5 Recherche et développement

Dans cette section, les différentes solutions proposées ainsi que les améliorations apportées au système initial sont exposées. Commençant par la segmentation, qui constitue la base de l'extraction des attributs nécessaires. Elle se poursuivra avec l'extraction des caractéristiques de couleur, qui permettent de distinguer des objets similaires présentant de variations de couleurs. Une autre partie sera consacrée aux attributs ajoutés, ceux obtenus à partir de la courbure et de la profondeur. Avant d'aborder la dernière partie, le classifieur employé et les techniques déployées pour renforcer ses performances seront abordés. Pour conclure, un système de détection de nouveauté en phase de prédiction sera présenté.

Dans le cadre de cette étude, le schéma de flux ci-dessous illustre les différentes étapes du processus. Ce schéma permet de visualiser la séquence des opérations réalisées, ainsi que la hiérarchie entre les différentes étapes.

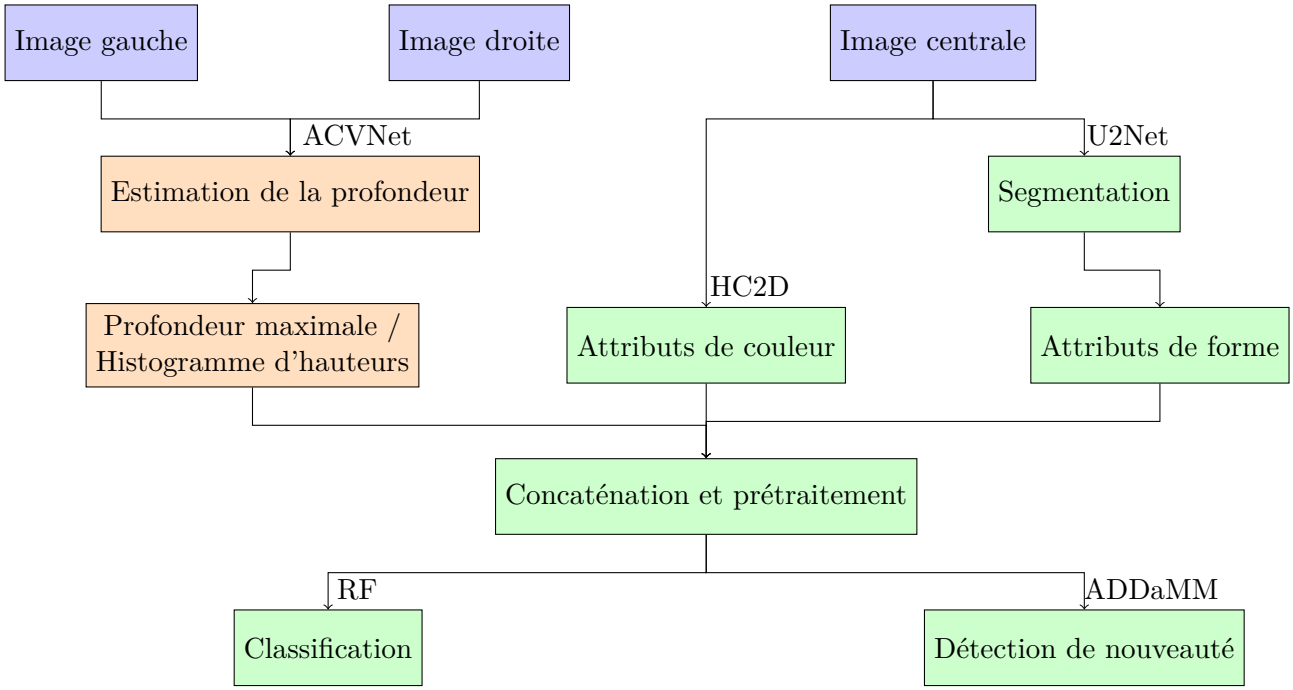


FIGURE 5.1 – Diagramme de flux du processus

On note que les cases de l'estimation de la profondeur sont en orange, car elles ne sont pas encore déployées en production. Plus de détails seront fournis dans les sections suivantes.

5.1 Amélioration de la segmentation avec U2-Net

5.1.1 Description de l'Architecture U2-Net

L'U2-Net [3] est une architecture de réseau profond utilisée dans la segmentation d'image. Son nom, "U2-Net", fait référence à sa structure en "U", elle comprend plusieurs de ces structures "U" imbriquées les unes dans les autres à différentes échelles, formant une sorte de "U" à plusieurs niveaux (Fig. 5.2). La spécificité de l'U2-Net réside dans l'utilisation de ces mul-

tiples niveaux pour capturer différentes caractéristiques à différentes échelles, ce qui améliore la précision de la segmentation.

Cette architecture se caractérise par son efficacité à capturer à la fois des informations de haut niveau (grandes structures) et de bas niveau (détails fins), grâce à l'usage de différentes tailles de champs récepteurs. De plus, l'U2-Net utilise une technique appelée "attention résiduelle" qui lui permet d'apprendre à se concentrer sur les régions d'intérêts de l'image lors de la réalisation de la segmentation.

Ainsi, l'U2-Net se distingue par sa capacité à produire des segmentations précises tout en étant relativement léger en termes de besoins en ressources de calcul, ce qui en fait un choix privilégié pour notre système.

