données synthétiques permet, de plus, de fournir des données supplémentaires d'entraînement pour le réseau de neurones. Durant ma thèse, j'ai collaboré avec un post-doctorant du LTM, dans le cadre du projet MADEin4, afin de prédire l'intégralité des dimensions (CD, Pitch, Hauteur, etc.) d'une structure périodique à l'aide de spectres de scatterométrie traités par un réseau de neurones. Dans cette approche, seules des données synthétiques, donc générées par modélisation RCWA, sont utilisées pour entraîner le réseau de neurones [11].

Conclusion

Pour des structures plus complexes (3D), l'approche de scatterométrie model-less utilisant un traitement par réseau de neurones montre des résultats convainquant avec des erreurs faibles (inférieures en moyenne au SEM-CD), et capable de traiter des données synthétiques (spectres générés par RCWA). Il est important de rappeler que la modélisation de ce type de structures est très lourde en temps de calcul, et n'est utilisable que dans le cas d'un procédé de fabrication figé. Notre

approche serait alors adaptée pour le suivi de CD de structure périodiques complexes de lots R&D au *process* rapidement variable.

Dans cette étude, nous avons étudié l'intérêt de notre approche sur des structures de métrologie entièrement périodiques, alors que le dispositif présent sur la plaque repose sur des réseaux apériodiques. Il serait donc intéressant de vérifier la capacité de notre approche pour des mesures réalisées sur ce type de structure.

V. Conclusion

Dans ce chapitre, nous avons montré que l'utilisation de réseaux de neurones pour traiter des données de scatterométrie brutes permet de réaliser un suivi de procédé versatile, précis et répétable, bien que moins précis qu'une approche modélisée. Elle est alors plus adaptée dans le cas où la modélisation n'est pas envisageable, notamment pour les structures complexes, surtout dans le cas de variations de procédés itératives induites par la R&D. L'analyse SWOT (*Strength, Weakness, Opportunity, Threat*) présenté dans le Tableau 5 permet de visualiser l'apport de notre approche métrospection par rapport à une approche standard utilisée en métrologie industrielle, tout en étant conscient de ses limitations.

Tableau 5 - Analyse SWOT de l'approche de scatterométrie model-less comparée à l'approche modélisé par RCWA

Force	Faiblesse
• Capacité de généralisation • Simple/Rapide à mettre en place	• Erreur supérieur à l'approche modélisée
Opportunité	**Risque**
• Utilisable en R&D • Baisse du coût en temps des ingénieurs spécialisés	• Dérive du modèle avec différents équipements

Références

[1] F. Anis, R. Gronheid, D. D. Van den Heuvel et F. Zach, «Identifying contributors to overlay variability using a model-less analysis method,» *Proceedings SPIE 11611, Metrology, Inspection, and Process Control for Semiconductor Manufacturing XXXV,* p. 116111D, 2021.

[2] M. Littau, C. Raymond, C. Gould et C. Gambill, «Novel implementations of scatterometry for lithography process control,» *Proceedings SPIE 4689, Metrology, Inspection, and Process Control for Microlithography XVI,* 2022.

[3] P. Reinig, R. Dost, M. Moert, T. Hingst, U. Mantz, J. Moffitt, S. Shakya, C. Raymond et M. Littau, «Metrology of deep trench etched memory structures using 3D scatterometry,,» *Proc. SPIE 5752, Metrology, Inspection, and Process Control for Microlithography XIX,* 2005.

[4] P. Durgapal, J. R. Ehrstein et N. V. Nguyen, «Thin film ellipsometry metrology,» *American Institute of Physics Conference Proceedings,* vol. 449, n° %11, pp. 121-131, 1998.

[5] F. P. J. Bernoux, B. Castellon, C. Defranoux, J. Lecat, P. Boher et J. Stehlé, «Ellipsométrie théorie,» chez *Techniques de l'ingénieur. Mesures et contrôle,*, 2003, p. R6490.

[6] R. Redheffer, «Inequalities for a Matrix Riccati Equation,» *Journal of Mathematics and Mechanics,* n° %18 (3), p. 349–367, 1959.

[7] R. Rumpf, «Improved formulation of scattering matrices for semi-analytical methods that is consistent with convention,» *Progress In Electromagnetics Research B,* n° %135, pp. 241-261, 2011.

[8] T. Germer, H. Patrick, R. Silver et B. Bunday, «Developing an uncertainty analysis for optical scatterometry,» *Proc. SPIE 7272, Metrology, Inspection, and Process Control for Microlithography XXIII,* vol. 72720T, 2009.

[9] M. Abadi, A. Agarwal et P. Barham, «TensorFlow, Large-scale machine learning on heterogeneous systems,» Zenodo, 2015.

[10] K. Yamashita, K. Kunitsu, T. Hattori, Y. Kuwahara et A. Saito, «Demonstration of a diffraction-based optical diffuser inspired by the Morpho butterfly,» *Opt. Express 29,* pp. 30927-30936, 2021.

[11] H.-L. Pham, T. Alcaire, S. Soulan, D. Le Cunff et J.-H. Tortai, «Efficient Rigorous Coupled-Wave Analysis Simulation of Mueller Matrix Ellipsometry of Three-Dimensional Multilayer Nanostructures,» *Nanomaterials,* n° %112, 2022.

Chapitre III –
Traitement avancé d'images

Un très grand nombre d'équipements industriels de suivi de *process* génèrent lors de leurs utilisations des images, soit de la région étudiée, soit de l'entière surface du wafer. Or, en général, toute l'information contenue dans ces images n'est pas utilisée. En effet, la plupart du temps, seules des données statistiques (moyenne, écart-type, etc.) extraites de ces images sont employées pour évaluer l'état du *process*. Dans d'autres cas, ces images sont traitées à l'aide d'algorithmes mathématiques spécifiques au produit et à l'étape étudiée, et donc rend ces traitements d'images peu versatiles. De plus, ce type d'approche s'avère extrêmement sensible à des variations d'intensité lumineuse et de contraste lors de l'acquisition. Il est alors d'intérêt d'évaluer des traitements avancés d'images permettant de réduire ces contraintes, en s'appuyant notamment sur l'utilisation de réseaux de neurones convolutifs. Dans le cadre de nos approche métrospection, ces outils seront évalués sur divers cas d'intérêts industriels, et comparées à ceux actuellement utilisées en production.

I. Introduction

Avant de décrire le traitement de ces images, il est nécessaire de comprendre comment sont organisées leurs données, et ce à quoi elles correspondent. Selon que l'image soit en niveaux de gris, ou en couleurs, ces données peuvent être représentées respectivement par des fonctions à 3 ou 4 dimensions. En effet, pour une image en niveaux de gris, à chaque point de l'image (pixel) est associée une coordonnée (x, y), ainsi que la valeur de ce pixel, usuellement comprise entre 0 et 255 (8-bits de données). Dans le cas d'une image en couleur, cette valeur est alors une combinaison des valeurs décomposées sur les 3 canaux : Rouge, Vert et Bleu (RGB en anglais). Une image est alors composée d'une superposition de 3 matrices de même dimension, définie par le nombre de pixels (Figure 68). Pour obtenir la valeur finale du pixel la plus fidèle à celle restituée par l'œil humain, l'équation suivante est couramment utilisée : $V = R \times 0.2126 + G \times 0.7152 + B \times 0.0722$.

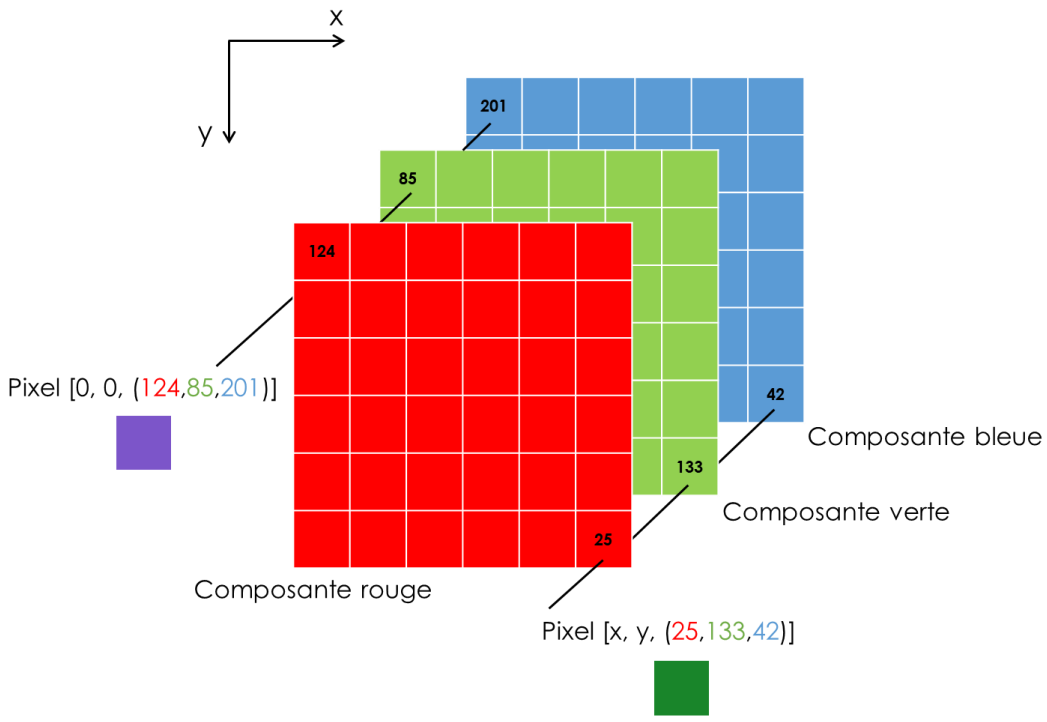

Figure 68 - Illustration matricielle d'une image RGB

Différentes opérations mathématiques peuvent alors être appliquées à ces matrices, permettant la mise en évidence de contours, textures et formes dans les images. Dans une première partie, nous nous intéresserons à l'application et l'optimisation de ce type de fonction pour la détection de dérives sur des images. Puis nous verrons dans des cas plus complexes, que l'utilisation de réseaux de neurones peut être nécessaire pour la classification automatique d'images, ou d'objets présents dans une image.

II. Analyse avancée d'images

Parmi les signatures de dérives couramment présentes sur les images collectées en microélectronique, nombreuses sont celles caractérisées par des lignes (rayures, rupture du réseau cristallin, etc.) et les équipements utilisés dans le suivi de procédé de ces dérives sont rarement capables de détecter correctement ces dernières. Il est alors d'intérêt d'évaluer l'apport de traitements avancés d'images capables de détecter ce type de dérive.

1. Algorithme de détection de ligne : Transformée de Hough

Notre objectif est ainsi la mise en place d'un traitement reposant sur la transformée de Hough [1], régulièrement utilisée comme descripteur pour l'analyse d'image ou la vision par ordinateur. Elle permet la détection de différentes formes (lignes, cercles, etc.) dans des images, et nous l'avons donc uniquement utilisée afin de détecter des lignes dans le cadre de cette étude.

En pratique, cet algorithme fonctionne en proposant un large nombre de lignes, caractérisées par leurs paramètres polaires (Figure 69a) : ρ (distance à l'origine du repère) et θ (angle entre la droite perpendiculaire à la droite passant par l'origine, et l'axe x). Les points (x, y) appartenant à cette droite obéissent alors à l'équation : $\rho = x\,cos(\theta) + y\,sin(\theta)$.

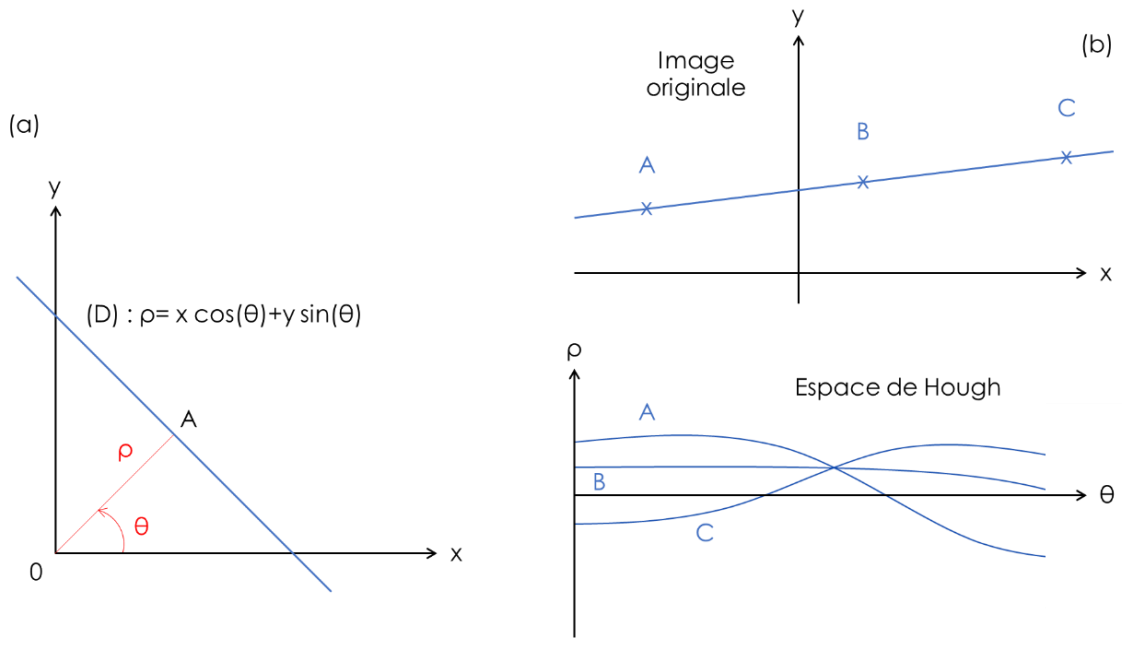

Figure 69 - (a) Équation d'une droite passant par un point A de coordonnées (x, y) dans un repère polaire et (b) représentation d'une droite passant par 3 points dans une image, et dans l'espace de Hough.

Comme la forme recherchée est une ligne, on peut alors uniquement considérer les pixels appartenant aux « contours » présents sur l'image. Ils sont caractérisés par un fort gradient local, et peuvent être extraits à l'aide d'un filtre de Canny [2] (Figure 70).

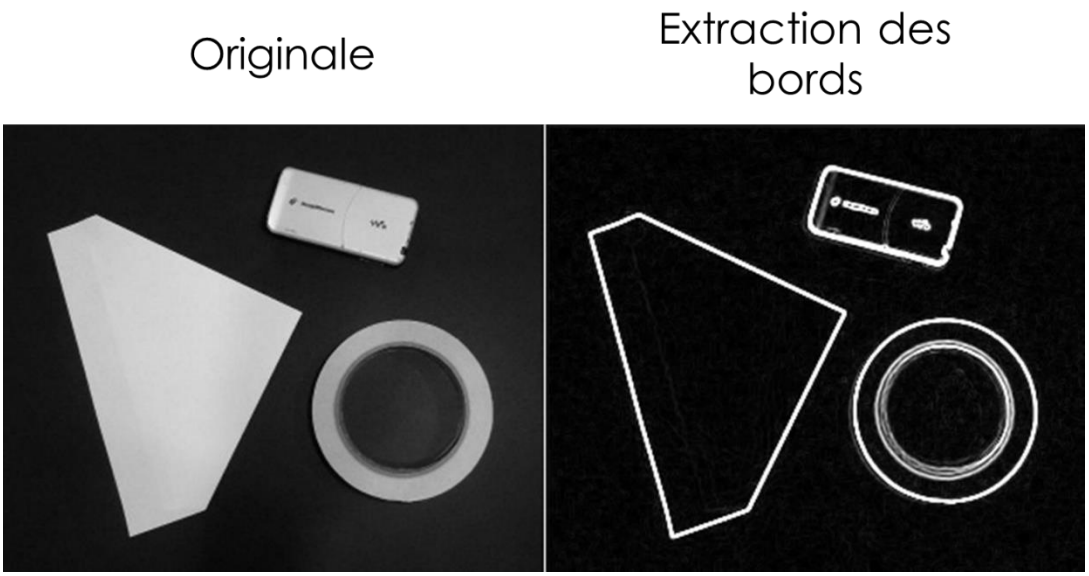

Figure 70 - Exemple d'extraction de contours réalisée avec le filtre de Canny.

Pour chacun de ces pixels restants, on détermine l'ensemble des droites passant par ce point (définies par leur ρ et θ spécifiques). À chaque point correspond alors une courbe $\rho = f(\theta)$, où θ ϵ [0, 2π] et $f = x\,cos(\theta) + y\,sin(\theta)$. L'ensemble de ces courbes sinusoïdales, générées à partir des points appartenant aux contours est représenté dans l'espace de Hough (ρ, θ) (Figure 69b). Les points d'intersections de ces courbes caractérisent donc les droites passant par les points d'où émergent ces courbes. Chaque point de l'espace de Hough est ainsi associé à une densité, correspondant à la quantité de courbes s'y coupant. Plus cette densité est forte, plus la droite définie par sa position dans le plan (ρ, θ) est probablement présente dans l'image. Aucune hypothèse de continuité n'est imposée pour la détection des lignes, ce qui rend ce traitement robuste, même si des points sur la ligne sont manquants (bruits dans l'image, masquage etc.).

Afin d'utiliser cet algorithme, il faut alors définir les variables suivantes :

- Discrétisation de θ : Nombres d'angles considérés lors de la création des courbes
- Résolution de ρ : Distance en pixel minimale entre deux points de l'image pouvant former une ligne
- Seuil de vote : Nombre d'intersections minimales dans l'espace de Hough pour que la droite ainsi définie soit retenue
- Dimensions limites des lignes : Taille minimale et intervalle maximale nécessaire pour combler les lignes à retenir pour la détection.

Un résumé du fonctionnement de l'algorithme de la transformée de Hough est présenté en Figure 71. On y observe bien que l'absence de point dans la droite n'empêche pas sa détection, bien que cela entraîne l'absence de courbes dans l'espace de Hough (bande noire).

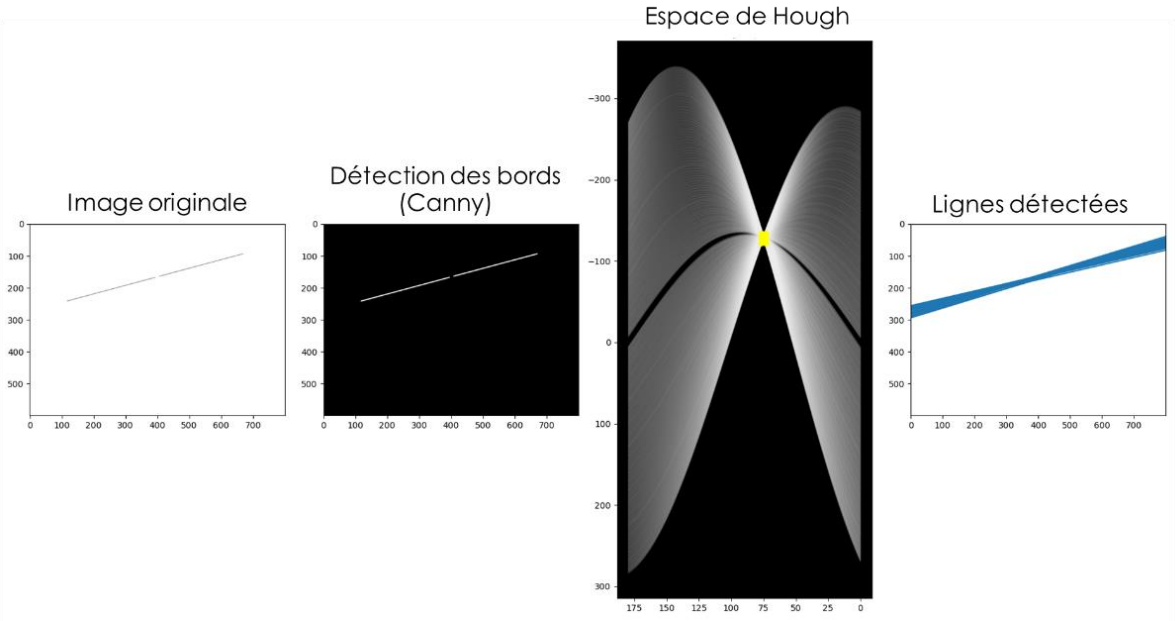

Figure 71 - Procédé de l'utilisation de l'algorithme de la transformée de Hough pour la détection de lignes.

Les caractéristiques de chacune des lignes détectées sont alors extractibles par l'algorithme. Afin d'optimiser les performances de ce traitement, les données d'entrées doivent être correctement préparées. En effet, comme les images full wafer fournies par les différentes techniques sont couramment de haute résolution (> 30M de pixels), le traitement par la transformée de Hough des données brutes serait très long (plusieurs heures de calculs), malgré la détection des contours réduisant en partie la quantité d'information présente dans l'image. De plus, ces images sont souvent bruitées, ce qui augmente aussi considérablement le temps de traitement par l'algorithme. Un protocole de prétraitement des données a été créé afin de rendre raisonnable l'utilisation de la transformée de Hough.

Ce protocole est alors une suite d'opérations réalisées sur les images à l'aide du logiciel « ImageJ » [3], référence en la matière dans le monde de la recherche, spécifiquement en biologie, pour sa facilité d'utilisation, son langage macro, et ses nombreux plug-ins. Comme les différentes techniques de suivi de procédé industrielles ont tendance à générer des images relativement différentes (présence ou non de design, sensibilité dans le bulk, résolution, etc.), ce traitement devra être optimisé pour chacune d'elles. Les différents prétraitements étaient fixes, mais les paramètres d'entrée de chacun pouvaient varier. Le protocole (Figure 72) est alors le suivant :

1. *Total variation denoising* [4] : Débruitage de l'image par minimisation de la variation totale, permettant une réduction du bruit, tout en conservant les contours.
2. *Sharpen* : Augmentation des contrastes et accentuation des détails. Peut augmenter le bruit, donc utilisé après le débruitage.
3. *Find edges* : Détection des contours dans l'image. Ce traitement est aussi réalisé lors du traitement de l'algorithme de Hough, mais celui réalisé par ImageJ est plus performant (Filtre de Sobel).

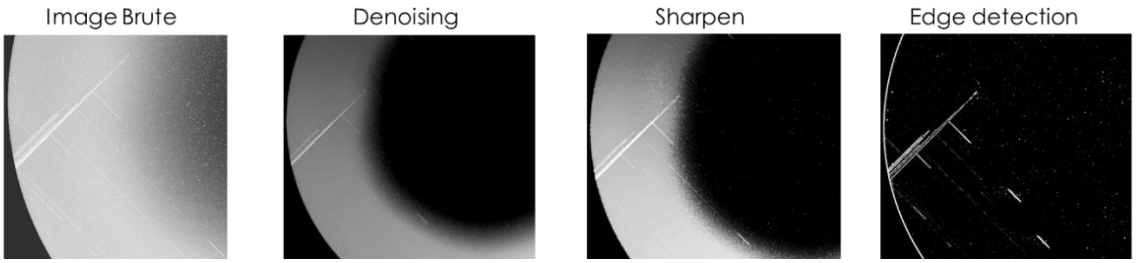

Figure 72 - Illustration du protocole de prétraitement des images avec ImageJ pour l'algorithme de transformée de Hough (région réduite pour meilleure visibilité des effets de chaque étape)

À partir des images obtenues par les équipements, on peut réaliser la détection des lignes grâce à l'algorithme de transformée de Hough, après optimisation des variables d'entrée de ce dernier (Discrétisation de θ, Résolution de ρ, Seuil de vote et Dimension limites des lignes). Cette approche permettrait alors d'obtenir des données fondamentales au suivi de dérives de procédés telles que les dimensions et le nombre de lignes présentes, impossible à obtenir précisément avec le traitement en place à ce jour.

2. Évaluation sur le cas d'intérêt des slip lines

Présentation du cas d'intérêt

Nous avons alors évalué l'intérêt de notre approche sur le cas des « slip lines », qui sont des ruptures du réseau cristallin du silicium, visibles en surface. Elles sont causées par des contraintes (recuit, stress etc.) et peuvent avoir de multiples impacts sur les performances finales des produits. Ces effets peuvent être des problèmes d'alignements en lithographie, par un glissement du réseau le long de la rupture, et pouvant aller jusqu'à la casse du wafer, fragilisé par un grand nombre de *slips*, lors de la manipulation des plaques ou lors de budgets thermiques élevés. Durant les dernières années, ces problèmes ont été exacerbés par l'utilisation grandissante de substrats faiblement concentrés en oxygène interstitiels. Les technologies IGBT (*Insulated Gate Bipolar Transistor*) combinant des forts budgets thermiques, des substrats pauvres en oxygène et une densité forte en tranchées profondes, entraînant du stress localisé et une déformation du wafer, sont particulièrement concernées par ces dérives. Plusieurs équipements sont capables de détecter ces slip lines, avec différentes sensibilités.

L'un d'eux est le Surfscan SP3, basé sur la réflectométrie, qui est un équipement de défectivité extrêmement sensible, mais limité à l'étude de plaques sans motif, et donc non-adaptée au suivi de cette dérive tout le long du *process*, mais elle peut servir de référence sur plaque *blanket (*sans motif*)*. L'En-vision, et son mode d'acquisition de MacroPhotoLuminescence (MacroPL), est aussi sensible à cette dérive [5], et est adapté pour des mesures sur plaque produit. En revanche, la photoluminescence n'est pas directement sensible aux slip lines, mais aux dislocations engendrées par ces dernières, tout en étant sensible à beaucoup d'autres effets parasites du silicium (contamination métallique, charge électrostatique de surface, etc.). Le dernier équipement est le PWG, utilisant un double interféromètre pour mesure la nanotopographie sur l'ensemble du wafer, et mettant ainsi en évidence les slip lines. C'est une technique rapide et capable d'étudier des wafers avec du design, mais avec une résolution limitée (100 µm). Un exemple de cartographie full wafer obtenu sur la même plaque (technologie IGBT) avec chaque équipement est présenté en Figure 73.

On y remarque que les slip lines sont logiquement alignées avec l'orientation du réseau cristallin du silicium, qui pour cette technologie sont orientés à 45° de l'axe du notch.

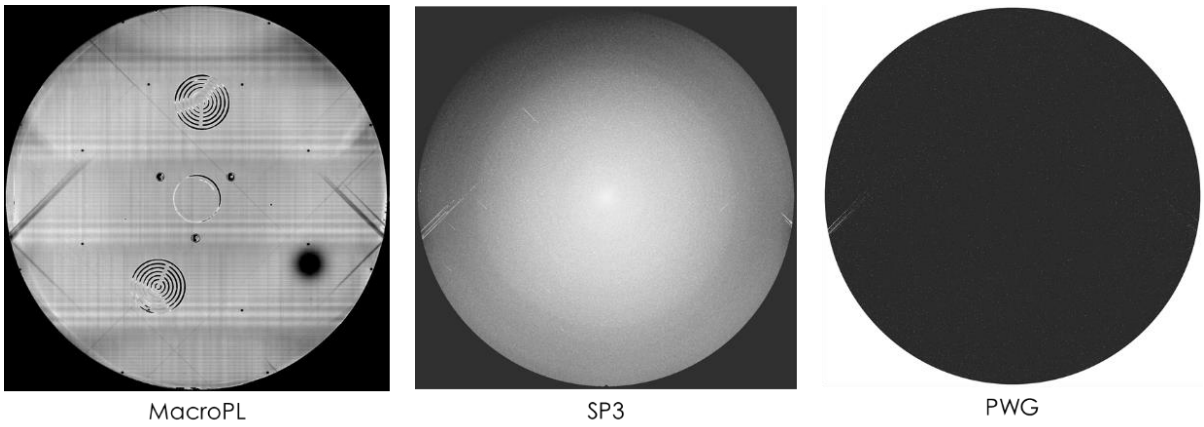

Figure 73 - Exemples de cartographies contenant des slip lines, obtenues avec les différentes techniques full wafer sur la même plaque

Parmi ces techniques, seule la MacroPL ne dispose pas d'un traitement automatique fourni par l'équipementier permettant la détection de ces dérives à partir des données *full wafer* récoltées. En effet, le SP3 est associé à un programme permettant dans un premier temps de repérer tous les pixels correspondants à un défaut (fort signal « Haze » – voir Chapitre I), et à les associer entre eux à un seul défaut s'ils sont localisés sur la même ligne. Le PWG possède quant à lui un algorithme similaire, mais comme ce dernier doit aussi être adapté à l'étude de wafers produits, il doit être capable de différencier les slip lines, des bords des puces. Ce dernier n'est efficace que pour les larges dérives, et n'est pas adapté pour un suivi rigoureux de cette dérive. De plus, l'information fournie en sortie de traitement est aussi très limitée : seule la longueur totale et la profondeur moyenne des slips détectés sont proposées. Il est d'intérêt de développer une solution intégrée permettant la détection des slip lines à partir des données du SP3, du PWG et de la MacroPL. On serait alors en mesure de caractériser l'état du *process* plus précisément, notamment à l'aide des données fournies en sortie par l'algorithme, où chaque défaut sera indépendamment décrit (localisation, taille et profondeur).

Afin d'évaluer la sensibilité relative de chacune des techniques aux slip lines, un lot de 10 wafers *blanket* (sans motif) a été mesuré par les 3 équipements, afin que les images obtenues soient traitées avec l'algorithme basée sur la transformée de Hough.

Optimisation de l'algorithme et résultats

Comme énoncé plus tôt, il est crucial d'optimiser les variables d'entrée de cette transformée pour chacun des équipements, les images obtenues par chacun étant radicalement différentes (Figure 73). Le temps de calcul de cet algorithme étant relativement rapide (< 1 seconde pour une image de 6000x6000 pixels), un script Python a été mis en place afin de faciliter cette optimisation, via l'utilisation de curseurs, permettant de voir l'incidence de chaque changement de paramètre sur la qualité de détection des lignes (Figure 74).

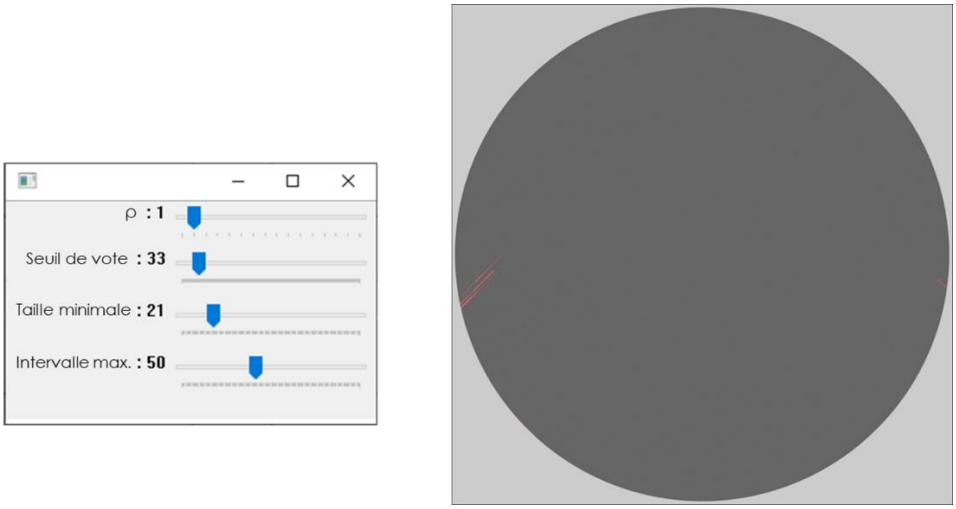

Figure 74 - Exemple d'optimisation des paramètres de la transformée de Hough pour détection des slip lines sur une image obtenue avec le PWG, où les lignes détectées sont surlignées en rouge.

Une fois l'optimisation terminée pour chacune des techniques, on peut retenir ces paramètres pour traiter toutes les images obtenues via ces dernières. On obtient ainsi, suite au protocole de prétraitement et à la détection des lignes via le script Python dédié, un fichier texte (ou excel) contenant les localisations et les dimensions de chacune des slip lines détectées. Ces résultats seront utilisés pour comparer l'efficacité et la sensibilité relative de chacune des techniques, afin de déterminer le contrôle de procédé optimal à mettre en place pour cette dérive.

Afin de vérifier l'exactitude des détections, nous avons manuellement identifié les 59 slip lines présentes sur les images issues du SP3, car comme précisé plus tôt, il possède la meilleure résolution, mais est limité à l'étude de plaques sans motif. Elles serviront de référence pour les autres techniques, où les slip lines seront aussi dénombrées pour valider les performances de l'approche. Les résultats obtenus sont alors présentés dans le Tableau 6. On y remarque que notre approche basée sur le traitement d'image et la transformée de Hough est capable de détecter la grande majorité, voire la totalité, des slip lines présentes sur les images issues du SP3 et du PWG, mais n'est pas aussi efficace pour les données obtenues avec la MPL, bien qu'elle soit aussi sensible que le SP3. Cela peut s'expliquer par le fait que la MPL est extrêmement sensible à diverses propriétés du silicium, et pas uniquement limitée à la surface contrairement aux deux autres techniques. Alors, malgré tout le prétraitement réalisé, il est difficile pour l'algorithme d'isoler les lignes représentant les slip lines, de celles issues entre autres de contact avec les équipements sur la face arrière.

Tableau 6 - Comparaison des slip lines détectables (présentes sur l'image) et détectées par le traitement basé sur la transformée de Hough pour chacune des techniques de suivi de procédé.

Technique	Détectables	Détectées
SP3	59	57
PWG	51	51
MPL	59	44

3. Conclusion

Alors, notre approche d'analyse d'image basée sur la transformée de Hough a été proposée et adoptée. Elle a permis d'améliorer le suivi de procédé actuellement en place pour le contrôle de la dérive des slip lines, en détectant la grande majorité de ces défauts pour la technique de référence (SP3), ainsi que pour la technique utilisable sur tout type de plaque (PWG). Le script python crée lors de ma thèse a été transmis aux ingénieurs en charge du PWG, et il est désormais utilisé pour le suivi de lots R&D afin d'optimiser le procédé de fabrication en contrôlant automatiquement la quantité de slip lines présentes.

Dans ce type de cas simples, il est possible d'utiliser des traitements d'images mathématiques car la caractéristique de l'image à extraire est relativement évidente. Dans des cas plus complexes, où des caractéristiques simples extractibles par des fonctions mathématiques ne peuvent pas être déterminées, il est plus puissant d'avoir recourt aux réseaux de neurones afin notamment de développer des modèles de classifications d'images.

III. Classification d'images : Réseaux de neurones convolutifs

Dans le cadre de la classification d'images, les réseaux de neurones convolutifs (*Convolutional Neural Networks* – CNN) sont devenus la référence pour cette tâche en tant que traitement IA. En effet, les traitements *machine learning* (SVM, *Random Forest*, etc.) utilisant des descripteurs (fonctions de traitements d'image pour extraire les caractéristiques ou *features* d'une image) se sont vus progressivement devancés en performance par les CNNs avec l'avènement de l'utilisation des GPU. De plus, leur simplicité d'utilisation a aussi contribué à leur croissance car il suffit de fournir des images, dont la classe est renseignée (la plupart du temps dans le nom des images ou du dossier les contenant) pour que les *frameworks* réalisent l'entraînement d'un CNN spécifique à ce *dataset*. Ces réseaux de neurones sont régulièrement utilisés dans l'industrie des semi-conducteurs pour la détection de défaut et l'analyse de cause principale [6] ainsi que la classification de motifs présents sur des cartographies de wafers [7].

1. Fonctionnement des réseaux de neurones convolutifs

Dans le chapitre 2, nous avons décrit les architectures des réseaux de neurones profonds afin de traiter des données dites « plates », donc à une dimension. Les images étant des données à 3 dimensions (positions X et Y des pixels, ainsi que leur valeur), des prétraitements sont nécessaires pour pouvoir utiliser ces architectures.

Convolutions

Les réseaux de neurones convolutifs utilisent, comme leur nom l'indique, des convolutions afin de réaliser cette tâche de prétraitement. Ce sont des opérations mathématiques simples où un noyau (*kernel*), qui est une matrice carrée, est déplacée sur l'ensemble de l'image pour en calculer chaque sortie (Figure 75).

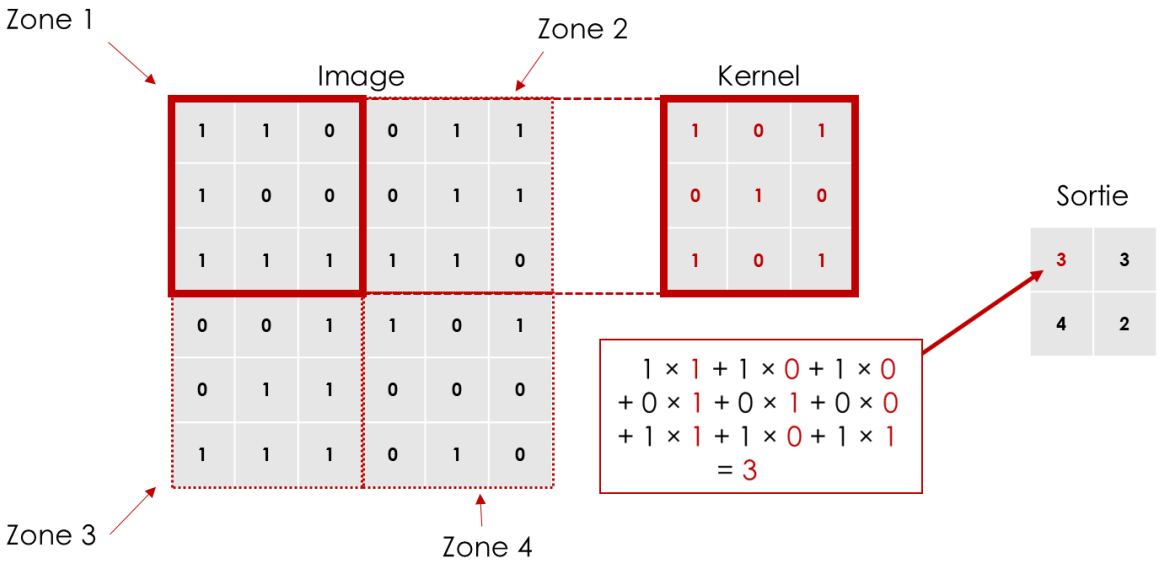

Figure 75 - Exemple de convolution par un kernel appliqué à une image, permettant la réduction de dimensionnalité ainsi que l'extraction de features.

