

# Images de microscopie annotées pour la détection de micro-organismes : traitement d'image classique versus approches par apprentissage automatique

Nicolas Detzen <sup>1,\*</sup>, Pierre-Louis Filoche <sup>1,\*</sup>

Encadrés par Mélanie Pietri <sup>2,3</sup> et Sakina Bensalem <sup>2</sup>

<sup>1</sup> Université Paris-Saclay, ENS Paris Saclay, Département SIEN

<sup>2</sup> Université Paris-Saclay, ENS Paris Saclay, CNRS, LuMIn

<sup>3</sup> Université Paris-Saclay, CNRS, ENS Paris Saclay, LMF

\* Auteurs dans l'ordre alphabétiques

## Résumé

Les microalgues, telles que *Chlamydomonas reinhardtii*, suscitent un intérêt croissant pour leurs applications en biotechnologie par exemple dans la production de biocarburants. Leur étude repose notamment sur l'analyse d'images de microscopie permettant de quantifier automatiquement leur concentration via une segmentation de l'image. Dans ce cadre, la segmentation de cellules sur un nombre conséquent d'images est une étape cruciale, mais délicate, en raison de la variabilité des conditions expérimentales (densité cellulaire, contraste, bruit, autofluorescence).

Ce travail propose une comparaison approfondie de plusieurs méthodes de segmentation appliquées à des images de microscopie optique de *Chlamydomonas reinhardtii* et en particulier à des images en fluorescence mettant en évidence l'autofluorescence rouge de ces cellules lorsqu'elles sont exposées à de la lumière. Cette propriété intrinsèque des microalgues est exploitée pour segmenter les cellules, dans des images caractérisées par des points lumineux (les cellules) sur fond sombre.

Deux grandes familles d'approches sont étudiées : d'une part, des méthodes classiques de traitement d'image (seuillage, Watershed, analyse de composantes connexes, *CellProfiler*) ; d'autre part, des approches par apprentissage profond, en particulier via le détecteur d'objets YOLOv5.

Les résultats expérimentaux, évalués à l'aide d'une vérité terrain annotée manuellement, montrent que les méthodes classiques sont performantes sur des images à forte densité, notamment après optimisation des paramètres et prétraitements. En particulier, *CellProfiler* permet une segmentation fiable et reproductible sans réglage manuel excessif. Toutefois, en présence d'images hétérogènes ou à faible contraste, les méthodes traditionnelles montrent des limites. Les approches par apprentissage automatique, bien qu'exigeantes en données annotées, offrent une meilleure robustesse dans ces cas complexes.

Ce rapport met ainsi en évidence les compromis entre performance, généralisation et accessibilité des méthodes.

# Table des matières

<b>I Introduction</b>	<b>3</b>
<b>II Matériel &amp; Méthode</b>	<b>4</b>
II.1 Culture des microalgues . . . . .	4
II.2 Description du jeu de données . . . . .	4
II.3 Logiciels utilisés . . . . .	5
II.4 Approche classique . . . . .	5
II.4.1 Segmentation par composantes connexes . . . . .	5
II.4.2 Segmentation par érosion . . . . .	8
II.4.3 Segmentation par détection de contours . . . . .	8
II.4.4 Segmentation par "Watershed" avec contrôle des "seeds" . . . . .	10
II.4.5 Segmentation via le logiciel <i>CellProfiler</i> . . . . .	12
II.5 Approche par apprentissage . . . . .	14
II.5.1 Augmentation et segmentation du dataset . . . . .	14
II.5.2 You Only Look Once (YOLO) . . . . .	17
<b>III Résultats &amp; Discussion</b>	<b>19</b>
III.1 Métriques utilisées . . . . .	19
III.2 Méthode classique . . . . .	21
III.2.1 Remarque générale . . . . .	21
III.2.2 Segmentation par composantes connexes . . . . .	21
III.2.3 Segmentation par érosion . . . . .	22
III.2.4 Segmentation par détection de contours . . . . .	23
III.2.5 Segmentation par <i>Watershed</i> avec contrôle des "seeds" . . . . .	25
III.2.6 Segmentation via le logiciel <i>CellProfiler</i> . . . . .	30
III.2.7 Conclusion méthodes classiques et choix de la méthode . . . . .	40
III.3 Méthode par apprentissage . . . . .	41
<b>IV Conclusion</b>	<b>44</b>
<b>V Annexe</b>	<b>48</b>

# I Introduction

L'analyse automatisée d'images de microscopie est devenue un outil incontournable dans les sciences du vivant, permettant une quantification rapide et reproductible des caractéristiques morphologiques et dynamiques des cellules. Parmi les organismes d'intérêt, les microalgues, telles que *Chlamydomonas reinhardtii*, occupent une place centrale en raison de leur rôle écologique et de leurs applications biotechnologiques variées, notamment dans la production de biocarburants, la dépollution et la nutrition humaine [1, 2]. *Chlamydomonas reinhardtii* est un modèle d'étude privilégié pour la photosynthèse, la motilité cellulaire et les réponses au stress environnemental. Sa facilité de culture et sa génétique bien caractérisée en font un organisme modèle pour de nombreuses recherches fondamentales et appliquées.

L'observation de ces microalgues se fait couramment en microscopie dite *brightfield* et en fluorescence. En *brightfield*, les images présentent un contraste faible, rendant la segmentation des cellules difficile. En fluorescence, les chloroplastes des microalgues absorbent des photons de haute énergie et réémettent des photons de plus faible énergie (dans le rouge), produisant des images caractérisées par des points rouges sur fond noir. Cette autofluorescence naturelle facilite la détection des cellules sans marquage additionnel. Ce seront ce type d'images qui seront utilisées pour le développement des méthodes de segmentation de ce papier.

La segmentation des cellules dans ces images est une étape cruciale pour l'analyse quantitative. Les méthodes classiques, telles que le seuillage, la détection de contours, la transformation de Hough ou l'algorithme de watershed, sont largement utilisées. Des logiciels comme *CellProfiler* intègrent ces techniques pour permettre une analyse automatisée des images biologiques.

Cependant, ces méthodes présentent des limites, notamment en présence de bruit, de chevauchement cellulaire ou de variations d'intensité. L'émergence des méthodes d'apprentissage profond, en particulier les réseaux de neurones convolutifs, a révolutionné la segmentation d'images. Des architectures comme *YOLO* (You Only Look Once) permettent une détection rapide et précise des objets, y compris des cellules, à condition de disposer de jeux de données annotés en quantité suffisante.