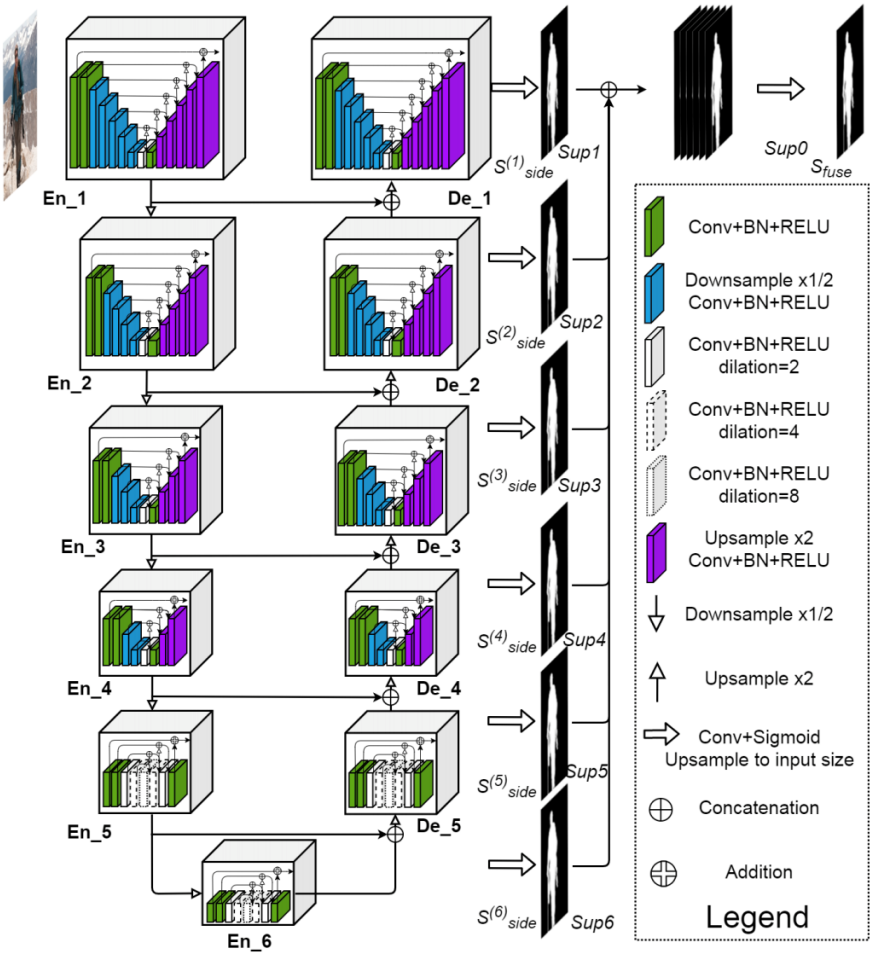


FIGURE 5.2 – Architecture de U2-Net [3]

5.1.2 Entraînement de U2-Net

L'entraînement d'un réseau de neurones est un processus itératif qui vise à optimiser les poids du réseau afin de minimiser une fonction de coût. Cette dernière représente la mesure de la différence entre les prédictions du réseau et les valeurs réelles.

Dans cette section, l'accent sera mis sur les éléments et les paramètres intervenant dans ce processus, en explorant leur impact sur la qualité de l'entraînement.

Données d'entraînement

Pour entraîner le réseau U2-Net, un ensemble d'images labellisées a été utilisé. Chaque image était associée à une segmentation d'image attendue. Étant donné l'indisponibilité de

données correspondant aux mêmes propriétés que nos images (instruments chirurgicaux), les annotations des images ont été réalisées manuellement grâce à l'outil en ligne gratuit MakeSense [4]. Cet outil permet l'étiquetage des objets, et les étiquettes préparées peuvent être téléchargées dans de nombreux formats, notamment le format COCO [5] (Common Objects in Context), format retenu pour ce travail.

Le processus d'annotation est long et exigeant, nécessitant concentration et minutie pour éviter les erreurs dans les étiquettes afin d'obtenir les meilleures performances après l'entraînement. Pour alléger et optimiser cette tâche, l'architecture existante a été mise à profit pour produire des annotations pour les objets imparfaitement segmentés (Fig. 4.1). Ces annotations ont ensuite été rectifiées avec l'outil MakeSense, conduisant à de nouvelles annotations corrigées. Ces dernières englobent trois catégories : objets, trous et mains, comme illustré dans la figure 5.3.

Une fois les annotations obtenues, les données au format COCO ont été converties en masques binaires, afin de construire le jeu de données composé des images provenant des caméras et de leurs masques binaires. Initialement, le modèle a été entraîné avec 600 images. Après corrections des images aux étiquettes inexactes, 360 images supplémentaires, illustrant des situations problématiques (objets réfléchissants, objets colorés, ou présentant de petits trous) ont été intégrées. Ce qui a conduit à un ensemble final de 960 images avec leurs masques binaires pour l'entraînement.

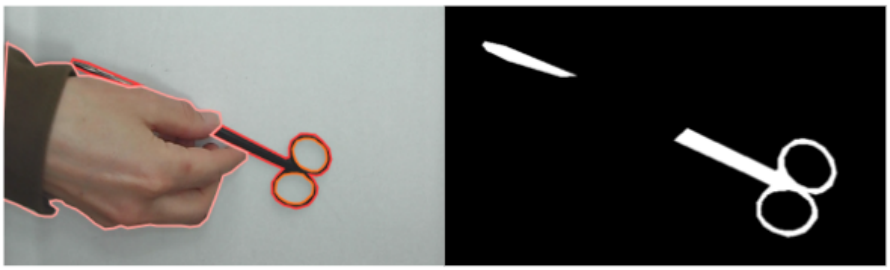


FIGURE 5.3 – Objet annoter sur MakeSense

Augmentation de données

Afin de diversifier l'ensemble de données, des transformations aléatoires ont été appliquées aux images d'entraînement, afin de créer de nouvelles variations et enrichir l'ensemble de données pour améliorer la robustesse du modèle.

Les techniques d'augmentation de données comprennent :

— **Flip horizontal** : Cette transformation permet de retourner l'image horizontalement, introduisant ainsi des variations dans l'orientation des objets.
— **Contraste aléatoire** : Cette transformation ajuste le contraste de l'image en appliquant une valeur de contraste aléatoire, favorisant ainsi des variations dans la clarté des instruments.
— **Gamma aléatoire** : Cette transformation ajuste le gamma de l'image en appliquant une valeur de gamma aléatoire, modifiant la luminosité globale de l'image.
— **Luminosité aléatoire** : Cette transformation ajuste la luminosité de l'image en appliquant une valeur de luminosité aléatoire, introduisant des variations dans l'éclairage des instruments chirurgicaux.
— **Variation de teinte, saturation et valeur** : Cette transformation modifie la teinte, la saturation et la valeur des pixels de l'image en appliquant des décalages aléatoires, créant ainsi des variations dans les couleurs des instruments.
— **Décalage, échelle et rotation aléatoires** : Cette transformation applique des décalages, des échelles et des rotations aléatoires à l'image, permettant de simuler des variations de

perspective et d'orientation des objets.
— **Mélange de canaux** : Cette transformation permute aléatoirement les canaux de l'image, introduisant des variations dans les relations de couleur des instruments.
— **Distorsion de grille** : Cette transformation applique une distorsion de grille à l'image, introduisant des variations locales dans les pixels.
— **Transformation élastique** : Cette transformation applique une transformation élastique à l'image, introduisant des déformations locales.
— **Distorsion optique** : Cette transformation applique une distorsion optique à l'image, simulant une déformation de lentille.
— **Mélange aléatoire de grille** : Cette transformation effectue un mélange aléatoire de la grille de pixels de l'image.
— **Ombre aléatoire** : Cette transformation ajoute une ombre aléatoire à l'image, simulant des variations d'éclairage.
— **Jitter de couleur** : Cette transformation applique un jitter de couleur aléatoire aux pixels de l'image, introduisant des variations de couleur.