À partir de chaque zone scannée par le *kernel*, on obtient ainsi une sortie unique. Plus la valeur de cette sortie est élevée, plus la région scannée est semblable au *kernel* utilisé. Différents *kernels* permettent d'extraire différentes caractéristiques de l'image. De plus, cette opération peut aussi réduire les dimensions des données d'entrée. Cette réduction est proportionnelle aux dimensions du noyau, ainsi qu'au pas utilisé (nombre de pixels de décalage entre deux applications du *kernel*). Dans le cas d'un pas de 1 pixel, on aurait alors une extraction de *features*, sans réduction de dimensionnalité, ce qui est couramment utilisé dans les larges réseaux de neurones afin de préserver le plus d'informations malgré le grand nombre de convolutions réalisées. Le calcul de cette sortie dépend donc des valeurs présentes dans la matrice utilisée pour scanner l'image, et de leur organisation dans la matrice. Chaque *kernel* permet d'extraire une information spécifique de l'image. Des exemples de ces derniers, et de la caractéristique qu'ils ont pour objectif d'extraire sont présentés en Figure 76. Dans un CNN, ces matrices sont donc extrêmement nombreuses, et de la même façon que les réseaux de neurones optimisent pendant l'entraînement le poids de chaque neurone pour obtenir la sortie souhaitée, les CNN vont favoriser les *kernels* dont les caractéristiques extraites permettent au mieux la classification des images fournies.

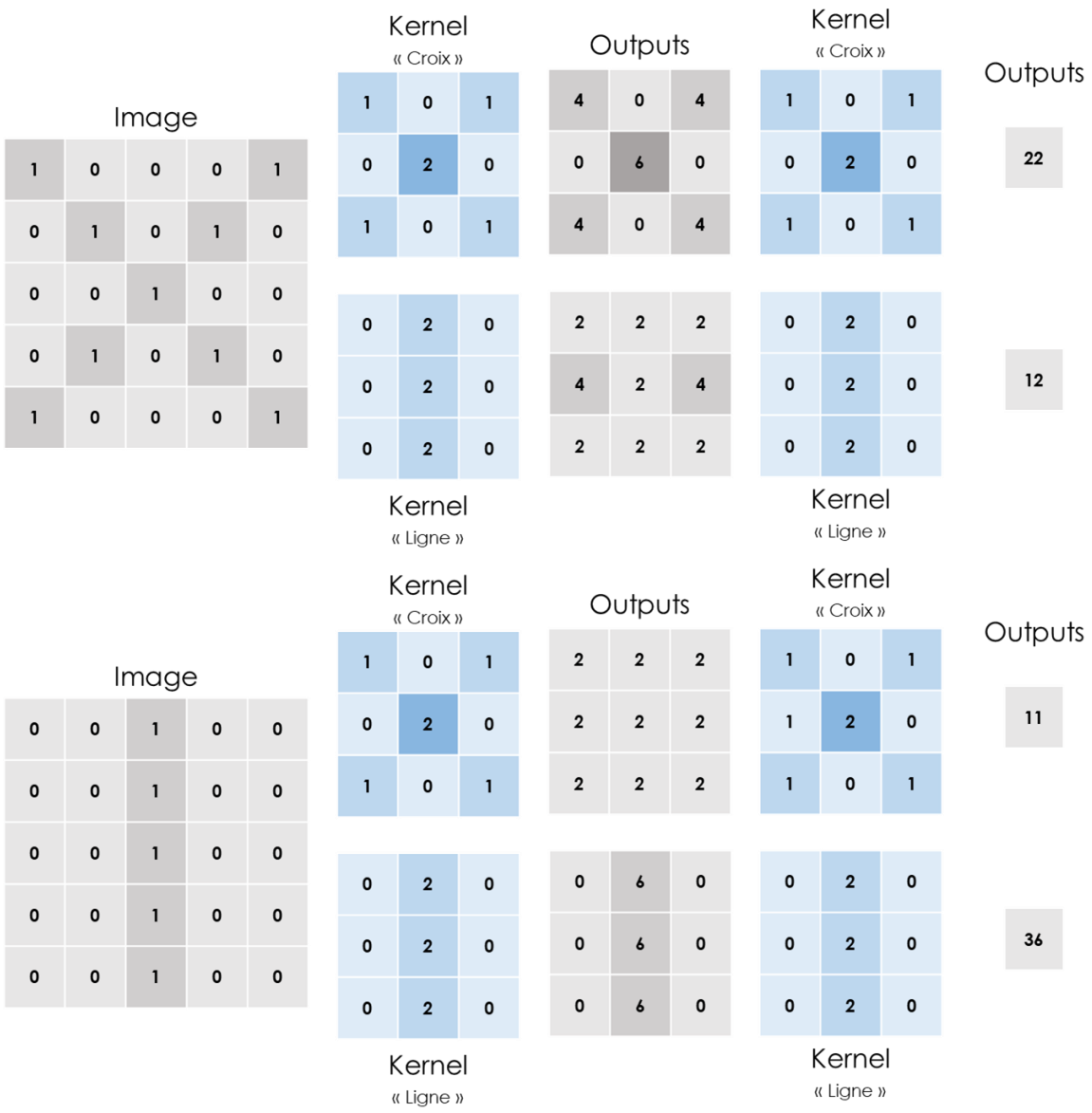

Figure 76 - Exemples de kernels permettant l'extraction de formes spécifiques présentes dans différentes images

Types de couches

D'autres type de couches peuvent être utilisées dans les architectures des CNN afin d'améliorer ses performances. Par exemple, il est commun d'utiliser, après une couche de convolution sans réduction de dimensionnalité (donc pas de 1), une couche dites de « *pooling* » (regroupement), qui réaliser alors cette tâche, en choisissant, par exemple, le pixel de valeur maximale dans un groupe de pixels. La Figure 77 montre un exemple de matrices 2x2 et 3x3 pixels respectivement, sans et avec chevauchement, induisant des sorties grandement différentes. La fonction de cette couche est d'éviter le surentraînement (*overfitting*) en réduisant davantage les dimensions des données les plus profondes dans le réseau.

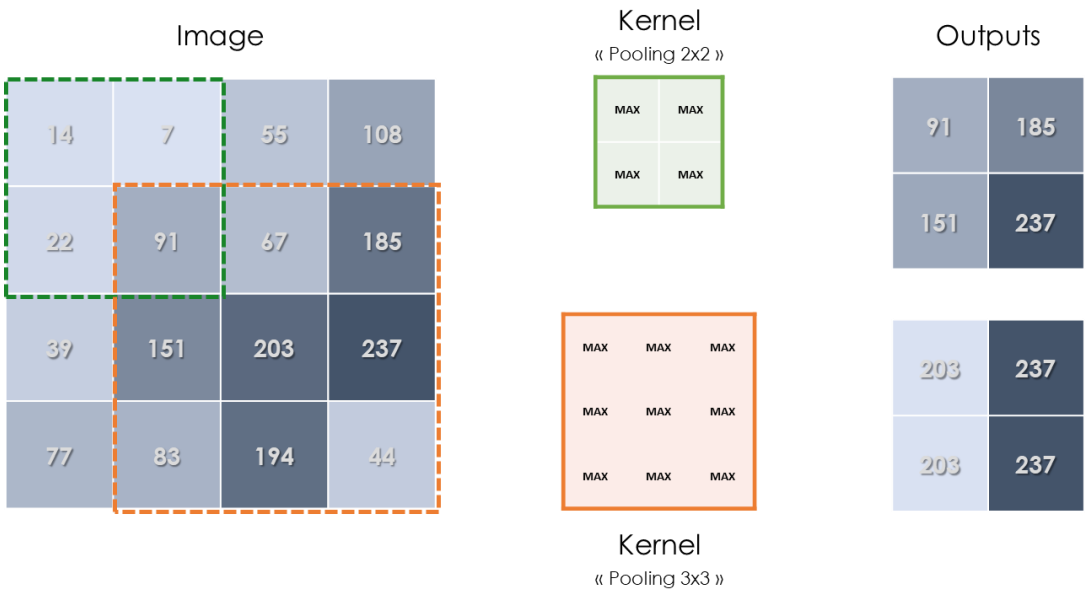

Figure 77 - Exemples d'application de kernels de pooling.

Ajoutons que, de la même façon que pour les réseaux de neurones non-convolutifs présentés dans le chapitre 2, ces couches sont suivies d'une fonction d'activation, permettant le filtrage des caractéristiques mineures. En effet, à la sortie de chaque convolution, afin que seules les régions possédant les *features* extraites par le *kernel* spécifique ne soient traitées par le reste du réseau. Seules les valeurs au-dessus du seuil fixé par la fonction sont conservées. Couramment, ce seuil est fixé à 0 car la plupart des *kernels* utilisés génèrent en sortie des valeurs positives dans le cas où la région considérée possède la caractéristique recherchée, et négative dans le cas contraire. La fonction le plus souvent utilisée est ReLU (*Rectified Linear Units* - Figure 78) qui ne retient donc que les valeurs positives : ReLU(x) = max(0, x). Cette fonction d'activation permet l'introduction de non-linéarité, et ainsi résoudre le problème de la disparition du gradient [8], très courants dans les larges réseaux de neurones.

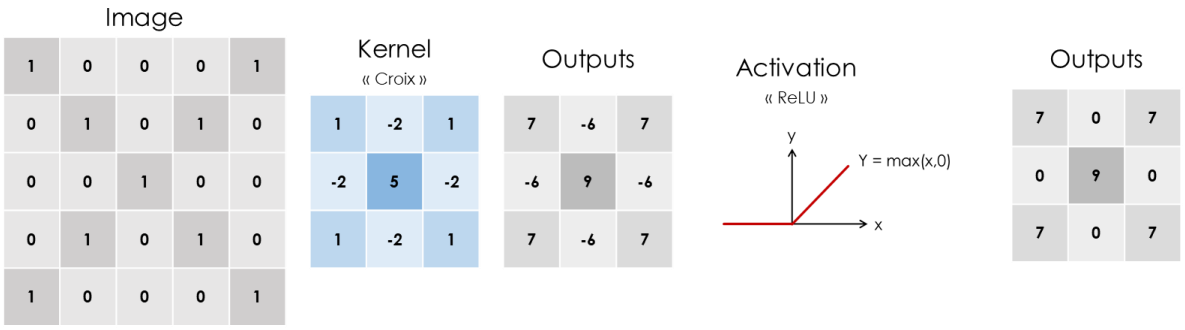

Figure 78 - Exemple d'utilisation du kernel d'activation (ici ReLU).

Augmentation de données

Ces réseaux de neurones nécessitent une large quantité de données pour être entraînés correctement. Il est courant, lorsque la quantité d'images n'est pas suffisante pour obtenir un modèle de classification satisfaisant, d'avoir recourt à de l'augmentation de données, où les images du *dataset*

d'entraînement sont légèrement modifiées afin de générer de nouvelles images. Bien qu'elles soient très proches des images originales, ces images augmentées sont fondamentalement différentes pour un réseau de neurones, et lui permettent alors de continuer son apprentissage. Des exemples de transformations les plus couramment utilisées sont présentées en Figure 79.

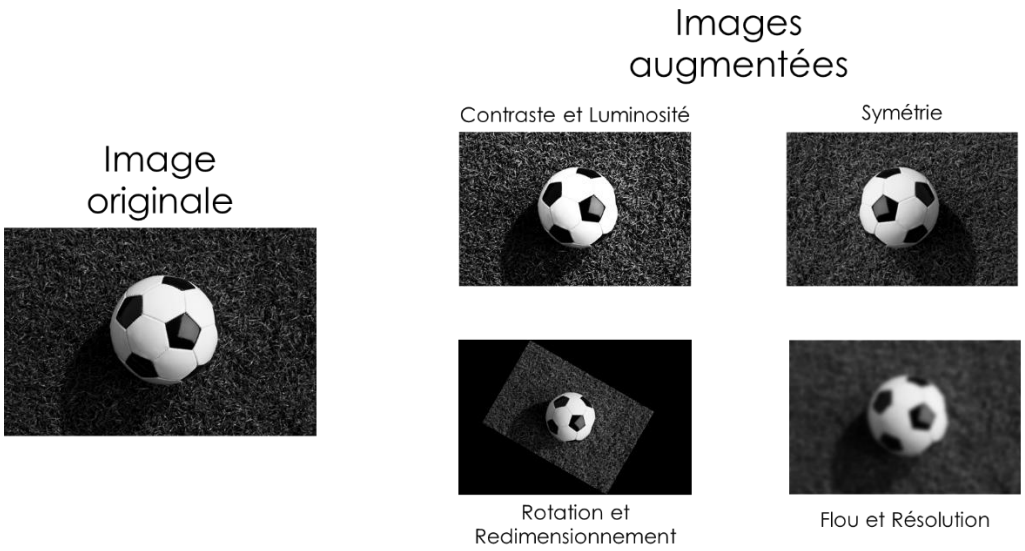

Figure 79 - Exemples d'images augmentées, et des transformations les ayant générées.

Architecture typique d'un CNN

Une fois que suffisamment de convolutions et *poolings* ont été réalisées sur l'image, et que les données sont sous la forme d'un vecteur à une dimension, elles sont traitées par une ou plusieurs couches dites « *fully connected* », où l'entièreté des neurones sont connectés entre eux d'une couche à la suivante, ainsi qu'à l'ensemble des neurones de sortie du réseau (voir Chapitre 2). Ceux-ci débouchent sur un vecteur de taille N (N étant nombre de classes à reconnaître) et la valeur de chacun des éléments de ce vecteur correspond à la probabilité de l'image à appartenir à la classe qu'il représente. Un résumé du fonctionnement d'un CNN est présenté Figure 80.

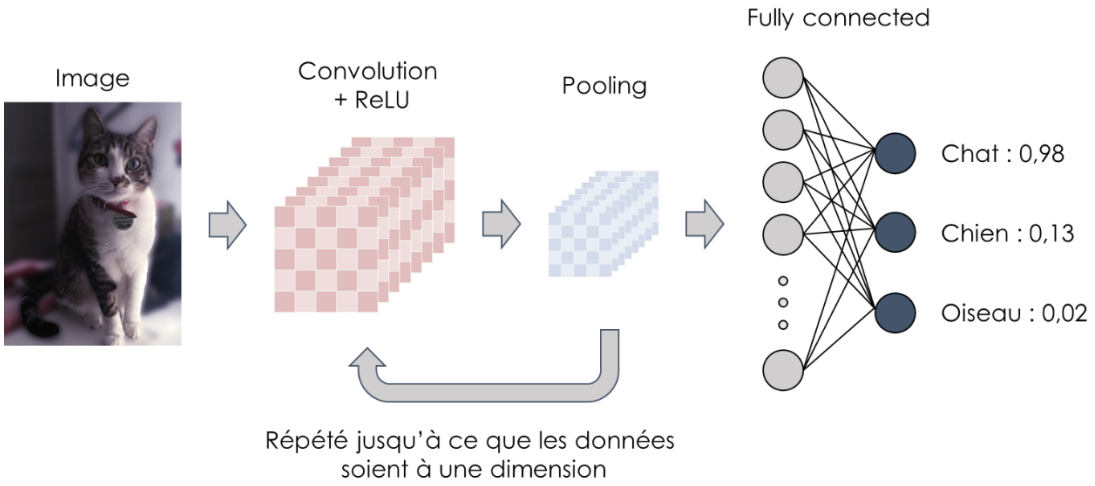

Figure 80 - Exemple de schéma fonctionnel d'un CNN, avec description des différentes couches le composant.

De la même façon que pour les réseaux de neurones non-convolutifs, on peut utiliser un *dataset* de validation pour éviter l'*overfitting* et optimiser l'entraînement. Ce dernier est encore une fois utilisé pour s'assurer de la bonne capacité de généralisation du réseau, et est alors régulièrement traité par le réseau pendant l'entraînement, jusqu'à ce que les performances de ce dernier régressent (moment à partir duquel on choisit d'arrêter l'entraînement du CNN). On obtient ainsi les courbes typiques de précision/erreur présenté dans le chapitre I.

Aussi, il est courant de créer un *dataset* de test pour évaluer les performances du réseau entraîné. Ces dernières sont la plupart du temps représentées par une matrice de confusion (Figure 81), où, pour l'ensemble des images du *dataset*, les classes prédites par le réseau et vraies sont comparées. Cela permet donc d'évaluer quelles classes sont correctement reconnues, et lesquelles sont confondues entre elles par le réseau. Ainsi, on peut identifier des voies d'amélioration du *dataset* d'entraînement à mettre en place pour remédier à ces erreurs, par exemple en augmentant le nombre ou la qualité des données fournies pour ces classes.

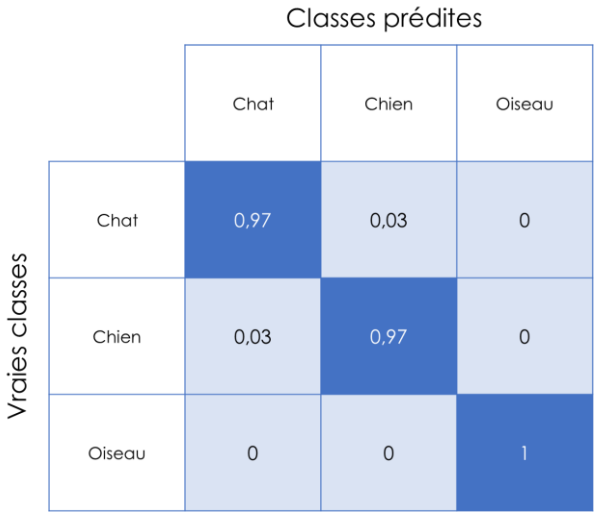

Figure 81 – Exemple de matrice de confusion d'un réseau de neurones à 3 classes. On y remarque une légère confusion entre les classes "Chien" et "Chat" de l'ordre de 3%. Ces résultats peuvent être présentés soit en ratio (comme ici), soit avec le nombre total de détection par classe.

Entraînement et *transfer learning*

Ces réseaux de neurones convolutifs, traitant des données à plus grandes dimensions, sont donc beaucoup plus lourds à entraîner en termes de puissance de calcul. En effet, il serait possible d'entrainer un réseau de neurones profond tels que ceux présentés dans le chapitre 2 à partir de poids aléatoirement initialisés et de rapidement (~ heures) converger vers un modèle prédictif avec des performances satisfaisantes. Néanmoins, plusieurs jours seraient nécessaires pour réaliser le même type d'entraînement pour un CNN.

Il est ainsi courant de recourir pour l'entraînement de ces derniers à du « *transfer learning* » : il s'agit d'utiliser des modèles pré-entrainés par des laboratoires ou entreprises spécialisés sur des gigantesques *datasets* (ex. la base de données *Imagenet* comporte plus d'un million d'images triées dans plus de 1000 classes [9]) afin de les adapter aux données que l'on souhaite traiter. En effet, comme la plupart des premières couches ont pour objectif d'extraire des *features* basiques communs à toutes les images (contours, motifs, gradients, etc.), elles peuvent être conservées (ou légèrement

ajustées), et seules les dernières couches, permettant d'optimiser le poids de ces *features* pour classifier nos données, sont réellement transformées.

De plus, ces modèles pré-entrainés sont issus de compétitions de classification d'images, et leur structure est alors optimisée pour obtenir les meilleures performances possibles. On profite d'un modèle capable de généraliser même avec un très large nombre de classes, possédant une architecture complexe permettant de maximiser les performances sur nos données, une fois ses capacités transférées à notre cas d'intérêt. Les *frameworks* proposent souvent des approches simples de *transfer learning,* à partir des réseaux pré-entraînés les plus performants (par exemple : VGG16, Inceptionv3, ResNet).

2. Évaluation sur le cas d'intérêt des striations

Présentation du cas d'intérêt

Dans les technologies imageurs, les résines colorées sont utilisées comme filtres optiques afin de produire les matrices de pixels RGB (*Red, Green, Blue*). Elles sont déposées sur les capteurs optiques (suivant une mosaïque de filtre Bayer), permettant la reproduction de l'image observée par l'appareil en combinant les signaux reçus par chaque canal. La prépondérance de pixels vert dans cette matrice s'explique par le fait que la luminance est principalement dépendante de la composante verte ($L = Rouge \times 0.2126 + Vert \times 0.7152 + Bleu \times 0.072$). Comme chaque pixel est filtré pour ne recevoir qu'une couleur, il est nécessaire d'utiliser un algorithme de démoisaïquage, appliqué aux pixels environnants afin d'en estimer la valeur RGB. Une représentation de ce type de matrice de pixels, ainsi que du processus de démoisaïquage sont présentées en Figure 82.

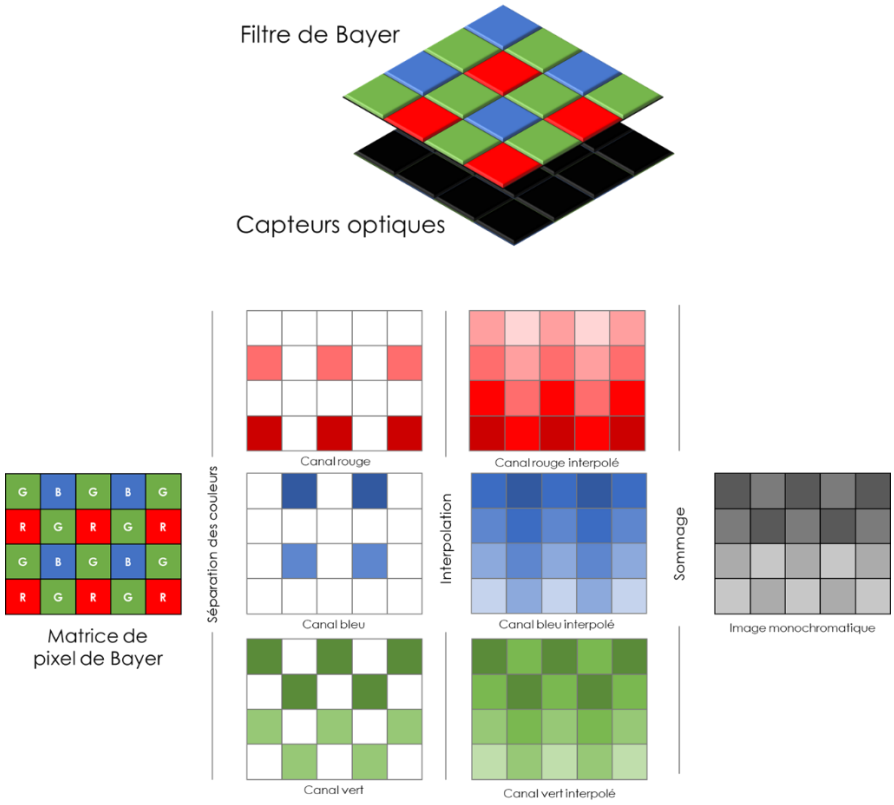

Figure 82 - Architecture des imageurs et résumé du processus de démoisaïquage, permettant d'interpoler les pixels manquant pour chaque canal de couleur pour déterminer la valeur finale de chaque pixel dans l'image finale.

Ces couches sont successivement déposées par *spin coating*, où la résine est déposée puis répartie sur l'entièreté du wafer par rotation de la plaque. Cette étape a lieu, pour cette technologie, relativement tard dans le cycle de fabrication, et une topographie complexe est donc présente en surface du wafer, qui nuit à l'écoulement uniforme des résines colorées. Dans ce contexte, des inhomogénéités de répartition et donc d'épaisseur de la résine peuvent apparaître, causant une signature radiale nommée « striation », prédominante en bord de wafer. À la suite de cette déposition, différentes étapes process sont encore réalisées, notamment la fabrication des microlentilles, qui sont déposées au-dessus de chaque pixel, afin de focaliser la lumière sur les capteurs optiques. Ces variations d'épaisseurs peuvent entraver la focalisation et ainsi engendrer des pertes de performances des capteurs optiques.

Du fait de leur fréquence spatiale large (et donc faible variation locale), elles sont difficilement détectables par les approches classiques de métrologie et de défectivité. En effet, les mesures de métrologie sont réalisées dans des cibles spécifiques en bord de produit, et sont alors de trop petites dimensions pour observer une variation d'épaisseur notable. La même limitation est valable pour les approches de comparaison puce-à-puce couramment utilisées en défectivité pour détecter des défauts physiques (particules, rayures, etc.), car la plupart des puces sont affectées, à différentes échelles, par cette déviation. Ces striations sont, pour l'instant, détectées au moment de l'EWS (*Electrical Wafer Sorting* : triage électrique des wafers), où les performances finales des capteurs sont individuellement testées, ce qui est bien trop tard pour corriger le problème en ligne de fabrication.

Nous avons évalué une approche de Metrospection pouvant être la solution pour contrôler cette dérive de procédé. L'objectif est ainsi d'obtenir des informations brutes sur la surface entière des wafers affectés par ces striations plus tôt dans le cycle de fabrication, et ainsi pouvoir prendre des décisions *process* sur les plaques atteintes.

Équipements utilisés

Afin d'obtenir ces données, deux équipements industriels ont été identifiés : L'interféromètre PWG et le réflectomètre Altair. Ils sont en effet capables d'imager l'intégralité de la surface du wafer et seront donc utilisés pour tenter de détecter cette dérive. L'objectif est donc d'étudier les wafers après le dépôt des résines afin d'évaluer s'ils sont affectés par des striations, et le cas échéant, de définir quelles puces en particulier le sont. Un exemple d'images obtenues par ces techniques sur des wafers à striations est présenté en Figure 83.

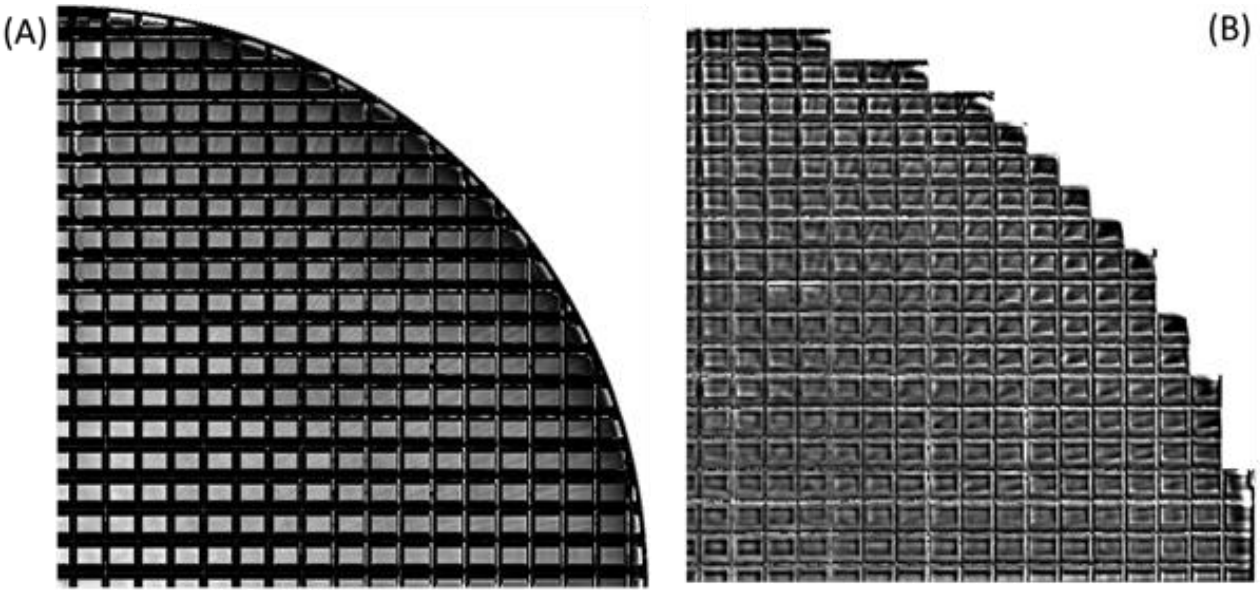

Figure 83 - Exemple d'images obtenues par l'Altair (A) et le PWG (B) sur des plaques à striations

Pour ce faire, des sous-images de puces individuelles sont extraites à l'aide d'un script Python, utilisant les informations de design des plaques pour obtenir les coordonnées de chaque puce et ce pour les deux techniques optiques. Afin de classifier automatiquement ces dernières, nous avons donc eu recourt à un CNN. Un exemple de chaque classe est présenté en Figure 84 et on y remarque qu'une 3ème classe (Erreur de reconstruction) est listée pour le PWG. En effet, cette technique reposant sur l'interférométrie, elle est adoptée pour caractériser des surfaces opaques ou de faible nanotopographie. Comme on étudie ici des résines colorées transparentes, les interférences optiques peuvent avoir lieu à différentes profondeurs dans ces matériaux, et la reconstruction du profil topographique est rendue inexploitable par l'équipement. L'image obtenue est alors extrêmement bruitée et ne sera pas traitable par l'algorithme, mais heureusement, ce problème est relativement rare (~5% des puces atteintes).

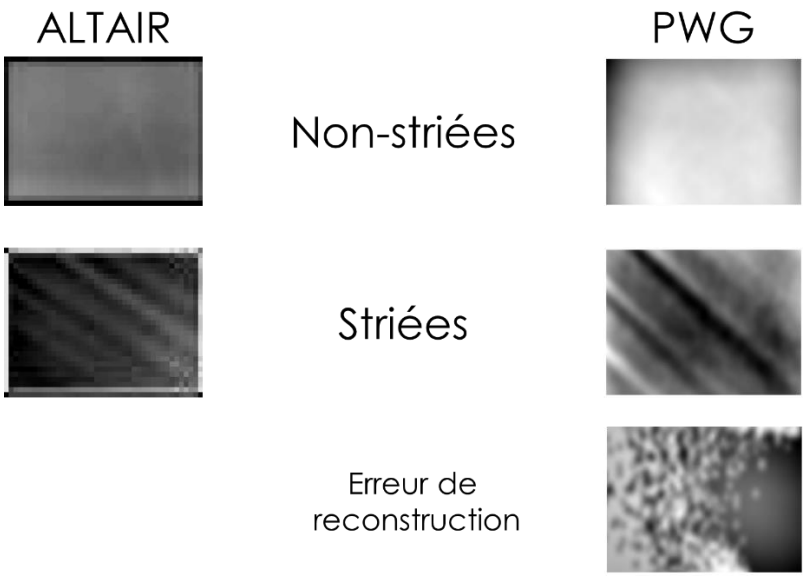

Figure 84 - Exemple de puces (région de la matrice de pixel) pour chaque classe, et chaque technique utilisée

Architecture du CNN

Comme expliqué plus tôt, nous avons adapté une approche utilisant le *transfer learning* de notre CNN à partir de nombreux modèles pré-entrainés, et nous avons choisi celui dont les performances nous paraissaient les plus adaptées à notre cas d'intérêt : ResNet [10]. Sa spécificité est qu'il « saute » certaines couches (convolutions ou activations), et cela a deux intérêts principaux : s'affranchir du problème de disparition du gradient (arrêt de l'apprentissage dû au trop grand nombres d'opérations/couches) et réduire le problème de dégradation (réduction de la qualité d'un réseau de neurones avec ajout de couches supplémentaires). Cela permet de pouvoir obtenir des réseaux de neurones performants avec des grandes profondeurs (>100 couches), permettant l'extraction de *features* complexes. Nous avons ainsi utilisé un réseau de type ResNet152 (152 couches de convolutions. Pour simplifier la représentation de ce grand réseau, nous avons représenté en Figure 85 la version réduite Resnet32. La différence tient en ce que les groupes de couches de convolutions, représentés par couleur dans la figure, sont simplement répétés 5 fois chacun avant de passer au prochain groupe.

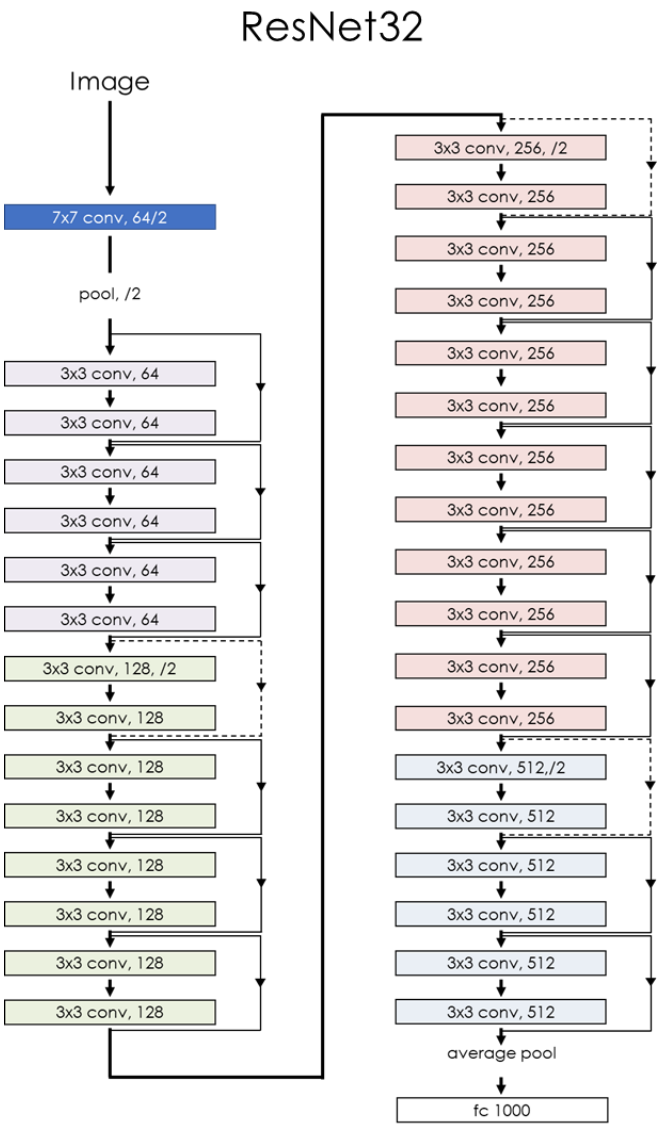

Figure 85 - Architecture du CNN Resnet32, composés de matrices de convolution de taille 3x3, les flèches désignant les sauts de couches, et qui se termine par une couche fully connected (fc) de 1000 neurones.

Cette architecture a depuis été dépassée par des réseaux plus complexes pour des applications à plusieurs milliers de classes, mais le gain en performance sur nos types d'études serait probablement minime.

a. Évaluation sur un lot R&D

Présentation du cas d'intérêt

Afin d'avoir à disposition pour l'entraînement et l'évaluation de notre réseau de neurones suffisamment de données, un lot R&D de 24 wafers a été produit avec différentes conditions de process, afin de déterminer celle qui génère le moins de striations, et donc avec les meilleures performances finales du produit. Un résumé de ces conditions de *process* est présenté en Figure 86. Les deux techniques optiques (PWG et Altair) ont alors été utilisées après l'étape de dépôt de résines colorées afin d'obtenir des images full wafer de toutes les plaques du lot.

		Wafers																						
Opération	Recette	3	8	13	15	19	24	17	23	11	6	14	18	4	22	12	7	5	2	20	10	16	1	9	21
Lithographie – Remplissage	Old	x	x	x	x	x	x	x	x	x	x	x	x												
	New													x	x	x	x	x	x	x	x	x	x	x	x
Lithographie – Planarisation	Standard	x	x	x	x	x	x							x	x	x	x	x	x						
Dépôt résine Verte	Standard	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Dépôt résine Bleue	Old	x	x	x				x	x	x				x	x	x				x	x	x			
	New				x	x	x				x	x	x				x	x	x				x	x	x
Dépôt résine Rouge	Standard	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

Figure 86 - Résumé des étapes de process réalisés sur chacune des plaques du lot R&D

Nous avons choisi d'utiliser le *framework* Tensorflow, comme lors de l'étude présentée en chapitre II (Traitement model-less de spectre). Comme explicité plus tôt, il est uniquement nécessaire, pour ce type de réseau de neurones, de fournir au *framework* utilisé un *dataset* d'images manuellement identifiées par classe (via nom du dossier ou de fichier). Il est donc nécessaire de classifier suffisamment d'images de puces striées et non-striées (ainsi qu'avec erreur de reconstruction pour le cas du PWG) pour générer nos *datasets* d'entraînement, de validation et de test. Il est usuellement recommandé d'avoir au minimum une centaine d'image par classe pour l'entraînement d'un réseau par *transfer learning*, mais au vu de la similarité de nos différentes classes entre elles, ainsi que de la quantité de données à notre disposition, nous avons choisi d'utiliser 250 images par classe. De plus, nous avons choisi de réaliser une augmentation des données avant l'entraînement, et nous avons utilisé des changements de luminosité et de contraste, ainsi que des floutages et changements de résolution. Les rotations et redimensionnements n'ont pas été utilisés car toutes les images ont la même taille et orientation. La répartition des données utilisée dans chacun des *datasets* est présentée dans le Tableau 7.

Tableau 7 - Répartition des données dans les datasets d'entraînement, de validation et de test pour les CNN (ResNet158) destinés à classifier les sous-images extraites respectivement des acquisitions Altair et PWG.

	ALTAIR	PWG
Dataset d'entrainement	250 images de puces striées 250 images de puces non-striées	250 images de puces striées 250 images de puces non-striées 250 images de puces « Erreur de reconstruction »
Dataset de validation	100 images par classe	100 images par classe
Dataset de test	1000 images (répartition aléatoire)	1000 images (répartition aléatoire)

Entraînement et validation

À partir de ces *datasets*, nous avons entraîné puis évalué les deux CNN (un pour chaque technique utilisée). Les *datasets* de validation ont alors été utilisés pour évaluer la progression de l'entraînement, et arrêter ce dernier après 30 epochs sans progression. Les courbes précision/erreur de chaque CNN sont présentées en Figure 87. On y remarque notamment qu'au début de l'entraînement, avant même que le modèle pré-entrainé puisse s'adapter à nos données, la précision est déjà de 65% pour le *dataset* Altair. Cela prouve la robustesse de ces réseaux, et leur capacité de généralisation forte. Ils convergent tous les deux rapidement vers des précisions très hautes (>95%) et des erreurs faibles (<10%).