Plusieurs jeux de données ont été développés pour l' entraînement et l'évaluation de ces algorithmes. Le jeu de données proposé par Zhou et al. [1] comprend 937 images et 4201 annotations de microalgues marines, avec des classes déséquilibrées et des échantillons mixtes pour simuler des conditions réelles. Orenstein et al. [2] ont développé un ensemble de 3,5 millions d'images pour la détection de phytoplancton. D'autres jeux de données, comme EMDS-5 [3] et CIDACC [4], offrent des images annotées pour diverses espèces de microalgues. Cependant, du fait de la spécificité de notre problématique, peu de données exploitables sont réellement disponibles. Il est en réalité long et fastidieux d'acquérir et d'annoter des images manuellement afin de constituer une véritable terrain exploitable pour le développement de méthodes de comptage automatique.

Afin d'améliorer la suffisance et la diversité des données d' entraînement, l'augmentation de données est devenue une étape incontournable pour l'application réussie des modèles d'apprentissage sur des images [16].

Il existe deux manières principales de mettre en œuvre la transformation du dataset existant [17] :

**En amont de l' entraînement (offline)** : les transformations sont appliquées une fois pour toutes, et les images augmentées sont enregistrées dans un nouveau dossier. Ce processus est simple à mettre en place et permet de vérifier visuellement les nouvelles images.

**Pendant l' entraînement (online)** : les transformations sont appliquées à la volée, de manière aléatoire à chaque epoch, directement dans le pipeline d' entraînement. Cette méthode permet de générer une grande diversité d'images sans augmentation du volume de stockage. Elle est particulièrement adaptée aux frameworks comme PyTorch ou TensorFlow, qui intègrent des outils de transformation d'images dans leurs *DataLoader*.

Ce travail vise donc à comparer différentes méthodes de segmentation pour le comptage de cellules de

*Chlamydomonas reinhardtii* à partir d’images de microscopie. Deux approches principales sont étudiées : les méthodes classiques basées sur des traitements d’images, et les méthodes d’apprentissage supervisé, notamment l’architecture *YOLO*. L’analyse porte sur la robustesse, la précision du comptage et la capacité de généralisation des méthodes sur des images acquises dans des conditions variées. Les résultats sont évalués à l’aide de vérités terrain manuelles et d’indicateurs quantitatifs, et discutés en fonction des propriétés intrinsèques des images. L’ensemble des détails concernant la bibliographie réalisée peut être retrouvée dans la partie *Annexe*, dans les tableaux 9 et ??.

## II Matériel & Méthode

Un jeu de données constitué d’images de microalgue a été constitué et annoté par Mélanie Pietri, doctante en deuxième année au sein des laboratoires LuMIn et LMF de l’ENS Paris-Saclay, à partir de cultures de la microalgue *C. reinhardtii*. Les données ont été récoltées avec les équipements de l’Institut d’Alembert (IDA). Des données supplémentaires avec des images contenant des microalgues et des bactéries ont également été récoltées.

### II.1 Culture des microalgues

Les cultures de la microalgue *C. reinhardtii* ont été réalisées au sein de l’IDA dans un incubateur MINI-TRON INFORS HT avec la souche UTEX 2244 CC-125 mt+, dont la nomenclature sur SAG est 34.89. Les cultures ont été réalisées dans un volume de travail de 50 mL dans une flasque Erlenmeyer d’une contenance de 250 mL. La vitesse de rotation du plateau d’agitation des flasques a été réglée à 100 min<sup>-1</sup>. La microalgue a été cultivée dans le milieu TAP Gibco<sup>TM</sup> (chaque quantité d’éléments est indiquée par la suite en mg.L<sup>-1</sup> : Chlorure d’ammonium 380, Molybdate d’ammonium (NH<sub>4</sub>)<sub>6</sub>Mo<sub>7</sub>O<sub>24</sub>–4 H<sub>2</sub>O 10.0, Acide borique H<sub>3</sub>BO<sub>3</sub> 10.0, Chlorure de calcium CaCl<sub>2</sub>–2 H<sub>2</sub>O 50.0, Chlorure de cobalt CoCl<sub>2</sub>–6 H<sub>2</sub>O 20.0, Sulfate de cuivre CuSO<sub>4</sub>–5 H<sub>2</sub>O 20.0, Sulfate de fer FeSO<sub>4</sub>–7 H<sub>2</sub>O 20.0, Sulfate de magnésium MgSO<sub>4</sub>–7 H<sub>2</sub>O 100.0, Chlorure de manganèse MnCl<sub>2</sub>–4 H<sub>2</sub>O 50.0, Phosphate monobasique de potassium KH<sub>2</sub>PO<sub>4</sub> 50.0, Phosphate dibasique de potassium K<sub>2</sub>HPO<sub>4</sub> 110.0, Sulfate de zinc ZnSO<sub>4</sub>–7 H<sub>2</sub>O 20.0, EDTA 50.0, Acide acétique glaciale 1100.0, Tris 2420.0). Le pH initial des cultures a été mesuré à pH = 7.23 pour une température régulée à 25°C. Le taux de CO<sub>2</sub> dans l’incubateur a été fixé à 1.5%.

### II.2 Description du jeu de données

Les images des microalgues ont été réalisée à l’aide d’un microscope NIKON en brightfield et en fluorescence avec une longueur d’onde excitation de 488 nm et une longueur d’onde d’émission de 645 nm. Les microalgues ont été placées sur des lames KOVA pour être observées. Les informations statistiques à propos du jeu de données sont conciliées sur la Figure 1.

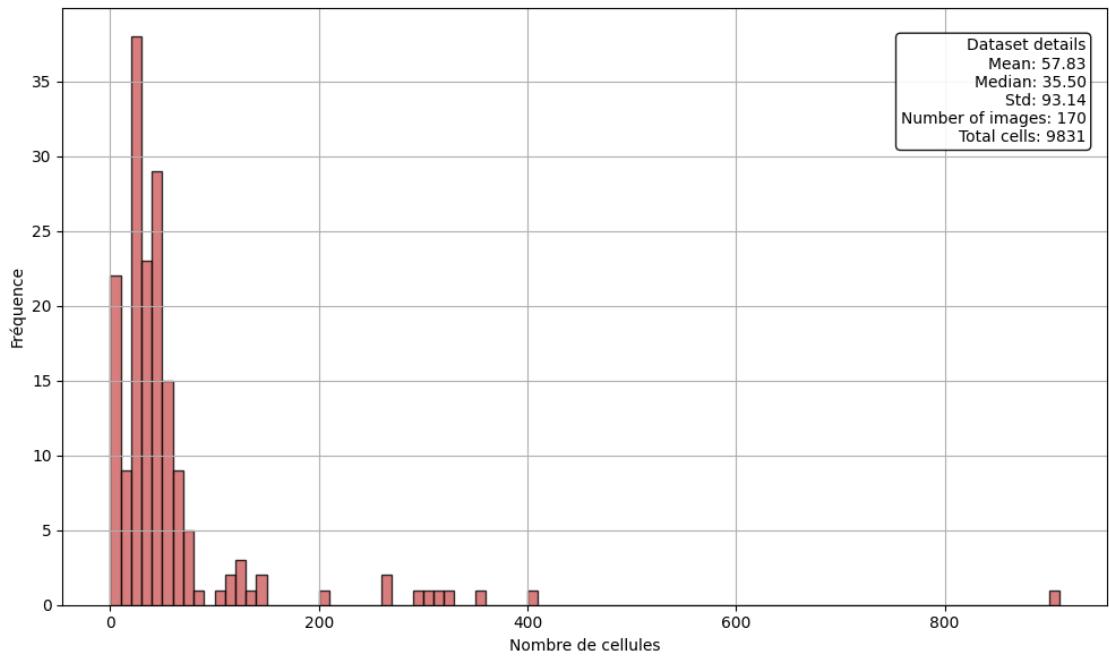


FIGURE 1 – Distribution de la densité cellulaire par images dans le jeu données.