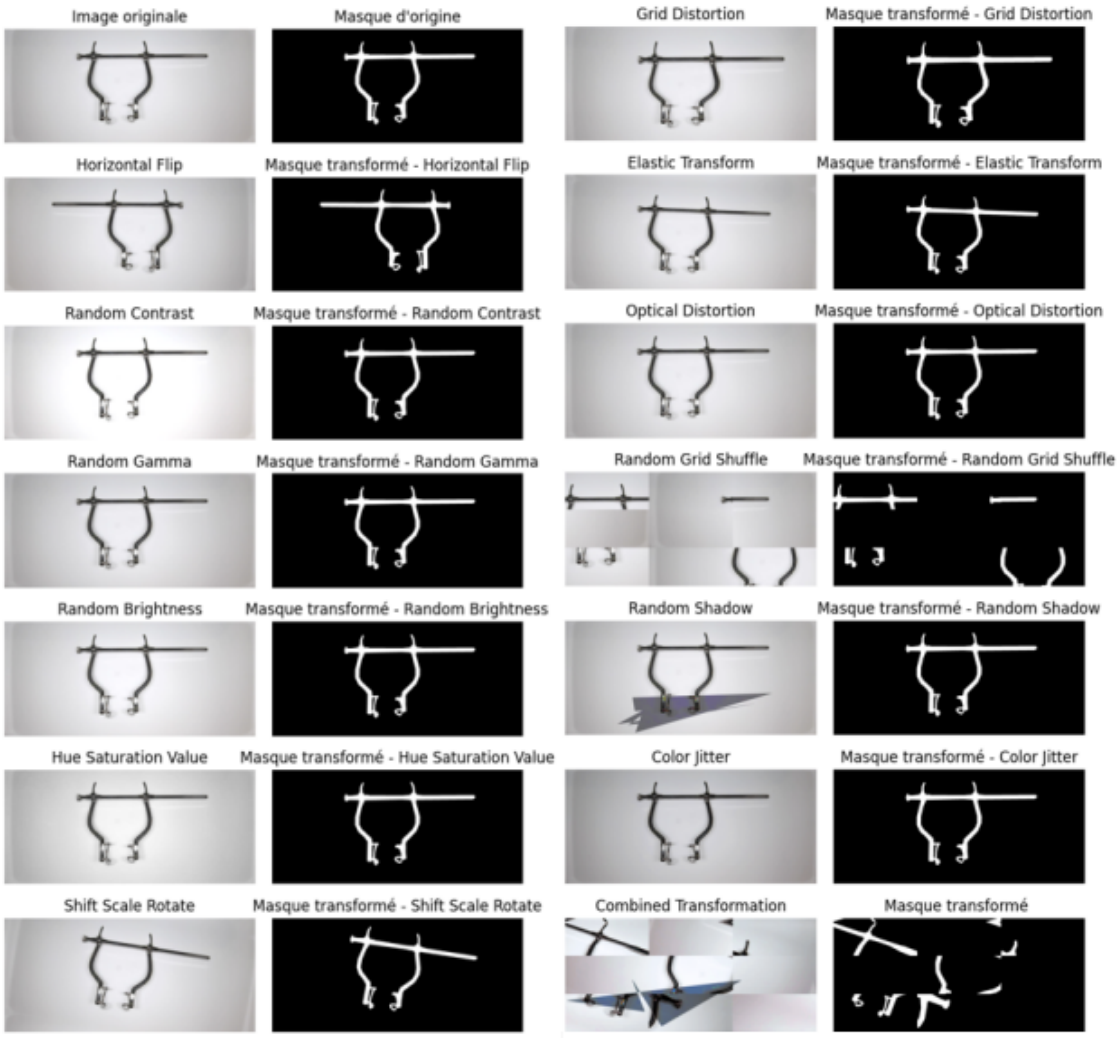


FIGURE 5.4 – Augmentations appliquées sur les données d'entraînement

Les transformations illustrées dans la figure 5.1.2 sont appliquées aux images d'entraînement ainsi qu'à celles de la validation à l'aide de la bibliothèque Albumentations [6]. Une autre transformation appliquée sur les données est le recadrage aléatoire centré sur le centre de masse de l'objet, qui permet d'entraîner le réseau sur des résolutions différentes de l'image.

Ajustement du modèle U2-Net

Le modèle est construit en utilisant l'API Keras de TensorFlow. Il prend en entrée des images de taille (320, 320, 3) et produit en sortie une carte de segmentation binaire. Le modèle est conçu avec des couches de convolution et de déconvolution, ainsi que des opérations de mise en commun (pooling) et de remise en échelle (upsampling) pour capturer les informations spatiales à différentes échelles (Fig. 5.2).