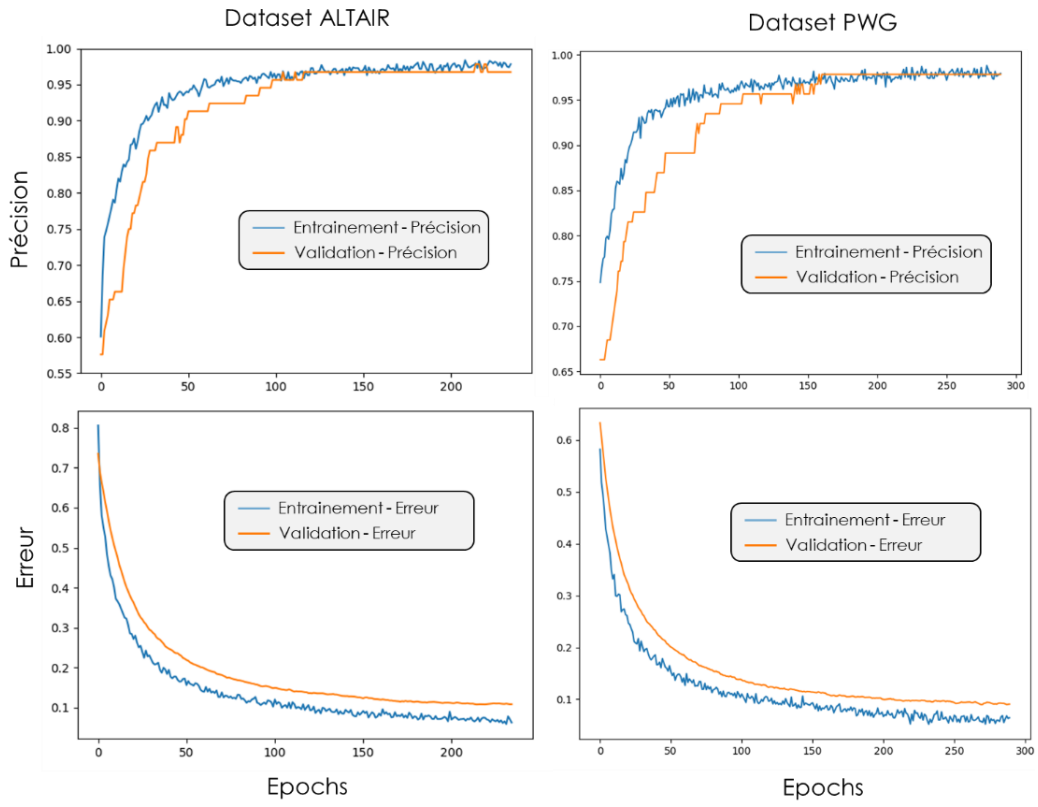

Figure 87 - Courbes d'entraînement (Erreur/Précision) du CNN "striations" basé sur ResNet156, pour le dataset Altair et PWG.

On utilise ces deux réseaux entraînés sur leur *dataset* de test afin d'évaluer leur performance sur des données jamais étudiées. On obtient ainsi une précision de 96,4 % pour le CNN entraîné à partir du *dataset* Altair, et 93,1% pour celui entraîné sur les données PWG (Tableau 8). Cette différence de performance peut notamment s'expliquer par la classe supplémentaire présente dans le *dataset* PWG.

Tableau 8 - Matrices de confusion obtenues sur les datasets de test pour le CNN Altair (gauche) et PWG (droite).

ALTAIR	Prédites : Striée	Prédites : Non-Striée
Vraies : Striée	597	17
Vraies : Non-Striée	19	367

PWG	Prédites : Striée	Prédites : Non-Striée	Prédites : Erreur de Recon.
Vraies : Striée	399	24	8
Vraies : Non-Striée	27	454	7
Vraies : Erreur de Recon.	3	0	78

Résultats

Au vu des performances robustes des deux algorithmes, on traite l'intégralité des images de puces obtenues sur le lot R&D, afin de les classifier comme striée ou non-striée chacune de ces dernières. Une comparaison des prédictions des CNN entraînés sur les deux *datasets* permet de vérifier une bonne adéquation (Figure 88) entre les traitements réalisés sur les images issues des deux techniques, la plupart des différences de prédictions étant dues aux erreurs de reconstruction. On y observe aussi que, comme attendu, cette déviation est plus forte en bord de wafer.

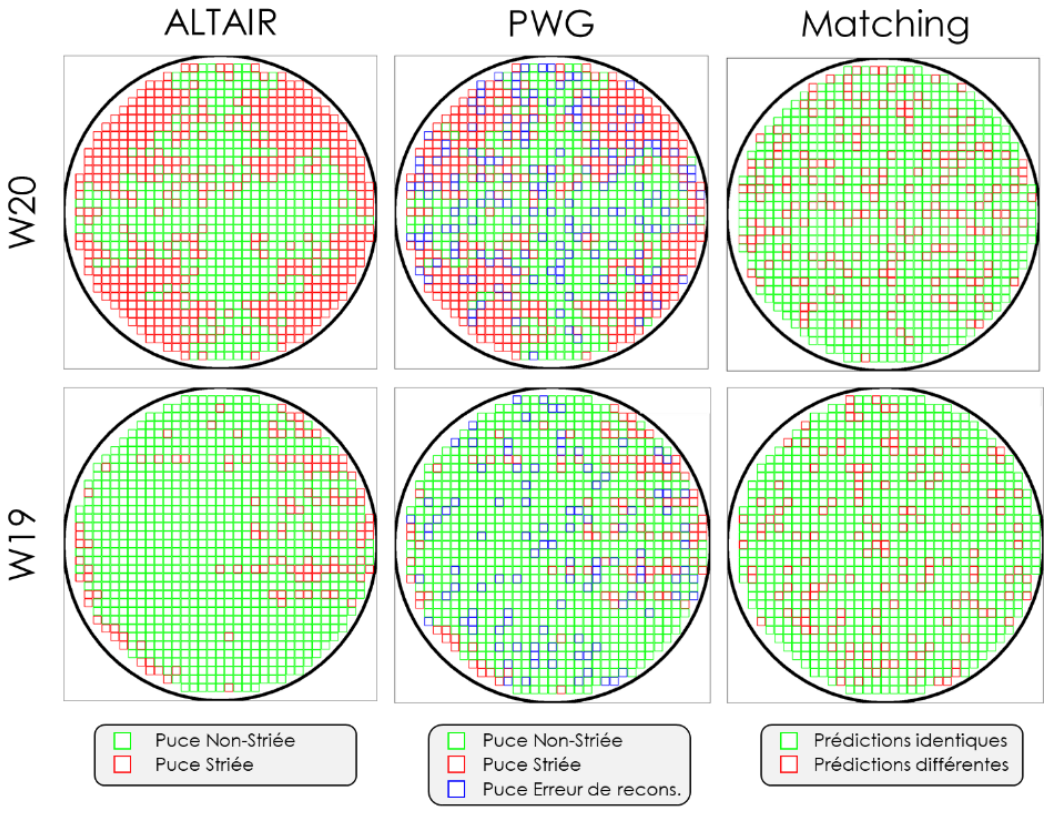

Figure 88 - Exemples de répartitions des puces affectées par les striations, à partir des prédictions faites par les CNN entraînés sur les datasets Altair et PWG. La concordance (Matching) des deux prédictions est présentée à droite.

Corrélations aux tests EWS

Une fois les analyses menées à l'étape de dépôt de résines colorées, le process de réalisation complet a permis de collecter les résultats EWS, permettant de déterminer les conditions optimales donnant les meilleures performances des capteurs optiques. Il serait alors aussi intéressant d'évaluer la corrélation de nos résultats (issus des prédictions des CNNs) aux résultats EWS, de sorte à savoir si notre approche peut, en plus de détecter automatiquement les puces possédant des striations, prédire les performances finales des produits. Pour ce faire, nous avons récupéré les résultats de rendement obtenus lors de ces tests (pourcentage de puces défectueuses), et nous les avons comparés au pourcentage de puces affectées par les striations par wafer. Les résultats sont présentés dans le Tableau 9 et les courbes de corrélation en Figure 89. Le tableau montre clairement une tendance similaire entre les deux métriques étudiées : pourcentage de puces défectueuses en EWS et pourcentage de puces striées déterminées par le CNN, et cela pour les deux techniques optiques utilisées. Ils concordent notamment sur le meilleur sous-lot, et donc la meilleure condition de process, à savoir les wafers 15/19/24. Cette tendance est vérifiée par les courbes qui montre une très forte corrélation ($R^2 > 0.9$), avec une meilleure performance dans le cas du *dataset* Altair, prévisible au vu de l'absence d'erreurs de reconstruction pour cette technique.

Tableau 9 - Comparaison des résultats obtenus aux tests finaux de rendements (EWS - Pourcentage de puces défectueuses) aux résultats obtenus par les CNNs entraînés sur les datasets Altair et PWG (Pourcentage de puces striées).

		W03	W08	W13	W15	W19	W24	W17	W23	W11	W06	W14	W18
EWS		34,14	54,77	63,1	18,49	14,83	16,41	50,11	58,42	52,56	17,13	18,5	25,77
CNN	ALTAIR	22,91	42,21	44,54	9,87	11,39	10,13	36,26	41,77	36,04	15,61	14,33	17,85
	PWG	33,06	44,32	49,51	19,56	16,07	17,6	42,68	46,56	44,48	22,3	24,81	23,5

		W04	W22	W12	W07	W05	W02	W20	W10	W16	W01	W09	W21
EWS		67,89	66,78	62,36	23,39	22,73	15,73	68,26	68,95	69,62	23,35	22,74	21,97
CNN	ALTAIR	45,26	48,29	49,13	13,01	16,59	12,64	48,29	51,75	54,2	21,27	20,91	20,68
	PWG	46,99	45,94	44,7	18,85	15,79	12,4	45,14	46,45	46,67	16,04	15,36	15,79

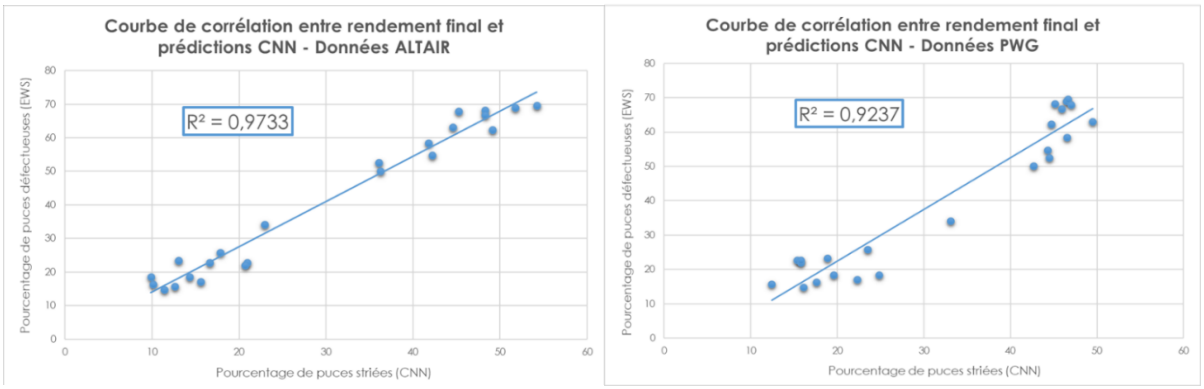

Figure 89 - Courbes de corrélations entre les résultats obtenus en EWS et avec le traitement CNN des images obtenues avec les équipements Altair et PWG.

Cette étude nous a permis d'évaluer la capacité des mesures optiques après le dépôt de résines colorées à, premièrement, détecter automatiquement les striations, puis, à prédire les performances finales des plaques étudiées. Bien que les deux techniques présentent des résultats satisfaisants, la technique de réflectométrie Altair, libre d'erreurs de reconstruction (contrairement à l'interféromètre PWG), a été retenue pour confirmer ces résultats et valider l'intérêt de cette approche dans le suivi de la dérive des striations.

b. Corrélation avec le ContourGT-X

Présentation du cas d'intérêt

Le CEA-LETI (Laboratoire d'électronique et de technologie de l'information) est l'un des principaux centres de recherche appliquée en microélectronique et nanotechnologies du monde, et l'un des partenaires clef de STMicroelectronics, dans des domaines allant des sciences des matériaux aux technologies logicielles. Dans ce contexte, le LETI a été commissionné afin d'optimiser le procédé de fabrication des technologies imageurs affectées par les striations. De la même manière que précédemment, un lot de 20 wafers (que l'on nommera « LETI ») a été produit avec des conditions de *process* variables afin de déterminer celle qui produira les performances optimales. Afin de valider notre approche de détection de striations par CNN, ces plaques ont été mesurées après le dépôt des résines colorées avec l'Altair sur le site d STMicroelectronics Crolles. De plus, dans l'optique de lier les résultats obtenus pour chacune de ces plaques à des mesures physiques, nous avons utilisé un équipement supplémentaire d'interférométrie présent au LETI : Le ContourGT-X. Cette technique locale étant relativement lente, seuls quelques champs lithographiques localisés sur une diagonale ont été considérés (Figure 90). Pour chaque champ, les douze puces contenues à l'intérieur ont été mesurées. La mesure consiste à relever la nanotopographie d'une zone identifiée comme particulièrement propice à l'apparition de striations lors d'études préliminaires, à savoir le coin supérieur droit de la puce, qui est une zone plane en dehors de la matrice de pixel. On obtient ainsi, parmi les nombreuses données d'intérêt extraites du profil nanotopographique de cette zone, les hauteurs maximales, et moyennes des striations détectées, mais aussi leur fréquence spatiale.

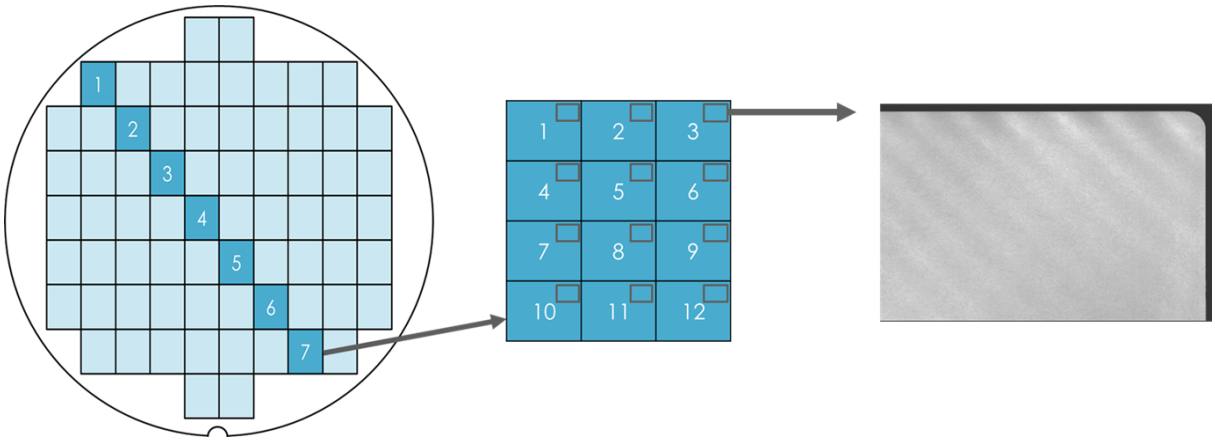

Figure 90 - Protocole de mesure pour le Contour GT-X, avec 7 champs lithographiques étudiés, contenant chacun 12 puces, dont la nanotopographie du coin supérieur droit sera mesuré. Une cartographie de nanotopographie typique possédant des striations est présentée à droite.

Cependant, ces mesures sont en faible nombre (84 par plaque), afin de pouvoir comparer ces résultats à ceux du pourcentage d puces affectées par les striations, obtenus par le traitement CNN des images Altair (plus de 1000 puces), les données obtenues par le ContourGT-X sur chaque puce ont dû être moyennée par champs, puis pondérées par le nombre de puces que ces derniers représentent à l'échelle du wafer (Figure 91).

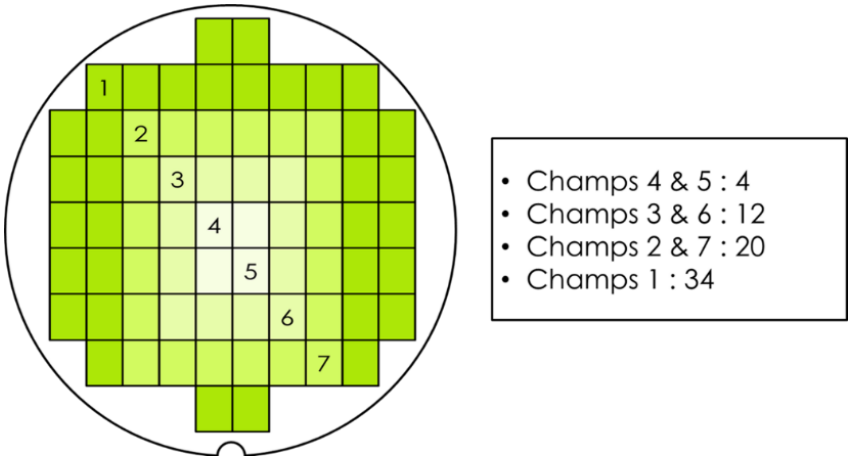

Figure 91 - Représentation de la pondération des données obtenus pour chaque champ lithographie selon le nombre de champs qu'ils représentent. Les poids associés à chacun de ces champs sont présentés à droite.

Résultats

Les résultats obtenus par ce traitement sont comparés, comme dans le cas de l'étude précédente, aux proportions de puces affectées par les striations, déterminées par l'approche CNN appliquée aux images issues de l'Altair. De plus, toutes les plaques du lot ont, en fin de *process*, été mesurées en *EWS* afin de déterminer leur rendement respectif. Premièrement, la comparaison entre les résultats obtenus à l'aide de l'Altair et du ContourGT-X sont présentées en Tableau 10 et en Figure 92. De toutes les métriques mesurées, on remarque que la largeur et la hauteur moyenne horizontale (préfixe « E ») montrent une forte corrélation avec le pourcentage de puces striées déterminées par le traitement CNN des images Altair. Cette corrélation forte à cette technique de référence permet de renforcer la validité de notre approche et permettre de mettre en place un suivi de procédé robuste des striations basé sur les réseaux de neurones convolutifs.

Tableau 10 - Résultats comparatifs entre le pourcentage de puces affectées par les striations à partir du traitement CNN des images Altair, et les différentes métriques pondérées obtenues par la mesure du ContourGT-X. Les données « N » et les données « E » sont extraites de profils orientés respectivement verticalement et horizontalement.

		W02	W03	W04	W05	W08	W09	W13
CNN (Altair)		79.87	77.86	74.84	83.27	86.92	65.79	56.35
ContourGT-X	N-Max (nm)	21.04	19.93	14.28	9.45	10.09	8.12	9.25
	N-Moyen (nm)	18.12	17.43	11.61	7.29	7.81	6.12	7.21
	E-Largeur (mm)	0.05	0.04	0.04	0.05	0.05	0.04	0.03
	E-Max (nm)	13.38	13.84	9.55	7.96	9.49	8.03	6.73
	E-Moyen (nm)	10.66	11.04	7.54	6.00	7.35	6.12	5.03
	N-Période (mm)	0.18	0.18	0.17	0.17	0.18	0.17	0.17
	N-Largeur (mm)	0.04	0.04	0.03	0.03	0.03	0.03	0.03

		W14	W16	W17	W18	W19	W20	W21
CNN (Altair)		54.72	56.10	67.67	69.18	87.67	88.18	84.28
ContourGT-X	N-Max (nm)	9.80	9.16	7.92	8.78	19.14	17.12	16.24
	N-Moyen (nm)	7.67	7.16	6.12	6.82	16.24	14.26	13.45
	E-Largeur (mm)	0.03	0.03	0.04	0.05	0.06	0.06	0.06
	E-Max (nm)	6.61	7.08	8.09	8.63	14.24	14.17	13.51
	E-Moyen (nm)	4.96	5.38	6.21	6.74	11.51	11.46	11.01
	N-Période (mm)	0.17	0.17	0.17	0.17	0.18	0.18	0.18
	N-Largeur (mm)	0.03	0.03	0.03	0.03	0.04	0.04	0.05

Tableau 11 - Tableau récapitulatif des corrélations entre les résultats obtenus avec le traitement CNN des images Altair, et des différentes métriques pondérées obtenues par la mesure ContourGT-X.

Métrique (ContourGT-X)	Corrélation aux résultats CNN (Altair)
N-max	0.609
N-Moyen	0.596
E-Largeur	0.943
E-Max	0.779
E-Moyen	0.938
N-Période	0.776
N-Largeur	0.807

Corrélations aux tests EWS

Comme pour le cas d'intérêt précèdent, on compare les résultats obtenus par CNN aux résultats de rendement finaux obtenus en EWS afin de valider la capacité de notre approche à prédire les performances finales des plaques à partir de la quantité de puces affectées par la dérive de striations. Les résultats obtenus sont présentés en Figure 92.

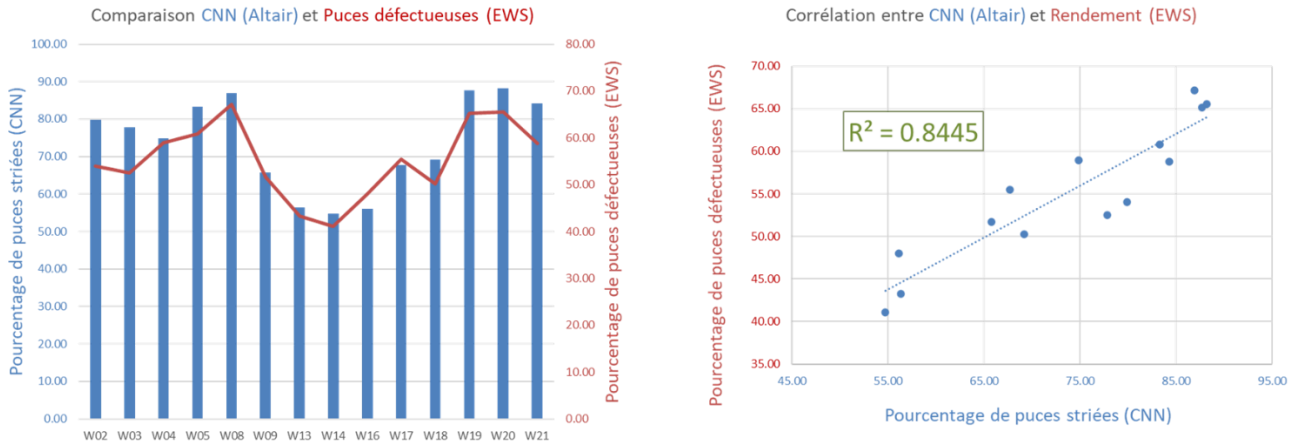

Figure 92 - Courbe comparative (gauche) et de corrélation (droite) entre les pourcentages de puces striées déterminées par le traitement CNN des images Altair, et les résultats de rendements finaux des plaques étudiées déterminés en EWS.

La corrélation observée entre les deux mesures étudiées est relativement bonne (R^2 = 0.84), mais tout de même inférieure à celles obtenues dans les évaluations précédentes. On peut expliquer cette baisse dans ce cas par le fait que d'autres dérives de *process* ont eu lieu après le dépôt de résine, entraînant un rendement fortement réduit (minimum de 40% de puces défectueuses) pour l'ensemble des plaques, et donc une relation moins évidente entre la dérive des striations, et les performances finales.

3. Conclusion

Notre approche basée sur l'utilisation de réseaux de neurones convolutifs pour traiter des images optiques de wafer est alors capable d'automatiquement détecter la dérive de striations, en classifiant les puces ayant subi cette dérive. Ce traitement permet ainsi de déterminer la quantité de puces par plaque affectées par cette déviation de *process*, et peuvent ensuite permettre d'évaluer les performances finales des plaques. La validité de notre approche est renforcée par la corrélation de nos résultats avec des propriétés physiques des striations, mesurées par interférométrie. L'utilisation d'un CNN pour traiter les images Altair pourrait fortement renforcer le suivi de procédé actuellement en place dans le cas de la dérive des striations, en permettant un contrôle automatique, robuste et rapide à mettre en place, versatile et capable de s'améliorer au fur et à mesure de son utilisation à l'aide de données supplémentaires obtenues en production.

IV. Classification d'objets : Réseaux de neurones convolutifs régionaux

Nous avons vu dans la partie précédente que les réseaux de neurones convolutifs sont extrêmement performants et robustes pour la classification automatique d'images, surtout lorsque la technique de *transfer learning* est utilisée. Ces outils sont donc très pratiques pour identifier des images de défauts (ex. *Automatic Defect Classification*), mais ne sont capable de traiter les images qu'entièrement, et sont donc limités à l'étude d'image de défauts uniques. En effet, si plusieurs défauts, de classes différentes, étaient présents dans ces images, un CNN ne pourrait pas efficacement traiter l'image pour en extraire cette information.

1. Présentation des RCNN

Ainsi, lorsque l'on cherche à identifier de multiples objets présents dans une image, il est courant d'utiliser une évolution de ces algorithmes, appelée réseaux de neurones convolutifs régionaux (*Region-based Convolutional Neural Network* – RCNN). Ils sont en effet capables, via un prétraitement de l'image, d'extraire des zones d'intérêts susceptibles de contenir des objets, puis de classifier ces objets indépendamment. Ces réseaux sont par exemple utilisés dans les véhicules autonomes pour la détection des obstacles [11], ou encore dans le milieu biomédical pour la classification de cellules [12], mais aussi dans l'industrie des semi-conducteurs afin d'assurer le contrôle de procédé et améliorer la précision de la détection de défauts [13] [14]. Comme ils sont basés sur des réseaux de neurones convolutifs, ces algorithmes intelligents sont mieux adaptés à l'étude d'images à contraste/luminosité variables que les traitements mathématiques couramment utilisés par les équipements de métrologie ou de défectivité.

Architecture des RCNNs

Le fonctionnement des RCNNs est donc très proche de celui des CNN, auquel est ajouté un réseau de proposition de régions (*Region Proposal Network* – RPN). La façon dont il fonctionne est similaire à la façon dont un CNN extrait les *features* d'une image. En effet, ce pré-réseau réalise de nombreuses convolutions de l'image d'origine pour en extraire des « *features maps* », soit des cartographies de caractéristiques, dont l'intensité va donc traduire la probabilité qu'un objet soit présent dans cette région. Le RPN va scanner les matrices finales extraites par convolution avec des *bounding boxs* (Bbox - cadre de limitation) pour déterminer où sont probablement localisés les objets,

et quelle est leur taille, en stockant ces informations dans le *feature vector* qui sera à son tour traitée par une couche *fully connected* pour sélection lesquelles de ces Bbox représente réellement des zones d'intérêts pouvant contenir des objets. Des sous-images sont extraites de ces Bbox pour être traitées par un CNN pour classification. Un résumé du fonctionnement de ce type de réseau est présenté en Figure 93. Comme pour les CNN, des réseaux pré-entrainés sont disponibles librement afin de pouvoir utiliser le *transfer learning* sur des *datasets* spécifiques. Nous avons donc choisi d'utiliser RetinaNet [15], l'un des meilleurs modèles de détection d'objet « ponctuel » (en une étape). Pour cette raison, il est couramment utilisé pour l'imagerie spatiale et aérienne.

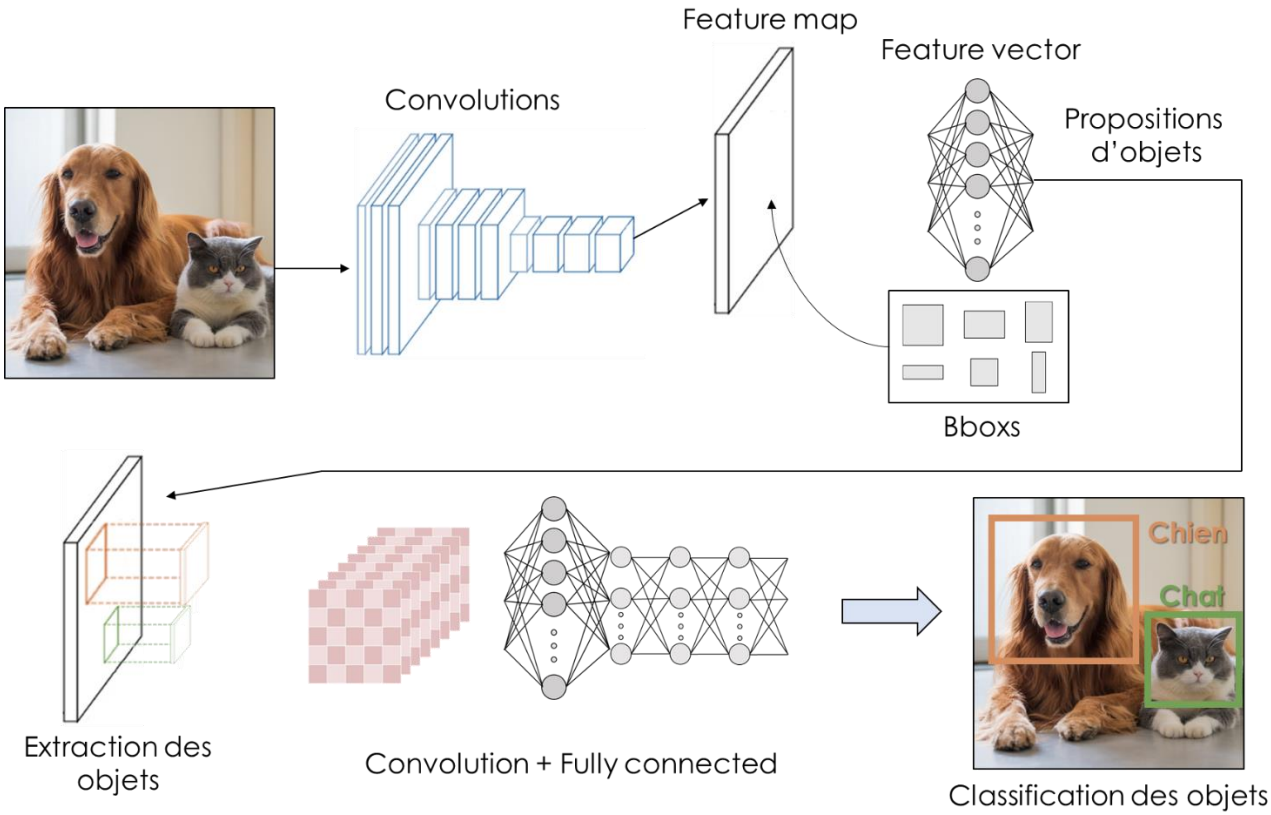

Figure 93 - Schéma résumant le fonctionnement d'un RCNN, débutant par une segmentation de l'image par le FPN, puis classification des objets extraits par un CNN.

Entraînement et annotations

Le fonctionnement d'un RCNN étant différent de celui d'un CNN, la génération d'un *dataset* d'entraînement pour un RCNN l'est donc aussi. En effet, les multiples objets appartenant à différentes classes étant répartis dans l'image, il n'est plus adapté d'indiquer une classe unique pour chaque image. Il faut donc fournir l'information du nombre, de la localisation, de la taille et de la classe de tous les objets que l'on cherche à détecter. Ces informations sont stockées dans des fichiers d'annotations crées par des logiciels spécialisés, qui contiennent les *bounding boxs* et les classes des objets à identifier. Un exemple d'annotation est décrit en Figure 94. L'augmentation de données est aussi possible pour ce type d'architecture, les *frameworks* permettant d'automatiquement transformer les annotations pour correspondre à la nouvelle localisation et taille des objets dans l'image augmentée.

Une fois ces annotations créées, le RCNN possède toutes les informations nécessaires à son entraînement, et la même logique d'utilisation d'un second *dataset* de validation pour éviter l'*overfitting* est aussi applicable ici. Pour entraîner ces algorithmes, nous avons choisi le *framework* « Detectron2 [16] », issu des laboratoires de recherche de Facebook, et basé sur PyTorch. Il est relativement simple d'utilisation et extrêmement modulable pour adapter les réseaux à entraîner aux spécificités des *datasets* qu'on lui fournit et nécessite que les annotations soient au format Json « COCO » (*Common Object in COntext*). Le logiciel d'annotations utilisé est « Coco annotator », qui est adapté à la génération de ce type de format. Ce type d'algorithme sera appliqué à deux cas d'intérêts industriels où la détection d'objets présents sur des images de wafer est critique pour le suivi de *process*.

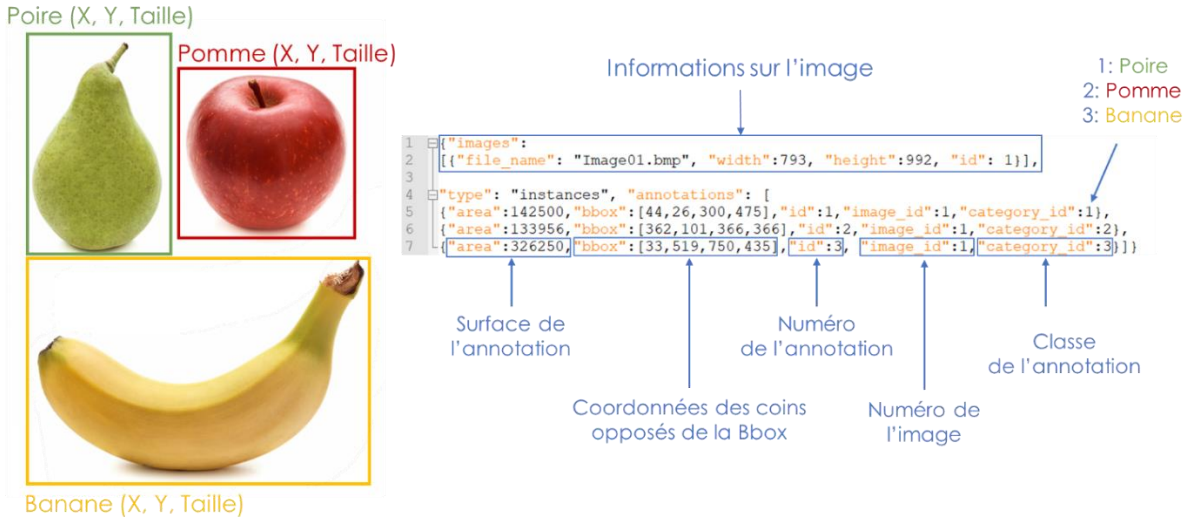

Figure 94 - Image annotée (gauche) et format du fichier d'annotation généré (format COCO), contenant toutes les informations nécessaires au RCNN pour s'entraîner à détecter et classifier les objets présents dans l'image.

2. Évaluation sur le cas d'intérêt des images de photoluminescence

Présentation du cas d'intérêt

Le premier cas d'intérêt est lié à la technique de MPL, utilisée entre autres dans les technologies imageurs pour détecter la présence de contaminants et de dislocations dans le silicium. En effet, comme expliqué dans le chapitre I, l'équipement de MPL est équipé d'un filtre passe-bande, qui peut être réglé sur la fréquence de photoluminescence du silicium et ainsi, pouvoir observer les chutes de ce signal bande-à-bande (BAB) à l'échelle du wafer. Ces pertes peuvent être dues à de multiples raisons, mais sont fondamentalement expliquées par l'apparition d'une bande « défaut » dans le diagramme de bandes, entraînant des recombinaisons à des plus faibles énergies (Figure 95), radiatives ou non.

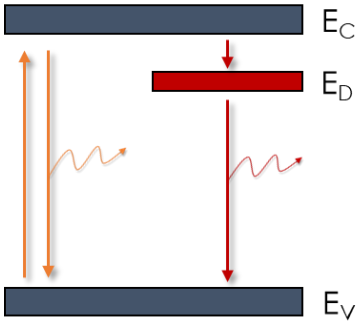

Figure 95 - Diagramme d'énergie montrant que l'apparition d'une bande « défaut » (E_D) entre la bande de conduction (E_C) et de valence (E_V) entraînent des recombinaisons à énergie réduite, et donc une réduction du signal BAB.

Ces pertes sont caractérisées sur les images obtenues par MPL (acquisition BAB) par des taches sombres. L'approche traditionnelle afin de fournir des résultats quantitatifs pour ce type d'image consisterait à déterminer le signal moyen de la plaque, ainsi que son écart-type, et à définir un seuil, à partir duquel la plaque serait en dehors des spécificités. Cette étude restitue alors très peu de l'information collectée sur la plaque, et plus spécifiquement ne permet pas la localisation de ces dérives sur la plaque. Afin de maximiser l'utilisation de ces données, une approche utilisant un RCNN pour détecter et localiser ces taches sombres a été mise en place. Il est important de noter que due à l'extrême sensibilité de cette technique aux défauts présents dans le volume du substrat, les images récoltées par la MPL possèdent toujours des taches sombres, non critiques pour le produit (ex. zone de contact face arrière avec les supports). La différentiation entre les taches « atypiques » et les taches communes est donc cruciale. Une étape critique dans le cycle de fabrication des technologies imageur a ainsi été choisie pour notre étude, et un large nombre d'images MPL a donc été récolté.

Identification des signatures atypiques

La première étape de notre approche est d'identifier les signatures communes, régulièrement présentes sur les images de MPL. Un large nombre d'images ont été superposées et deux classes de taches communes furent identifiées : *triangle interne*, et *triangle externe* (Figure 96). La signature *triangle externe* est composée de 3 points qui peuvent être localisés à deux positions/configurations en bord de wafer, tandis que la signature *triangle interne* est plus proche du centre et « tourne » autour de celui-ci avec un rayon fixe. Le traitement RCNN devra donc être capable de classifier ces signatures comme communes, et toute autre tache devra être identifiée comme atypique, entraînant la revue du wafer affecté par la dérive.

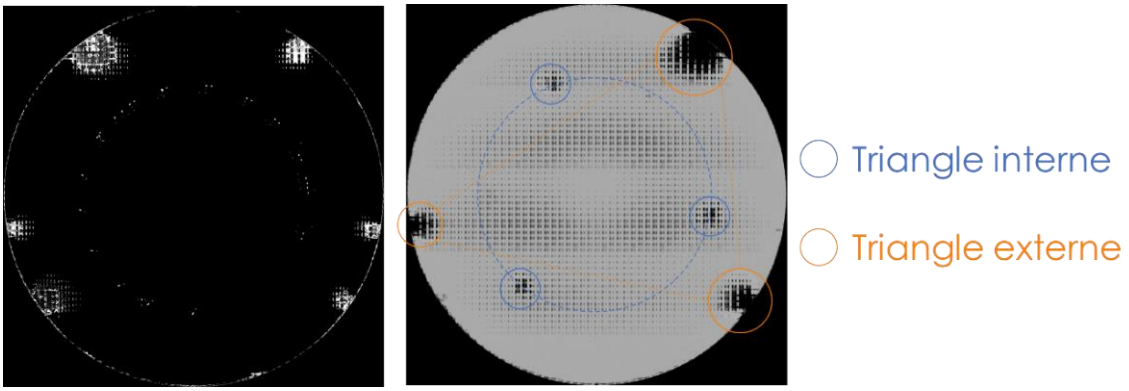

Figure 96 - Image stackée des images MPL montrant deux classes de taches communes (gauche), identifiées sur une image typique (droite).