### II.3 Logiciels utilisés

Plusieurs environnements logiciels ont été mobilisés au cours de ce travail :

- *Python* (via l'environnement *Spyder*) a été utilisé pour l'implémentation des méthodes classiques de traitement d'image, incluant les filtres, le seuillage, la détection de contours, et la segmentation par composantes connexes ou par *watershed*.
- *CellProfiler* a été utilisé comme alternative aux scripts Python pour la segmentation classique. Ce logiciel open-source dédié à l'analyse d'images biologiques permet la création de pipelines de traitement robustes et reproductibles, sans avoir à coder manuellement chaque étape.
- *Roboflow* a été employé pour la partie dédiée à l'apprentissage automatique. Cette plateforme en ligne permet l'annotation, l'augmentation de données, l'entraînement de modèles de détection d'objets (type *YOLO*), ainsi que l'évaluation des performances de ces modèles de manière intuitive.

### II.4 Approche classique

L'ensemble des méthodes utilisées dans cette section ont été implémentées via le langage de programmation *Python*. L'ensemble des codes utilisées ainsi que les informations liées peuvent être retrouvées sur Github : [https://github.com/melastik/microalgae\\_detection](https://github.com/melastik/microalgae_detection)

#### II.4.1 Segmentation par composantes connexes

La première méthode de segmentation utilisée repose sur la détection des composantes connexes dans l'image binaire, à l'aide de la fonction `connectedComponents` de la librairie *cv2*. Une composante connexe désigne un ensemble de pixels ayant la même valeur (par exemple, des pixels blancs) et étant connectés entre eux selon une certaine règle de voisinage. On distingue généralement deux types de connexité :

- la 4-connexité, où un pixel est connecté à ses voisins situés horizontalement et verticalement ;
- la 8-connexité, où un pixel est également connecté à ses voisins diagonaux.

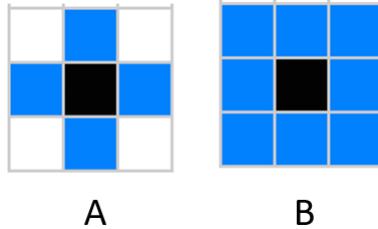


FIGURE 2 – Exemple de 4-connexité (A) et de 8-connexité (B)

Dans le cadre du comptage cellulaire, cette approche est pertinente car, après seuillage, chaque cellule bien segmentée apparaît comme une "tâche" isolée de pixels blancs dans l'image binaire. Ainsi, chaque cellule est naturellement identifiée comme une composante connexe distincte. Cette hypothèse est valable tant que les cellules sont suffisamment séparées et que le bruit est raisonnablement filtré.

Avant l'utilisation de la fonction `connectedComponents`, il est indispensable de réaliser plusieurs pré-traitements sur l'image brute. Ces étapes visent à simplifier et structurer l'information contenue dans l'image afin de faciliter une segmentation fiable et automatisée.

Une première étape consiste en une conversion en niveaux de gris. Cette opération réduit l'image à une seule composante d'intensité par pixel, simplifiant ainsi l'analyse tout en conservant l'information structurelle essentielle à la détection des objets d'intérêt. Elle permet également une réduction du volume de données à traiter, ce qui améliore l'efficacité et la rapidité des traitements ultérieurs.

Après conversion en niveaux de gris, nous appliquons ensuite un seuillage automatique basé sur la méthode d'Otsu [7], une technique non supervisée de binarisation largement utilisée en traitement d'image. Cette méthode repose sur l'analyse de l'histogramme de l'image, c'est-à-dire la distribution des niveaux de gris des pixels.

L'objectif de la méthode est de trouver un seuil d'intensité qui permet de séparer l'image en deux classes : les pixels appartenant à l'objet d'intérêt (par exemple, les cellules), et ceux appartenant à l'arrière-plan. Pour cela, *Otsu* explore tous les seuils possibles et, pour chacun, calcule la variance intra-classe (c'est-à-dire la dispersion des intensités à l'intérieur de chaque classe). Le seuil optimal est celui qui minimise la variance intra-classe totale, ce qui revient à maximiser la séparation entre les deux classes. Cette approche est entièrement automatique : elle ne nécessite aucun paramètre préalable ni ajustement manuel, ce qui en fait une méthode particulièrement robuste et reproductible. Elle est donc bien adaptée à des traitements en série ou à l'analyse d'images acquises dans des conditions variables.