L'architecture du modèle U2-Net est personnalisée avec les paramètres suivants :

Paramètres	**Description**	**Valeur dans le modèle initial**	**Valeur dans le modèle amélioré**
filter_num_down	Nombre et taille de filtres dans les couches de convolution descendantes	[64, 64, 64, 64]	[32, 64, 128, 256]
filter_num_up	Nombre et taille de filtres dans les couches de déconvolution ascendantes	[64, 64, 64, 64]	[32, 64, 128, 256]
filter_mid_num_down	Nombre et taille de filtres dans les couches de milieu de réseau de convolution descendantes	[16,16,16,16]	[16,32,64,64]
filter_mid_num_up	Nombre et taille de filtres dans les couches de milieu de réseau de déconvolution ascendantes	[16,16,16,16]	[16,32,64,64]
filter_4f_num	Nombre et taille de filtres dans les couches de convolution avec un facteur de 4	[64,64]	[64,64]
filter_4f_mid_num	Nombre et taille de filtres dans les couches de milieu de réseau de convolution avec un facteur de 4	[32,32]	[32,32]
activation	Fonction d'activation utilisée dans le modèle	ReLU	ReLU
output_activation	Fonction d'activation de la couche de sortie du modèle	sigmoïde	sigmoïde
batch_norm	Indique si la normalisation par lots est utilisée	True	True
pool	Type de mise en commun utilisé	max pooling	max pooling
unpool	Type de remise en échelle utilisé	bilinéaire	bilinéaire
deep_supervision*	Indique si une supervision profonde est utilisée	False	True

La supervision profonde [7] (deep supervision) est une technique utilisée pour améliorer la performance et la stabilité de l'apprentissage. Elle consiste à ajouter des branches de supervision

supplémentaires à différents niveaux du réseau, permettant ainsi une supervision intermédiaire. Sans cette dernière, l'erreur de prédiction n'est généralement rétro propagée que depuis la sortie finale du réseau vers les couches internes. Toutefois, avec la supervision profonde dans U2-Net, l'erreur est rétro propagée depuis la sortie de chaque niveau Sup_i avec $i \in [1, 6]$ (Fig. 5.2). Cela signifie que des pertes intermédiaires sont calculées à partir des sorties des couches internes et utilisées pour guider l'apprentissage de ces couches, dans la fin de cette partie, on va voir l'évolution des métriques sur ces couches intermédiaires.

Le modèle est compilé avec l'optimiseur Adam et un taux d'apprentissage (learning rate) de 0.005. La fonction de perte utilisée est l'entropie croisée binaire (binary cross-entropy) et les métriques d'évaluation du modèle sur les données de validation comprennent l'exactitude (accuracy), le coefficient de Dice (Dice coefficient) et le coefficient IoU (Intersection over Union) pour évaluer la similarité entre les masques prédits et les masques réels.

— **Binary Cross Entropy** : L'entropie croisée binaire est une mesure de la différence entre la distribution de probabilité prédite par le modèle et la distribution de probabilité réelle pour un problème de classification binaire, Dans notre cas il s'agit d'une tâche de classification pixel par pixel, chaque pixel de l'image est classifié comme appartenant à l'objet d'intérêt ou au fond.
Elle est calculée comme suit :

$$\textbf{Binary Cross\_Entropy} = -\frac{1}{N}\sum_{i=1}^{N} y_i \log(p_i) + (1 - y_i)\log(1 - p_i) \tag{5.1}$$

où N est le nombre d'échantillons, y_i est la vraie valeur de l'échantillon i (0 ou 1), et p_i est la probabilité prédite par le modèle pour l'échantillon i.

— **Accuracy** : L'exactitude est une mesure de la précision globale du modèle dans la classification des échantillons. Elle est calculée comme le rapport entre le nombre d'échantillons correctement classés et le nombre total d'échantillons :

$$\textbf{Accuracy} = \frac{Nombre\ d'chantillons\ correctement\ classes}{Nombre\ total\ d'chantillons} \tag{5.2}$$

— **Dice Coefficient** : Le coefficient de Dice est une mesure de similarité entre deux ensembles. Il est calculé comme le rapport entre le double de l'intersection des ensembles et la somme des tailles des ensembles :

$$\textbf{Dice Coefficient} = \frac{2 \times Intersection}{Taille\ de\ l'ensemble\ prdit + Taille\ de\ l'ensemble\ rel} \tag{5.3}$$

Pour évaluer le Dice coefficient dans le cas de la segmentation binaire, l'ensemble prédit correspond aux pixels prédits comme faisant partie de l'objet d'intérêt, et l'ensemble réel correspond aux pixels réels de l'objet d'intérêt.

— **Intersection over Union (IoU)** : L'Intersection sur l'Union, aussi connue sous le nom de coefficient de Jaccard, est une autre mesure de similarité entre deux ensembles. L'IoU est calculé comme le rapport entre l'intersection des deux ensembles (les pixels correctement prédits comme appartenant à la classe) et l'union des deux ensembles (tous les pixels qui sont réellement dans la classe plus ceux qui sont prédits comme étant dans la classe) :

$$\textbf{IoU} = \frac{Intersection}{Union} = \frac{Intersection}{Ensembleprdit + Ensemblerel - Intersection} \tag{5.4}$$

L'évolution des métriques pour chaque niveau de U2-net a été analysée afin d'évaluer les performances du modèle pendant l'entraînement. Les graphiques sur la figure 5.5 illustrent les variations des métriques au fil des époques, mettant en évidence les résultats obtenus pour les différents niveaux de U2-net. Chaque graphique présente à la fois les valeurs d'en-

traînement et de validation. Ces graphiques fournissent des informations essentielles sur la convergence du modèle et sa capacité de généralisation aux données de validation.