Le RCNN en lui-même n'est pas capable d'utiliser l'information de la localisation absolue des objets dans l'image pour la classification, il est donc nécessaire, pour ce cas d'intérêt, de mettre en place un protocole adapté. L'objectif du RCNN est de détecter toutes les taches présentes, et un second algorithme géométrique étudiera quant à lui la localisation et la répartition spatiale des taches pour les classifier individuellement. Ce second algorithme sera mis en place avec un script Python (Figure 97).

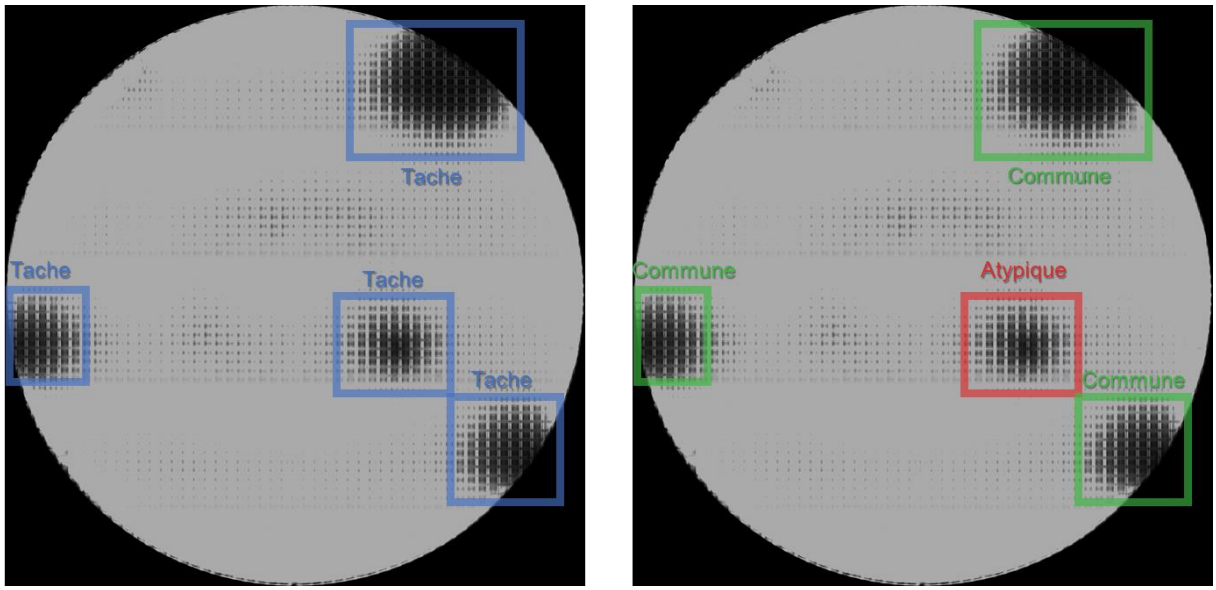

Figure 97 - Illustration du protocole de détection des taches par le RCNN (gauche) et classification de ces dernières par le script Python (droite).

Entraînement du RCNN

Nous avons généré un *dataset* d'entraînement/validation à partir d'images annotées composé de 275 images, contenant 1420 annotations de tache. Afin d'évaluer les performances de l'algorithme, 670 annotations supplémentaires issues de 142 images ont été générées pour le *dataset* de test. Les courbes d'entraînements obtenues sont présentées en Figure 98.

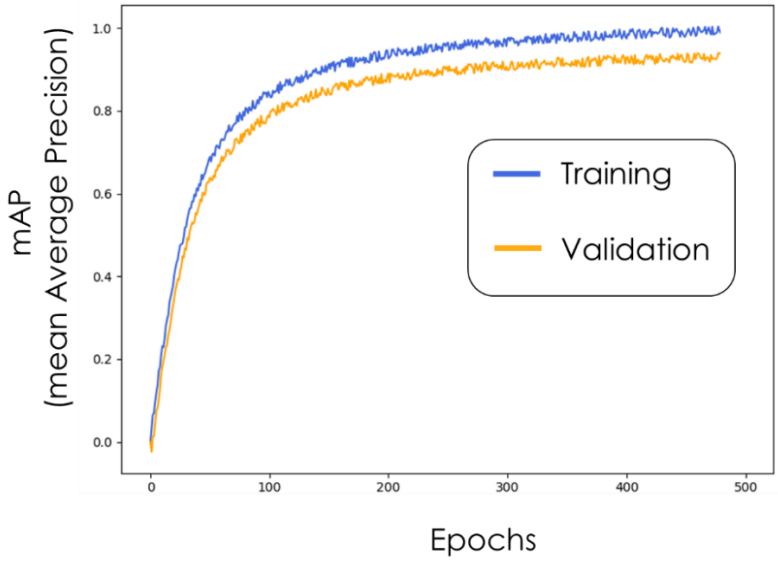

Figure 98 - Courbes d'entraînement du RCNN sur les données MPL, représentées par la moyenne de la précision moyenne (mAP), pour les deux datasets de training (entraînement) et validation.

La grandeur représentée est ici la « moyenne de la précision moyenne » (mAP – *mean Average Precision*), déterminée par la précision moyenne obtenue pour chaque classe entraînée. Comme dans le cas des RCNN, les objets détectés sont localisés via des Bbox, ces dernières sont comparées à celles définies lors de l'annotation, et leur surface en commun permet de définir une métrique nommée *IoU* (*Intersection over Union* – Figure 99). La mAP est alors déterminée pour un seuil d'IoU précis, par exemple, dans notre cas, entre 0,5 et 0,95 (Soit 5% à 95%). Après 482 epochs, notre réseau converge vers une mAP de 0,97 sur l'entraînement, arrêté après 30 epochs sans progrès sur le *dataset* de validation.

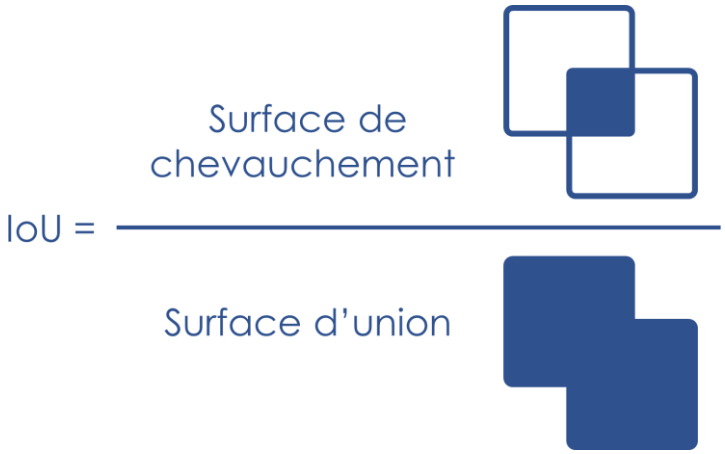

Figure 99 - Définition de l'IoU, représentant le ratio entre la surface de chevauchement (Intersection) sur la surface d'union, permettant d'évaluer la qualité de segmentation du pré-réseau.

Résultats sur *dataset* de test

Les résultats obtenus avec le RCNN entraîné sur le *dataset* de test sont indiqués dans le Tableau 12, sous la forme d'une matrice de confusion, bien qu'ici, la classification commune/atypique soit faite par un algorithme à part. Le résultat est que, sur les 670 annotations, le RCNN a été capable d'en détecter 659 (98.3%), ne manquant que quelques signatures communes très faibles. Un exemple de résultat d'inférence est présenté en Figure 100. C'est tout à fait remarquable car pour la plupart des signatures atypiques, une approche standard de métrologie, contrôlant les signaux moyens de photoluminescence et son écart-type n'aurait pas été capable de détecter la dérive, ces anomalies n'induisant pas une variation suffisamment large du signal global de photoluminescence de la plaque.

Tableau 12 - Matrice de confusion obtenue sur le dataset MPL de test.

MPL		Prédites: Commune	Prédites: Atypique	Prédites: Non-détectées
Vraies	Commune	542	0	11
	Atypique	0	117	0

Au vu des performances obtenues de notre approche à cette dérive, une collaboration avec Semilab, équipementier industriel qui a fourni le MPL, est en cours pour mettre en place ce traitement,

basé sur le RCNN, en sortie de mesure, afin de pouvoir traiter automatiquement les images obtenues et permettre un suivi de procédé robuste.

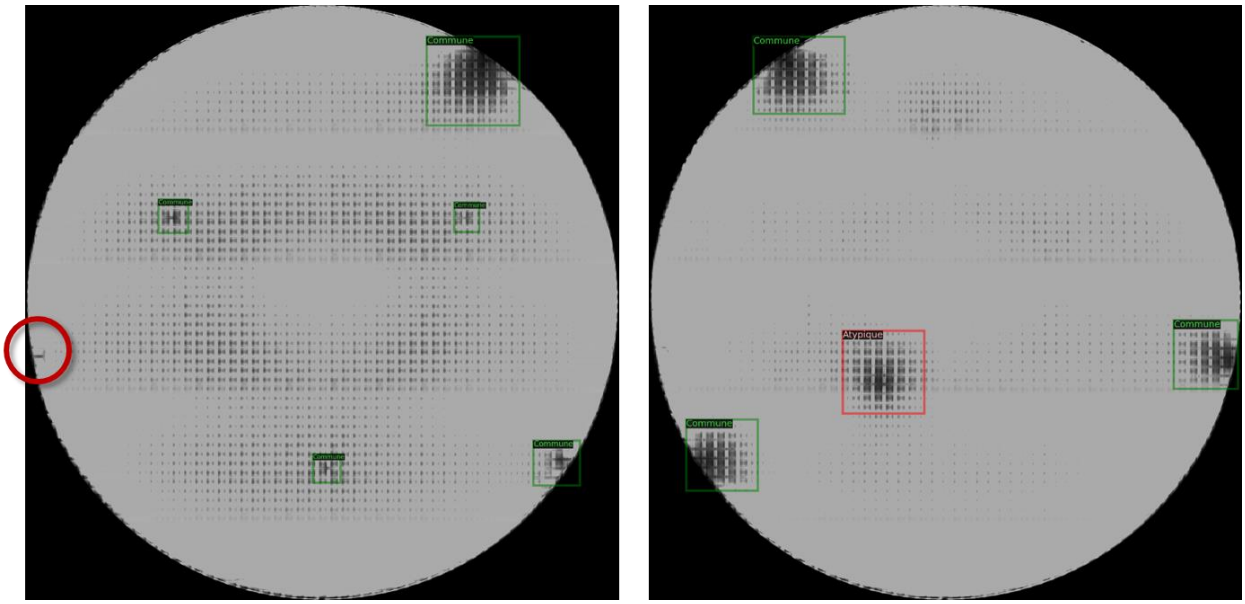

Figure 100 - Exemples d'inférences réalisées sur le dataset de test, avec non-détection d'une tache commune (gauche) et bonne détection d'un tache atypique (droite).

3. Évaluation sur le cas d'intérêt des images de microscopie acoustique

Présentation du cas d'intérêt

Afin de contrôler le bon collage des plaques dans le cadre de l'intégration 3D [17], la microscopie acoustique (SAM – *Scanning Acoustic Microscopy*) est régulièrement utilisée pour sa haute résolution et sa grande sensibilité pour contrôler cette étape importante. Bien que cette technique soit capable de rendre visible la plupart des « *voids* » (poches de vide/air entre les plaques), l'algorithme de traitement automatique de l'équipement ne les détecte pas tous. De plus, afin de réaliser un *process control* rigoureux, ces défauts doivent être classifiés selon leur forme et leur localisation sur le wafer, ce qui n'est pas possible avec les traitements mathématiques d'images actuellement en place sur l'équipement. De plus, ces traitements sont extrêmement sensibles aux conditions d'acquisitions (luminosité/contraste) et une utilisation d'un algorithme RCNN sur ce sujet serait alors bénéfique, afin d'améliorer les capacités de détection, et de classification, et ainsi d'obtenir une solution de suivi de procédé permettant de résoudre ces dérives, soit par *reworking* (réélaboration de l'étape de collage), soit par encrage des puces concernées. Dans ce dernier cas, la plaque continue son *process*, à l'exception des puces encrées qui sont ignorées.

Contrairement au cas d'intérêt précédent, le RCNN sera dans ce cas-ci directement utilisé pour classifier les voids selon 6 classes : Particule incluse, Design, Krone, Scribe, Bord, et Starry Night (Figure 101). Il est important de noter que ce type d'algorithme est normalement prévu pour détecter une dizaine d'objets environ par image, alors que les images SAM (6100 x 6100 pixels) peuvent en contenir plusieurs centaines, d'une dizaine de pixel chacune. Une optimisation de la structure, ainsi que des hyperparamètres du RCNN a donc dû être réalisée pour obtenir des résultats satisfaisants. Notamment le nombre de Bbox proposées par le RPN, qui est par défaut à 5000 (pour une dizaine d'objets par image), et qui a donc dû être drastiquement augmenté à 35 000. Mais aussi la taille des

Bbox proposées, étant donné que nous recherchons des petits objets dans une grande image. Toute ces modifications sont aisément implémentables dans le *framework* Detectron2 via des fonctions de configuration du modèle à entraîner.

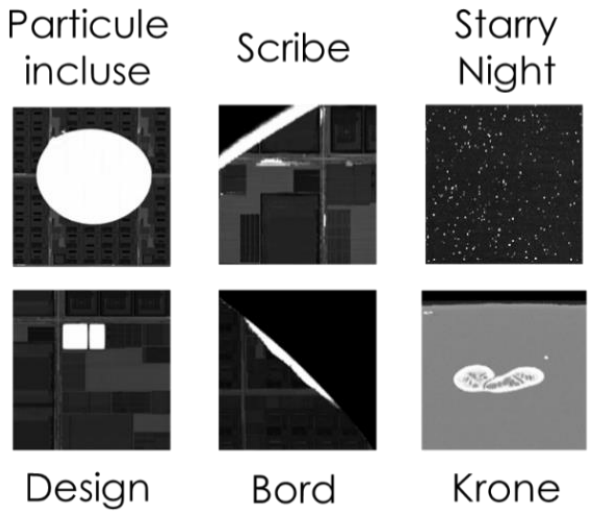

Figure 101 - Exemple de chaque classe de voids, caractérisées par leur forme/localisation, qui seront à détecter et classifier par le RCNN.

Entraînement du RCNN

Comme dit précédemment, chaque image SAM possède un large nombre de voids, et le *dataset* d'entraînement devra donc être suffisamment riche pour que le RCNN soit capable de tous les détecter. Ainsi, 155 images, contenant 12 644 annotations ont été traitées manuellement pour générer le *dataset* d'entraînement/validation. Deux exemples sont présentés en Figure 102. Enfin, un *dataset* de test ayant pour but l'évaluation des performances du réseau a été créé à partir de 2287 annotations, contenues dans 32 images.

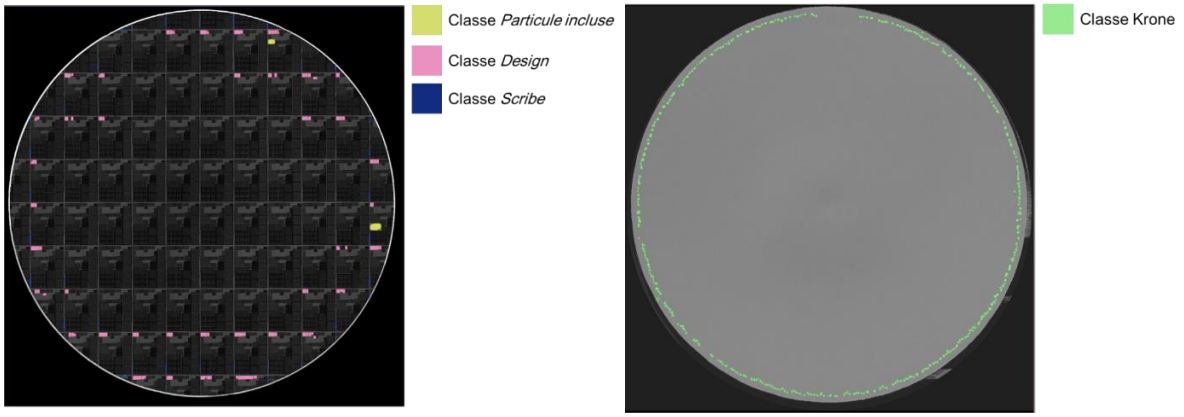

Figure 102 - Exemple d'images SAM annotées pour l'entraînement du RCNN. Chacune de ces cartographies possède en moyenne plusieurs dizaines (voir une centaine) de défauts à précisement identifier manuellement.

Ces *datasets* ont finalement été utilisés pour réaliser le *transfer learning* de RetinaNet à notre cas d'intérêt, et les courbes d'entraînement sont présentées en Figure 103. Après 1172 epochs, le RCNN obtient une mAP de 0,94 sur le *dataset* d'entraînement, arrêté après 75 epochs sans progrès sur le *dataset* de validation (augmenter par rapport aux autres évaluations pour prendre en considération la complexité accrue de ce cas d'intérêt).

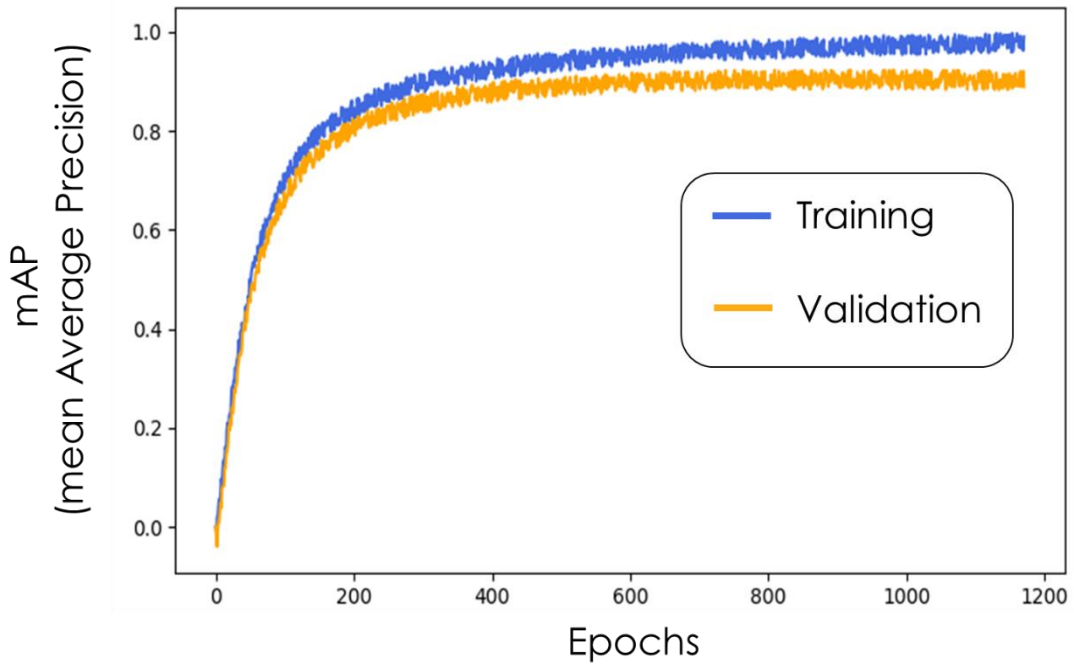

Figure 103 - Courbes d'entraînement du RCNN sur les données SAM, représentées par la moyenne de la précision moyenne (mAP), pour les deux datasets d'entraînement (training) et validation.

Résultats sur *dataset* de test

Après entraînement, le RCNN est utilisé sur le *dataset* de test et les résultats sont présentés dans le Tableau 13. Ce réseau est ainsi capable de détecter 96.19% (2200/2287) des voids présents sur les images SAM, en les classifiant correctement dans 99.09% (2180/2200) des cas. Cela apparaît remarquable si l'on considère le fait que les classes ne paraissent pas significativement différentes les unes des autres (petites taches blanches). Les performances globales du RCNN sont explicables par le très grand nombre de défauts par image, qui permet d'obtenir une très bonne qualité de prédiction de classification, mais réduit les performances de détection. Des exemples d'inférences réalisées sur le *dataset* de test sont présentées en Figure 104 et Figure 105.

Tableau 13 - Matrice de confusion obtenue sur le dataset de test de SAM, contenant 32 images et 2287 annotations.

		Prédites						
	SAM	Particule incluse	Design	Bord	Scribe	Starry Night	Krone	Non-détectée
Vraies	Particule incluse	96	0	0	0	0	0	0
	Design	5	564	0	14	0	0	4
	Bord	0	0	73	0	0	0	0
	Scribe	0	1	0	781	0	0	25
	Starry Night	0	0	0	0	412	0	51
	Krone	0	0	0	0	0	254	7

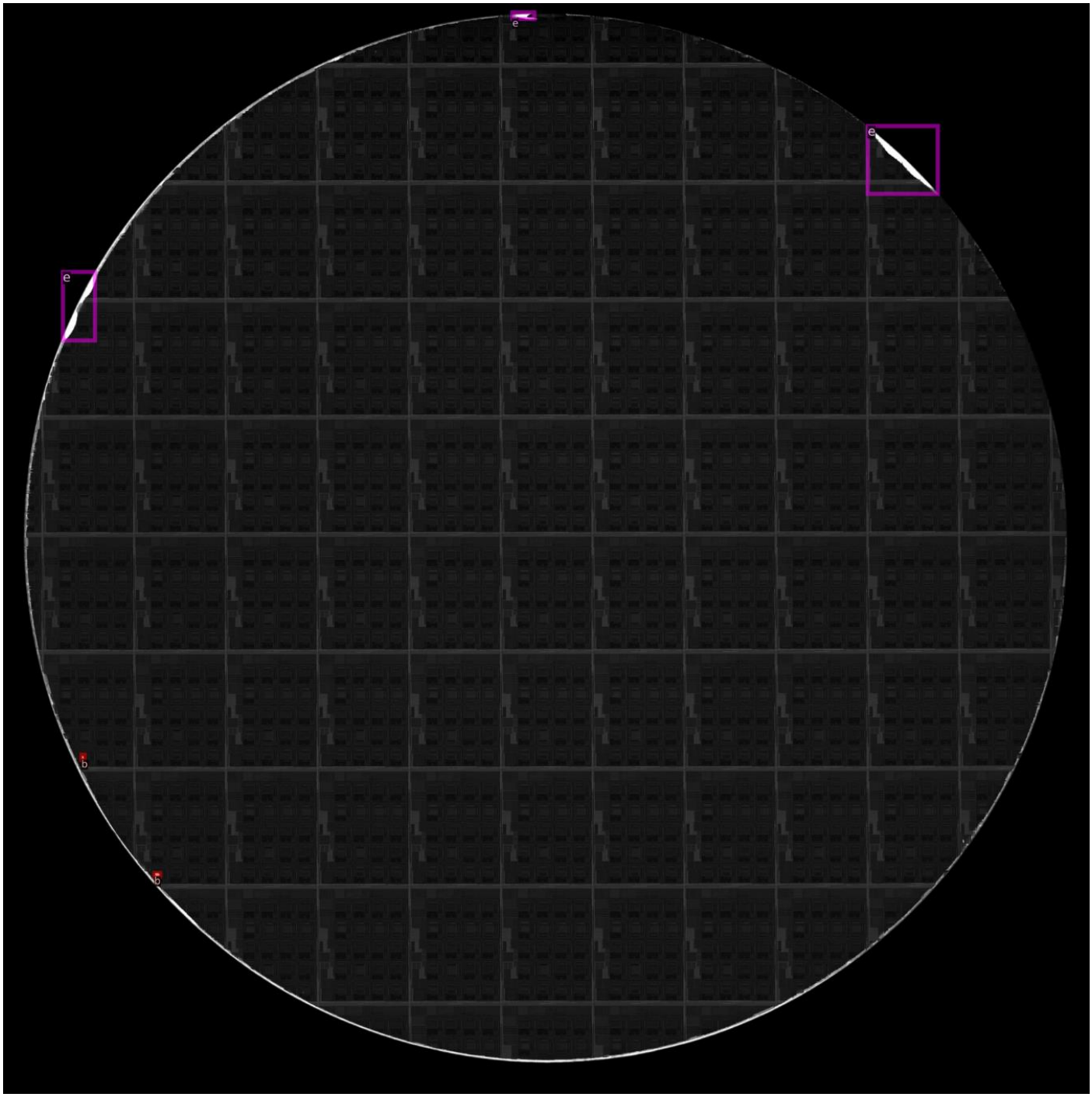

Figure 104 - Exemple d'inférence réalisée sur une image obtenue par SAM, montrant la détection et la classification des voids présentes (« b » en rouge : Design, « e » en violet : Bord).

Ces performances vont bien au-delà de celles permises par l'équipement, uniquement capable de détecter, sans classifier, une faible partie des voids présents. De ce fait, une mise en place industrielle de cette approche est en cours, afin de pouvoir suivre ces étapes de *bonding* automatiquement, et avec une haute qualité de détection et classification des défauts de collages présents sur les images issues du microscope acoustique.

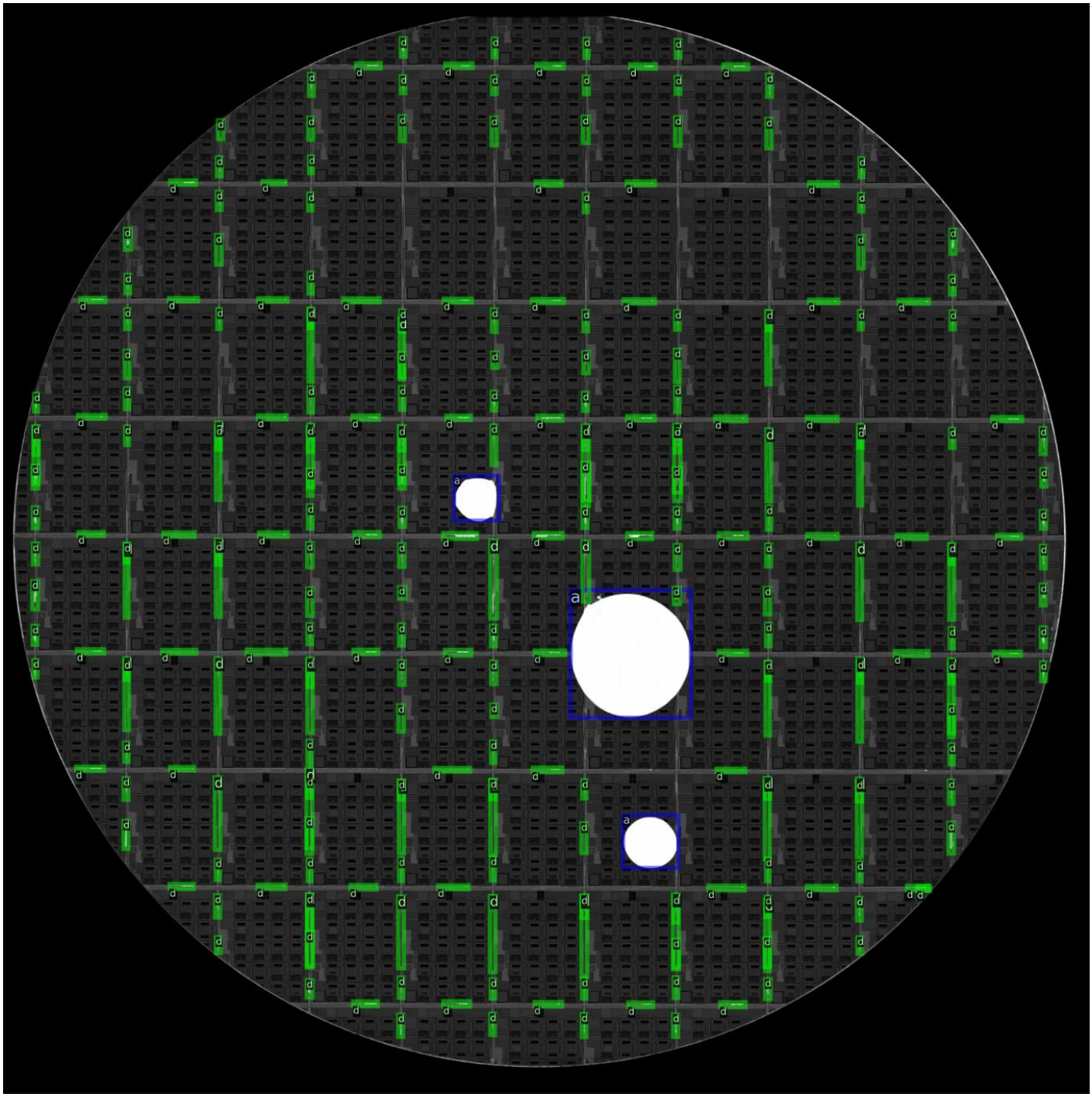

Figure 105 - Exemple d'inférence réalisée sur une image obtenue par SAM, montrant la détection et la classification des voids présentes (« a » en bleu : Particule incluse, « d » en vert : Scribe).

Détection de nouveaux défauts

L'apparition de nouveaux types de défauts est courante dans l'industrie des semi-conducteurs, et dont la détection est particulièrement critique, nous avons alors évalué la capacité du notre approche à répondre à cette attente. Pour ce faire, la méthodologie mise en place est basée sur le fait de s'appuyer sur l'excellente qualité de classification des RCNN. En effet, étant donné que les erreurs de classification sont extrêmement rares (inférieures à 1% dans notre approche, et largement

réductibles avec un entraînement plus conséquent), si plusieurs réseaux de neurones entraînés sur le même *dataset* classifie un défaut différemment, il est donc fort probable qu'il n'appartienne à aucune des classes déjà existantes dans le *dataset*.

Afin d'évaluer cette hypothèse et la capacité de ce type d'approche à détecter des nouveaux défauts, nous avons entraînés 4 RCNN à l'aide des *datasets* précèdent, mais en supprimant toutes les annotations et images contenant la classe « Krone ». Ces images sont traitées en parallèle par les réseaux entraînés, et les résultats des inférences sont présentés en Figure 106.

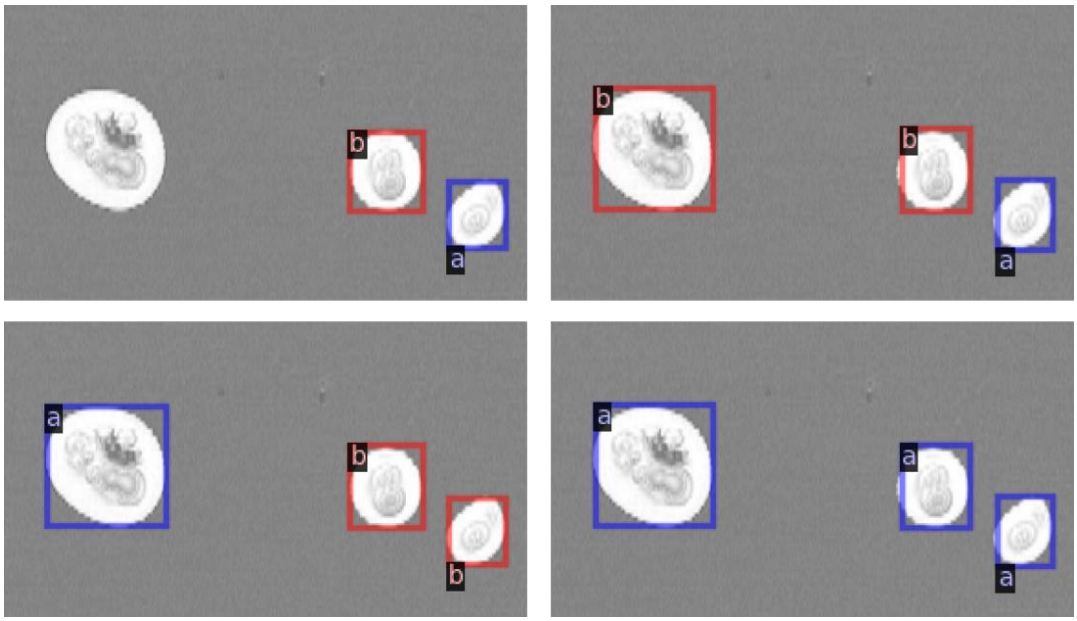

Figure 106 – Exemple d'inférences réalisées par les 4 réseaux de neurones entraînés sans la classe « Krone », et qui sont alors détectées, mais incorrectement identifiées comme d'autres classes (« a » en bleu : Particule incluse, « b » en rouge : Design).

Notre hypothèse semble alors validée, car les « nouveaux » défauts (Krone) sont la plupart du temps détectés mais classifiés différemment par chacun des réseaux de neurones. Ce type d'approche à plusieurs réseaux en parallèle peut donc être utilisé pour, non seulement renforcer la robustesse du suivi de procédé, mais aussi pour détecter de nouvelles signatures.

4. Conclusion

L'utilisation de réseaux de neurones convolutifs régionaux permet ainsi, quand appliquée à des images obtenues par des équipements de suivi de procédé, contenant plusieurs signatures caractéristiques de dérives de procédés, de classifier automatiquement ces dernières. Ces algorithmes pourraient alors considérablement améliorer le suivi de procédé actuellement en place, en apportant une capacité de détection et de classification robuste et améliorable aisément en utilisant les prédictions générées sur des nouvelles images pour renforcer l'entraînement. De plus, ils sont aussi capables dans certains cas de détecter l'apparition de nouveaux défauts, courant dans l'industrie de la microélectronique et très dangereux car souvent détectés trop tard, au moment des tests de rendement des plaques.

V. Conclusion

Les équipements de suivi de procédé industriels générant des images sont une source de données extrêmement riche dont la plupart n'est pas valorisée suffisamment. L'utilisation d'outils de traitement d'images avancés permet donc dans de multiples cas d'améliorer le suivi de procédé de fabrication actuellement en place. En effet, les approches métrospection évaluées dans ce chapitre permettent de répondre à des besoins forts, notamment en apportant des capacités de détection ou de classification absentes dans les approches actuellement utilisées.

Références

[1] R. O. Duda et P. E. Hart, «Use of the Hough transformation to detect lines and curves in pictures,» *Association for Computing Machinery,* vol. 15, p. 11–15, 1972.

[2] J. Canny, «A Computational Approach To Edge Detection,» *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 8, pp. 679-698, 1986.

[3] C. A. Schneider, W. S. Rasband et K. W. Eliceiri, «NIH Image to ImageJ: 25 years of image analysis,» *Nature Methods,* n° %19(7), p. 671–675, 2012.

[4] L. I. Rudin, S. Osher et E. Fatemi, «Nonlinear total variation based noise removal algorithms,» *Physica D. 60,* n° %11–4, p. 259–268, 1992.

[5] I. Mica, R. Duru, J. Frascaroli et P. Bellanger, «Photoluminescence imaging for slip line detection and characterization in silicon substrates,» *Proc. Advanced Semiconductor Manufacturing Conference,* 2022.

[6] K. Lee, S. Cheon et C. O. Kim, «A Convolutional Neural Network for Fault Classification and Diagnosis in Semiconductor Manufacturing Processes,» *IEEE Transactions on Semiconductor Manufacturing,* vol. 30, n° %12, pp. 135-142, 2017.

[7] T. Nakazawa et D. Kulkarni, «Wafer Map Defect Pattern Classification and Image,» *IEEE Transactions on Semiconductor Manufacturing,* vol. 31, n° %12, pp. 309-314, 2018.

[8] X. Glorot, A. Bordes et Y. Bengio, «Deep Sparse Rectifier Neural Networks,» *AISTATS,* 2011.

[9] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. Berg et L. Fei-Fei, «ImageNet Large Scale Visual Recognition Challenge,» *International Journal of Computer Vision,* 2015.

[10] K. He, X. Xiangyu Zhang, S. Ren et J. Sun, «Deep Residual Learning for Image Recognition,» *Computer Vision and Pattern Recognition (CVPR),* pp. 770-778, 2016.

[11] R. Bin Issa, M. Das, M. S. Rahman, M. Barua, M. K. Rhaman, K. S. N. Ripon et M. G. R. Alam, «Double deep Q-learning and faster R-Cnn-based autonomous vehicle navigation and obstacle avoidance in dynamic environment,» *Sensors,* vol. 21, n° %14, p. 1468, 2021.

[12] J. Zhang, H. Hu, S. Chen, Y. Huang et Q. Guan, «Cancer cells detection in phase-contrast microscopy images based on faster R-CNN,» *International symposium on computational intelligence and design (ISCID),* vol. 1, pp. 363-367, 2016 .

[13] S. Ahmad, N. Enshaei, F. Naderkhani et A. Awasthi, «Integrated Deep Learning and Statistical Process Control for Online Monitoring of Manufacturing Processes,» *IEEE International Conference on Prognostics and Health Management (ICPHM),* pp. 1-6, 2020.

[14] J. Han et S. Hong, «A New Backbone Network for Instance Segmentation: Application on a Semiconductor Process Inspection,» *IEEE Access,* vol. 8, pp. 218110-218121, 2020.

[15] T. -Y. Lin, P. Goyal, R. Girshick, K. He et P. Dollár, «Focal Loss for Dense Object Detection,» *IEEE International Conference on Computer Vision (ICCV),* pp. 2999-3007, 2017.

[16] Y. Wu, A. Kirillov, M. F., W.-Y. Lo et R. Girshick, «Detectron2,» 2019. [En ligne]. Available: https://github.com/facebookresearch/detectron2.

[17] M. Sadaka et D. C. L. , «Building blocks for wafer-level 3D integration,» *Solid State Technology,* vol. vol. 52, n° %110, p. 20+, 2009.

Chapitre IV – Imagerie ellipsométrique

I. Introduction

Nous avons vu dans les précédents chapitres que les techniques permettant d'obtenir des images à large échelle des plaques permettent, grâce à des traitements intelligents, de réaliser dans l'industrie des semi-conducteurs, un contrôle du procédé de fabrication robuste et réactifs aux évolutions rapides induites par la R&D. L'utilisation d'équipements de métrologie très sensibles permet ainsi une sensibilité accrue aux déviations, tandis que l'utilisation d'algorithmes IA évitent le développement fastidieux de modèles physiques rigoureux. Actuellement, ces techniques optiques « *full wafer* » reposent principalement sur l'interférométrie et la réflectométrie, qui, bien que sensibles en général, ne le sont en pratique que pour les couches de surface. Il serait alors intéressant d'obtenir de telles images à large échelle mais en utilisant l'ellipsométrie, celle-ci étant une technique sensible à la fois à la topographie de surface, mais aussi à la composition des matériaux des couches présentes enterrées.