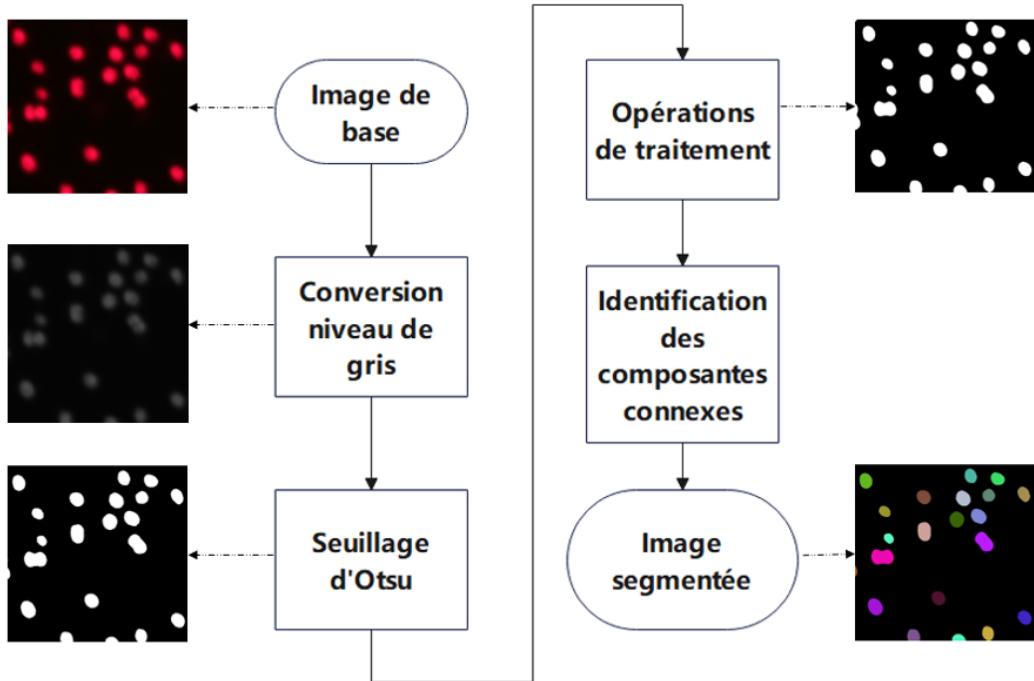


FIGURE 3 – Diagramme de flux de la méthode par composantes connexes

Une opération de traitement est cependant nécessaire pour filtrer les pixels résiduels, réduire le bruit et combler d'éventuelles lacunes au sein des cellules seuillées dues à des variations d'intensité locales. Cette opération, que nous appliquerons de manière systématique dans les différentes méthodes de segmentation proposées, repose sur une combinaison d'opérations morphologiques et de filtrage spatial. Elle se décompose en plusieurs étapes :

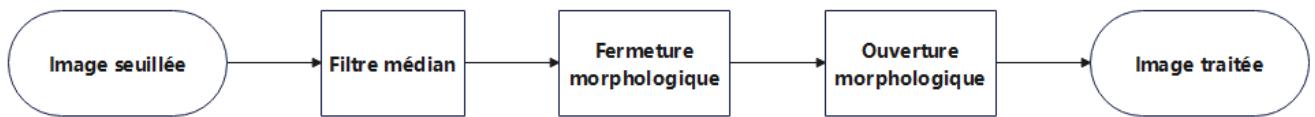


FIGURE 4 – Détails du bloc *Opérations de traitement*

- l'utilisation d'un filtre médian : cette étape permet d'éliminer les pixels isolés ou aberrants, souvent issus du bruit de fond ou d'artefacts liés à l'acquisition.
- une fermeture morphologique suivie d'une ouverture morphologique : la fermeture morphologique est une opération composée d'une dilatation suivie d'une érosion. Elle a pour effet de combler les petites discontinuités, comme les trous ou ruptures internes aux objets segmentés, en assurant leur continuité. À l'inverse, l'ouverture morphologique, composée d'une érosion suivie d'une dilatation, permet de supprimer les petits objets ou irrégularités résiduelles qui ne sont pas reliés à des structures significatives.

Appliquées de manière successive, ces deux opérations permettent de régulariser les formes segmentées : la fermeture consolide les structures internes, tandis que l'ouverture supprime les détails parasites et affine les contours. L'ensemble du processus améliore ainsi la qualité de la segmentation binaire, en assurant une représentation plus fidèle et exploitable des cellules.

Une fois que l'opération de traitement a abouti, l'image binaire obtenue est suffisamment propre et structurée pour permettre une segmentation efficace. Celle-ci est alors réalisée à l'aide de la fonction *connectedComponents*, qui permet d'identifier et d'indexer automatiquement les régions connexes présentes dans l'image. La fonction *connectedComponents* analyse l'image binaire et attribue un label unique à chaque groupe de pixels adjacents appartenant à une même composante connexe, c'est-à-dire formant un objet

cohérent spatialement. Cette méthode permet une segmentation rapide et efficace, notamment dans des cas où les cellules sont bien séparées après prétraitement.

#### II.4.2 Segmentation par érosion

La segmentation par érosion reprend une grande partie des étapes de la méthode précédente :

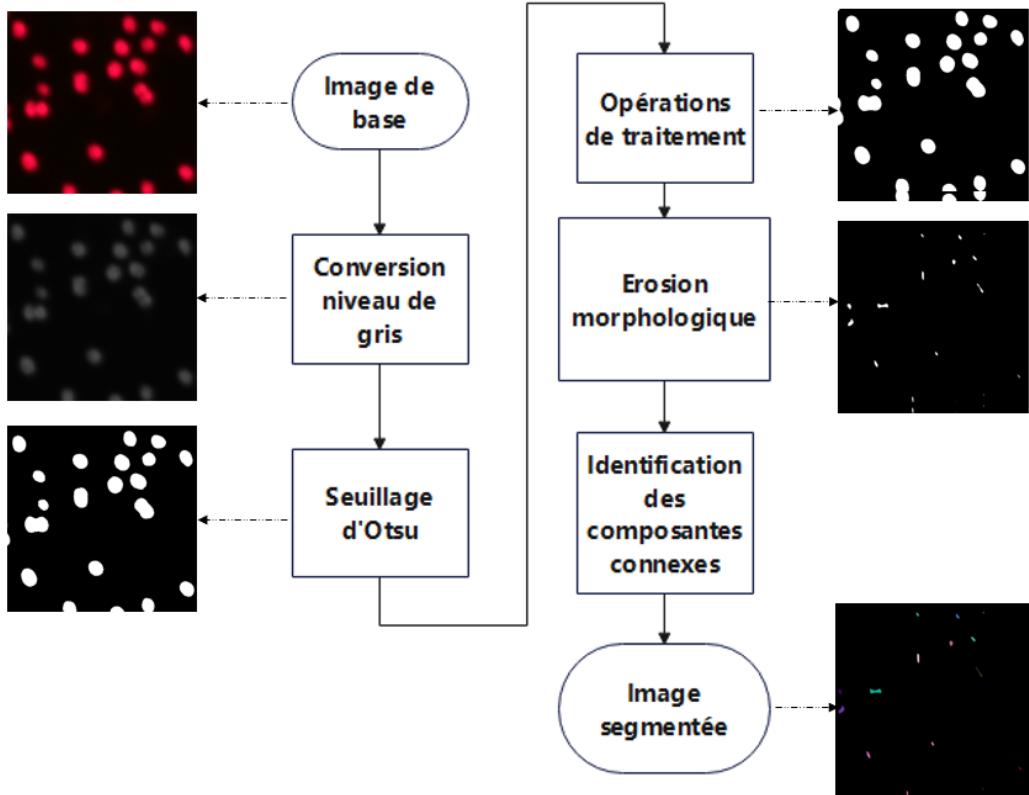


FIGURE 5 – Diagramme de flux de la méthode par érosion

L'inconvénient majeur d'une segmentation reposant uniquement sur la détection de composantes connexes apparaît lorsque plusieurs cellules se touchent ou sont fortement adjacentes. Cette détection, basée sur la simple continuité spatiale des pixels, considère alors l'ensemble des cellules connectées comme une seule et unique composante. Il en résulte une sous-segmentation : plusieurs objets distincts biologiquement sont fusionnés en un seul segment, faussant ainsi le comptage et l'analyse morphologique.