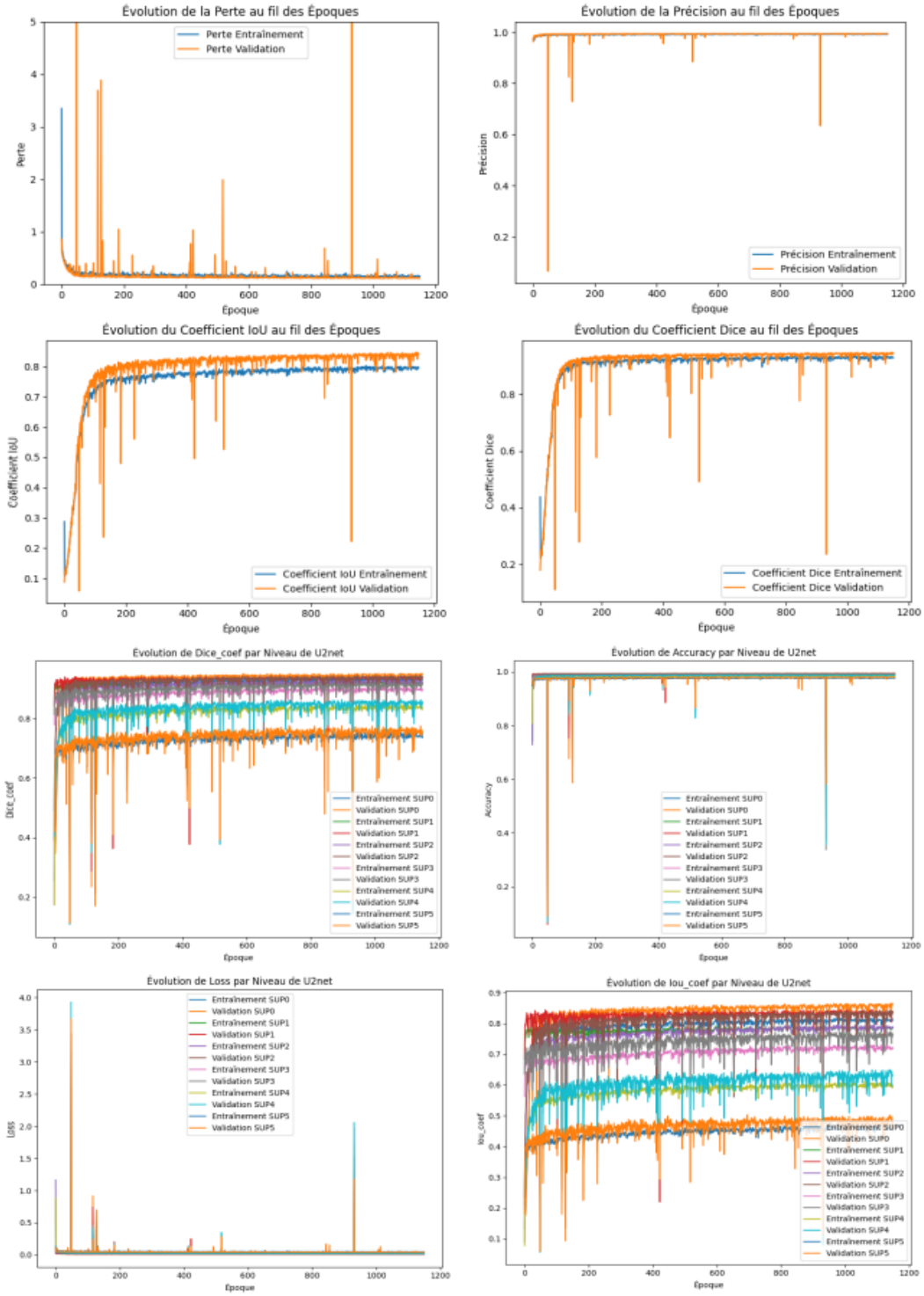


FIGURE 5.5 – Évolution de la perte et des métriques pendant l'entraînement de U2-Net

Il est important de noter que les métriques affichées sur les graphiques ci-dessus ne correspondent pas à un modèle entraîné à partir de zéro, mais à un modèle qui a été affiné (fine-tuned) à partir d'un modèle pré-entraîné sans supervision profonde qui n'était pas performant, ce qui explique la convergence rapide.

Les deux premières figures représentent respectivement la perte (loss) et l'exactitude (accuracy). On peut conclure que le modèle est performant, mais cela n'est pas suffisant pour

affirmer s'il s'agit d'un surajustement (overfitting) ou non. Cependant, les graphes de la deuxième ligne représentent respectivement les coefficients de Dice et d'IoU. On constate que le modèle est plus performant sur les données de validation, ce qui est normal étant donné que nous disposions d'un petit nombre de données pour la validation, qui ne présentaient pas beaucoup de variétés par rapport aux données d'entraînement. Cela confirme que le modèle peut être généralisé à de nouvelles données.

Finalement, les quatre dernières figures représentent l'évolution de la perte, de l'exactitude, et des coefficients de Dice et d'IoU pour les niveaux intermédiaires. Ces coefficients présentent des valeurs importantes pour les couches supérieures et la valeur maximale correspond à la couche de sortie SUP_0.

Évaluation du modèle U2-Net

Pour évaluer le modèle sur de nouvelles données qui diffèrent des données d'entraînement, des données synthétisées sur Blender [8] ont été utilisées. Ces données (50 échantillons) ont été créées pour ressembler aux instruments chirurgicaux et représenter des scénarios réalistes (Fig. 5.6).