Dans l'approche traditionnelle en métrologie, l'ellipsométrie spectroscopie (SE) est utilisée pour la mesure d'épaisseur et d'indices optiques des couches composants un empilement étudié en modélisant la réponse optique de la zone mesurée à l'aide des lois de Fresnel. Cette méthode requiert la connaissance détaillée du nombre de couches et des matériaux constituant l'empilement, afin de pouvoir générer un modèle rigoureux. En simulant la réponse ellipsométrique de cette structure théorique, dans le cadre de la résolution du problème inverse, il est possible d'extraire les paramètres d'intérêt (épaisseurs et/ou indices optiques des matériaux). Comme beaucoup d'approches reposent sur la modélisation, elles peuvent être très consommatrices en ressources et nécessitent un lourd travail de pré-caractérisation et d'analyse croisée, comme nous l'avons montré dans le chapitre 2 avec la modélisation RCWA. Elles nécessitent de plus la réalisation de cibles de métrologie spécifiques placées dans les lignes de découpe de la plaque. Ces empilements présents dans les cibles sont de plus simplifiés, tout en restant représentatif des variations subies par le produit.

Développer une approche non-modélisée permettrait d'exploiter l'extrême sensibilité intrinsèque de l'ellipsométrie aux variations d'épaisseurs (~ quelques Angström), aux variations d'indices optiques (~ 0.01 variation de n), ainsi qu'aux variations de pattern (formes, *dishing*, etc.). Cela permet alors d'ouvrir la voie à des acquisitions directes dans le produit où la modélisation rigoureuse est impossible au vu de la complexité des structures réelles présentes dans les produits. Une nouvelle approche consiste donc à imager une zone du wafer à l'aide d'un ellipsomètre. A ce jour, il existe dans l'industrie des solutions d'imagerie ellipsométriques [1] mais elles se concentrent sur l'acquisition de zones de faibles surfaces et ont pour but de mesurer les épaisseurs et indices optiques d'empilements uniformes à l'aide d'une modélisation mathématique. Notre objectif était de développer une technique d'imagerie ellipsométrique à large échelle, sans développement de modèle, où les données brutes récoltées à la volée par l'équipement seraient utilisées pour générer les images. L'évaluation de cette approche sera donc appuyée sur 3 cas d'études de dérives de procédés impactant chacun des propriétés différentes des wafers.

II. Présentation de l'équipement et protocole expérimental de l'imagerie ellipsométrique

1. Ellipsomètre Impact (LTM)

Les équipements industriels couramment utilisés pour la métrologie ellipsométrique sont complètement inadaptés à des acquisitions à large échelle. En effet, ils fournissent des mesures ponctuelles (~30x30 µm) sur le wafer, et une cartographie sur la surface entière avec une haute résolution serait bien trop longue à réaliser. Ils sont dédiés à la mesure de structures spécifiques sur le wafer, suffisamment petites pour être placées dans les lignes de découpes, ce qui explique leur taille de spot très faible. Les équipements académiques, quant à eux, ont pour objectif de déterminer très précisément les épaisseurs et indices optiques des matériaux étudiés, et possède pour cela des spots bien plus larges (> 100 µm).

Parmi les différentes technologies ellipsométriques, l'ellipsométrie à modulation de phase est la plus adapté à l'étude académique car son gain de mesure, par principe, est indépendant de la longueur d'onde utilisée, grâce à l'utilisation d'un photomultiplicateur. De plus, les acquisitions spectroscopiques sont réglables finement à l'aide d'un monochromateur. Cela induit malheureusement des temps d'acquisitions plus long (> 2 minutes pour 100 longueurs d'onde), car elles sont mesurées séquentiellement en sortie de monochromateur. Dans le cas où une seule longueur d'onde est sélectionnée, des mesures cinétiques sont possibles, permisent par le rapide taux de mesure (< 25ms par période) du photomultiplicateur, ainsi que la modulation à haute fréquence crée par le modulateur photoélastique (~ 50kHz). Le schéma fonctionnel de ce type d'ellipsomètre est présenté dans le chapitre 1, et en détail dans [2]. Les données brutes récoltées par les ellipsomètres à modulation de phase (en configuration II – voir Chapitre 1) sont « Is » et « Ic », angles ellipsométriques aux valeurs contenues entre -1 et 1, et qui sont liés aux angles ellipsométriques Δ et Ψ par les équations suivantes :

$$Is = sin\,(2\Psi)\,sin\,(\Delta)$$

$$Ic = sin\,(2\Psi)\,cos\,(\Delta)$$

L'équipement utilisé durant ma thèse est un ellipsomètre à modulation de phase académique qui est un prototype de recherche, installé en salle blanche du LETI et financé par l'ANR au travers de l'EQUIPEX IMPACT. Cet outil est un UVISEL deuxième génération d'Horiba (UVISEL 2) permettant la réalisation de mesures sur des wafers de 300 mm de diamètre. Il couvre une très large gamme de longueurs d'onde (150 nm – 2 µm) grâce à ses deux sources lumineuses : Xenon et Deuterium.

Figure 107 - Plateforme expérimentale Impact avec ellipsomètre (gauche) et une chambre spectromètre Raman/Photoluminescence (droite).

De nombreuses tailles de spot peuvent être sélectionnées pour réaliser les acquisitions, mais durant ma thèse, les principales utilisées étaient les deux extrêmes (Figure 108) : La plus petite (300 x 300 µm, et carrée) et la plus large (8 x 5 mm, et elliptique).

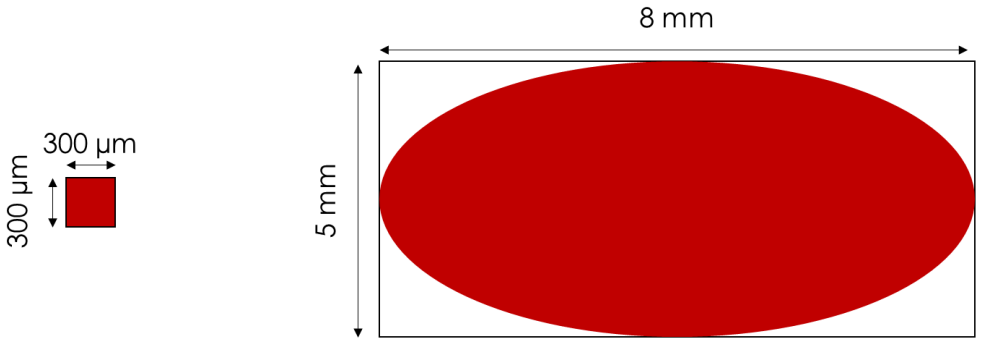

Figure 108 - Exemples de tailles de spot (échelle non-respectée) utilisables pour les acquisitions avec l'ellipsomètre Impact.

Cet équipement est dédié à l'analyse et la caractérisation de matériaux avancés, déposés sur des plaques de 300 mm, le chargement des plaques étant automatiquement géré par une chambre de transfert placée sous vide pour éviter toute contamination de surface. Il est capable de réaliser des acquisitions cinétiques, permettant des mesures à des fréquences jusqu'à 100 Hz. Cependant, pour réaliser ces mesures à la volée, une longueur d'onde unique doit être sélectionnée

2. Protocole d'acquisition et de génération des images

a. Sélection de la longueur d'onde

Comme indiqué plus tôt, l'imagerie ellipsométrique ne peut se faire qu'à une longueur d'onde sur cette plateforme, et le choix de cette dernière est déterminante pour s'assurer que la réponse optique est sensible à la dérive étudiée. Pour faire ce choix, deux protocoles ont été mis en place : Un basique, puis un avancé, basé sur l'utilisation de substrats virtuels.

Protocole basique :

La première étape de ce protocole est de réaliser une acquisition spectroscopique (*Is* et *Ic* pour une large gamme de longueur d'onde) dans la zone d'intérêt de l'étude (bord de wafer, ou puce typique) avec un spot large pour en obtenir la signature optique moyenne. À partir du spectre obtenu,

on détermine alors la longueur d'onde qui sera utilisée pour l'acquisition à large échelle en repérant les points d'intersections des deux courbes *Is* et *Ic* et en sélectionnant une longueur d'onde possédant la plus grande pente locale. Cette dernière est déterminée par la dérivée maximale calculée pour les deux signaux. Le raisonnement derrière ce choix repose sur l'hypothèse que les deux signaux mesurés vont, dans le cas d'une faible variation de propriété optique du matériau étudié, subir une variation forte, normalisée par la valeur à l'intersection des courbes. Un exemple de ce protocole de sélection est présenté en Figure 109. Cette approche n'étant basée sur aucune étude théorique, elle ne peut donc pas être considérée comme robuste, et n'est adaptée qu'aux problèmes « simples ». Une autre approche a été élaborée pour améliorer ce protocole de sélection de longueur d'onde.

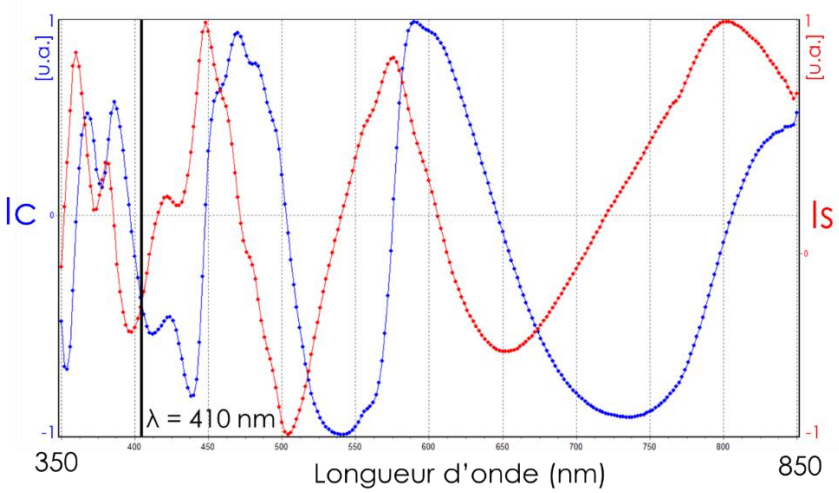

Figure 109 - Exemple de sélection de longueur d'onde en utilisant le protocole basique. Plusieurs intersections sont présentes sur ce spectre, mais celle à 410 nm possède la plus forte pente (déterminé par dérivation) et est donc sélectionnée.

Protocole avancé :

Afin d'augmenter la fiabilité du protocole de sélection de longueur d'onde, une alternative basée sur les substrats virtuels a alors été explorée. La première étape est alors d'obtenir un spectre ellipsométrique sur la zone d'intérêt, comme pour l'approche basique, mais cette fois à deux étapes du process : avant et après l'étape étudiée. Cela permet d'utiliser le premier comme substrat virtuel, et ainsi de créer une couche virtuelle permettant d'obtenir le second spectre, cette couche représentant alors la variation optique induite par l'étape. On réalise cette étape en utilisant le logiciel « DeltaPsi2 » fourni par Horiba, qui permet de générer des spectres ellipsométriques théoriques à partir des paramètres de l'échantillon mesuré(épaisseurs/indices) et d'ajuster ces spectres à ceux obtenus expérimentalement. On peut faire varier l'épaisseur et/ou la rugosité de cette couche afin d'évaluer la variation induite sur le spectre ellipsométrique, et ainsi repérer la longueur d'onde pour laquelle cette variation est la plus forte, et donc qui sera le plus sensible aux dérives process pour cette étape. De plus, comme la même zone est mesurée avant et après l'étape, aucune variation du stack présent n'est possible (empilement de couches enterrées), et tout changement de signal ellipsométrique est donc dû à la dernière étape du process. Pour les étapes de procédé très inhomogènes, où les propriétés optiques des matériaux peuvent varier fortement entre différentes régions du wafer, cette approche pourra être appliquée à plusieurs sites afin d'obtenir la longueur d'onde optimale pour l'ensemble de la plaque. Un résumé de cette approche est présenté en Figure 110.

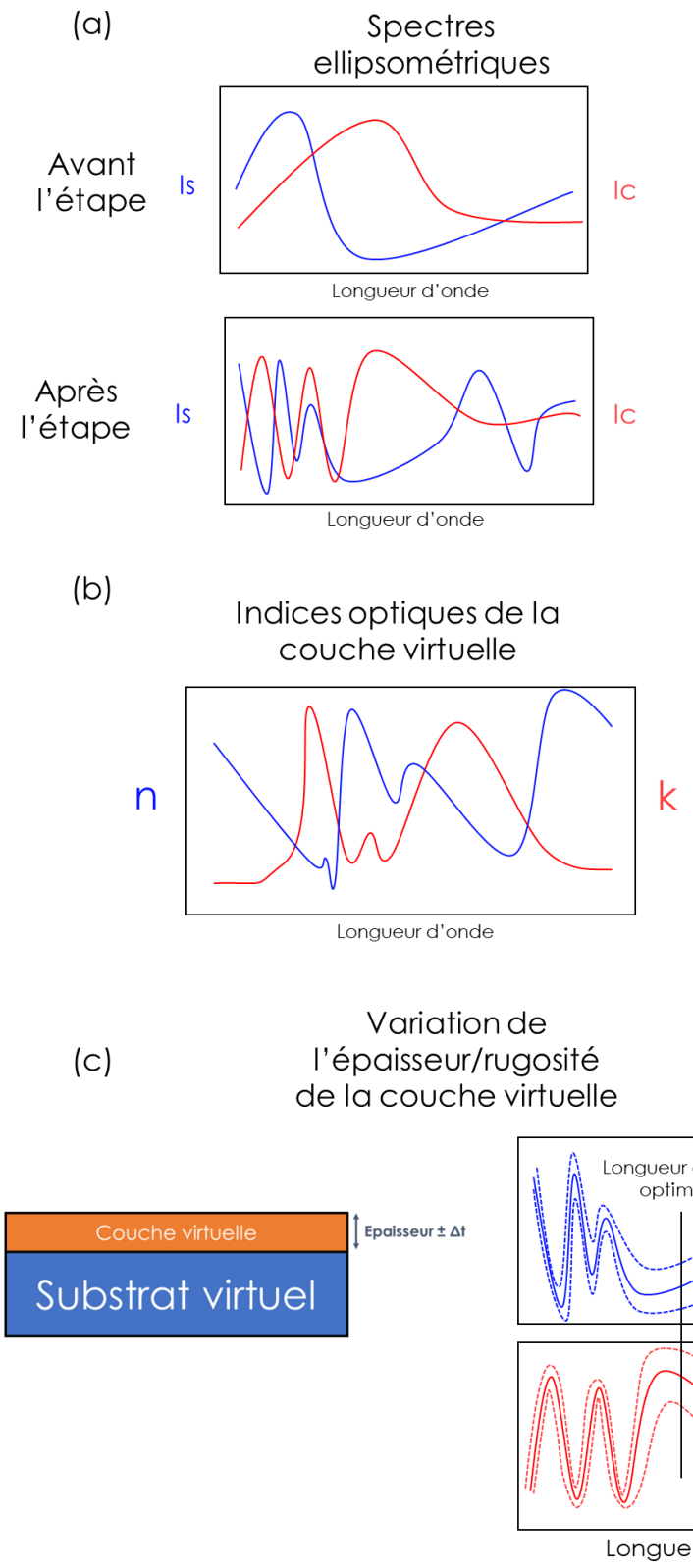

Figure 110 - Protocole avancé de sélection de longueur d'onde, avec deux scans spectroscopiques (a), création d'une couche virtuelle représentant l'étape process (b) et variation des propriétés de cette couche, avec évaluation de la variation induite (c). Le maximum de variation induite est calculée, et la longueur d'onde optimale peut alors être choisie.

b. Acquisition des données ellipsométriques

Définition des régions d'acquisitions

Une fois la longueur d'onde optimale sélectionnée, différentes approches sont possibles pour récolter les données ellipsométriques. Avant l'implémentation du mode d'acquisition à la volée (« Raster »), les mesures étaient faites point-par-point, et étaient donc beaucoup plus longues. Une grille dense de points définissait les acquisitions réalisées sur la zone d'intérêt (Figure 111), et cette approche n'était applicable qu'aux dérives induisant une signature à longue portée, et donc ne nécessitant pas une haute résolution spatiale.

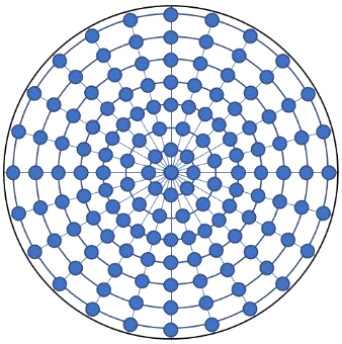

Figure 111 - Exemple de grille de points d'acquisition utilisée pour les dérives à faible fréquence spatiale.

Comme précisé plus tôt, l'ellipsomètre Impact est capable de réaliser des acquisitions à la volée pendant le déplacement de la plateforme, chaque point étant mesuré de façon cinétique. Les mesures sont enregistrées à une fréquence temporelle définie, et les coordonnées sont fournies par le servomoteur de la plateforme. Cette méthode permet de grandement réduire le temps d'acquisition, tout en augmentant considérablement la résolution des images ellipsométriques générées. Ce protocole est illustré en Figure 112.

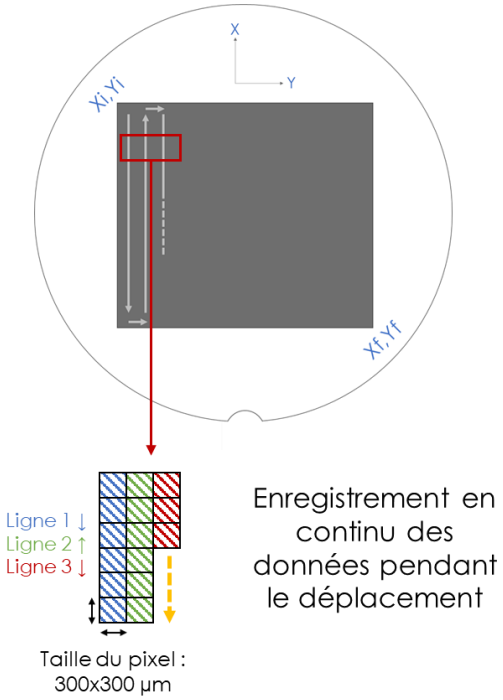

Figure 112 - Protocole d'acquisition en mode Raster, où la surface d'intérêt est scannée par un spot de 300x300µm (dont le signal définit un pixel final de l'image) par lignes successives descendantes puis montantes.

Les paramètres d'acquisition à fournir à l'équipement sont alors :

- Les coordonnées Xi, Yi du point de départ et Xf, Yf du point d'arrivée, limitant les surfaces d'acquisitions possible à des rectangles.
- L'intervalle de mesure (résolution en X) et le nombre de lignes (résolution en Y). Ces valeurs sont définies selon la surface étudiée afin d'obtenir des images de résolution correspondant à la taille du spot utilisé (300x300 µm), sachant que la résolution en X peut être augmentée sans entrainer un temps d'acquisition plus long, contrairement à celle en Y.

Mise en place du protocole de correction des aberrations :

La plateforme de l'ellipsomètre n'étant pas originellement prévue pour ce type d'étude cinétique, les moteurs contrôlant le déplacement de la platine sont uniquement optimisés pour des mouvements de point à point, et non pas pour des déplacements continus à vitesse fixée. Lors de la génération des images à partir des données obtenues par acquisition raster, il est ainsi courant d'observer des aberrations dues aux accélérations et ralentissements du support, mais aussi à divers points de grippage du moteur, induisant des changements de vitesse non-commandés qui nuisent à la localisation précise du point mesuré. Pour pallier ces erreurs, un script Python a été créé pour corriger les coordonnées X des points mesurés (Figure 113).

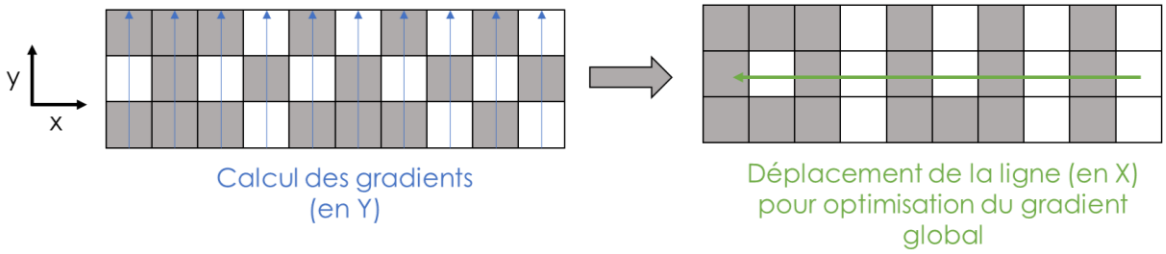

Figure 113 - Exemple simplifié de corrections des coordonnées en X pour minimiser le gradient en Y, permettant la correction des aberrations de déplacement induis par la plateforme.

Afin de déterminer ces corrections, l'algorithme compare alors chaque ligne à sa ligne adjacente, avec pour objectif de minimiser le gradient entre chaque pixel mesuré. Pour modifier les coordonnées X des points de la ligne étudiée, une composition de fonctions mathématiques a été créée, en tentant de modéliser les variations de vitesse globale de la plaque – accélération en début de ligne, et décélération en fin de ligne – tout en étant capable de prendre en compte les points de grippage. La fonction utilisée pour aligner les lignes entre elles est composée de deux sigmoïdes aux extrémités de lignes, et d'une fonction d'ordre 3 entre ces deux dernières (Figure 114).

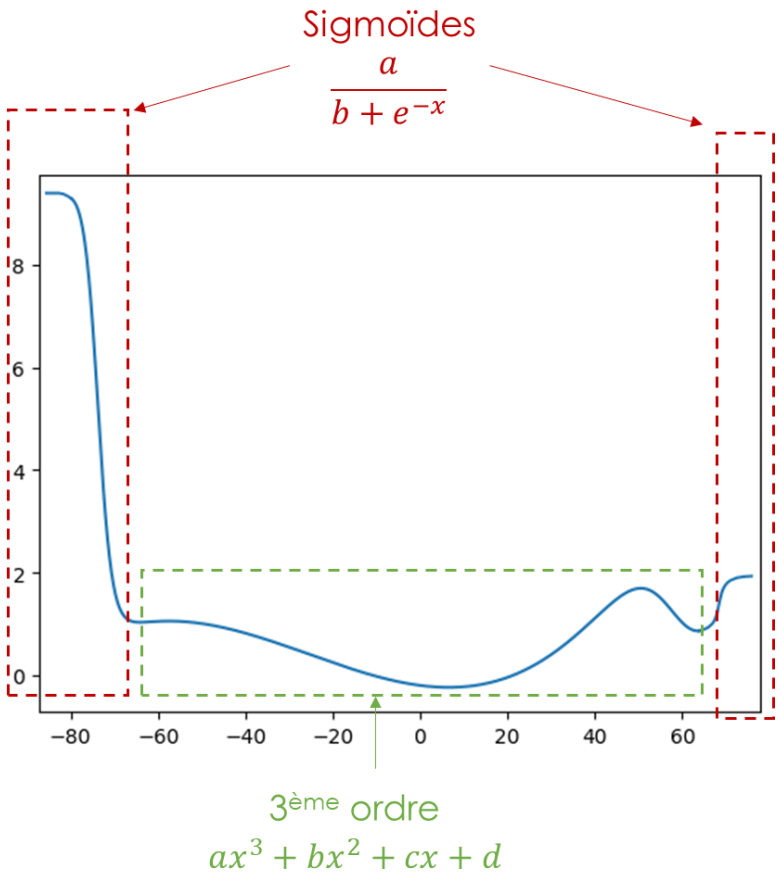

Figure 114 - Exemple de fonction composite utilisée pour corriger les aberrations en adaptant les coordonnées en X des points de mesure.

Un optimiseur va faire varier tous les paramètres de la fonction composite et calculer le gradient en Y pour chacune des lignes, afin de déterminer la correction optimale. Cela permet d'obtenir des images ellipsométriques de haute résolution, où la grande majorité des aberrations sont retirées. Ces images peuvent être comparées à des acquisitions full wafer optiques, ou aux designs des produits pour valider le protocole de correction.

Génération des images ellipsométriques :

Les images ellipsométriques sont générées en assignant chaque valeur de *Is* et *Ic*, mesurées à la longueur d'onde optimale sélectionnée, à leurs coordonnées respectives (Figure 115). Dans le cas d'acquisition raster, on remarque que la correction des aberrations permet bien l'obtention d'images ellipsométrique de larges zones à haute résolution. Dans le cas des acquisitions point par point, il faut au préalable réaliser une interpolation cubique pour compléter les données manquantes entre les points de mesures. Ce traitement a été implémenté par un script Python, et n'a donc pas eu à être mis en place pour les acquisitions Raster, car la région entière est scannée, chaque empreinte de spot correspondant à un pixel de l'image finale.

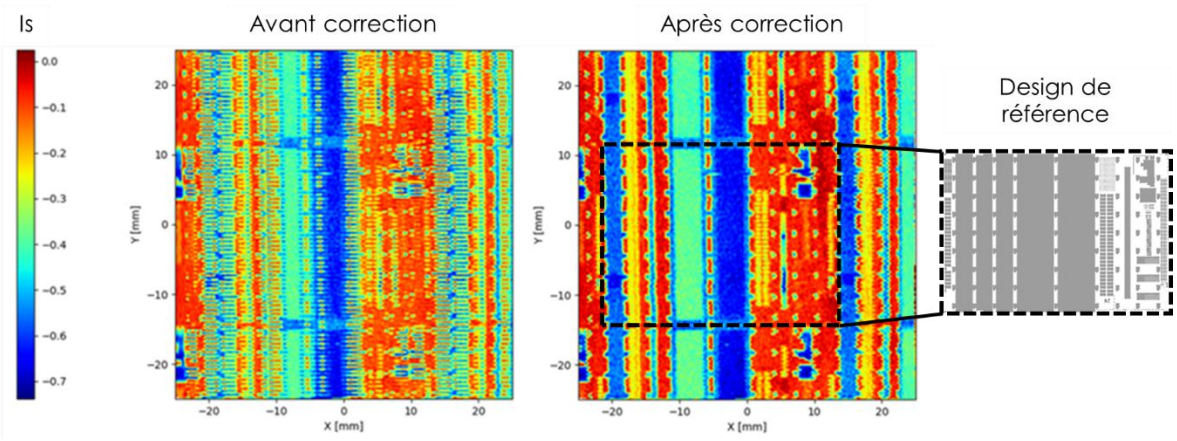

Figure 115 - Exemple de correction d'aberrations présents sur les images ellipsométriques, dues aux problèmes de grippage de la plateforme.

En suivant ce protocole expérimental, nous sommes alors capables d'utiliser l'ellipsométrie à large échelle pour obtenir des images, qui seront étudiées pour évaluer la sensibilité de cette approche à diverses dérives de procédés fournies par STMicroelectronics.

III. Évaluation de la sensibilité de l'imagerie ellipsométrique à diverses propriétés

Dans cette partie, nous nous sommes intéressés à évaluer l'intérêt de cette approche pour détecter plusieurs caractéristiques physiques à grande échelle : Variations de propriétés des matériaux, variations d'épaisseur et aussi variations des dimensionnels.

1. Variation de propriétés physiques et optiques : Cas du nitrure de passivation

Afin d'évaluer la sensibilité de notre approche d'imagerie ellipsométrique aux propriétés optiques et physique des matériaux, un cas d'étude a été identifié. Celui des nitrures de passivations, qui sont des couches sont utilisées comme films diélectriques agissants comme isolants entre les pads de contact en aluminium (Figure 116).

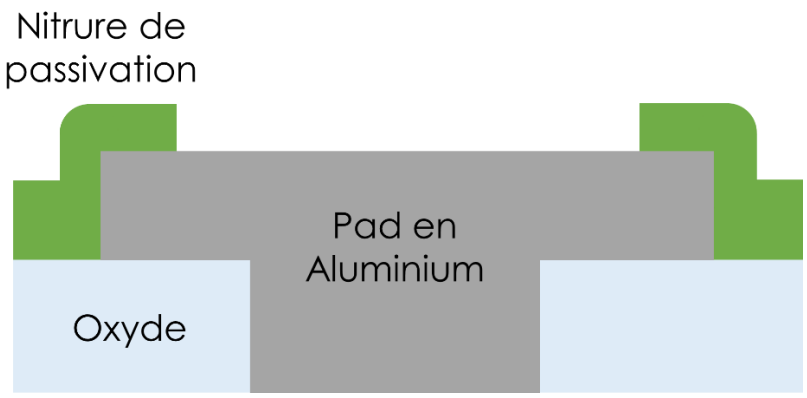

Figure 116 - Représentation des couches présentes en fin de procédés, montrant la couche de nitrure de passivation, isolant le pad en Aluminium.

Cette étape finale permet ainsi de s'assurer du bon fonctionnement de chacun de ces contacts, et l'étape de dépôt de ces nitrures de silicium est alors critique, malgré leur faible épaisseur (~600 nm). Cette étape est réalisée par PECVD (*Plasma Enhanced Chemical Vapor Deposition* – Dépôt chimique en phase vapeur assisté par plasma), et une dérive du centrage des plaques dans la chambre de déposition peut engendrer du stress dans ces couches, entrainant des fortes pertes de performances des puces présentes sur la plaque (Figure 117).

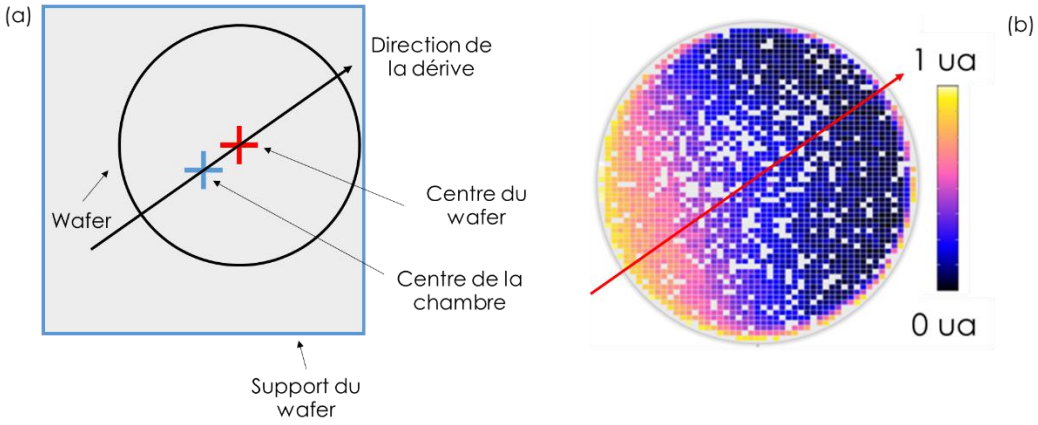

Figure 117 – Exemple de dérive de centrage (a) dans la chambre de dépôt du nitrure, entrainant des pertes de performances des puces, visibles sur la cartographie de rendement normalisée (b), montrant une signature diagonale (Blanc : Pas de résultats de test - Bleu est bon, Jaune est mauvais).

Nous avons volontairement fait subir cette dérive lors du dépôt de nitrure à deux plaques : Une où uniquement cette étape fut réalisée (sur silicium vierge), l'autre sur une plaque en fin de production. L'objectif est d'être capable de détecter cette dérive à l'aide de l'imagerie ellipsométrique, en mettant en évidence la signature diagonale caractéristique de cette dernière.

Au vu de la signature générée par cette dérive, le protocole basique d'acquisition a été choisi, et une grille de mesure de 6000 points répartis de façon homogène sur la surface du wafer à 300mm a été créé. Afin de maximiser l'information récoltée, le spot de 8 mm x 5mm a été utilisé. Le temps d'acquisition fut de 3h30 par plaque.

a. Analyse du nitrure : Plaque témoin

Dans un premier temps, nous nous sommes intéressés à la plaque où seule la couche de nitrure est déposée sur le wafer en silicium. Ce wafer a été, dans une étude préliminaire [3], caractérisé en utilisant un système d'ellipsométrie de métrologie industrielle, dont les résultats sont présentés en Figure 118. Ils avaient, quant à eux, à leur dispositions une plaque n'ayant pas subi la dérive, et on remarque alors que cette dérive n'a que très peu d'effet sur l'épaisseur de nitrure déposée, mais que l'indice optique est fortement modifié, suivant la signature diagonale observée sur la cartographie de rendement précédente.

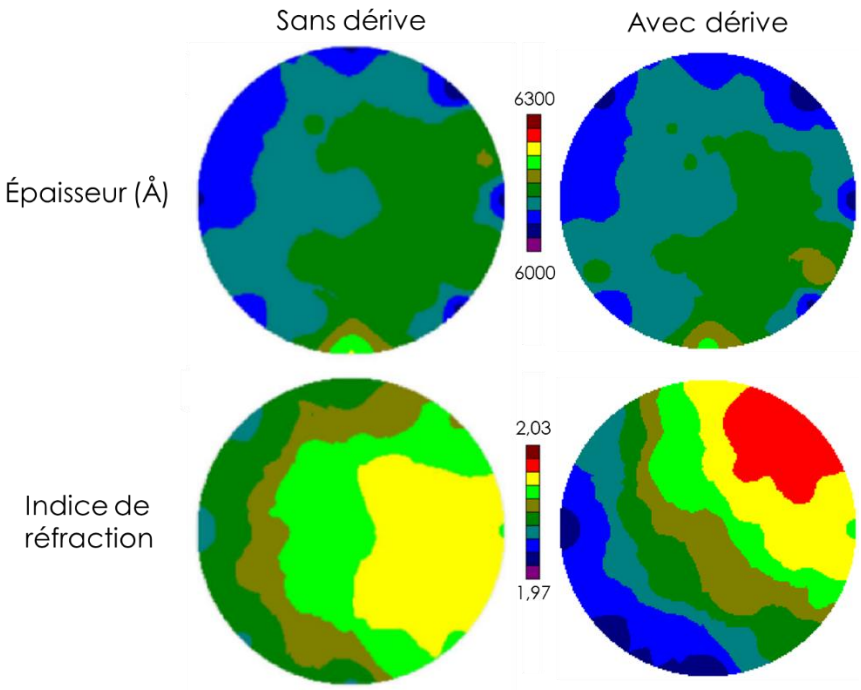

Figure 118 - Résultats des études ellipsométrique industrielle des plaques (avec et sans dérive de centrage – 37 points par plaque) où seul le dépôt de Nitrure de Silicium est réalisé, avec cartographies d'épaisseur (en Angström – haut) et d'indice de réfraction (à 632 nm – bas).

Nous avons suivi le protocole de sélection de longueur d'onde basique, et donc réalisé un scan spectroscopique au centre de la plaque. Le résultat de cette mesure est présenté en Figure 119. On y remarque que plusieurs longueurs d'ondes auraient pu être sélectionnées avec des pentes locales comparables, mais, afin de maximiser l'intensité lumineuse de la source, une longueur d'onde de 500nm a été retenue.

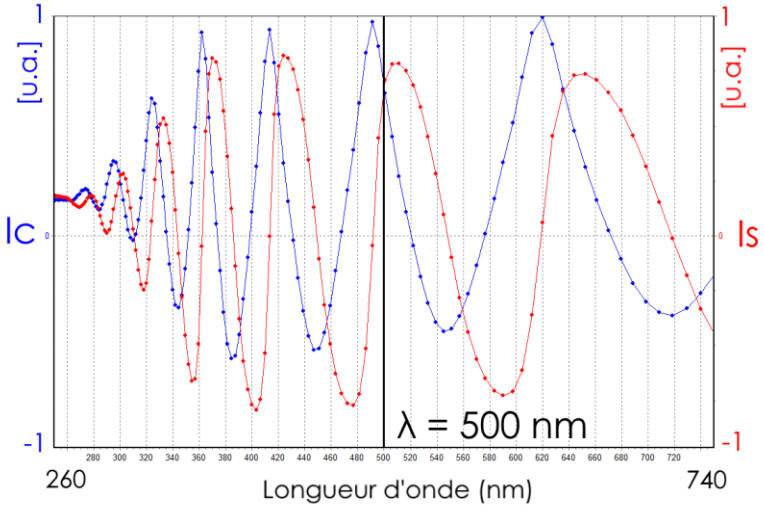

Figure 119 - Spectre ellipsométrique obtenu par scan spectroscopique (Is et Ic) réalisé au centre de la plaque de nitrure.

La réponse ellipsométrique à 500 nm a donc été collectée en utilisant une grille de mesure de 6000 points répartis uniformément sur la surface du wafer, à l'aide du spot le plus large (8 mm x 5 mm) et le résultat de cette acquisition est présenté en Figure 120. En plus des études ellipsométriques,

l'étude préalable a également utilisé un équipement d'interférométrie industriel (PWG) pour mesurer le stress sur la surface entière de la plaque. Cette technique de référence sera utilisée pour valider la sensibilité de notre approche à cette dérive.

On observe que la signature obtenue grâce à l'imagerie ellipsométrique model-less est extrêmement similaire à celle obtenue lors de mesures industrielles de stress. Cela indique donc une sensibilité de l'approche aux variations de propriétés des couches de nitrures de passivation déposées, sans avoir recourt au développement de modèle théorique. Il est important de rappeler que des mesures d'épaisseurs ont été préalablement réalisées, et que la cartographie obtenue ne possédait pas de signature diamétrale, contrairement à la cartographie d'indices de réfraction obtenues par ellipsométrie industrielle, ainsi qu'à celles obtenues avec notre approche d'imagerie ellipsométrique. On peut alors en conclure que la sensibilité de cette dernière est bien liée aux propriétés optiques de la couche de nitrure, et pas à l'épaisseur locale.