Pour atténuer ce phénomène, une étape d'érosion morphologique est introduite en amont de la détection des composantes connexes. L'érosion permet de réduire légèrement la taille des cellules segmentées, et ainsi de briser les ponts de connexion trop fins ou artificiels entre cellules adjacentes. Cette contraction des objets vise à favoriser leur séparation en entités distinctes lors de l'analyse de connectivité.

#### II.4.3 Segmentation par détection de contours

Une seconde approche, complémentaire à la méthode précédente, repose sur l'extraction des zones colorées caractéristiques des cellules dans l'espace HSV (*Hue*, *Saturation*, *Value*), qui sera défini par la suite. Cette méthode tire parti d'une particularité visuelle des images analysées : les cellules apparaissent principalement en nuances de rouge et de rose. Cela s'explique par la nature des microalgues étudiées, riches en pigments fluorescents rouges, comme mentionné en introduction. Il est important de noter que cette méthode s'applique spécifiquement aux images acquises en fluorescence, où ces pigments sont excités, et ne serait pas directement utilisable sur des images en lumière transmise (*brightfield*).

Elle s'affranchit donc d'une binarisation globale pour se concentrer sur les composantes chromatiques pertinentes.

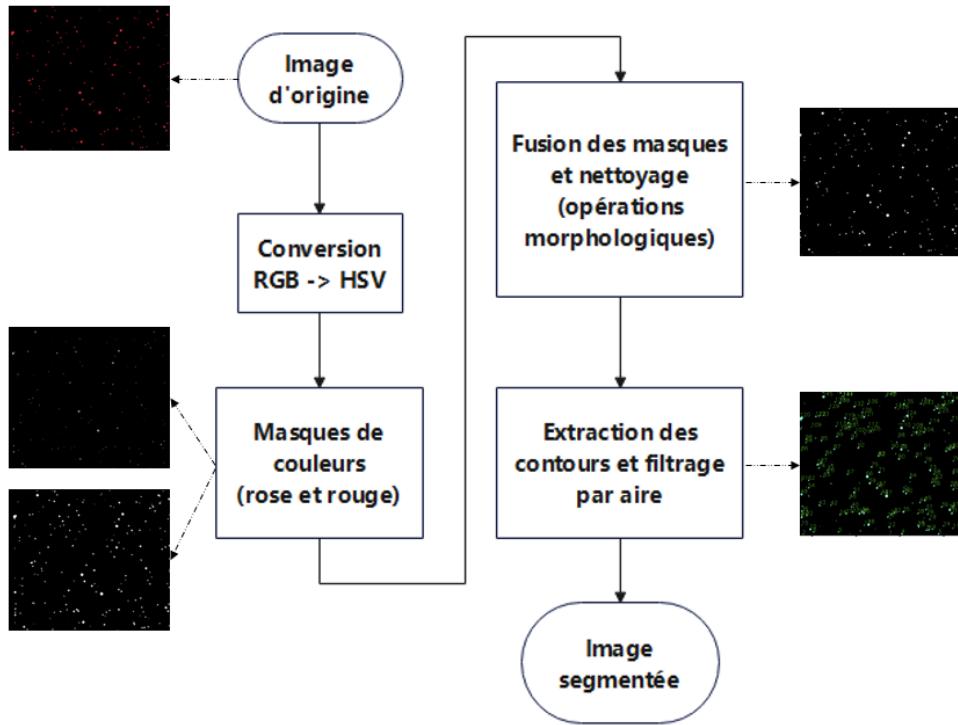


FIGURE 6 – Diagramme de flux de la méthode par détection de contours

Dans un premier temps, l'image est convertie en espace HSV afin de faciliter l'identification des plages de teinte associées aux cellules. Trois plages sont définies : deux pour les différentes nuances de rouge, et une troisième pour le rose. Cette étape permet de générer un masque binaire isolant les pixels correspondant à ces plages de couleurs.

**Espace HSV** L'espace de couleur HSV est utilisé dans ce code pour segmenter les cellules en fonction de leur teinte. Contrairement à l'espace RGB, où les composantes de couleur sont fortement corrélées, l'espace HSV sépare la couleur pure (Hue) de l'intensité lumineuse (Value) et de la saturation (7). La séparation des composantes chromatiques et lumineuses permet une détection plus robuste aux variations d'éclairage. Par exemple, une cellule rouge foncé et une cellule rouge clair auront des teintes similaires (*Hue*) mais des valeurs (*Value*) différentes, ce qui rend leur détection plus cohérente dans HSV que dans RGB.

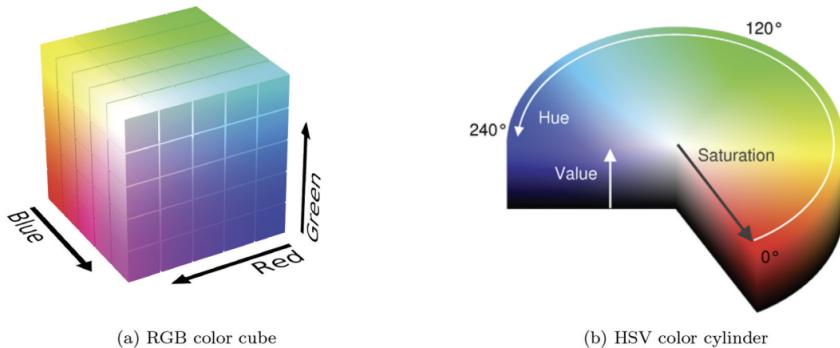


FIGURE 7 – RGB et HSV représentation - /24/ figure 5

Le code convertit chaque image de BGR (format natif d'OpenCV) en HSV via la fonction `cv2.cvtColor`. Ensuite, il applique un seuillage avec des plages définies pour détecter :

- les rouges classiques (plages  $[0^\circ - 10^\circ]$  et  $[170^\circ - 180^\circ]$ ) ;
- les teintes rosées (plage autour de  $140^\circ - 170^\circ$ ).

Ces plages sont choisies empiriquement pour capter les couleurs typiques des cellules dans les images du jeu de données.

Dans notre code nous avons donc 3 masques, deux rouges et un rose pour s'adapter le mieux à nos données. En effet, cela nous permet d'isoler les cellules du bruit de couleur différentes présents sur certaines images. Dans le cas où on aurait plusieurs types de cellules (micro-algues et bactéries par exemple) cela permettrait de segmenter par rapport à la couleur aussi. Ici on choisit les plages  $[0, 10]$  et  $[170, 180]$  pour les deux masques rouges (voir Remarque suivante pour explications sur les valeurs) et  $[140, 170]$  pour le masque rose (ce qui revient à élargir le deuxième masque rouge).

Le masque ainsi obtenu est ensuite filtré par des opérations morphologiques, notamment une ouverture, afin d'éliminer le bruit et les petites structures non pertinentes.