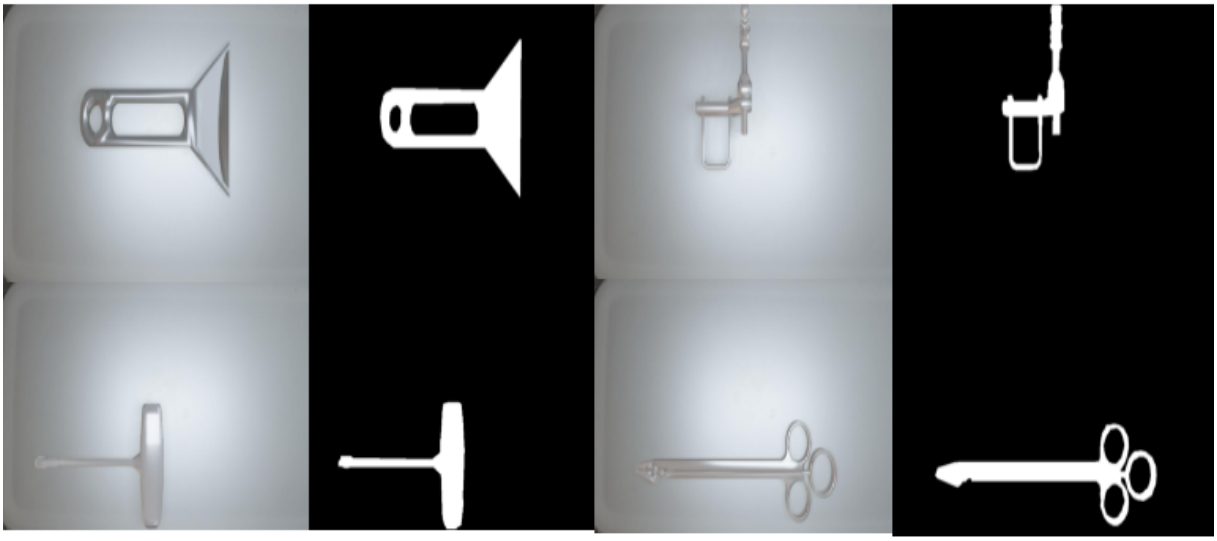


FIGURE 5.6 – Données de validation générées par Blender

Pour l'évaluation, plusieurs métriques ont été utilisées. La précision (precision) mesure la capacité du modèle à prédire correctement les pixels positifs. Le rappel (recall) mesure la capacité du modèle à détecter tous les pixels positifs. Le score F1 (F1-score) est une mesure combinée de la précision et du rappel, fournissant une évaluation globale des performances du modèle. Enfin, l'aire sous la courbe ROC (AUC-ROC) a été utilisé pour évaluer la capacité de discrimination du modèle entre les classes positives (pixels blancs) et négatives (pixels noirs).

Métrique	**Formule**	**Valeur dans le modèle initial**	**Valeur dans le modèle amélioré**
Précision	$P = \frac{TP}{TP + FP}$ (5.5)	96.36%	96,19%
Rappel (Recall - R)	$R = \frac{TP}{TP + FN}$ (5.6)	63,98 %	87,64 %
Score F1	$F1 - S = \frac{2 \times P \times R}{P + R}$ (5.7)	76,91 %	94,72 %
ROC AUC	Aire sous la courbe ROC	81,95 %	93,76 %

Pour mieux interpréter ces valeurs, la figure 5.7 illustre les courbes Précision-Rappel, F1-Score et la courbe ROC pour les deux modèles.

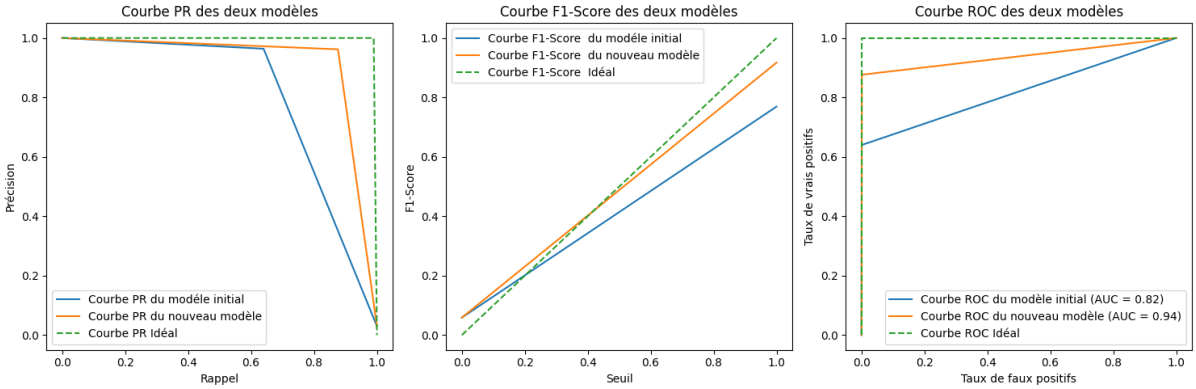


FIGURE 5.7 – Courbes d'évaluation du modèle de segmentation

Les courbes du modèle amélioré montrent une proximité avec les courbes idéales dans les trois graphiques, ce qui indique de bonnes performances sur de nouvelles données. Cela confirme que le modèle a réussi à généraliser et à obtenir des résultats satisfaisants sur de nouvelles données.

Optimisation du modèle U2-Net

La transition vers une architecture de réseau de neurones plus complexe a entraîné une augmentation significative du temps nécessaire pour effectuer les inférences.

Initialement, le modèle est entraîné avec l'API Keras [9] de TensorFlow [10], ce qui conduit à l'enregistrement du modèle au format H5 (Hierarchical Data Format 5). Ce format spécifique à TensorFlow permet de sauvegarder l'architecture du modèle, les poids des neurones et les autres paramètres nécessaires pour effectuer des prédictions ultérieures ou des ajustements.

Cependant, pour obtenir des inférences plus rapides, le format ONNX [11] (Open Neural Network Exchange) a été adopté. ONNX est un format de fichier ouvert qui facilite l'interopérabilité entre différents frameworks d'apprentissage automatique en permettant l'échange de modèles entre ces frameworks. En utilisant la bibliothèque ONNX Runtime dé-

veloppée par Microsoft pour exécuter des modèles ONNX, il est possible de bénéficier d'un environnement d'exécution optimisé offrant plusieurs fonctionnalités d'optimisation pour améliorer les performances des modèles d'apprentissage automatique. Ces optimisations sont regroupées en trois niveaux : basique, étendu et optimisations de mise en page.
Au niveau basique, ONNX effectue des optimisations de graphe telles que :

— **Le pliage constant (constant folding)** : calcule de manière statique les parties du graphe dépendant d'initialisations constantes.
— **L'élimination des nœuds redondants (Redundant node elimination)** : l'élimination des nœuds redondants supprime les nœuds inutiles sans modifier la structure du graphe (Identity, Slice, Unsqueeze and Dropout Elimination)
— **La fusion des nœuds (nodes fusion)** : regroupe plusieurs nœuds en un seul, réduisant ainsi les calculs superflus (Fusion de Conv-Add, Conv-Mul et Conv-BatchNorm (Fig. 5.8), etc...)