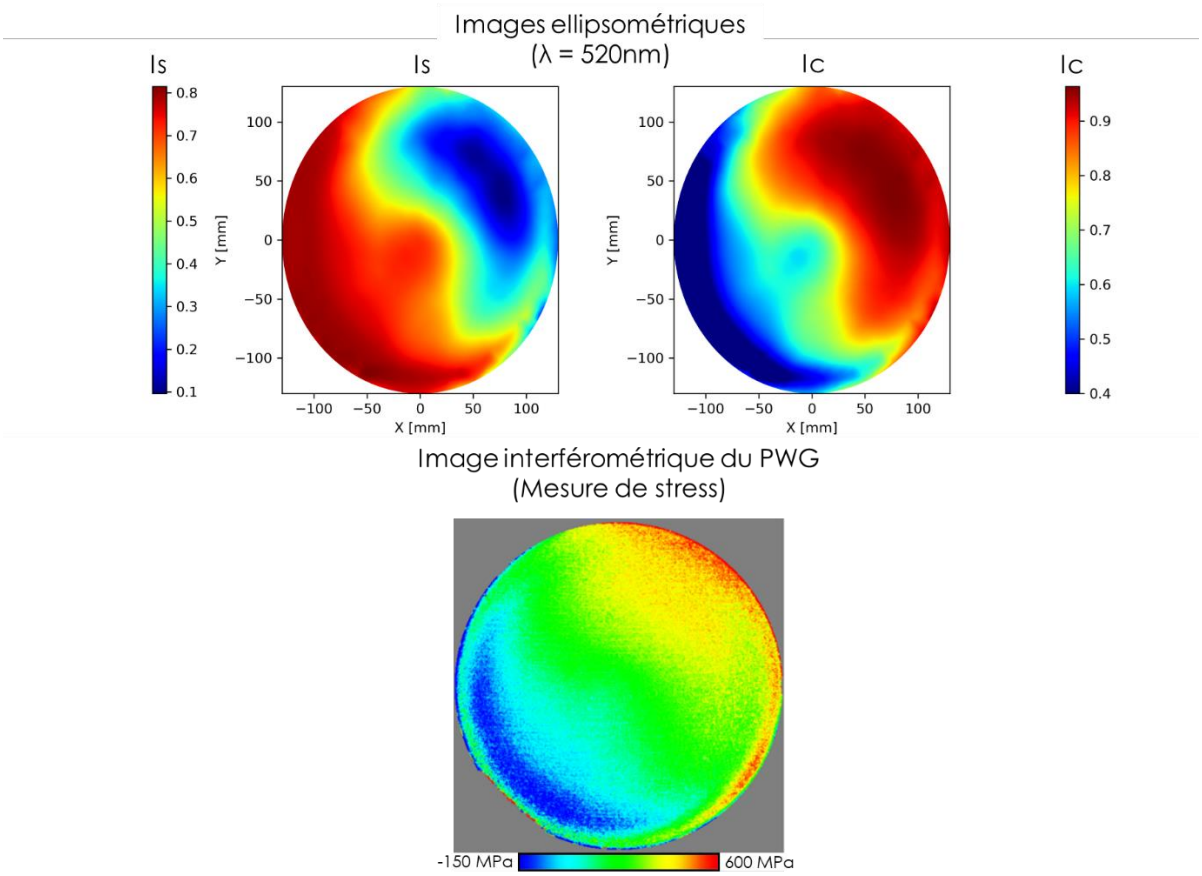

Figure 120 - Résultats de l'acquisition d'imagerie ellipsométrique réalisé sur la plaque témoin (a), comparée à la cartographie de stress obtenue par le PWG (b).

b. Evaluation en condition réelle de production : Plaque produit

Contrairement aux plaques témoins, les plaques produits sont composées d'un empilement complexe de couches et de motifs, pour lesquels la modélisation d'un modèle rigoureux serait extrêmement complexe, rendant l'approche non-modélisée de l'imagerie ellipsométrique d'autant plus pertinente). Les structures de métrologies utilisées pour les études in-line (notamment chez STMicroelectronics) sont très limitées en nombre, et peu présentes en bord de plaque, ce qui rend la détection de ce type de dérive – à large échelle et en bord de plaque – très difficile par les approches usuelles de métrologie. En effet, ces mesures sont d'autant plus compliquées à ces étapes avancées

du procédé, les empilements présents dans les structures de métrologie étant très complexes (empilement de nitrures et d'oxydes), la détermination précise des indices optiques est souvent impossible.

De la même façon que lors de l'étude de la plaque témoin, le protocole expérimental débute par une mesure spectroscopique afin de déterminer la longueur d'onde optimale à utiliser pour l'imagerie ellipsométrique de la plaque (Figure 121). Afin de moyenner la signature optique des couches sous-jacentes, l'acquisition a été réalisée avec le spot le plus large de 8 mm x 5 mm. Nous avons aussi privilégié une longueur d'onde dans l'UV (338 nm) afin d'augmenter la sensibilité à la surface de la plaque.

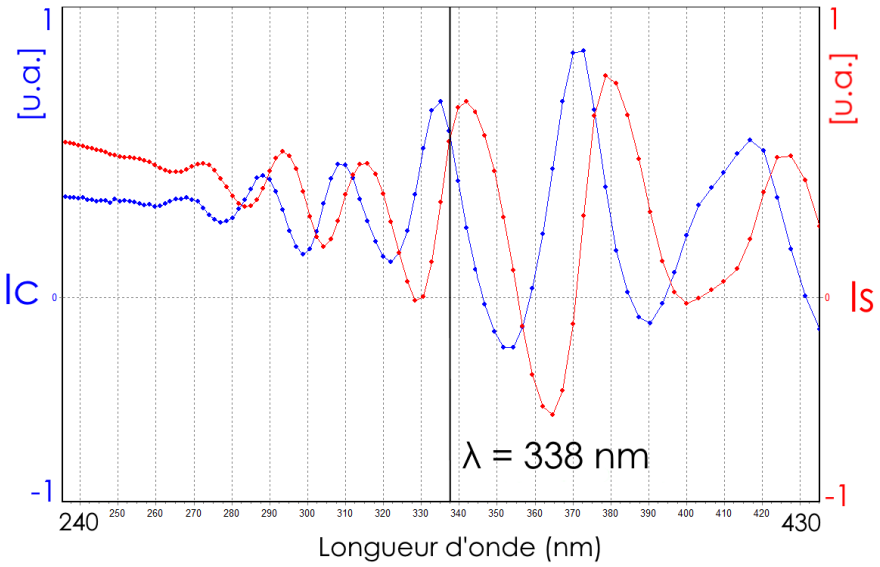

Figure 121 - Spectre ellipsométrique obtenu par scan spectroscopique (Is et Ic) réalisé au centre de la plaque produit.

On utilise la même grille de 6000 points de mesure que précédemment pour récolter les données brutes ellipsométriques, et ainsi obtenir les images présentées en Figure 122. Une signature diamétrale est alors visible sur ces dernières, comparable à celle obtenue lors de l'étude de la plaque témoin. La différence d'orientation entre les deux signatures est expliquée par l'absence d'une étape d'orientation de la plaque dans la chambre lors de l'étape de déposition de nitrure. Ces résultats ont encore une fois été comparés aux caractérisations obtenues sur la même plaque à l'aide d'une technique de métrologie basée sur l'interférométrie (PWG). La cartographie obtenue par cet équipement est présentée en Figure 122 et indique une signature fortement similaire à celle obtenue par notre approche model-less.

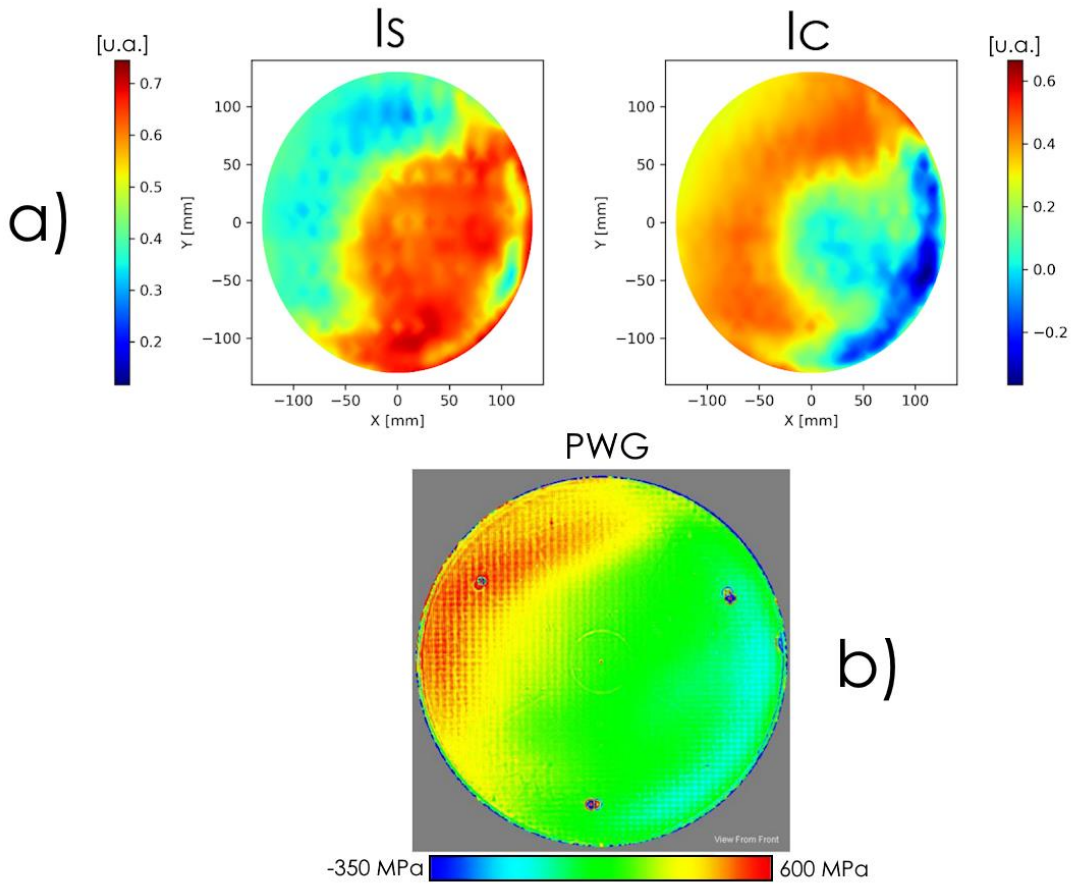

Figure 122 - Résultats de l'acquisition d'imagerie ellipsométrique réalisé sur la plaque produit (a) et comparaison à la cartographie de stress obtenue par le PWG (b).

Notre approche d'ellipsométrie model-less a ainsi été capable de détecter des changements de propriétés de matériaux induit par une dérive de procédé à l'échelle du wafer, d'abord dans le cas d'une plaque blanket, mais surtout, pour une plaque produit. Il est alors nécessaire de valider la sensibilité apparente de notre approche à ce type de déviation de process sur des plaques possédant différentes propriétés (e.g. variation d'épaisseur des couches sous-jacentes au nitrure) pour en évaluer la robustesse ainsi que sa répétabilité.

2. Variation d'épaisseur : Cas des résines colorées sur les imageurs

Pour évaluer la sensibilité de l'imagerie ellipsométrique à des variations d'épaisseur à large échelle, un second cas d'intérêt industriel a été identifié : celui de la variation d'épaisseur des résines colorées. Cette dérive nommée "Striations" a été présentée plus en détail dans le chapitre III.3.b. Pour rappel, ces résines sont utilisées comme filtres dans les technologies imageur afin de réaliser la matrice de pixel, et la dérive de striations est caractérisée par une signature radiale, plus prononcée en bord de plaque. Dans ce contexte, il était nécessaire d'augmenter radicalement la résolution des images générées par ellipsométrie model-less, le mode d'acquisition *Raster* a donc été utilisé. Comme la dernière étape de process est un dépôt, nous avons alors aussi pu mettre en place le protocole avancé de sélection de longueur d'onde basé sur l'utilisation de substrats virtuels. L'intérêt d'une approche model-less est dans ce cas d'intérêt renforcé par l'extrême complexité des propriétés optiques des résines utilisées.

Un scan spectroscopique a été réalisé avant l'étape de dépôt des résines colorées, sur une puce en bord de plaque, zone susceptible d'être davantage atteinte par l'apparition de striations après dépôt de la résine colorée. Puis, la même puce a été scannée de nouveau après le dépôt de résine, de façon à s'assurer que toute différence entre les deux spectres soit entièrement due à l'étape réalisée. Des études préliminaires (via ContourGT-X présentée au chapitre 3) ont déterminé que la hauteur moyenne des striations était de l'ordre de 20 nm, que nous avons alors choisi comme variation de notre couche virtuelle. Un résumé du procédé de sélection de longueur d'onde par substrat virtuel est présenté en Figure 123. La longueur d'onde retenue pour les acquisitions est de 412 nm.

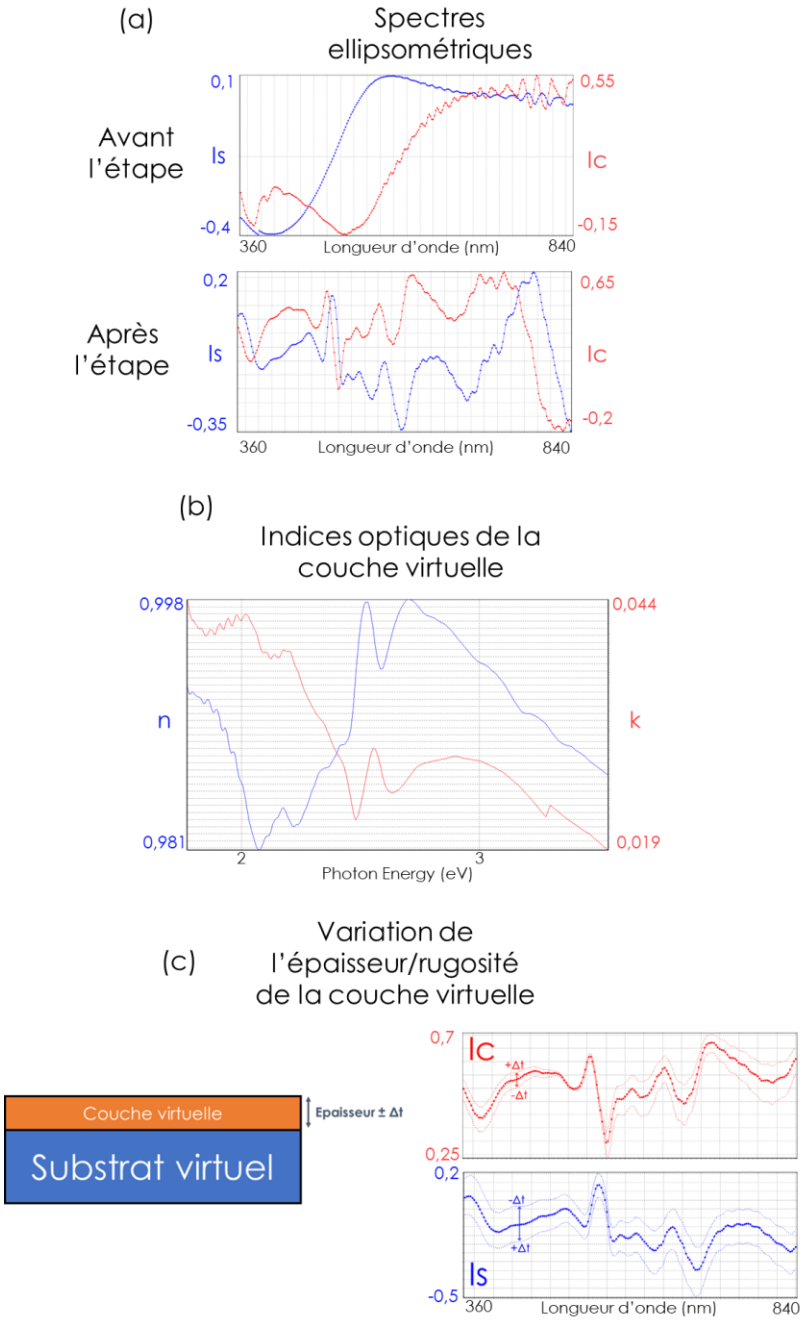

Figure 123 - Résumé du protocole de sélection de longueur d'onde par utilisation de substrat virtuel : a) Scans spectroscopiques avant et après l'étape étudiée, b) création d'une couche virtuelle (indices optiques du matériau fictif représenté) permettant de représenter la variation induite par l'étape et c) variation de l'épaisseur de la couche créée pour en observer l'incidence sur les spectres, afin de déterminer la longueur d'onde optimale, ici 412 nm.

Afin de tester cette sensibilité, une acquisition rapide a été réalisée sur une puce en périphérie de wafer, et les résultats sont présentés en Figure 17. On peut clairement y observer la signature des striations recherchée ce qui confirme la sensibilité de notre approche à cette dérive à la longueur d'onde choisie.

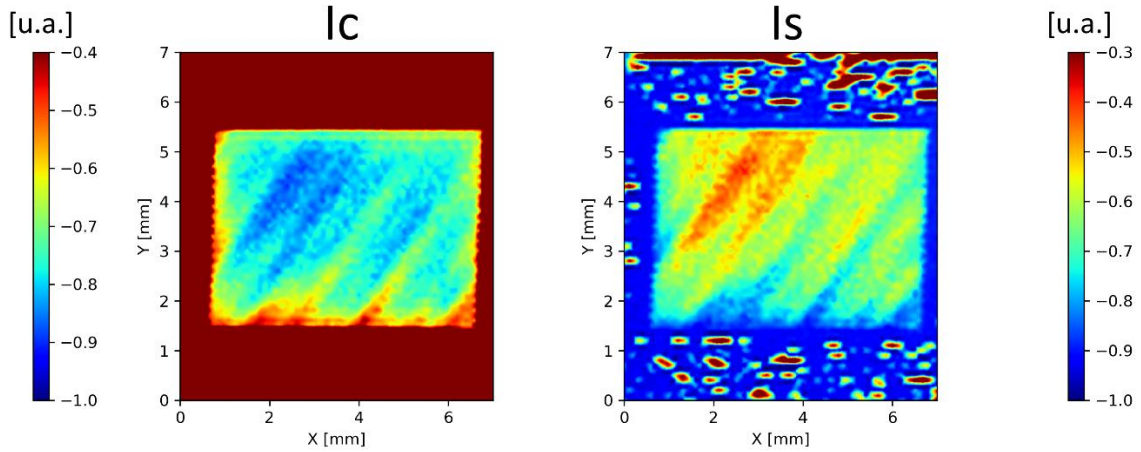

Figure 124 - Résultat de l'acquisition d'imagerie ellipsométrique sur une puce en périphérie de wafer.

Ensuite, une acquisition raster (~1.5 millions de points en 4 heures) a permis d'imager une large échelle du wafer. Après corrections des aberrations dues aux erreurs de calculs des positions lors du scan à la volée, on obtient les images ellipsométriques présentées en Figure 125. Ces images sont à comparer aux cartographies de nano-topographies obtenus par le PWG.

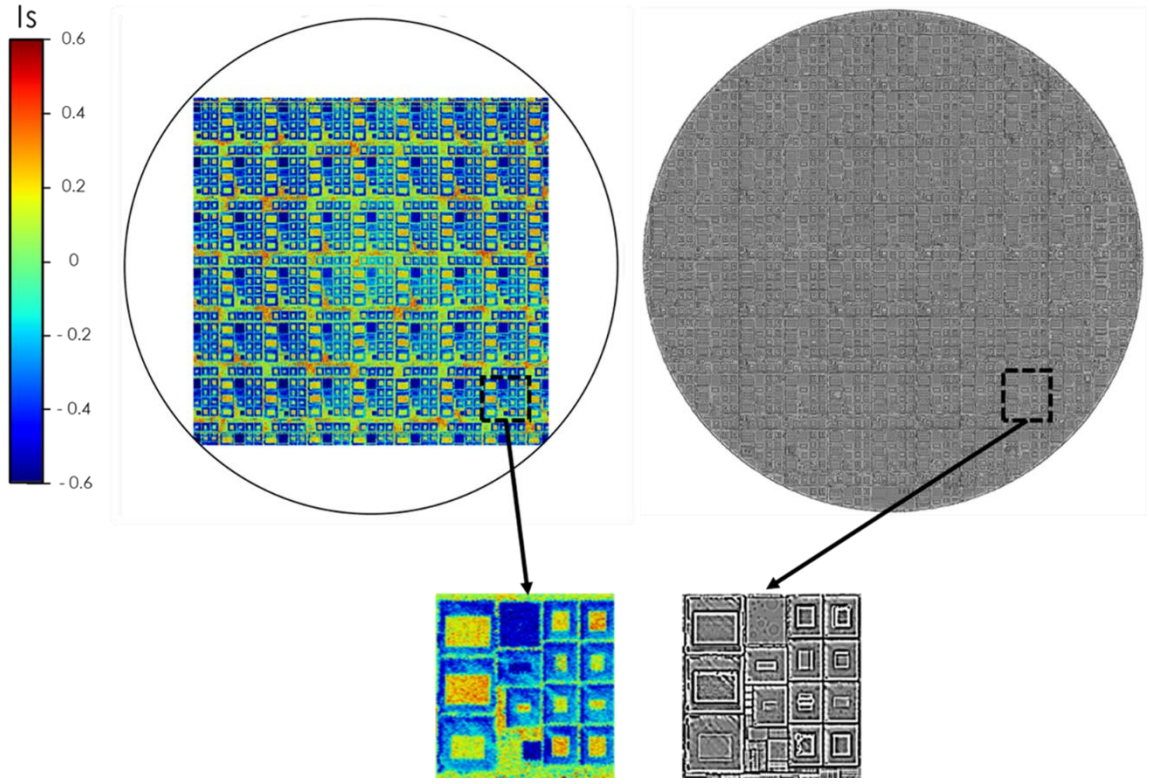

Figure 125 - Résultat de l'acquisition en mode raster (Uniquement Is représenté) à 412 nm (1.5 millions de points) sur la plaque à résines colorées, comparées à la cartographie de nanotopographie obtenue par le PWG.

L'image ellipsométrique obtenue par acquisition en mode Raster permet alors d'obtenir une image de haute résolution, dont la sensibilité est assurée par l'utilisation du protocole avancé de sélection de longueur d'onde basé sur les substrats virtuels. On y remarque une signature de striations semblable à celle obtenue par le PWG, qui comme vu dans le chapitre précèdent, n'est pas adapté à l'étude de surface non-opaques.

Notre approche est alors capable, dans ce cas d'intérêt, de détecter des variations d'épaisseur de résine sur un stack très complexe, ce qui serait une fois de plus impossible avec une approche traditionnelle d'ellipsométrie in-line, les cibles utilisées étant bien trop petites et rares en bord de wafer.

3. Variation de géométrie : Cas des vias périodiques

Pour déterminer la sensibilité de l'imagerie ellipsométrique à des variations de dimensions critiques, nous nous sommes appuyés sur le cas applicatif des vias périodique. Ces structures, formant un réseau périodique, sont présentes sur les wafers à technologies optiques (voir Figure 126), et leurs buts est de transformer un faisceau ponctuel incident afin de le diffuser (technologie ODIFF déjà présentée dans le chapitre 2). L'étude de ces structures, particulièrement dans le produit, où bien que l'espacement entre chaque via soit constant, les dimensions de chacune d'entre elles sont apériodiques, n'est alors pas envisageable par une approche modélisée. En effet, nous avons montré dans le chapitre II que la modélisation de ce type de structure, même dans le cas de dimensions constantes, était extrêmement complexe. Notre approche sans modélisation permettrait un suivi de dérive des dimensions de ces structures, à large échelle, et surtout directement dans le produit. Pour rappel, nous avons généré deux plaques, ayant été fabriquées avec une *energy matrix*, comportant donc une variation de CD intentionnelle sur le diamètre horizontal du wafer.

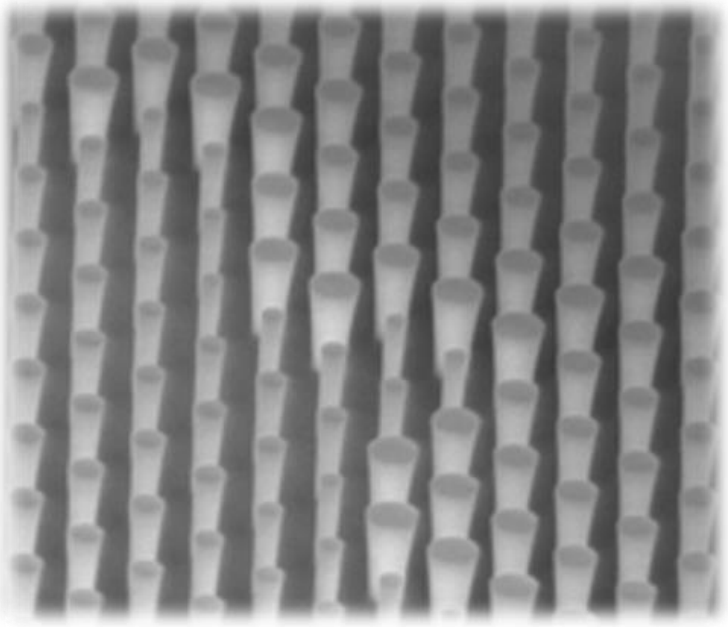

Figure 126 - Image SEM de structures 3D périodiques utilisées dans les technologies ODIFF.

Ces plaques sont étudiées à la fin de l'étape de gravure de ces structures. L'approche basée sur les substrats virtuels n'est alors pas applicable ici, du fait de l'absence de l'ajout d'une couche supplémentaire. On a eu recourt au protocole basique de sélection de longueur d'onde pour imager le wafer. Un scan spectroscopique a été réalisé dans ces structures et deux longueurs d'ondes

possédants des sensibilités semblables ont été identifiées (pentes locales en *Is* et *Ic* similaire) : 410 et 518 nm (Figure 127).

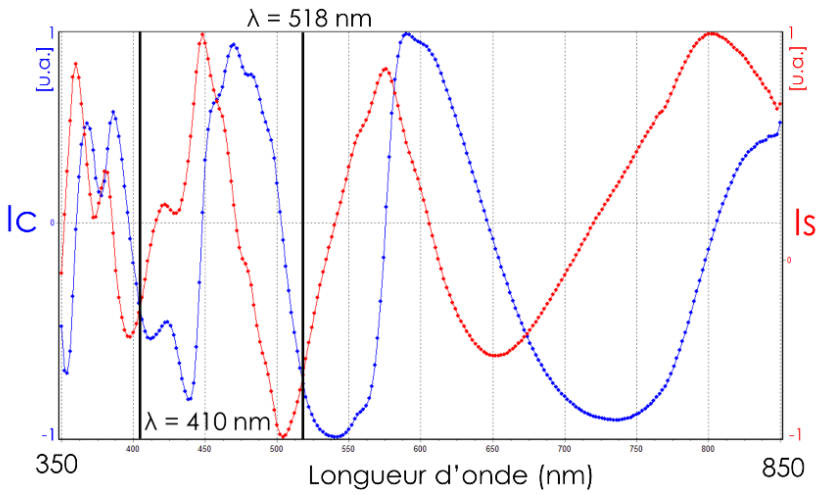

Figure 127 - Scan spectroscopique réalisé sur les composants optiques (trous périodiques), montrant deux points d'intersections possédant des pentes locales similaires.

Une acquisition en mode Raster a ainsi été réalisé en utilisant les deux longueurs d'ondes identifiées ce qui permet d'obtenir deux images d'une large surface du wafer. Nous avons ainsi pu évaluer la sensibilité de chacune d'entre elles aux variations de CD induites par l'*energy matrix*. Les résultats obtenus (uniquement *Is* par soucis de clarté) à 410 nm sont présentés en Figure 128.

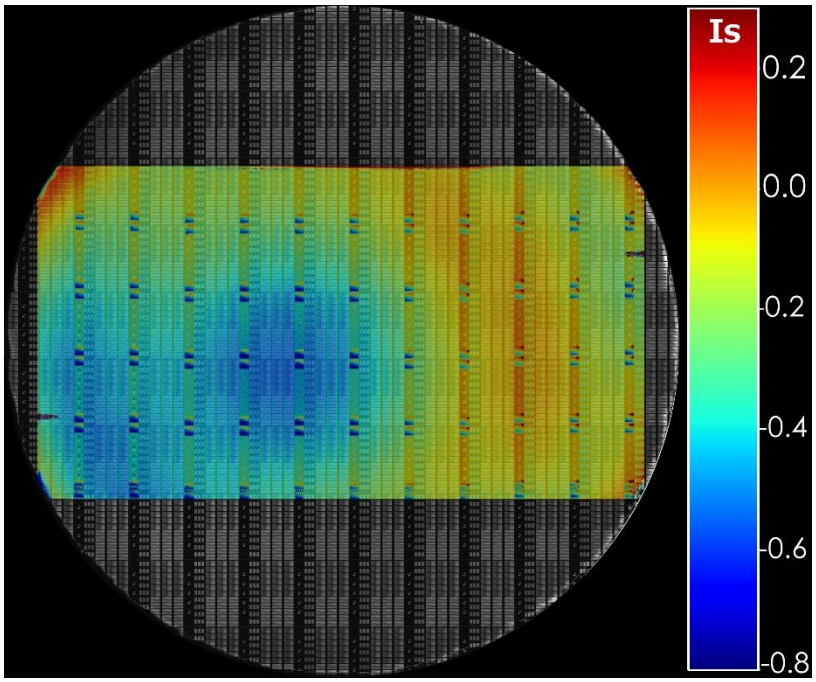

Figure 128 - Résultat de l'imagerie ellipsométrique (uniquement Is) en utilisant le mode raster à 410 nm (400 000 points).

Malheureusement, la signature observée à cette longueur d'onde ne corrèle pas avec les variations de CD mesurés en SEM-CD, présentés en Figure 128. Cette cartographie interpolée à partir de 70 points montre une signature constituée d'une composante bord/centre usuelle dans les procédés de fabrication de microélectronique, combinée à la variation horizontale induite par l'EM.

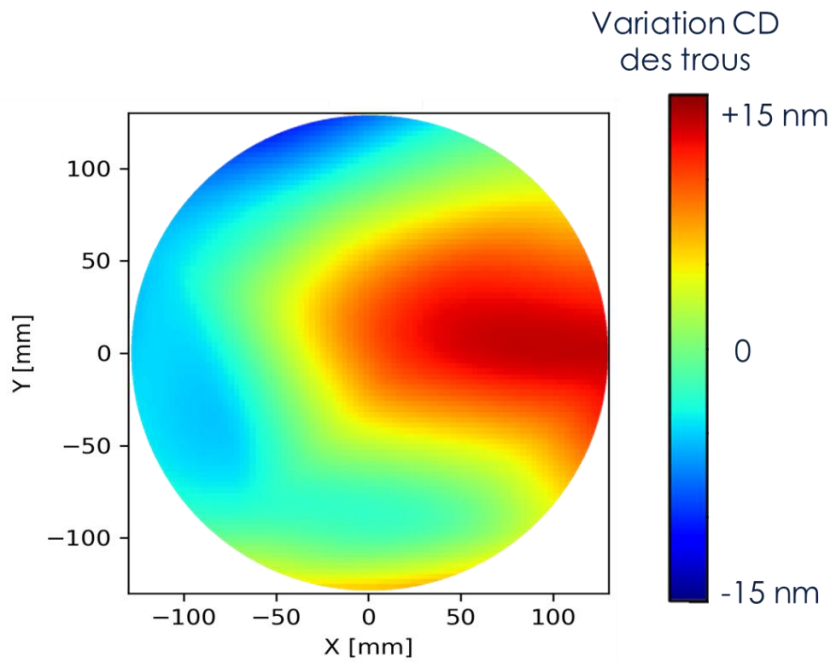

Figure 129 – Cartographie de variation de CD mesurés en SEM-CD (70 points interpolés - variation de 37% entre les CD maximums et minimum) montrant une signature combinée d'une centre/bord usuelle, et de l'horizontale due à l'EM.

La seconde longueur d'onde de 518 nm est sélectionnée pour l'acquisition Raster, et les résultats obtenus sont présentés en FIG. Cette fois, une signature horizontale claire est visible sur l'image ellipsométrique générée, mais aussi une signature habituelle centre/bord dans l'axe verticale, de la même façon que sur la cartographie SEM-CD. Il est prévu de valider cette apparente sensibilité à l'aide d'une acquisition sur une plus large zone à cette longueur d'onde.

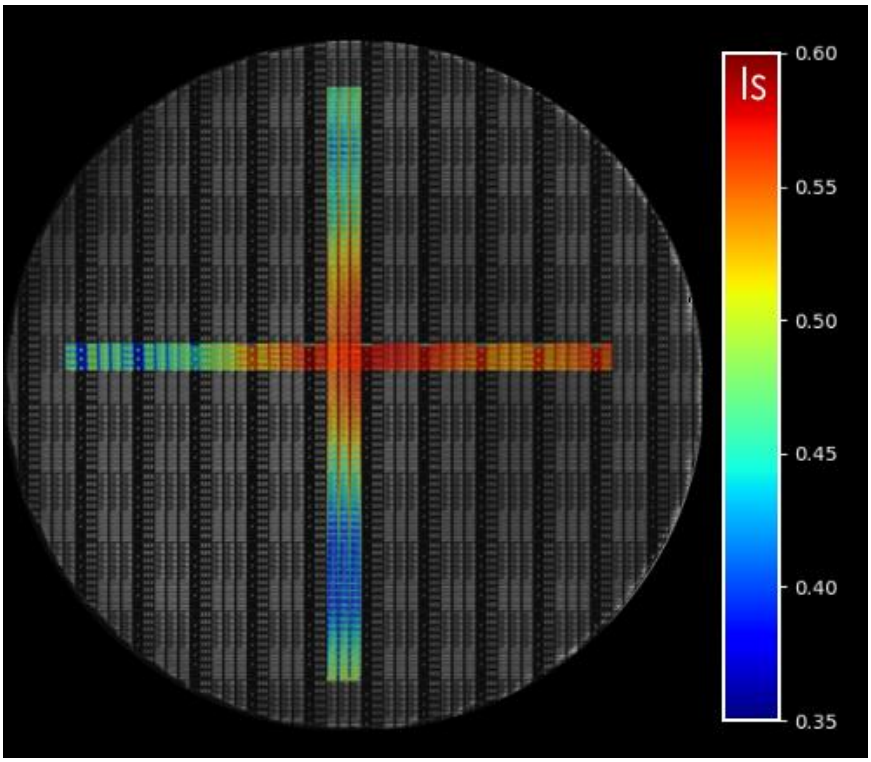

Figure 130 - Résultat de l'imagerie ellipsométrique (uniquement Is) en utilisant le mode raster à 518 nm (83 000 points).

Ce dernier cas d'étude dont l'objectif était de valider la sensibilité de l'imagerie ellipsométrique pour le suivi de variation dimensionnelle de structures périodiques montre que le choix de la longueur d'onde est crucial. Pour le moment, nous n'avons pas encore déterminé de protocole permettant de choisir correctement la longueur d'onde. Cependant, dans ce cas d'intérêt, une voie d'amélioration pourrait être basée sur la modélisation RCWA (présentée en chapitre 2) des structures, afin de pouvoir simuler la réponse optique de ces dernières, et ainsi déterminer quelles longueurs d'ondes seront les plus sensibles à des variations dimensionnelles.

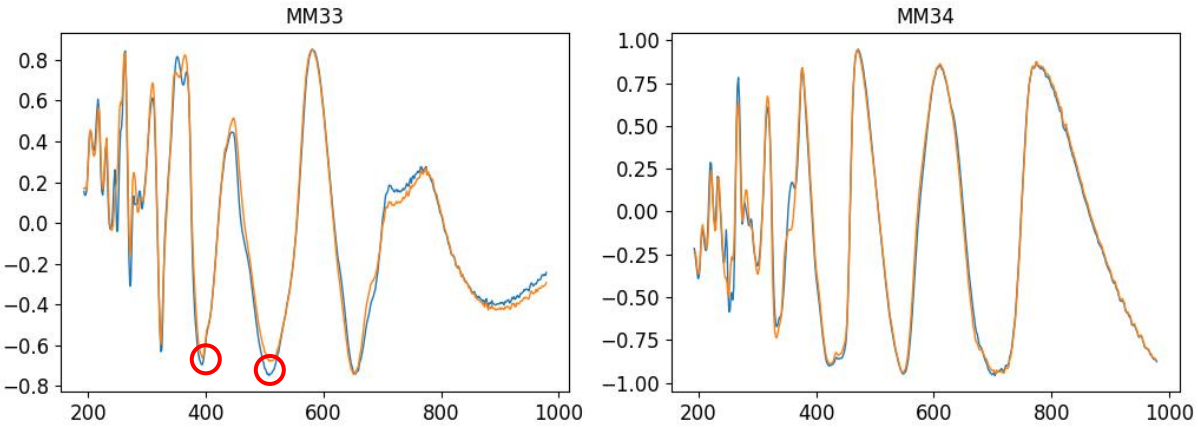

Figure 131 - Simulation RCWA (MM33 : Is ; MM34 : Ic) de deux structures aux CD proches (293 et 296 nm) pouvant permettre de déterminer une longueur d'onde sensible pour l'imagerie ellipsométrique.

IV. Conclusion

L'utilisation de l'ellipsométrie afin de générer des images à large échelle permet d'être capable de détecter des déviations de procédé de fabrication et ceci avec une approche sans modèle. Ces dérives peuvent être d'origine multiple : variation des propriétés des matériaux, d'épaisseur ou de nanotopographie de surface à large échelle, ou encore de dimensions (CD). Afin de réduire les temps d'acquisitions, tout en augmentant la résolution des images obtenues par ces dernières, un mode raster permettant la récolte des données à la volée a été mis en place, permettant d'imager en haute résolution de larges zones du wafer. Cela s'avère particulièrement utile dans le cas de déviation à large échelle comme les striations. Ces acquisitions requièrent une caractérisation préalable de l'échantillon à étudier, afin de déterminer la longueur d'onde optimale à utiliser pour la récolte des données ellipsométrique sur une large zone. Plusieurs protocoles de sélection ont été évalués et dans le cas de dépôt de couches, un protocole avancé basé sur l'utilisation de substrats virtuels a permis une détermination plus fiable des longueurs d'ondes optimale que l'approche basique.

Références

[1] P. Braeuninger, S. Funke, R. Wang, P. Thiesen, D. Tasche, W. Viöl et S. Hofmann, «Fast, Non-Contact, Wafer-Scale, Atomic Layer Resolved Imaging of 2D Materials by Ellipsometric Contrast Micrography,» *ACS Nano,* vol. 8, p. 8555–8563, 2018.

[2] A. Canillas, E. Pascual et E. Bertran, «Calibration improvement of Fourier transform infrared phase-modulated ellipsometry,» *J. Opt. Soc. Am. A ,* n° %113, pp. 2461-2467, 1996.

[3] V. Brouzet, V. Gredy, F. Chenevas-Paule, K. Le-Chao, D. Guiheux, A. Laurent, V. Coutellier et D. Le-Cunff, «Full Wafer Stress Metrology for Dielectric Film Characterization: Use Case,» *30th Annual SEMI Advanced Semiconductor Manufacturing Conference (ASMC),* pp. 1-7, 2019.

Conclusion générale

Les progrès envisagés par la microélectronique, que ce soit la miniaturisation extrême, où l'intégration de produits avancés, nécessitent une évolution considérable des techniques et approches de suivi de procédés.

L'utilisation de techniques de métrologie extrêmement sensibles, à large échelle et sans développement de modèle physique rigoureux, avec une approche plus semblable à de l'inspection peut permettre de répondre aux attentes de l'industrie des semi-conducteurs. Ces travaux de thèse ont permis d'évaluer ces approches de métrospection, à la frontière de ces deux domaines, en utilisant des données brutes, traitées par des algorithmes intelligents, et cela afin d'en évaluer la sensibilité à différentes dérives de procédés, influant sur divers paramètres physiques des matériaux constituants les produits. Les résultats obtenus sont encourageants, et peuvent inciter à poursuivre ce type d'approche, ainsi qu'à leur mise en production pour du HVM.