Une fois le masque nettoyé, les contours sont extraits à l'aide de la fonction `findContours` de la bibliothèque `opencv-python`, avec l'option `RETR_EXTERNAL` permettant de ne conserver que les structures externes, typiquement associées à des objets bien définis comme les cellules. Un filtre sur la taille des contours permet d'écartier les éléments trop petits ou trop grands, qui pourraient correspondre à du bruit ou à des agrégats. Enfin, chaque contour retenu est encadré d'un rectangle et numéroté afin de visualiser les cellules détectées.

#### II.4.4 Segmentation par "Watershed" avec contrôle des "seeds"

La méthode de segmentation par "lignes de partage des eaux" (watershed) est une technique classique de segmentation, basée sur les régions et utilisant la morphologie mathématique. Elle a été introduite par Digabel et Lantuejoul dans le domaine du traitement d'image à la fin des années 1970. [10]. Ensuite, Vincent, Beucher et al. ont établi la théorie du *Watershed*, qui a depuis suscité un large intérêt dans le domaine de la segmentation d'images. [11]. L'algorithme du *Watershed* peut être considéré comme une carte topographique. La valeur de gris des pixels de l'image représente le relief : plus la valeur de gris est élevée, plus le point est considéré comme un sommet ; plus elle est faible, plus il est considéré comme une vallée. Si l'eau s'écoule depuis un point élevé, elle se dirigera vers les zones plus basses du bassin jusqu'à atteindre une région locale, et au final, toute l'eau sera répartie dans différents bassins. La ligne de partage des eaux correspond à la crête entre ces bassins. Dans la segmentation d'image, le principal objectif de l'algorithme du *Watershed* est d'identifier les différents bassins versants et lignes de partage des eaux. Les bassins représentent des zones cibles ayant des caractéristiques différentes, tandis que les lignes de partage des eaux représentent les contours ou bords des objets. Le but de la segmentation est donc d'identifier ces lignes de partage.

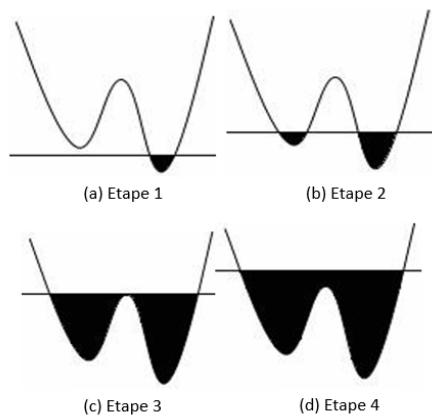


FIGURE 8 – Exemple des étapes de la méthode *Watershed* Figure issue de [13]

Dans la Figure 8, nous supposons qu'il y a deux bassins adjacents. Nous élevons progressivement le plan horizontal, à partir du point le plus bas du bassin. En continuant d'augmenter le niveau, à un

moment donné (niveau d'eau 3, comme illustré dans la figure 4(c)), les eaux des deux bassins vont se rencontrer. Le niveau d'eau 3 correspond alors à la ligne de partage des eaux.

Cependant, si la segmentation par *Watershed* est appliquée directement à l'image du gradient de magnitude (mesure l'intensité des variations de luminosité dans une image. Autrement dit, il indique à quel point la couleur ou la luminosité change rapidement d'un pixel à l'autre), elle entraîne presque toujours une sur-segmentation, en raison des variations d'intensité à l'intérieur des objets ainsi que dans l'arrière-plan. Au lieu de laisser l'eau monter depuis chaque minimum de l'image, nous pouvons restreindre la montée de l'eau uniquement à certains points marqués comme des graines (seeds) : [8]

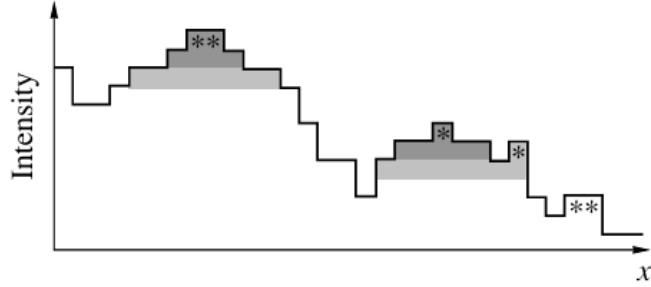


FIGURE 9 – Exemple de maxima en 1D - évolution de l'intensité suivant la position. Figure issue de [8]

Les maxima locaux dans une image 1D, ou dans un profil d'intensité, sont marqués par des étoiles et représentent le résultat de la transformation h-maxima avec  $h = 1$  sur la Figure 9. Si  $h = 2$  ou  $h = 3$ , le résultat correspondra respectivement aux régions en gris foncé ou gris clair. Un faible  $h$  donnera lieu à de nombreuses petites régions, tandis qu'un  $h$  plus élevé produira moins de régions, mais plus grandes.

La méthode utilisée dans cette partie faisant intervenir l'algorithme du *Waterhed* décrite à travers les étapes de la Figure 10. Les premières étapes de pré-traitement et de seuillage sont similaires à celles décrites précédemment. La transformée de distance est utilisée pour calculer, pour chaque pixel, sa distance au bord du fond de l'image préalablement identifié. L'algorithme de *Watershed* est ensuite appliqué sur cette carte de distances : chaque minimum local correspond à une seed (graine) à partir de laquelle la "montée des eaux" peut débuter. Afin de limiter les phénomènes de sur-segmentation (qui seront détaillés dans la section Résultats et discussions), une analyse de la circularité et de l'aire de chaque segment a été effectuée. Si l'une de ces deux caractéristiques est inférieure à un seuil prédéfini, la seed associée est supprimée, comme suggéré dans [12]. Un second *Watershed* est alors exécuté, cette fois à partir des seeds conservées.