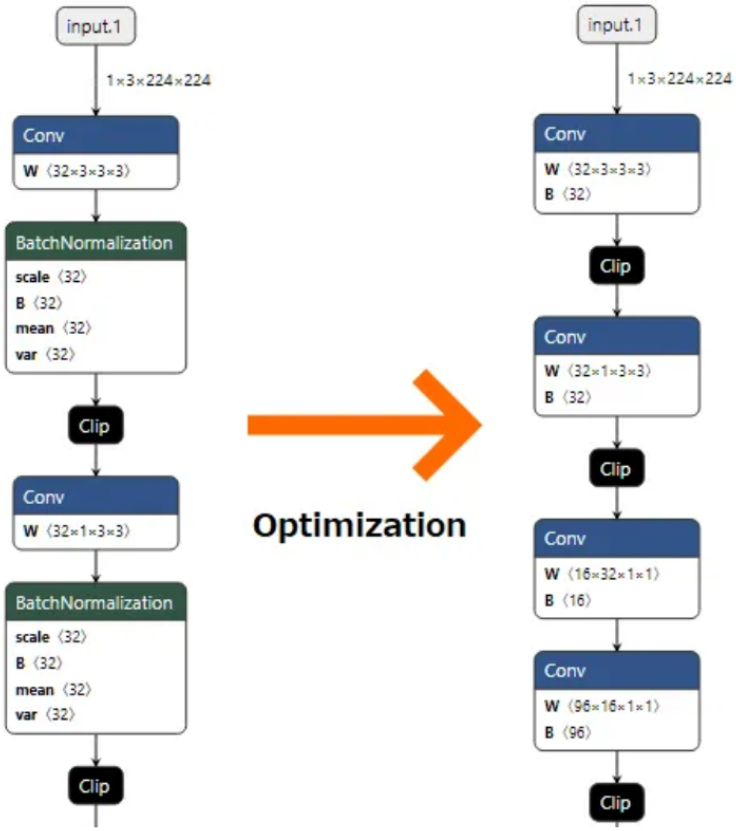
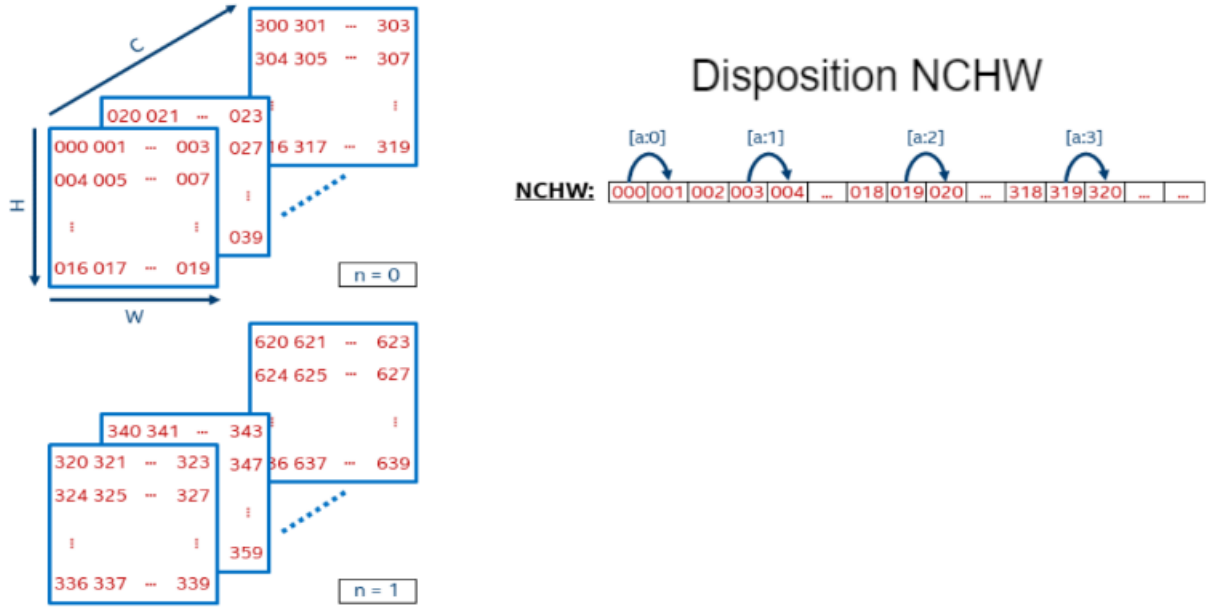


FIGURE 5.8 – Fusion de Convolution et Normalisation par Batch [12]

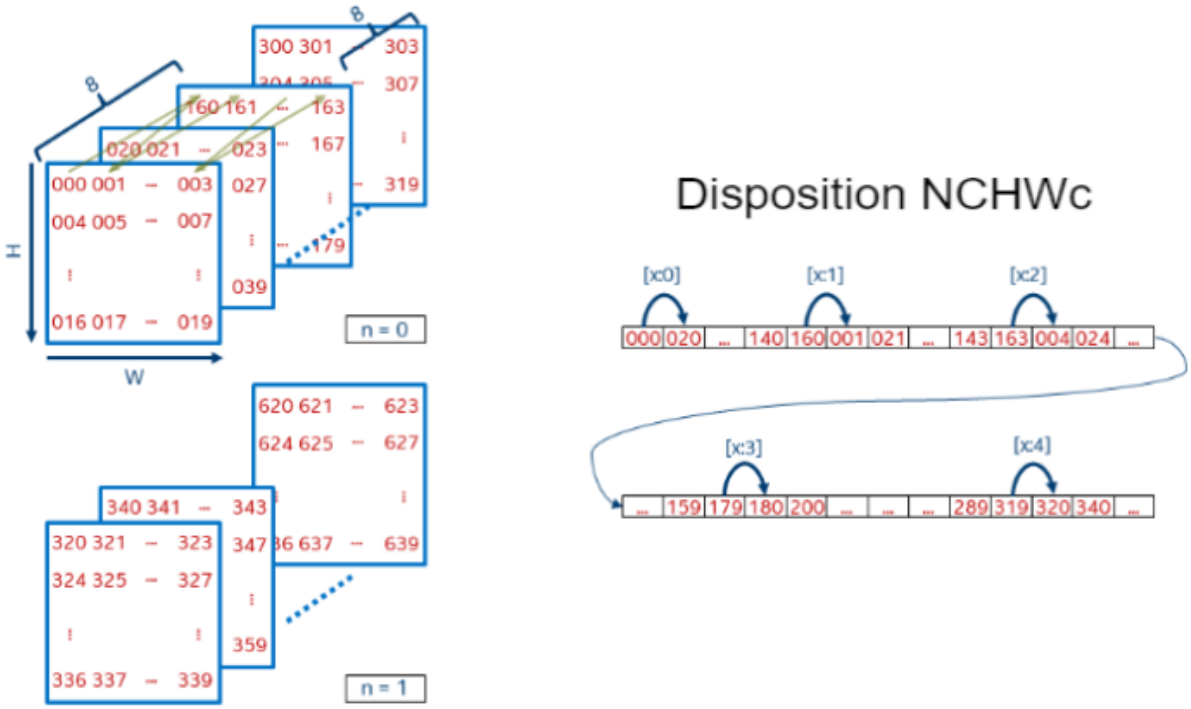
Les optimisations étendues comprennent des fusions de nœuds complexes qui sont exécutées après la partition du graphe et appliquées aux nœuds assignés au CPU. Ces optimisations, telles que la fusion GEMM (General Matrix Multiply) Activation qui permet de combiner opérations de multiplication de matrices et d'activation en une seule opération, et la fusion Conv Activation, qui combine les opérations de convolution et d'activation en une seule opération.