Dans cette thèse, nous sommes partis d'une technique optique extrêmement sensible, l'ellipsométrie, pour l'appliquer à large échelle, et déterminer si cette sensibilité était possible sans lourd développement de modèle. Pour ce faire, un équipement de recherche académique à été adapté pour permettre des mesures relativement rapides de larges zones des plaques. Les données ellipsométriques récoltées sur ces surfaces ont donc permis, via un script Python, de générer des images ellipsométriques des zones étudiées. Via ces images, on a alors évalué la sensibilité de cette approche à diverses dérives de procédés, impactant l'épaisseur de couches, les propriétés des matériaux présents, ou les dimensions des structures, sur des régions bien plus grandes que celles mesurées couramment par les équipements d'ellipsométrie de métrologie industrielle. Une fois cette sensibilité avérée, des premières approches de traitement avancés de données ont alors été utilisées pour traiter les images obtenues.

Ce type de traitement a été appliqué à d'autre sources d'images de wafer, soit des équipements de métrologie ou de défectivité. En effet, afin de répondre aux attentes de suivi de process auxquelles aucun traitement usuel n'était applicable, divers traitements avancés d'images ont été mis en place. Premièrement, des réseaux de neurones convolutifs ont été utilisés pour classifier des images de puces marquées par une signature de dérive de procédé. Puis, un traitement mathématique d'image a été mis en place pour automatiser le suivi de rupture du réseau cristallin du silicium, et appliquées sur trois équipements industriels, afin de déterminer le protocole optimal de contrôle de cette dérive. Enfin, afin d'automatiquement classifier des signatures mises en évidence par de l'imagerie de photoluminescence industrielle, un réseau de neurones convolutif régional a été entrainé pour détecter et localiser les signatures atypiques. Au vu de l'efficacité de cette approche, nous avons alors, pour classifier les défauts de collage rendu visibles par de l'imagerie par microscopie acoustique, utiliser le même type d'algorithme.

La capacité des réseaux de neurones à générer des modèles complexes à partir de données brutes a ensuite été mis à contribution dans le cadre de la scatterométrie. Cette technique requiert usuellement le développement de modèle physique complexes de structures périodiques afin d'obtenir les dimensions et indices de ces dernières. L'utilisation de ces algorithmes intelligent a alors permis de se passer de cette étape, en utilisant les spectres bruts récoltés par l'ellipsomètre, et associés aux mesures des dimensions critiques réalisés par microscopie électronique. Sur des cas complexes de structure à 3 dimensions, extrêmement lourdes à modéliser rigoureusement, cette approche a été capable de prédire les dimensions de ces structures avec une haute précision.

Durant cette thèse, de multiples sources de données industrielles, sous des formats différents (spectres, données de nanotopographie, images, etc.) ont été traités afin de répondre à des

manquements de suivi de procédés, auquel aucune solution couramment utilisée n'était applicable. Dans les cas d'intérêt étudiés, les approches permettaient d'apporter un suivi automatique et robuste de la dérive étudiée, améliorant grandement les informations à dispositions des ingénieurs responsables de ce suivi, pouvant ainsi augmenter la qualité de fabrication globale des puces.

Ces travaux sont poursuivis par une thèse orientée vers la détermination des propriétés de microlentilles par scatterométrie model-less. Des approches de métrospection sont aussi attendues pour répondre à des attentes sur l'étude des données des équipements de production.

Annexe

I. Scripts Python

1. Entrainement du FCNN et inférence (Chapitre II)

```
#########################################################################
##                Entrainement du réseau de neurones                   ##
##              à partir des spectres de Scatterométrie                ##
##                          et des mesures SEM-CD                      ##
#########################################################################

## Importation des librairies
import pandas as pd
import glob
import random
import numpy as np
import os

if __name__ == '__main__':

    ##XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX##
    ##      Entraînement du FCNN      ##
    ##XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX##

    ###################################
    ##    Préparation du dataset     ##
    ###################################

    ## Lecture du fichier contenant les données SEM-CD
    Data_SEMCD = pd.read_xlsx(r'Path/to/xlsx')

    ## Récupération de la liste des données
    file_list = glob.glob(r"Path/to/files")

    ## Mélange aléatoire des données pour l'entrainement
    file_list = random.sample(file_list)
    L = len(file_list)

    ## Initialisation des variables pour récupérer les données des spectres
       scattérométriques (Xtrain, Xval, et Xtest)
    ## et de SEM-CD (Ytrain, Yval, et Ytest)
    Xtrain = []
    Xval = []
    Xtest = []
    Ytrain = []
    Yval = []
    Ytest = []

    ## Initialisation du compteur pour répartition des données
    count= 0

    ## Itération sur tous les fichiers de mesure scatterométrique, et lien à la
       mesure SEM-CD
    for file_OCD in file_list:
        ## Répartition des données pour l'entrainement, validation
         ## et test : 70/15/15%
        if count<=int(L*0.70):
            data = pd.read_csv(file_OCD)
            ## Ajout de toutes les matrices de Mueller du spectre ouvert aux
               données d'entrainement
            Xtrain.append(data.values)
            ## Récupération de la mesure SEM à partir du nom du fichier OCD,
               contenant la localisation de la mesure '
            Puce = os.path.basename(file_OCD).split('_')[4:6]
            Y = Data_SEMCD.loc[Data_SEMCD["Puce"]==Puce]
            ## Ajout de la mesure SEM-CD aux données d'entrainement
```

```
            Ytrain.append(float(Y))
            count+=1

        elif ((count<=int(L*0.70)) & (count>int(L*0.85))):
            data = pd.read_csv(file_OCD)
            ## Ajout de toutes les matrices de Mueller du spectre ouvert aux
               données de validation
            Xtrain.append(data.values)
            ## Récupération de la mesure SEM à partir du nom du fichier OCD,
               contenant la localisation de la mesure '
            Puce = os.path.basename(file_OCD).split('_')[4:6]
            Y = Data_SEMCD.loc[Data_SEMCD["Puce"]==Puce]
            ## Ajout de la mesure SEM-CD aux données validation
            Ytrain.append(float(Y))
            count+=1

        elif (count>int(L*0.85)):
            data = pd.read_csv(file_OCD)
            ## Ajout de toutes les matrices de Mueller du spectre ouvert aux
               données de test
            Xtrain.append(data.values)
            ## Récupération de la mesure SEM à partir du nom du fichier OCD,
               contenant la localisation de la mesure '
            Puce = os.path.basename(file_OCD).split('_')[4:6]
            Y = Data_SEMCD.loc[Data_SEMCD["Puce"]==Puce]
            ## Ajout de la mesure SEM-CD aux données de test
            Ytrain.append(float(Y))
            count+=1

## Formattage des données des différents datasets
Xtrain = np.array(Xtrain)
Xval = np.array(Xval)
Xtest = np.array(Xtest)

Ytrain = np.array(Ytrain)
Yval = np.array(Yval)
Ytest = np.array(Ytest)

############################
##     Entrainement       ##
############################

import tensorflow as tf
from keras.optimizers import Adam
from keras.models import Sequential
from keras.layers import Dense
import matplotlib.pyplot as plt

tf.keras.backend.clear_session()

## Création de l'architecture du réseau neuronal
model = Sequential()
model.add(Dense(5476, input_dim=5476, kernel_initializer='normal',
            activation='relu')) ##10956,5478,10714
model.add(Dense(4500, kernel_initializer='normal', activation='relu'))
model.add(Dense(3000, kernel_initializer='normal', activation='relu'))
model.add(Dense(2000, kernel_initializer='normal', activation='relu'))
model.add(Dense(1000, kernel_initializer='normal', activation='relu'))
model.add(Dense(250, kernel_initializer='normal', activation='relu'))
model.add(Dense(50, kernel_initializer='normal', activation='relu'))
model.add(Dense(1, kernel_initializer='normal'))

# Entrainement du modèle avec les hyper-paramètres spécifiés
opt = Adam(learning_rate=0.0000001, beta_1=0.9, beta_2=0.999, epsilon=1e-7)
model.compile(loss='mean_absolute_error', optimizer=opt)
callback = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=30)
```

```
history = model.fit(Xtrain, Ytrain, validation_data=(Xval,Yval), batch_size=50,
                        epochs=10000, callbacks=[callback], verbose=1)

# Récapitulatif de l'entrainement, et dessin des courbes
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()

##XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX##
##   Inférence avec le FCNN    ##
##XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX##

from keras.models import load_model

## Initialisation de la variable pour récuperer les prédictions des CD
Yinf = []

## Lecture du modèle entrainé
model = load_model("Path/to/model")

for X in Xtest:
    ## Prédiction du CD à partir des spectres du dataset de test
    Yinf.append(model.predict(X))

## Création d'une variable contenant les mesures SEM-CD et les
## CD prédits par le FCNN pour sauvegarde dans un excel
Resultats = np.array(Ytest,Yinf)
Resultats.to_excel(excel_writer = "Results_RCWA.xlsx")
```

2. Image de nanotopographie PWG (Chapitre III)

```
#########################################################################
##             Création d'images full wafer à partir                   ##
##             des données de nanotopographie du PWG                    ##
##               pour détection des slip lines                          ##
#########################################################################

## Importation des librairies
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import cm as CM
import matplotlib.patches as patches
import glob
import os
import pandas as pd
from scipy import interpolate

if __name__ == '__main__':

    ## Chemin vers dossier contenant les données
    csv_dir = r'Path/to/csv'

    ## Itération sur les fichiers (csv) pour chaque wafer
    for file in glob.glob(csv_dir+"/*.csv"):
        name = os.path.basename(file)[:-4]
        ## Lecture des données
        data = pd.read_csv(file, delimiter=',', header=None, skiprows = 47,
                           usecols=[0,1,2], names=['X','Y','Z']).dropna()


        ## Interpolation des données manquantes
        tix = np.linspace(-149,149,3000)
        tiy = np.linspace(-149,149,3000)
        XI,YI = np.meshgrid(tix,tiy)
        pos = np.zeros((len(data),2))
        Z = np.zeros(len(data))
        pos[:,0] = data['X']
        pos[:,1] = data['Y']
        Z[:] = data['Z']
        ZI = interpolate.griddata(pos,Z,(XI,YI),method='nearest')

        ## Géneration de l'image
        fig, ax = plt.subplots()

        plt.axis('off')
        ax.set_axis_off()
        ax.set_aspect('equal')
        fig.add_axes(ax)
        im = ax.pcolormesh(XI,YI,ZI,cmap=CM.gray)
        ## Ajout d'un patch pour effacer l'extérieur
        ##    du cercle créer par l'interpolation
        patch1 = patches.Circle((0, 0), radius=147, transform=ax.transData)
        im.set_clip_path(patch1)

        plt.savefig(os.path.join(csv_dir,name) + '.png')

        plt.close()
```

3. Optimisation des paramètres de la transformée de Hough (Chapitre III)

```
#########################################################################
##        Optimisation des différents paramètres permettant            ##
## la détection de lignes par transformée de Hough à l'aide de sliders ##
#########################################################################

## Importation des librairies
import cv2 as cv
import numpy as np
import math

def nothing():
    pass

if __name__ == '__main__':

    ## Chemin vers image
    img_path = r"Path/to/img.png"
    ## Lecture de l'image
    src = cv.imread(img_path)
    gray_img = cv.cvtColor(src, cv.COLOR_BGR2GRAY)

    ## Création de la fenêtre pour la detection des bords (Edge detection)
           par Canny
    ## Création des sliders pour ajustement du kernel de flou Gaussien et
           des tresholds de Canny
    cv.namedWindow("bar")
    cv.createTrackbar("kernel_size", "bar", 1, 50, nothing)
    cv.createTrackbar("low_threshold", "bar", 1, 200, nothing)
    cv.createTrackbar("high_threshold", "bar", 1, 500, nothing)


    dst = cv.equalizeHist(gray_img)
    while True:
        ## Itérations du flou Gaussien et de Canny avec les valeurs
               choisies sur les sliders
        kernel_size = 2*cv.getTrackbarPos("kernel_size", "bar") + 1
        low_threshold = cv.getTrackbarPos("low_threshold", "bar")
        high_threshold = cv.getTrackbarPos("high_threshold", "bar")
        blur_gray = cv.GaussianBlur(gray_img,(kernel_size, kernel_size),0)
        edges = cv.Canny(blur_gray, low_threshold, high_threshold)

        ## Changement de taille de l'image pour affichage
        scale_percent = 30
        width = int(edges.shape[1] * scale_percent / 100)
        height = int(edges.shape[0] * scale_percent / 100)
        dim = (width, height)
        edges = cv.resize(edges, dim, interpolation = cv.INTER_AREA)
        cv.imshow("edges", edges)
        if cv.waitKey(1) & 0xFF == 27:
            break
    cv.destroyAllWindows()

    edges = cv.Canny(blur_gray, low_threshold, high_threshold)


    ## Résolution angulaire de la grille de Hough
```

```
theta = np.pi / 250

## Création de la fenêtre pour la detection des lignes par transformée
    de Hough
## Création des sliders pour ajustement de Hough
cv.namedWindow('bar',cv.WINDOW_AUTOSIZE)
cv.createTrackbar("rho", "bar", 1, 15, nothing)
cv.createTrackbar("threshold", "bar", 10, 350, nothing)
cv.createTrackbar("min_line_length", "bar", 10, 120, nothing)
cv.createTrackbar("max_line_gap", "bar", 50, 120, nothing)
dst = cv.equalizeHist(gray_img)

while True:
    ## Itérations de la transformée de Hough avec les valeurs choisies
            par les sliders
    threshold = cv.getTrackbarPos("threshold", "bar")
    rho = cv.getTrackbarPos("rho", "bar")
    min_line_length = cv.getTrackbarPos("min_line_length", "bar")
    max_line_gap = cv.getTrackbarPos("max_line_gap", "bar")
    line_image = np.copy(dst) * 0

    lines = cv.HoughLinesP(edges, rho, theta, threshold, np.array([]),
                           min_line_length, max_line_gap)

    ## Filtrage des lignes détectées en fonction de leur angle (à
          définir selon l'orientation cristalline)
    sliplines = []
    for line in lines:
            for x1,y1,x2,y2 in line:
                angle = math.atan2(x2-x1,y2-y1)*180/math.pi
                if (44.0 <angle < 46.0 or 134.0 <angle < 136.0):
                        cv.line(line_image,(x1,y1),(x2,y2),(255,0,0),2)
                        sliplines.append((float(x1), float(y1),
                                          float(x2), float(y2)))
                        lines_edges = cv.addWeighted(dst, 0.8,
                                                     line_image, 1, 0)

    ## Changement de taille de l'image pour affichage
    scale_percent = 40
    width = int(lines_edges.shape[1] * scale_percent / 100)
    height = int(lines_edges.shape[0] * scale_percent / 100)
    dim = (width, height)
    lines_edges = cv.resize(lines_edges, dim, interpolation =
                            cv.INTER_AREA)
    cv.imshow("lines_edges", lines_edges)
    if cv.waitKey(1) & 0xFF == 27:
        break

    cv.destroyAllWindows()
```

4. Détection des lignes par transformée de Hough (Chapitre III)

```
#########################################################################
##        Détection de lignes par transformée de Hough à partir        ##
##     des paramètres optimisés déterminés par le script à sliders     ##
#########################################################################

## Importation des librairies
import cv2
import numpy as np
import math
import glob
import os
import pandas as pd

if __name__ == '__main__':

    imgs_dir = r'Path/to/imgs'

    ## Lecture des images (format jpg) contenues dans le dossier
    for file in glob.glob(imgs_dir + "/*.jpg"):

        name = os.path.basename(file)[:-4]
        ## Lecture de l'image
        img = cv2.imread(file)
        gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)

        ## Flou Gaussien et détection des bords (Canny)
        kernel_size = 3
        blur_gray = cv2.GaussianBlur(gray,(kernel_size, kernel_size),0)
        low_threshold = 20
        high_threshold = 105
        edges = cv2.Canny(blur_gray, low_threshold, high_threshold)

        ## Paramètres pour la transformée de Hough
        rho = 2
        theta = np.pi / 1000
        threshold = 250
        min_line_length = 100
        max_line_gap = 35
        line_image = np.copy(img) * 0

        lines = cv2.HoughLinesP(edges, rho, theta, threshold, np.array([]),
                                min_line_length, max_line_gap)

        ## Filtrage des lignes détectées en fonction de leur angle
        ##     (à définir selon l'orientation cristalline)
        sliplines = []
        for line in lines:
            for x1,y1,x2,y2 in line:
                angle = math.atan2(x2-x1,y2-y1)*180/math.pi
                d_milieu = math.sqrt(((x2+x1)/2 - 1500)**2 + ((y2+y1)/2 -
                                     1500)**2)
                if (44.0 <angle < 46.0 or 134.0 <angle < 136.0) and
                                                    d_milieu < 1475:
                    cv2.line(line_image,(x1,y1),(x2,y2),(255,0,0),2)
                    sliplines.append((float(x1), float(y1), float(x2),
                                                        float(y2)))
```

```
        lines_edges = cv2.addWeighted(img, 0.8, line_image, 1, 0)

        ## Sauvegarde de l'image avec détection des lignes
        cv2.imwrite(os.path.join(imgs_dir,name) + "_detection_output.png",
                    lines_edges)

        ## Calcul des longueurs et surfaces des lignes détectées
        Ltot = 0
        Stot = 0
        Swafer = math.pi*300**2
        for line in sliplines:
            x1,y1,x2,y2 = line
            Ltot += (math.sqrt((x2-x1)**2+(y2-y1)**2))
            Stot += abs((x2-x1)/100*(y2-y1)/100)
        Ltot = Ltot/100
        Def_area = Stot/Swafer*100
        Nbtot = len(sliplines)

        ## Création d'un DataFrame pour stockage des données
        if file == glob.glob(imgs_dir + "/*.jpg")[0] :
            df = pd.DataFrame([[name,Nbtot,Ltot,Def_area]],
                              columns=['Wafer','Nb tot', 'Long. tot
                                                  [mm]','Def area [%]}'])
        else:
            df2 = pd.DataFrame(([[name, Nbtot, Ltot, Def_area]]),
                               columns=['Wafer', 'Nb tot', 'Long. tot
                                                   [mm]', 'Def area [%]'])
            df = df.append(df2,ignore_index=True)

    ## Sauvegarde des données extraites de la détection des lignes
    df.to_excel(imgs_dir + "/Sliplines_data.xlsx",index=False)
```

5. Découpe des images ALTAIR en puces individuelles

```
#################################################################
#           Script pour génération des images de puces          #
#         individuelles à partir des acquisitions ALTAIR        #
#################################################################

## Importation des librairies
import PIL.Image
import PIL.ImageEnhance
import pandas as pd
import os
import glob

if __name__ == '__main__':
    ## Choix du dossier contenant les cartographies ALTAIR
    imgs_dir = r'Path/to/imgs'

    ## Lecture du fichier contenant les coordonnées des puces
    coords = pd.read_excel(r"Path/to/coords.xlsx",header=None,
                           skiprows=1, usecols=[0,1],
                           names=['X','Y']).values

    ## Création du dossier "Dies" pour stocker les puces indiv.
    if not os.path.exists(imgs_dir+"/Dies"):
        os.mkdir(imgs_dir+"/Dies")

    ## Traitement des images (format jpg) contenues dans le dossier
    for file in glob.glob(imgs_dir + "/*.jpg"):

        name = os.path.splitext(os.path.basename(file))[0]

        ##Ouverture de l'image
        colorImage  = PIL.Image.open(file)

        ## Itération sur les coordonnées pour découpe de l'image
        for X,Y in coords:
                ## Extraction de la puce dans l'image à l'aide des
                       coordonnées
                left = 970 + 57*X
                right = 1025 + 57*X
                up = 990 - 55*Y
                down = 1038 - 55*Y

                ## Découpe de la die et amélioration de l'image (contraste,
                     luminosité, etc.)
                cropped = PIL.Image.Image.crop(colorImage,
                                               box=(left,up,right,down))
                enhancer = PIL.ImageEnhance.Sharpness(cropped)
                cropped = enhancer.enhance(1)
                cropped = PIL.ImageEnhance.Contrast(cropped).enhance(2.5)
                cropped = PIL.ImageEnhance.Brightness(cropped).enhance(1.2)
                cropped = cropped.convert('LA')

                ## Sauvegarde de l'image avec informations de localisation
                cropped.save(imgs_dir + 'Dies/' + name + '_' + str(X) + '_'
                             + str(Y) + '.png')

        print("Traitement du dossier %s terminé"%name)
```

6. Génération des images de puces à partir des données PWG

```
#################################################################
#           Script pour génération des images de puces          #
#           individuelles à partir des acquisitions PWG         #
#             avec soustraction de la puce moyenne              #
#################################################################

## Importation des librairies
import pandas as pd
import numpy as np
import matplotlib
matplotlib.use('Agg')
import matplotlib.pyplot as plt
from matplotlib import cm as CM
from scipy import interpolate
import os
import glob

if __name__ == '__main__':
    plt.ioff()

    ## Lecture du fichier contenant les coordonnées des puces
    coords = pd.read_excel(r'Path/to/xlsx',header=None,
                           skiprows=1, usecols=[0,1], names=['X','Y'])

    ## Chemin vers le dossier contenant les données PWG
    data_path = r'Path/to/csv'

    ## Itération sur les données récoltées sur chaque wafer
    for filename in (glob.glob(data_path + '/*.csv')):

        print(os.path.basename(filename))
        ## Récupération du numéro du wafer
        wafer = os.path.basename(filename).split('W')[1][:2]

        ## Création du dossier "Dies" pour stocker les puces indiv.
        if not os.path.exists(data_path+"/Dies"):
            os.mkdir(data_path+"/Dies")

        data = pd.read_csv(filename,header=None,delimiter=',', skiprows=47,
                           usecols=[0,1,2], names=['X','Y','Z'])
        data = data.dropna()


        ## Moyennage des 36 puces centrales pour soustraction de la puce
               moyenne
        k = 0
        for i in range(-3,4):
            for j in range(-3,4):
                print(i,j)
                x_min = -5.55 + 8.43*i
                x_max = 0.28  + 8.43*i
                y_min = -1.2  + 8.12*j
                y_max = 2.4   + 8.12*j
```

```
        data0 = data.loc[(data['Y'] >= y_min) & (data['Y'] <=
                                                 y_max)]
        data1 = data0.loc[(data['X'] >= x_min) &
                          (data['X'] <=
                                  x_max)].reset_index(drop=True)

        xmi = min(data1['X'])
        xMa = max(data1['X'])
        ymi = min(data1['Y'])
        yMa = max(data1['Y'])

        positions = np.zeros((len(data1),2))
        Z = np.zeros(len(data1))
        positions[:,0] = data1.values[:,0] - xmi
        positions[:,1] = data1.values[:,1] - ymi
        Z[:] = data1.values[:,2]


        tix = np.linspace(0,(xMa-xmi),600)
        tiy = np.linspace(0,(yMa-ymi),600)
        XI,YI = np.meshgrid(tix,tiy)
        ZI = interpolate.griddata(positions, Z, (XI,YI),
                                  method='cubic')

        if k == 0:
            ZI_sum = ZI
        else:
            ZI_sum +=ZI
        k+=1

## Puce moyenne à soustraire pour faire ressortir la signature de
     striations
ZI_sum = ZI_sum/k

## Itération sur les coordonnées pour géneration
## des images de puces individuelles
fig, ax = plt.subplots()
plt.axis('off')
for i in range(len(coords)):
        ## Extraction des données de la puce à l'aide des
                                                 coordonnées
        x0 = -5.55 + 8.43*coords['X'][i]
        x1 = 0.28  + 8.43*coords['X'][i]
        y0 = -1.2  + 8.12*coords['Y'][i]
        y1 = 2.4   + 8.12*coords['Y'][i]

        ## Mise à jour des data en Y que si Y change pour gain de
                                                        temps
        if(i==0 or coords['Y'][i-1]!=coords['Y'][i]):
            data0 = data.loc[(data['Y'] >= y0) & (data['Y'] <= y1)]

        data1 = data0.loc[(data['X'] >= x0) & (data['X'] <= x1)]

        xmi = min(data1['X'])
        xMa = max(data1['X'])
        ymi = min(data1['Y'])
        yMa = max(data1['Y'])

        positions = np.zeros((len(data1),2))
```

```
            Z = np.zeros(len(data1))
            positions[:,0] = data1.values[:,0] - xmi
            positions[:,1] = data1.values[:,1] - ymi
            Z[:] = data1.values[:,2]

            tix = np.linspace(0,(xMa-xmi),600)
            tiy = np.linspace(0,(yMa-ymi),600)
            XI,YI = np.meshgrid(tix,tiy)
            ZI = interpolate.griddata(positions, Z, (XI,YI),
                                                    method='cubic')
            ZI = ZI - ZI_sum

            ax.cla()
            ax = plt.Axes(fig, [0., 0., 1., 1.])
            ax.set_axis_off()
            fig.add_axes(ax)
            ## Génération de l'image de puce et sauvegarde
            im = ax.pcolormesh(XI,YI,ZI,cmap=CM.gray)
            plt.savefig(data_path+'/Dies/%s@%.0f,%.0f.png'% (wafer,
                coords['X'][i], coords['Y'][i]), bbox_inches = 'tight',
                                                    pad_inches = 0)
            plt.pause(0.01)
            fig.clf()
    plt.show()
```

7. Entrainement du CNN sur images de puces

```
#########################################################################
##           Entrainement du CNN à partir des images de puces          ##
##         ALTAIR ou PWG, pour classification des puces striées        ##
#########################################################################

## Importation des librairies
import matplotlib.pyplot as plt
import os
import tensorflow as tf

if __name__ == '__main__':

    ## Chemin vers les dossiers contenant les images
    Img_path = "Path/to/imgs/folders"
    train_dir = os.path.join(Img_path, 'train')
    validation_dir = os.path.join(Img_path, 'validation')

    ## Nombre d'images envoyé simultanément dans le CNN
    BATCH_SIZE = 32

    ## Définition de la taille (en pixels) des images fournies
    IMG_SIZE = (54, 48) ## PWG : (640, 480)     ALTAIR : (54, 48)

    ## Création des datasets
    train_dataset = tf.keras.utils.image_dataset_from_directory(train_dir,

shuffle=True,

batch_size=BATCH_SIZE,

image_size=IMG_SIZE)

    validation_dataset =
tf.keras.utils.image_dataset_from_directory(validation_dir,

shuffle=True,

batch_size=BATCH_SIZE,

image_size=IMG_SIZE)

    class_names = train_dataset.class_names

    val_batches = tf.data.experimental.cardinality(validation_dataset)
    test_dataset = validation_dataset.take(val_batches // 5)
    validation_dataset = validation_dataset.skip(val_batches // 5)

    print('Number of validation batches: %d' %
tf.data.experimental.cardinality(validation_dataset))
    print('Number of test batches: %d' %
tf.data.experimental.cardinality(test_dataset))

    AUTOTUNE = tf.data.AUTOTUNE

    train_dataset = train_dataset.prefetch(buffer_size=AUTOTUNE)
    validation_dataset = validation_dataset.prefetch(buffer_size=AUTOTUNE)
```

```
    test_dataset = test_dataset.prefetch(buffer_size=AUTOTUNE)

    ## Augmentation des données
    data_augmentation = tf.keras.Sequential([
      tf.keras.layers.RandomFlip('horizontal_and_vertical'),
      tf.keras.layers.Contrast(0.95,1.05),
      tf.keras.layers.Brightness(0.95,1.05)
    ])

    preprocess_input = tf.keras.applications.mobilenet_v2.preprocess_input

    IMG_SHAPE = IMG_SIZE + (3,)

    ## Transfer learning à partir de ResNet : Entrainement des dernières
couches

    base_model = tf.keras.applications.ResNet(input_shape=IMG_SHAPE,
                                                  include_top=False,
                                                  weights='imagenet')

    image_batch, label_batch = next(iter(train_dataset))
    feature_batch = base_model(image_batch)
    base_model.trainable = False

    global_average_layer = tf.keras.layers.GlobalAveragePooling2D()
    feature_batch_average = global_average_layer(feature_batch)
    print(feature_batch_average.shape)

    prediction_layer = tf.keras.layers.Dense(1)
    prediction_batch = prediction_layer(feature_batch_average)

    inputs = tf.keras.Input(shape=(54, 48, 3))
    x = data_augmentation(inputs)
    x = preprocess_input(x)
    x = base_model(x, training=False)
    x = global_average_layer(x)
    x = tf.keras.layers.Dropout(0.2)(x)
    outputs = prediction_layer(x)
    model = tf.keras.Model(inputs, outputs)
    callback = tf.keras.callbacks.EarlyStopping(monitor='val_loss',
patience=20)
    base_learning_rate = 0.0001

    ## Compilation du modèle

model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=base_learnin
g_rate),

loss=tf.keras.losses.BinaryCrossentropy(from_logits=True),
                  metrics=['accuracy'])

    initial_epochs = 500

    loss0, accuracy0 = model.evaluate(validation_dataset)

    print("initial loss: {:.2f}".format(loss0))
    print("initial accuracy: {:.2f}".format(accuracy0))

    ## Entrainement du modèle
```

```
    history = model.fit(train_dataset,
                        epochs=initial_epochs,
                        callbacks=[callback],
                        validation_data=validation_dataset)

    print(history.history.keys())

    ## Récapitulatif de l'entrainement et dessin des courbes

    plt.plot(history.history['accuracy'])
    plt.plot(history.history['val_accuracy'])
    plt.show()

    plt.subplots()
    plt.plot(history.history['loss'])
    plt.plot(history.history['val_loss'])
    plt.title('model loss')
    plt.ylabel('loss')
    plt.xlabel('epoch')
    plt.legend(['train', 'test'], loc='upper left')
    plt.show()

    ## Transfer learning : Entrainement premières couches (Léger)
    base_model.trainable = True

    fine_tune_at = 100

    for layer in base_model.layers[:fine_tune_at]:
      layer.trainable = False


model.compile(loss=tf.keras.losses.BinaryCrossentropy(from_logits=True),
                  optimizer =
tf.keras.optimizers.RMSprop(learning_rate=base_learning_rate/10),
                  metrics=['accuracy'])

    fine_tune_epochs = 10
    total_epochs =  initial_epochs + fine_tune_epochs

    ## Entrainement du modèle
    history_fine = model.fit(train_dataset,
                             epochs=total_epochs,
                             initial_epoch=history.epoch[-1],
                             validation_data=validation_dataset)

    ## Récapitulatif de l'entrainement et dessin des courbes

    plt.plot(history.history['accuracy'])
    plt.plot(history.history['val_accuracy'])
    plt.show()
    # summarize history for loss
    plt.subplots()
    plt.plot(history.history['loss'])
    plt.plot(history.history['val_loss'])
    plt.title('model loss')
    plt.ylabel('loss')
    plt.xlabel('epoch')
    plt.legend(['train', 'test'], loc='upper left')
    plt.show()
```

```
    model.save('Models/CNN')

    i+=1

import pandas as pd

data = pd.DataFrame(list(zip(Wl, Xl, Yl, PNNl)), columns
=['Wafer','X','Y','Class'])
data.to_excel('Res_CNN.xlsx')
```

8. Inférence du CNN sur les images de puces ALTAIR ou PWG

```
##########################################################
##          Inférence à partir du CNN entrainé.         ##
##   Les images traitées sont classées dans le dossier   ##
##            correspondant à leur classe               ##
##########################################################

## Importation des librairies
import fastai
import fastai.vision as fv
import glob
import os

## Définition du "Learner" permettant l'inférence
def getLearner(directoryName: str, fileName: str) -> fastai.train.Learner:
    return fv.load_learner(path=fv.Path(directoryName),fname=fileName)

if __name__ == '__main__':
    ## Chemin vers le dossier contenant le modèle
    model_path = r'model/to/path'
    ## Nom du fichier du modèle
    model_name = r'nom_du_modèle'

    ## Chemin vers dossier contenant les images
    imgs_path = r'Path/to/imgs'

    ## Importation du modèle
    model = getLearner(directoryName = model_path,fileName = model_name)

    ## Création du fichier de résultat
    fichier_resultats = open(imgs_path + "\\" + model_name.split(".")[0] +
".csv", "w")

    ## Liste des images (format png) contenues dans le dossier
    List_images = glob.glob(os.path.join(imgs_path,"*.png"))

    ## Itération sur les images
    for file in List_images:

        ## Récupération du nom de l'image uniquement
        name = os.path.basename(file)

        ## Ouverture de l'image
        img = fv.open_image(file)

        ## Prédiction à l'aide du modèle entrainé
        [cat, indice, preds] = model.predict(img)

        ## Ecriture du résultat dans le fichier csv
        fichier_resultats.write(name + ',' + str(cat) + ',' + str(indice) +
',' + str(preds) + '\n' )

        ## Créations des dossiers de sauvegarde des images par classe
        class_folder = imgs_path + + "\\"+ str(cat).split(" ")[-1] + "\\"
        if not os.path.exists(class_folder):
            os.makedirs(class_folder)
```

```
## Déplacent de l'image dans le dossier cible
NewPath_file = os.path.join(class_folder,name)
os.rename(file,NewPath_file)
```

9. Stacking des images MPL

```
##########################################################################
##                    Stacking des images Macro PL pour                 ##
##             détermination des taches communes et atypiques           ##
##########################################################################

## Importation des librairies
import cv2
import glob

if __name__ == '__main__':

    ## Chemin vers dossier contenant les images
    imgs_dir = r'Path/to/imgs'

    ## Seuil (en niveaux de gris) sous lequel les données des images sont
conservées
    seuil = 65

    ## Initialisation à partir de la 1ère image du dossier
    file0 = glob.glob(imgs_dir+'/*.png')[0]
    im0 = cv2.imread(file0,0)
    im0 = cv2.normalize(im0, None, alpha=0, beta=255,
norm_type=cv2.NORM_MINMAX)

    ## Seuillage de l'image et normalisation
    ret,th0 = cv2.threshold(im0,seuil,255,cv2.THRESH_TOZERO_INV)
    im0 = cv2.normalize(th0, None, alpha=0, beta=255,
norm_type=cv2.NORM_MINMAX)

    ## Itérations sur toutes les images (format png) du dossier
    for file in glob.glob(imgs_dir+'/*.png'):
        if (file==file0):
            continue
        ## Seuillage de l'image et normalisation
        im1 = cv2.imread(file,0)
        im1 = cv2.normalize(im1, None, alpha=0, beta=255,
norm_type=cv2.NORM_MINMAX)
        ret,th1 = cv2.threshold(im1,seuil,255,cv2.THRESH_TOZERO_INV)
        im1 = cv2.normalize(th1, None, alpha=0, beta=255,
norm_type=cv2.NORM_MINMAX)

        ## Ajout de l'image au stack et normalisation
        Image_add = cv2.absdiff(im0,im1)
        im0 = Image_add
        im0 = cv2.normalize(im0, None, alpha=0, beta=255,
norm_type=cv2.NORM_MINMAX)

    ## Sauvegarde de l'image stackée
    name = imgs_dir + '/Stack.png'
    cv2.imwrite(name,im0)
```

10.Entrainement d'un RCNN sur images MPL ou SAM

```
#########################################################################
##      Entrainement d'un RCNN (via GPU) en utilisant detectron2       ##
##      à partir d'annotations COCO, sur une base "faster_rcnn"        ##
#########################################################################

## Importation des librairies
import detectron2
from detectron2.utils.logger import setup_logger
setup_logger()
import os
import torch, gc
gc.collect()
torch.cuda.empty_cache()
from detectron2 import model_zoo
from detectron2.config import get_cfg
from detectron2.data import MetadataCatalog
from detectron2.data.catalog import DatasetCatalog
from detectron2.data import transforms as T
from detectron2.data.datasets import register_coco_instances
from detectron2.engine import DefaultTrainer
from detectron2.evaluation import COCOEvaluator
from detectron2.data.build import build_detection_train_loader
from detectron2.data import DatasetMapper

## Créations des dictionnaires, avec fichier d'annotations COCO
##          et chemin vers les images annotées
register_coco_instances("Nom_du_dataset_de_training", {},
                            "Path/to/json",
                                "Path/to_imgs")

register_coco_instances("Nom_du_dataset_de_validation", {},
                            "Path/to/json",
                                "Path/to_imgs")

my_dataset_train_metadata =
MetadataCatalog.get("Nom_du_dataset_de_training")
dataset_dicts = DatasetCatalog.get("Nom_du_dataset_de_training")

my_dataset_val_metadata =
MetadataCatalog.get("Nom_du_dataset_de_validation")
dataset_dicts_val = DatasetCatalog.get("Nom_du_dataset_de_validation")

## Création du "Trainer" pour gérer l'entrainement (Augmentation,
Evaluation, etc.)
class CocoTrainer(DefaultTrainer):
    def build_train_loader(cls, cfg):
            if "GeneralizedRCNN" in cfg.MODEL.META_ARCHITECTURE:
                mapper = DatasetMapper(cfg, is_train=True,
                                       augmentations=build_sem_seg_train_aug(cfg))
            else:
                mapper = None
            return build_detection_train_loader(cfg, mapper=mapper)
    @classmethod
    def build_evaluator(cls, cfg, dataset_name, output_folder=None):
        if output_folder is None:
                    os.makedirs("coco_eval", exist_ok=True)
```

```
                output_folder = "coco_eval"
        return COCOEvaluator(dataset_name, cfg, False, output_folder)

## Définition des augmentations à faire sur les images
def build_sem_seg_train_aug(cfg):
    augs = [T.ResizeShortestEdge(cfg.INPUT.MIN_SIZE_TRAIN,
                  cfg.INPUT.MAX_SIZE_TRAIN,
cfg.INPUT.MIN_SIZE_TRAIN_SAMPLING)]
    augs.append(T.RandomFlip(prob=0.5))
    augs.append(T.RandomBrightness(0.9,1.1))
    augs.append(T.RandomRotation(-1,1))
    return augs

## Définiton des paramètres de l'entrainement
## (Modèle de base, Epochs, Learning Rate, etc.)
cfg = get_cfg()
os.makedirs(cfg.OUTPUT_DIR, exist_ok=True)
cfg.merge_from_file(model_zoo.get_config_file("COCO-
Detection/faster_rcnn_X_101_32x8d_FPN_3x.yaml"))
cfg.DATASETS.TRAIN = ("my_dataset_train",)
cfg.DATASETS.TEST = ("my_dataset_val",)
cfg.DATALOADER.NUM_WORKERS = 3
cfg.MODEL.WEIGHTS = model_zoo.get_checkpoint_url("COCO-
Detection/faster_rcnn_X_101_32x8d_FPN_3x.yaml")
cfg.SOLVER.IMS_PER_BATCH = 5
cfg.SOLVER.BASE_LR = 0.005
cfg.SOLVER.GAMMA = 0.0002
cfg.SOLVER.MAX_ITER = 5000
cfg.SOLVER.STEPS = (1000,2000,3000,4000)
cfg.INPUT.MIN_SIZE_TRAIN = 1500
cfg.INPUT.MAX_SIZE_TRAIN = 3000
cfg.MODEL.RPN.POST_NMS_TOPK_TRAIN = 25000
cfg.MODEL.RPN.PRE_NMS_TOPK_TRAIN = 25000
cfg.MODEL.ROI_HEADS.NUM_CLASSES = 8 ## Nombre de classes d'objets
cfg.TEST.EVAL_PERIOD = 250

## Lancement de l'entrainement, et sauvegarde du modèle entrainé
trainer = CocoTrainer(cfg)
trainer.resume_or_load(resume=False)
```