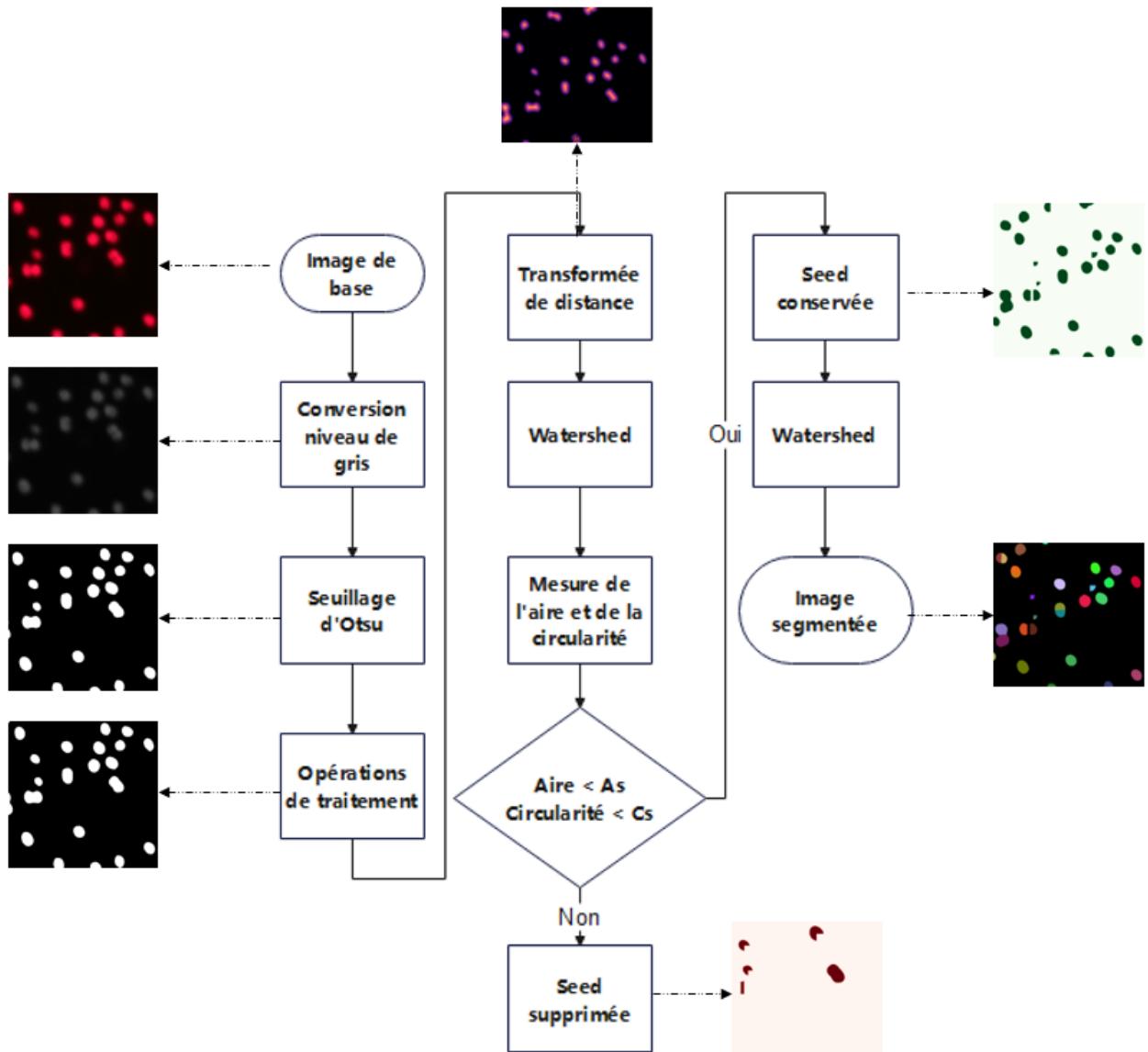


FIGURE 10 – Diagramme de flux de la méthode par Watershed avec contrôle des "seeds"

#### II.4.5 Segmentation via le logiciel *CellProfiler*

*CellProfiler* est un logiciel libre et open-source dédié à l'analyse quantitative d'images biologiques, développé initialement par le *Broad Institute of MIT and Harvard*. Conçu pour le traitement d'images microscopiques en grand nombre, il permet l'extraction automatique de caractéristiques morphologiques, d'intensité et de texture à différentes échelles. Basé sur une interface modulaire, *CellProfiler* facilite la construction de pipelines d'analyse sans nécessiter de compétences en programmation, tout en offrant une grande flexibilité pour adapter les traitements aux spécificités des échantillons. Il propose des algorithmes avancés de segmentation et d'analyse d'images.

Par rapport aux méthodes précédentes, une étape de redimensionnement de l'intensité a été ajoutée. Elle consiste en l'étirement de l'histogramme pour utiliser l'ensemble des valeurs possibles du noir au blanc. Cet ajout est généralement conseillé dans le cas de la segmentation d'image [14]. Les mêmes opérations de traitement (filtre médian et ouvertures/fermetures morphologiques) ont été implémentées. Un filtre gaussien a également été ajouté et son utilisation sera justifiée dans la partie *Résultats et discussions*. La segmentation de l'image est réalisée via le module *IdentifyPrimaryObjects*. Parmi l'ensemble des méthodes proposées, celle portant le numéro 34 dans l'article [15] a été retenue.

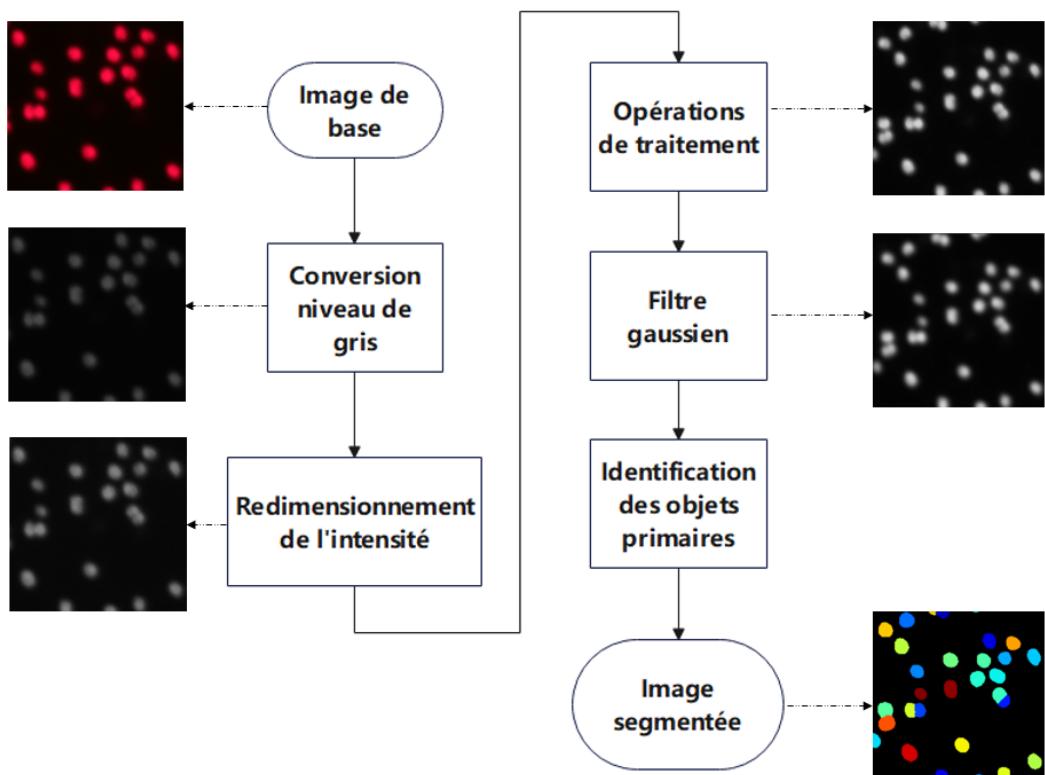


FIGURE 11 – Diagramme de flux via le logiciel "CellProfiler"