Finalement, les optimisations de mise en page se concentrent sur la modification de la disposition des données pour les nœuds assignés au CPU. Par exemple, l'optimiseur NCHWc optimise le graphe en utilisant la mise en page NCHWc au lieu de la mise en page NCHW, en effet Le format NCHW (Nombre de lots, Nombre de canaux, Hauteur, Largeur) est couramment utilisé pour représenter les dimensions des tenseurs. Il spécifie la taille du lot, le nombre de canaux, ainsi que la hauteur et la largeur des tenseurs d'entrée. Les données sont organisées séquentiellement, avec les canaux suivis des coordonnées spatiales

(Fig. 5.18a). Cependant, dans la mise en page NCHWc, les canaux sont regroupés en blocs contenant un nombre défini 'c' de canaux consécutifs. Cette organisation en blocs (Fig. 5.18b) améliore l'accès à la mémoire et au cache, réduisant ainsi les temps d'attente et améliorant les performances de calcul sur les CPUs.



(a) Format NCHW



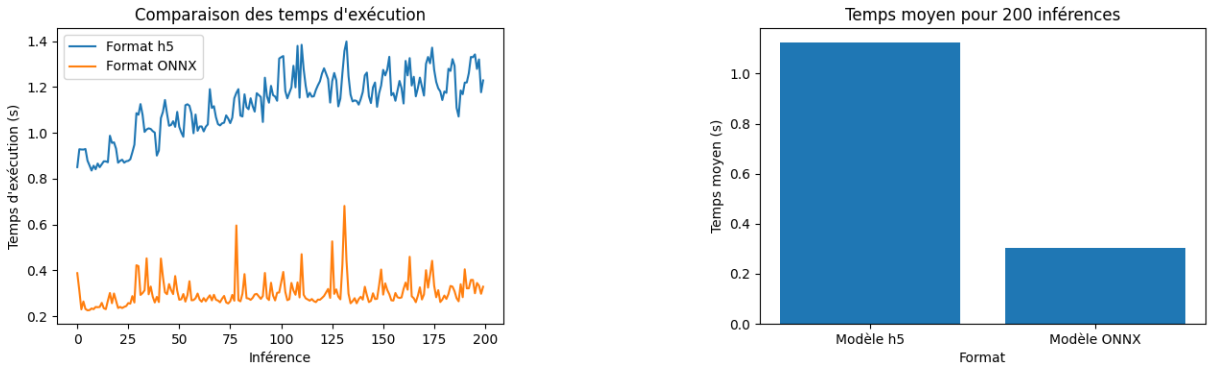
(b) Format NCHWc avec c=8

FIGURE 5.9 – Formats de représentation des tenseurs [13]

En plus de ces optimisations qui permettent de réduire la quantité de calcul, il existe d'autres techniques d'optimisation telles que la quantification et la précision mixte. La quantification réduit la complexité du modèle en réduisant le nombre de bits utilisés pour

représenter les paramètres, tandis que la précision mixte utilise des formats de données à virgule réduite pour accélérer les calculs. Toutefois, dans notre cas, nous nous sommes concentrés sur les optimisations de calcul fournies par ONNX vu qu'ils ne dégradent pas les performances du modèle.

La figure 5.10 présente deux graphiques : le graphique 5.10a montre le temps d'exécution en fonction du nombre d'inférences pour le modèle en formats H5 et ONNX, et un graphique à barres 5.10b représentant le temps moyens d'exécution pour 200 inférences pour les deux formats.



(a) Temps d'exécution (H5 vs. ONNX)

(b) Temps moyens d'exécution pour 200 inférences

FIGURE 5.10 – Comparaison de temps d'inférence pour les formats H5 et ONNX

La figure 5.10 illustre l'efficacité supérieure du modèle ONNX par rapport au modèle H5 en termes de temps d'exécution, avec un rapport de temps d'exécution moyen de 3.71. Cette performance supérieure justifie notre choix du format ONNX.

5.2 Caractéristiques du système amélioré

Cette section est dédiée à l'extraction de caractéristiques à partir des images.Deux types d'attributs se distinguent : ceux liés aux couleurs dans la première partie et ceux liés à la forme, notamment les attributs dérivés de la courbure et l'estimation de la profondeur. Les méthodes employées pour extraire ces attributs seront examinées en détail, ainsi que leur pertinence pour le développement.

5.2.1 Attributs colorimétriques

Dans le contexte de la reconnaissance d'instruments chirurgicaux, il est souvent essentiel de pouvoir distinguer plusieurs instruments ayant des formes similaires, mais présentant de légères différences de couleur. Ces différences de couleur sont souvent marquées par de petites bandes colorées sur les instruments, comme illustré dans la Figure 4.3a. Dans d'autres cas, des objets similaires peuvent également être différenciés par leur couleur, comme illustré dans la Figure 4.3b.

Afin d'extraire les informations liées à la couleur, l'histogramme de couleur bidimensionnel (HC2D) [14], également appelé carte thermique de densité a été utilisé. L'HC2D est une représentation graphique qui permet d'illustrer la distribution conjointe de deux variables numériques.

Dans le contexte de la couleur, l'histogramme bidimensionnel est calculé en analysant la distribution des composantes de couleur dans une image. En utilisant un espace de couleur approprié, tel que l'espace chromatique Lab* dans notre cas, chaque pixel de l'image est attribué à un bin bidimensionnel correspondant aux valeurs de ses composantes de couleur. Dans notre cas, nous avons choisi d'utiliser 7 bins pour représenter l'histogramme