11.Inférence du RCNN entrainé

```
################################################################
##    Inférence à partir du RCNN entrainé, avec sortie        ##
##      d'une image annotée, et d'un KLARF contenant          ##
##        les informations sur les objets détectées           ##
################################################################

## Importation des librairies
import detectron2
import os
from detectron2.utils.logger import setup_logger
setup_logger()
import numpy as np
import cv2
import glob
import matplotlib
matplotlib.use('Agg')
import matplotlib.pyplot as plt
from detectron2.data import DatasetCatalog, MetadataCatalog
from detectron2.evaluation import COCOEvaluator
from detectron2 import model_zoo
from detectron2.engine import DefaultPredictor,DefaultTrainer
from detectron2.config import get_cfg
from detectron2.utils.visualizer import Visualizer
from detectron2.data.datasets import register_coco_instances
from detectron2.utils.visualizer import ColorMode

## Créations des dictionnaires, avec fichier d'annotations COCO
##          et chemin vers les images annotées
register_coco_instances("Nom_du_dataset_de_training", {},
                            "Path/to/json",
                                "Path/to_imgs")
register_coco_instances("Nom_du_dataset_de_validation", {},
                            "Path/to/json",
                                "Path/to_imgs")

my_dataset_train_metadata =
MetadataCatalog.get("Nom_du_dataset_de_training")
dataset_dicts = DatasetCatalog.get("Nom_du_dataset_de_training")

my_dataset_val_metadata =
MetadataCatalog.get("Nom_du_dataset_de_validation")
dataset_dicts_val = DatasetCatalog.get("Nom_du_dataset_de_validation")


## Création du "Trainer" (nécessaire pour l'infence même si non utilisé)
class CocoTrainer(DefaultTrainer):
  @classmethod
  def build_evaluator(cls, cfg, dataset_name, output_folder=None):
    if output_folder is None:
        os.makedirs("coco_eval", exist_ok=True)
        output_folder = "coco_eval"
    return COCOEvaluator(dataset_name, cfg, False, output_folder)

## Définiton des paramètres utilisé pendant l'entrainement
## (Modèle de base, Epochs, Learning Rate, etc.)
cfg = get_cfg()
os.makedirs(cfg.OUTPUT_DIR, exist_ok=True)
```

```
cfg.merge_from_file(model_zoo.get_config_file("COCO-
Detection/faster_rcnn_X_101_32x8d_FPN_3x.yaml"))
cfg.MODEL.WEIGHTS =
os.path.join('./Models/faster_rcnn_X_101_32x8d_FPN_3x',"Path/to/trained_mod
el.pth")
cfg.DATASETS.TRAIN = ("my_dataset_train",)
cfg.DATASETS.TEST = ("my_dataset_val",)
cfg.DATALOADER.NUM_WORKERS = 3
cfg.SOLVER.IMS_PER_BATCH = 5
cfg.SOLVER.BASE_LR = 0.0050
cfg.SOLVER.GAMMA = 0.0002
cfg.SOLVER.MAX_ITER = 5000
cfg.SOLVER.STEPS = (1000,2000,3000,4000)
cfg.INPUT.MIN_SIZE_TRAIN = 1500
cfg.INPUT.MAX_SIZE_TRAIN = 3000
cfg.MODEL.RPN.POST_NMS_TOPK_TRAIN = 25000
cfg.MODEL.RPN.PRE_NMS_TOPK_TRAIN = 25000
cfg.MODEL.ROI_HEADS.NUM_CLASSES = 8
cfg.TEST.EVAL_PERIOD = 250

## Création du "predictor" permettant l'inférence
## à partir du RCNN entrainé
predictor = DefaultPredictor(cfg)

## Chemin vers dossier contenant les images à inférer
path_imgs = r'Path/to/imgs'

## Itération sur les images (format bmp) contenues dans le dossier
for imageName in glob.glob(path_imgs + '/*.bmp'):
    nm = os.path.basename(imageName)
    plt.cla()

    ## Lecture de l'image et inférence à partir du modèle entrainé
    im = cv2.imread(imageName)
    outputs = predictor(im)

    ## Création de l'image pour visualisation de l'inférence
    im_cropped = cv2.imread(imageName)
    x,y = 1000,1000
    mask = np.zeros(im_cropped.shape, dtype=np.uint8)
    cv2.circle(mask, (x,y), 990, (255,255,255), -1)
    ROI = cv2.bitwise_and(im, mask)
    mask = cv2.cvtColor(mask, cv2.COLOR_BGR2GRAY)
    x,y,w,h = cv2.boundingRect(mask)
    result = ROI[y:y+h,x:x+w]
    mask = mask[y:y+h,x:x+w]
    result[mask==0] = (255,255,255)

    v = Visualizer(im[:, :, ::-1], metadata=my_dataset_train_metadata,
                   scale=1,instance_mode=ColorMode.SEGMENTATION)
    out = v.draw_instance_predictions(outputs["instances"].to("cpu"))
    plt.axis('off')
    plt.imshow(out.get_image()[:, :, ::-1])
    ## Sauvegarde de l'image annotée
    plt.savefig(imageName + '_Result',bbox_inches='tight',
pad_inches=0,dpi=500)

    ## Récupération des données des objets obtenus par prédiction
    predictions = outputs["instances"].to("cpu").pred_classes[:].numpy()
```

```
    ## Ouverture du fichier de base pour le KLARF
    ##     et écriture des données obtenues
    File_object = open(r'klarf_void.txt','w+')
    File_object.write('DefectRecordSpec 6 DEFECTID XREL YREL XSIZE YSIZE
CLASSNUMBER ;\n')
    File_object.write('DefectList\n')
    XC = 3050
    YC = 3050

    ## Itérations sur les objets détectés
    for K in range(len(predictions)):
        X1,Y1,X2,Y2 =
outputs["instances"].pred_boxes.tensor.cpu().numpy()[K].tolist()
        Class = outputs["instances"].to("cpu").pred_classes[K].numpy()+1
        XREL = (X1 - XC)/20.
        YREL = (YC - Y2)/20.
        XSIZE = abs(X1-X2)/20.
        YSIZE = abs(Y1-Y2)/20.
        File_object.write(str(K+1) +' '+ str(XREL) +' '+
    str(YREL) + ' ' + str(XSIZE) + ' ' + str(YSIZE) + ' ' + str(Class)+'
\n')
    File_object.close()

    plt.pause(0.01)
plt.show()
```

12.Génération d'image ellipsométrique et correction des aberrations

```
############################################################
##         Traitement des données RASTER pour             ##
##     corréctions des abbérations de déplacement         ##
############################################################

## Importation des librairies
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import cm as CM
import tkinter as tk
from tkinter.filedialog import askopenfilename
from scipy import interpolate
import scipy.optimize as optimization
from scipy.special import expit

if __name__ == '__main__':

    ########################################
    ##   Choix du répertoire de travail   ##
    ########################################

    root = tk.Tk()
    root.withdraw()
    filename = askopenfilename()

    ########################################################
    ##   Lecture du fichier contenant les coordonnées   ##
    ########################################################

    positions_init = pd.read_csv(filename, sep='\t', header = None,
skiprows = 1,usecols=[2,3], names=['X','Y'])
    positions = pd.read_csv(filename, sep='\t', header = None, skiprows =
1,usecols=[2,3], names=['X','Y'])
    IsIc = pd.read_csv(filename, sep='\t',header = None, skiprows =
1,usecols=[4,5,6], names=['Is','Ic','S0'])
    IsIc = pd.read_csv(filename, sep='\t',header = None, skiprows =
1,usecols=[4,5,6], names=['Is','Ic','S0'])


    time = pd.read_csv(filename, sep='\t', header = None, skiprows =
1,usecols=[1], names=['ms'])
    t=time['ms'][:]
    mint=min(t)
    maxt=max(t)

    ## Positions du 1er champ
    positions_champ1=positions[positions['X']<-60]
    positions_champ1=positions_champ1[positions_champ1['Y']<30]


    K = len(set(positions_init['Y'][:]))
    L = len(positions_init)
    M=len(set(positions_init['X'][:]))


    positions_corrected=np.zeros(L)
```

```
positions_corrected=positions['X']

Is_corrected=IsIc['Is']
Is_corrected[Is_corrected>1]=1
Is_corrected[Is_corrected<-1]=-1

Ic_corrected=IsIc['Ic']
Ic_corrected[Ic_corrected>1]=1
Ic_corrected[Ic_corrected<-1]=-1

Is=np.zeros(L)
Is[:]=IsIc['Is'][:]
Ic=np.zeros(L)
Ic[:]=IsIc['Ic'][:]
S0=np.zeros(L)
S0[:]=IsIc['S0'][:]

positions_bis=positions
IsIc_bis=IsIc

shift_backup=np.zeros(L)

line_0=np.zeros(int(L/K))
Is_0=np.zeros(int(L/K))
Ic_0=np.zeros(int(L/K))

line_1=np.zeros(int(L/K))
Is_1=np.zeros(int(L/K))
Ic_1=np.zeros(int(L/K))

line_2=np.zeros(int(L/K))
Is_2=np.zeros(int(L/K))
Ic_2=np.zeros(int(L/K))

x_list_init = positions_init['X'][:]
y_list_init = positions_init['Y'][:]

minX_init=min(x_list_init)
maxX_init=max(x_list_init)
Champ_init=maxX_init-minX_init
x_center=(maxX_init-abs(minX_init))/2

minY_init=min(y_list_init)
maxY_init=max(y_list_init)

##############################################
##   Traitement unique des lignes paires    ##
##############################################

k=int(0)
n_tot=1
n_tot=int(K/2-1)
shift_trace=np.zeros(n_tot)
a_trace=np.zeros(n_tot)
b_trace=np.zeros(n_tot)
c_trace=np.zeros(n_tot)

ad_trace=np.zeros(n_tot)
xd_trace=np.zeros(n_tot)
```

```
    sigmad_trace=np.zeros(n_tot)

    ag_trace=np.zeros(n_tot)
    xg_trace=np.zeros(n_tot)
    sigmag_trace=np.zeros(n_tot)

    error_trace=np.zeros(n_tot)

    shift_opt_1=0

    for k in range(0,n_tot):


        m=int(2*k)

        if m<0:m=0

        print('Lines with multiguess=', m,m+1,m+2)

        ###############################################
        ##               Première ligne              ##
        ###############################################

        for i in range(m*int(L/K),(m+1)*int(L/K)):
            line_0[i-m*int(L/K)]=positions_corrected[i]
            Is_0[i-m*int(L/K)]=Is_corrected[i]
            Ic_0[i-m*int(L/K)]=Ic_corrected[i]

        if line_0[0]==min(line_0) or line_0[-1]==max(line_0):
            line_0=line_0
            Is_0=Is_0
            Ic_0=Ic_0
        else:
            line_0=np.flipud(line_0)
            Is_0=np.flipud(Is_0)
            Ic_0=np.flipud(Ic_0)

        ###############################################
        ##              Deuxième ligne               ##
        ###############################################

        for i in range((m+1)*int(L/K),(m+2)*int(L/K)):
            line_1[i-(m+1)*int(L/K)]=positions_corrected[i]
            Is_1[i-(m+1)*int(L/K)]=Is_corrected[i]
            Ic_1[i-(m+1)*int(L/K)]=Ic_corrected[i]

        if line_1[0]==min(line_1) or line_1[-1]==max(line_1):
            line_1=line_1
            Is_1=Is_1
            Ic_1=Ic_1
        else:
            line_1=np.flipud(line_1)
            Is_1=np.flipud(Is_1)
            Ic_1=np.flipud(Ic_1)

        weight=1

#########################################################################
```

```
##    Fonction simulant le shift en X                                    ##
##                                                                       ##
##    Deux sigmoïdes en bord de scan, un polynôme d'ordre 3 centré       ##
##                                                                       ##
##    Interpole la sortie:                                               ##
##                                                                       ##
##        Is + np.gradient(Is) + np.gradient(np.gradient(Is))            ##
##                                                                       ##
###########################################################################


###########################################################################
##     Fonction d'interpolation entre le shift et les données fittées    ##
###########################################################################
        def func_1(x,a,b,c,shift,ad,xd,sigmad,ag,xg,sigmag):
         return (
             np.interp(x+ad*expit((x-xd)/(sigmad))+ag*(1-expit((x-
xg)/sigmag))+c*(x-x_center)**3+b*(x-x_center)**2+a*(x-
x_center)+shift,line_1,Is_1)
                 +np.interp(x+ad*expit((x-xd)/(sigmad))+ag*(1-expit((x-
xg)/sigmag))+c*(x-x_center)**3+b*(x-x_center)**2+a*(x-
x_center),line_1,weight**0.5*np.gradient(Is_1))
                  +np.interp(x+ad*expit((x-xd)/(sigmad))+ag*(1-expit((x-
xg)/sigmag))+c*(x-x_center)**3+b*(x-x_center)**2+a*(x-
x_center),line_1,weight**0.5*np.gradient(np.gradient(Is_1)))
                  )

        xdata=line_0


###########################################################################
##        Ici, ce sont les data à minimiser --> Jusqu'au Laplacien       ##
###########################################################################

ydata=Is_0+weight**0.5*np.gradient(Is_0)+weight**0.5*np.gradient(np.gradien
t(Is_0))

        shift=0

        if (k%2)==0:
            x0_guess=np.arange(-2,2,0.10)
            a0=-10**-3
            b0=10**-6
            c0=10**-8
            xd0=max(xdata)-2
            xg0=min(xdata)+2
            ad0=-2.5
            ag0=2.5
            sigmad0=0.0005
            sigmag0=1

        else:
            x0_guess=np.arange(-2,2,0.10)
            a0=-10**-3
            b0=10**-6
            c0=10**-8
            xd0=max(xdata)-2
            xg0=min(xdata)+2
            ad0=-2.5
```

```
            ag0=2.5
            sigmad0=0.0005
            sigmag0=1

        l_guess=len(x0_guess)
        error_tmp=np.zeros(l_guess)
        shift_opt_tmp=np.zeros(l_guess)
        a_opt_tmp=a0*np.ones(l_guess)
        b_opt_tmp=b0*np.ones(l_guess)
        c_opt_tmp=c0*np.ones(l_guess)

        ad_opt_tmp=ad0*np.ones(l_guess)
        ag_opt_tmp=ag0*np.ones(l_guess)

        xd_opt_tmp=xd0*np.ones(l_guess)
        xg_opt_tmp=xg0*np.ones(l_guess)

        sigmad_opt_tmp=sigmad0*np.ones(l_guess)
        sigmag_opt_tmp=sigmag0*np.ones(l_guess)

        sigma_1=0.005*np.ones(len(ydata))


#############################################################################
##             Définitions des bornes de variation des paramètres          ##
#############################################################################
        # Lower bonds
        # Polynome
        lb_a0=-np.inf
        lb_b0=-np.inf
        lb_c0=-np.inf
        lb_x0=-15           # Shift moyen
        # Sigmoide droite
        lb_ad0=-5
        lb_xd0=xd0-20
        lb_sigmad0=0.000000001
        # Sigmoide gauche
        lb_ag0=0
        lb_xg0=xg0-10
        lb_sigmag0=0.01
        # Upper bonds
        # Polynome
        ub_a0=np.inf
        ub_b0=np.inf
        ub_c0=np.inf
        ub_x0=15           # Shift moyen
        # Sigmoide droite
        ub_ad0=20
        ub_xd0=xd0+20
        ub_sigmad0=5
        # Sigmoide gauche
        ub_ag0=20
        ub_xg0=xg0+10
        ub_sigmag0=5


#############################################################################
##        Fin des définitions des bornes de variation des paramètres       ##
#############################################################################
```

```
        for u in range(0,l_guess):
            x0=x0_guess[u]


bounds_parameters=[(lb_a0,lb_b0,lb_c0,lb_x0,lb_ad0,lb_xd0,lb_sigmad0,lb_ag0
,lb_xg0,lb_sigmag0),(ub_a0,ub_b0,ub_c0,ub_x0,ub_ad0,ub_xd0,ub_sigmad0,ub_ag
0,ub_xg0,ub_sigmag0)]

            xFit_1,cov = optimization.curve_fit(func_1, xdata, ydata,
(a0,b0,c0,x0,ad0,xd0,sigmad0,ag0,xg0,sigmag0),
sigma_1,maxfev=500000,bounds=bounds_parameters)
            error_1=(
                np.sum(

((func_1(xdata,xFit_1[0],xFit_1[1],xFit_1[2],xFit_1[3],xFit_1[4],xFit_1[5],
xFit_1[6],xFit_1[7],xFit_1[8],xFit_1[9])-ydata)**2

+0*(np.gradient((func_1(xdata,xFit_1[0],xFit_1[1],xFit_1[2],xFit_1[3],xFit_
1[4],xFit_1[5],xFit_1[6],xFit_1[7],xFit_1[8],xFit_1[9]))))-
np.gradient((ydata)))**2

+0*(np.gradient(np.gradient((func_1(xdata,xFit_1[0],xFit_1[1],xFit_1[2],xFi
t_1[3],xFit_1[4],xFit_1[5],xFit_1[6],xFit_1[7],xFit_1[8],xFit_1[9])))))-
np.gradient((np.gradient((ydata)))))**2)**1
                    )
                )

            error_tmp[u]=error_1

            a_opt_tmp[u]=xFit_1[0]
            b_opt_tmp[u]=xFit_1[1]
            c_opt_tmp[u]=xFit_1[2]
            shift_opt_tmp[u]=xFit_1[3]

            ad_opt_tmp[u]=xFit_1[4]
            xd_opt_tmp[u]=xFit_1[5]
            sigmad_opt_tmp[u]=xFit_1[6]

            ag_opt_tmp[u]=xFit_1[7]
            xg_opt_tmp[u]=xFit_1[8]
            sigmag_opt_tmp[u]=xFit_1[9]

        fig = plt.figure()
        ax = fig.subplots()
        ax.plot(shift_opt_tmp,error_tmp,'o')

        shift_opt_1=shift_opt_tmp[error_tmp==min(error_tmp)][0]
        a_opt_1=a_opt_tmp[error_tmp==min(error_tmp)][0]
        b_opt_1=b_opt_tmp[error_tmp==min(error_tmp)][0]
        c_opt_1=c_opt_tmp[error_tmp==min(error_tmp)][0]

        ad_opt_1=ad_opt_tmp[error_tmp==min(error_tmp)][0]
        xd_opt_1=xd_opt_tmp[error_tmp==min(error_tmp)][0]
        sigmad_opt_1=sigmad_opt_tmp[error_tmp==min(error_tmp)][0]

        ag_opt_1=ag_opt_tmp[error_tmp==min(error_tmp)][0]
        xg_opt_1=xg_opt_tmp[error_tmp==min(error_tmp)][0]
        sigmag_opt_1=sigmag_opt_tmp[error_tmp==min(error_tmp)][0]
```

```
        # Mise à jour des conditions initiales
        a0=a_opt_1
        b0=b_opt_1
        c0=c_opt_1

        # Deux gaussiennes
        ad0=ad_opt_1
        xd0=xd_opt_1
        sigmad0=sigmad_opt_1

        ag0=ag_opt_1
        xg0=xg_opt_1
        sigmag0=sigmag_opt_1
        shift_f_de_x=c_opt_1*(xdata-x_center)**3+b_opt_1*(xdata-
x_center)**2+a_opt_1*(xdata-x_center)+shift_opt_1+ad_opt_1*expit((xdata-
xd_opt_1)/(sigmad_opt_1))+ag_opt_1*(1-expit((xdata-
xg_opt_1)/(sigmag_opt_1)))

        fig = plt.figure()
        plt.plot(xdata,shift_f_de_x)

        line_1_shifted=line_1-shift_f_de_x
        line_0_shifted=line_0

        ##############################################
        ##              Troisième ligne             ##
        ##############################################

        for i in range((m+2)*int(L/K),(m+3)*int(L/K)):
            line_2[i-(m+2)*int(L/K)]=positions_corrected[i]
            Is_2[i-(m+2)*int(L/K)]=Is_corrected[i]
            Ic_2[i-(m+2)*int(L/K)]=Ic_corrected[i]

        if line_2[0]==min(line_2) or line_2[-1]==max(line_2):
            line_2=line_2
            Is_2=Is_2
            Ic_2=Ic_2
        else:
            line_2=np.flipud(line_2)
            Is_2=np.flipud(Is_2)
            Ic_2=np.flipud(Ic_2)

        shift_opt_2=0
        line_2_shifted=line_2-shift_opt_2
        fig = plt.figure()
        ax = fig.subplots()
        ax.plot(line_0,Is_0,'k')
        ax.plot(line_1_shifted,Is_1,'r')
        ax.plot(line_2,Is_2,'b')
        plt.title('After correction red curve')


#########################################################################
##           Coordonnées en X triées par ordre croissant               ##
#########################################################################

        for i in range((m+2)*int(L/K),(m+3)*int(L/K)):
            positions_corrected[i]= line_2_shifted[i-(m+3)*int(L/K)]
```

```
            Is_corrected[i]=Is_2[i-(m+3)*int(L/K)]
            Ic_corrected[i]=Ic_2[i-(m+3)*int(L/K)]

        for i in range(m*int(L/K),(m+1)*int(L/K)):
            positions_corrected[i]= line_0_shifted[i-(m+1)*int(L/K)]
            Is_corrected[i]=Is_0[i-(m+1)*int(L/K)]
            Ic_corrected[i]=Ic_0[i-(m+1)*int(L/K)]

        for i in range((m+1)*int(L/K),(m+2)*int(L/K)):
            positions_corrected[i]= line_1_shifted[i-(m+2)*int(L/K)]
            Is_corrected[i]=Is_1[i-(m+2)*int(L/K)]
            Ic_corrected[i]=Ic_1[i-(m+2)*int(L/K)]

        shift_trace[k]=shift_opt_1
        a_trace[k]=a_opt_1
        b_trace[k]=b_opt_1
        c_trace[k]=c_opt_1

        ad_trace[k]=ad_opt_1
        xd_trace[k]=xd_opt_1
        sigmad_trace[k]=sigmad_opt_1

        ag_trace[k]=ag_opt_1
        xg_trace[k]=xg_opt_1
        sigmag_trace[k]=sigmag_opt_1

        error_trace[k]=error_tmp[error_tmp==min(error_tmp)][0]

    ######################################################################
    ##      Optimisation autour des points optimaux précédents          ##
    ######################################################################


    for k in range(k+1,n_tot):

        m=int(2*k)
        if m<0:m=0

        print(' lines without=', m,m+1,m+2)

        ###############################################
        ##               Première ligne              ##
        ###############################################
        for i in range(m*int(L/K),(m+1)*int(L/K)):
            line_0[i-m*int(L/K)]=positions_corrected[i]
            Is_0[i-m*int(L/K)]=Is_corrected[i]
            Ic_0[i-m*int(L/K)]=Ic_corrected[i]

        if line_0[0]==min(line_0) or line_0[-1]==max(line_0):
            line_0=line_0
            Is_0=Is_0
            Ic_0=Ic_0
        else:
            line_0=np.flipud(line_0)
            Is_0=np.flipud(Is_0)
            Ic_0=np.flipud(Ic_0)
```

```
        ##########################################
        ##             Deuxième ligne           ##
        ##########################################
        for i in range((m+1)*int(L/K),(m+2)*int(L/K)):
            line_1[i-(m+1)*int(L/K)]=positions_corrected[i]
            Is_1[i-(m+1)*int(L/K)]=Is_corrected[i]
            Ic_1[i-(m+1)*int(L/K)]=Ic_corrected[i]

        if line_1[0]==min(line_1) or line_1[-1]==max(line_1):
            line_1=line_1
            Is_1=Is_1
            Ic_1=Ic_1
        else:
            line_1=np.flipud(line_1)
            Is_1=np.flipud(Is_1)
            Ic_1=np.flipud(Ic_1)

        weight=1


#########################################################################
##    Fonction simulant le shift en X                                  ##
##                                                                     ##
##    Deux sigmoides en bord de scan, un polynome d'ordre 3 centré     ##
##                                                                     ##
##    Interpole la sortie:                                             ##
##                                                                     ##
##        Is + np.gradient(Is) + np.gradient(np.gradient(Is))          ##
##                                                                     ##
#########################################################################




#########################################################################
##   Fonction d'interpolation entre le shift et les données fittées    ##
#########################################################################

        def func_1(x,a,b,c,shift,ad,xd,sigmad,ag,xg,sigmag):
         return (
             np.interp(x+ad*expit((x-xd)/(sigmad))+ag*(1-expit((x-
xg)/sigmag))+c*(x-x_center)**3+b*(x-x_center)**2+a*(x-
x_center)+shift,line_1,Is_1)
                 +np.interp(x+ad*expit((x-xd)/(sigmad))+ag*(1-expit((x-
xg)/sigmag))+c*(x-x_center)**3+b*(x-x_center)**2+a*(x-
x_center),line_1,weight**0.5*np.gradient(Is_1))
                  +np.interp(x+ad*expit((x-xd)/(sigmad))+ag*(1-expit((x-
xg)/sigmag))+c*(x-x_center)**3+b*(x-x_center)**2+a*(x-
x_center),line_1,weight**0.5*np.gradient(np.gradient(Is_1)))
                  )

        xdata=line_0


#########################################################################
##        Ici, ce sont les data à minimiser --> Jusqu'au Laplacien     ##
#########################################################################
```

```
ydata=Is_0+weight**0.5*np.gradient(Is_0)+weight**0.5*np.gradient(np.gradien
t(Is_0))
        shift=0

        if (k%2)==0:
            x0=np.mean(shift_trace[0:2])
            a0=np.mean(a_trace[0:2])
            b0=np.mean(b_trace[0:2])
            c0=np.mean(c_trace[0:2])
            xd0=np.mean(xd_trace[0:2])
            xg0=np.mean(xg_trace[0:2])
            ad0=np.mean(ad_trace[0:2])
            ag0=np.mean(ag_trace[0:2])
            sigmad0=np.mean(sigmad_trace[0:2])
            sigmag0=np.mean(sigmag_trace[0:2])

        else:
            x0=np.mean(shift_trace[0:2])
            a0=np.mean(a_trace[0:2])
            b0=np.mean(b_trace[0:2])
            c0=np.mean(c_trace[0:2])
            xd0=np.mean(xd_trace[0:2])
            xg0=np.mean(xg_trace[0:2])
            ad0=np.mean(ad_trace[0:2])
            ag0=np.mean(ag_trace[0:2])
            sigmad0=np.mean(sigmad_trace[0:2])
            sigmag0=np.mean(sigmag_trace[0:2])

        l_guess=1
        error_tmp=np.zeros(l_guess)
        shift_opt_tmp=np.zeros(l_guess)
        a_opt_tmp=a0*np.ones(l_guess)
        b_opt_tmp=b0*np.ones(l_guess)
        c_opt_tmp=c0*np.ones(l_guess)

        ad_opt_tmp=ad0*np.ones(l_guess)
        ag_opt_tmp=ag0*np.ones(l_guess)

        xd_opt_tmp=xd0*np.ones(l_guess)
        xg_opt_tmp=xg0*np.ones(l_guess)

        sigmad_opt_tmp=sigmad0*np.ones(l_guess)
        sigmag_opt_tmp=sigmag0*np.ones(l_guess)

        sigma_1=0.005*np.ones(len(ydata))

        for u in range(0,l_guess):
            xFit_1,cov = optimization.curve_fit(func_1, xdata, ydata,
(a0,b0,c0,x0,ad0,xd0,sigmad0,ag0,xg0,sigmag0),
sigma_1,maxfev=500000,bounds=bounds_parameters)
            error_1=(
                np.sum(

((func_1(xdata,xFit_1[0],xFit_1[1],xFit_1[2],xFit_1[3],xFit_1[4],xFit_1[5],
xFit_1[6],xFit_1[7],xFit_1[8],xFit_1[9])-ydata)**2

+0*(np.gradient((func_1(xdata,xFit_1[0],xFit_1[1],xFit_1[2],xFit_1[3],xFit_
```

```
1[4],xFit_1[5],xFit_1[6],xFit_1[7],xFit_1[8],xFit_1[9])))-
np.gradient((ydata)))**2

+0*(np.gradient(np.gradient((func_1(xdata,xFit_1[0],xFit_1[1],xFit_1[2],xFi
t_1[3],xFit_1[4],xFit_1[5],xFit_1[6],xFit_1[7],xFit_1[8],xFit_1[9])))))-
np.gradient((np.gradient((ydata)))))**2)**1
                    )
                )

            error_tmp=error_1

            a_opt_tmp=xFit_1[0]
            b_opt_tmp=xFit_1[1]
            c_opt_tmp=xFit_1[2]
            shift_opt_tmp=xFit_1[3]

            ad_opt_tmp=xFit_1[4]
            xd_opt_tmp=xFit_1[5]
            sigmad_opt_tmp=xFit_1[6]

            ag_opt_tmp=xFit_1[7]
            xg_opt_tmp=xFit_1[8]
            sigmag_opt_tmp=xFit_1[9]

        fig = plt.figure()
        ax = fig.subplots()
        ax.plot(shift_opt_tmp,error_tmp,'o')

        shift_opt_1=shift_opt_tmp
        a_opt_1=a_opt_tmp
        b_opt_1=b_opt_tmp
        c_opt_1=c_opt_tmp

        ad_opt_1=ad_opt_tmp
        xd_opt_1=xd_opt_tmp
        sigmad_opt_1=sigmad_opt_tmp

        ag_opt_1=ag_opt_tmp
        xg_opt_1=xg_opt_tmp
        sigmag_opt_1=sigmag_opt_tmp

        # Mise à jour des conditions initiales
        a0=a_opt_1
        b0=b_opt_1
        c0=c_opt_1

        #  Deux gaussiennes
        ad0=ad_opt_1
        xd0=xd_opt_1
        sigmad0=sigmad_opt_1

        ag0=ag_opt_1
        xg0=xg_opt_1
        sigmag0=sigmag_opt_1

        shift_f_de_x=c_opt_1*(xdata-x_center)**3+b_opt_1*(xdata-
x_center)**2+a_opt_1*(xdata-x_center)+shift_opt_1+ad_opt_1*expit((xdata-
xd_opt_1)/(sigmad_opt_1))+ag_opt_1*(1-expit((xdata-
xg_opt_1)/(sigmag_opt_1)))
```

```
        fig = plt.figure()
        plt.plot(xdata,shift_f_de_x)
        line_1_shifted=line_1-shift_f_de_x
        line_0_shifted=line_0

        ####################################################
        ##            Troisième ligne (paire)             ##
        ####################################################

        for i in range((m+2)*int(L/K),(m+3)*int(L/K)):
            line_2[i-(m+2)*int(L/K)]=positions_corrected[i]
            Is_2[i-(m+2)*int(L/K)]=Is_corrected[i]
            Ic_2[i-(m+2)*int(L/K)]=Ic_corrected[i]


        if line_2[0]==min(line_2) or line_2[-1]==max(line_2):
            line_2=line_2
            Is_2=Is_2
            Ic_2=Ic_2
        else:
            line_2=np.flipud(line_2)
            Is_2=np.flipud(Is_2)
            Ic_2=np.flipud(Ic_2)

        shift_opt_2=0
        line_2_shifted=line_2-shift_opt_2
        fig = plt.figure()
        ax = fig.subplots()
        ax.plot(line_0,Is_0,'k')
        ax.plot(line_1_shifted,Is_1,'r')
        ax.plot(line_2,Is_2,'b')
        plt.title('After correction red curve')


#########################################################################
##           Coordonnées en X triées par ordre croissant               ##
#########################################################################
        for i in range((m+2)*int(L/K),(m+3)*int(L/K)):
            positions_corrected[i]= line_2_shifted[i-(m+3)*int(L/K)]
            Is_corrected[i]=Is_2[i-(m+3)*int(L/K)]
            Ic_corrected[i]=Ic_2[i-(m+3)*int(L/K)]

        for i in range(m*int(L/K),(m+1)*int(L/K)):
            positions_corrected[i]= line_0_shifted[i-(m+1)*int(L/K)]
            Is_corrected[i]=Is_0[i-(m+1)*int(L/K)]
            Ic_corrected[i]=Ic_0[i-(m+1)*int(L/K)]

        for i in range((m+1)*int(L/K),(m+2)*int(L/K)):
            positions_corrected[i]= line_1_shifted[i-(m+2)*int(L/K)]
            Is_corrected[i]=Is_1[i-(m+2)*int(L/K)]
            Ic_corrected[i]=Ic_1[i-(m+2)*int(L/K)]

        shift_trace[k]=shift_opt_1
        a_trace[k]=a_opt_1
        b_trace[k]=b_opt_1
        c_trace[k]=c_opt_1
```

```
        ad_trace[k]=ad_opt_1
        xd_trace[k]=xd_opt_1
        sigmad_trace[k]=sigmad_opt_1

        ag_trace[k]=ag_opt_1
        xg_trace[k]=xg_opt_1
        sigmag_trace[k]=sigmag_opt_1

        error_trace[k]=error_1

#####################################################
##           Représentation Graphique              ##
#####################################################

positions_init = pd.read_csv(filename, sep='\t', header = None, skiprows =
1,usecols=[2,3], names=['X','Y'])
x_list_init = positions_init['X'][:]
y_list_init = positions_init['Y'][:]
x_list_init=x_list_init[0:int((m+2)*L/K)]
y_list_init =y_list_init[0:int((m+2)*L/K)]

step_X=np.gradient(x_list_init)

IsIc = pd.read_csv(filename, sep='\t',header = None, skiprows =
1,usecols=[4,5], names=['Is','Ic'])
IsIc =IsIc [0:int((m+2)*L/K)]

S0 = pd.read_csv(filename, sep='\t',header = None, skiprows =
1,usecols=[6], names=['S0'])
S0 = S0 [0:int((m+2)*L/K)]

positions_init =positions_init[0:int((m+2)*L/K)]

#########################################
##           Sans correction           ##
#########################################

fig, (axs1,axs2) =
plt.subplots(2,1,figsize=(12,4),gridspec_kw={"width_ratios":[1]})

minX = min(x_list_init)
maxX = max(x_list_init)
Champ=maxX_init-minX_init
minY = min(y_list_init)
maxY = max(y_list_init)

rez=3
tix = np.linspace(minX,maxX,int(len(x_list_init)**0.5)*rez)
tiy = np.linspace(minY,maxY,int(len(x_list_init)**0.5)*rez)
XI, YI = np.meshgrid(tix, tiy)
XII, YII = np.meshgrid(tix, tiy)
Z0 = IsIc['Is']
Z1 = IsIc['Ic']
Z2=S0['S0']
minZ = np.min([Z0,Z1])
maxZ = np.max([Z0,Z1])
minS0=np.min(S0)
maxS0=np.max(S0)
```

```
ZI = interpolate.griddata(positions_init,Z0,(XI,YI),method='cubic')
ZII = interpolate.griddata(positions_init,Z1,(XI,YI),method='cubic')
ZS0 = interpolate.griddata(positions_init,Z2,(XI,YI),method='cubic')

axs1.cla()
axs2.cla()

im  =
axs1.pcolormesh(XI,YI,ZI,vmin=min(Is_corrected[0:int((m+2)*L/K)]),vmax=max(
Is_corrected[0:int((m+2)*L/K)]),cmap=CM.jet)
im2 =
axs2.pcolormesh(XI,YI,ZII,vmin=min(Ic_corrected[0:int((m+2)*L/K)]),vmax=max
(Ic_corrected[0:int((m+2)*L/K)]),cmap=CM.jet)

axs1.set_title('Is')
axs2.set_title('Ic')
fig.suptitle('Cartographie ellipsométrique de tous les champs sans
correction')
plt.show()

x_list = positions_corrected[:]
x_list_end=x_list

for p in range(0,int(K)):
    if p % 2 == 0:
        x_list_end[p*int(L/K):(p+1)*int(L/K)]=
x_list[p*int(L/K):(p+1)*int(L/K)]
    else:
        x_list_end[p*int(L/K):(p+1)*int(L/K)]=
np.flipud(x_list[p*int(L/K):(p+1)*int(L/K)])

x_list_end=x_list_end[0:int((m+2)*L/K)]
x_list=x_list_end
y_list = positions['Y'][0:int((m+2)*L/K)]

step_X=np.gradient(x_list)

########################################
##          Avec corrections          ##
########################################

fig, (axs1,axs2) =
plt.subplots(2,1,figsize=(12,4),gridspec_kw={"width_ratios":[1]})
minX = min(x_list)
maxX = max(x_list)
Champ=maxX-minX
minY = min(y_list)
maxY = max(y_list)
rez=3

tix = np.linspace(minX,maxX,int(len(x_list_init)**0.5)*rez)
tiy = np.linspace(minY,maxY,int(len(x_list_init)**0.5)*rez)

XI, YI = np.meshgrid(tix, tiy)
XII, YII = np.meshgrid(tix, tiy)
Z0 = IsIc['Is']
Z1 = IsIc['Ic']
Z2=S0['S0']
```

```
ZI =
interpolate.griddata(positions_bis[0:int((m+2)*L/K)],Z0,(XI,YI),method='cub
ic')
ZII =
interpolate.griddata(positions[0:int((m+2)*L/K)],Z1,(XI,YI),method='cubic')
axs1.cla()
axs2.cla()
im  = axs1.pcolormesh(XI,YI,ZI,vmin=min(Z0),vmax=max(Z0),cmap=CM.jet)
im2 = axs2.pcolormesh(XI,YI,ZII,vmin=min(Z1),vmax=max(Z1),cmap=CM.jet)
axs1.set_title('Is')
axs2.set_title('Ic')
fig.suptitle('Cartographie ellipsométrique de tous les champs avec
correction')
plt.show()
```