L'interface graphique du logiciel avec les modules utilisées pour réaliser la segmentation décrite figure 11 est détaillée sur la figure 12 :

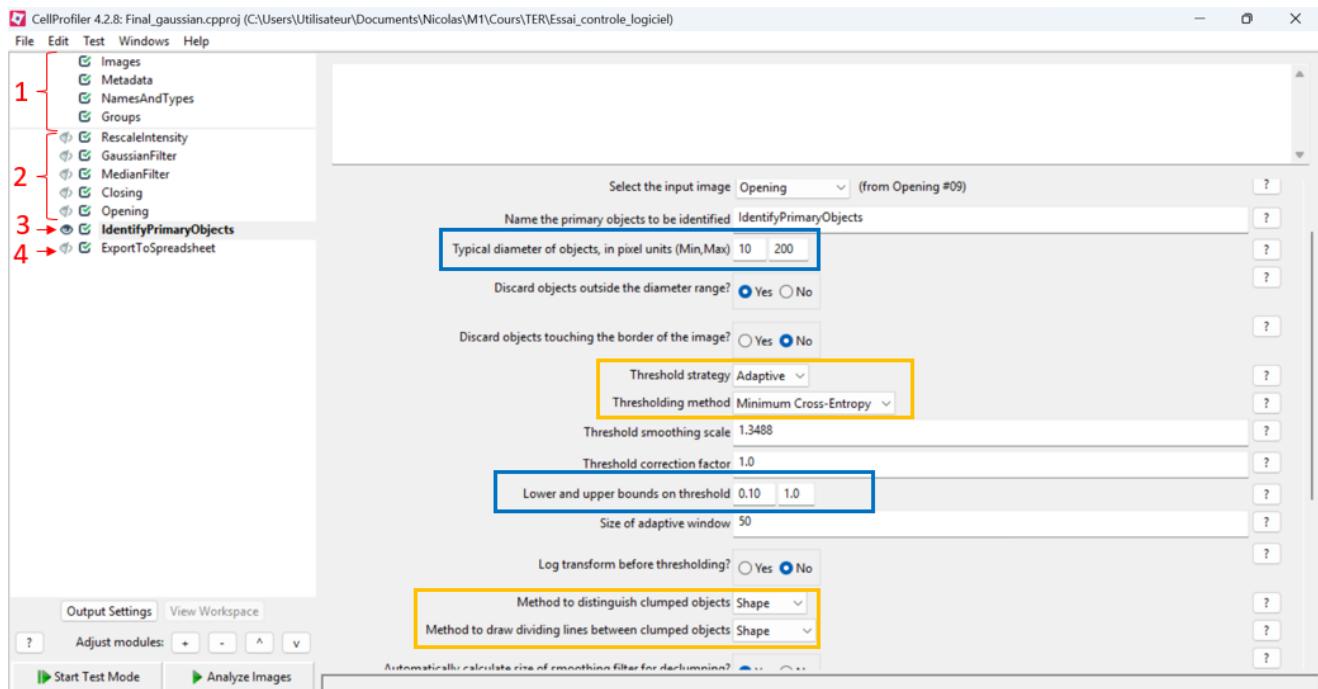


FIGURE 12 – Interface graphique du logiciel *CellProfiler*

- (1) : modules pour l'ouverture de l'image
- (2) : modules pour l'opération de traitement de l'image
- (3) : module pour réaliser la segmentation
- (4) : export des informations souhaitées

## II.5 Approche par apprentissage

Nous avons également décidé de mettre en place plusieurs modèles d'apprentissage automatique (*machine learning*) pour le comptage des cellules. Ce choix se justifie par la complexité des images et la variabilité des formes et tailles des cellules, rendant difficile l'élaboration de méthodes de comptage entièrement fondées sur des règles fixes.

L'ensemble des entraînements et des expériences numériques a été réalisé sur un ordinateur équipé d'un GPU Nvidia Quadro RTX 4000 (8 Go de RAM) et d'un processeur Xeon W disposant de 164 Go de mémoire vive.

L'une des principales difficultés pour l'apprentissage automatique dans notre cas est la faible taille de notre jeu de données. Cette limitation s'explique non seulement par le temps et l'effort nécessaires pour annoter manuellement les images, chaque cellule devant être identifiée et comptée individuellement, mais également par le temps et l'effort requis pour développer des cultures de microalgues et prendre des images de qualité sans dégrader les échantillons.

Afin de pallier cette contrainte, nous avons exploré plusieurs méthodes d'augmentation de données, permettant de générer artificiellement de nouvelles images d'entraînement à partir des images existantes.

### II.5.1 Augmentation et segmentation du dataset

Dans un premier temps, nous avons appliqué une stratégie d'augmentation offline, afin de générer de nouveaux exemples avant l'entraînement et d'évaluer visuellement la qualité des images synthétiques.

Trois jeux de données ont ainsi été constitués : **Le jeu de données original**, composé de 170 images au total, dont 120 utilisées pour l'entraînement. **Un premier jeu de données augmenté** via la plateforme RoboFlow [23], en appliquant trois transformations distinctes :

- *Flip horizontal et vertical* : les images sont retournées pour enrichir la diversité géométrique.
- *Modification de la saturation* : un facteur de saturation aléatoire compris entre  $-25\%$  et  $+25\%$  est appliqué.
- *Flou gaussien* : un flou jusqu'à 2.5 pixels est appliqué pour simuler des variations de mise au point.

Ainsi, chaque exemple original produit trois sorties augmentées, ce qui triple artificiellement la taille du dataset d'entraînement et favorise la généralisation.

**Un troisième jeu de données** combinant les augmentations RoboFlow avec une méthode de type *Copy-Paste* décrit sur la Figure 13. Notre programme permet de générer automatiquement de nouvelles images annotées à partir d'un dataset existant, dans le but d'augmenter les données pour entraîner un modèle de détection. L'idée est de créer des images artificielles en copiant les bounding boxes des cellules détectées dans différentes images originales, puis en les collant aléatoirement sur un fond noir. Cela permet de recombiner des éléments existants pour produire des exemples variés, tout en gardant un format compatible avec l'annotation YOLO. En répétant ce processus avec des combinaisons différentes, on enrichit efficacement le jeu de données sans avoir besoin de nouvelles images réelles. On choisit de former une nouvelle image à partir de 3 images originales en prenant aléatoirement  $\frac{1}{3}$  des bounding boxes de chaque image originale.

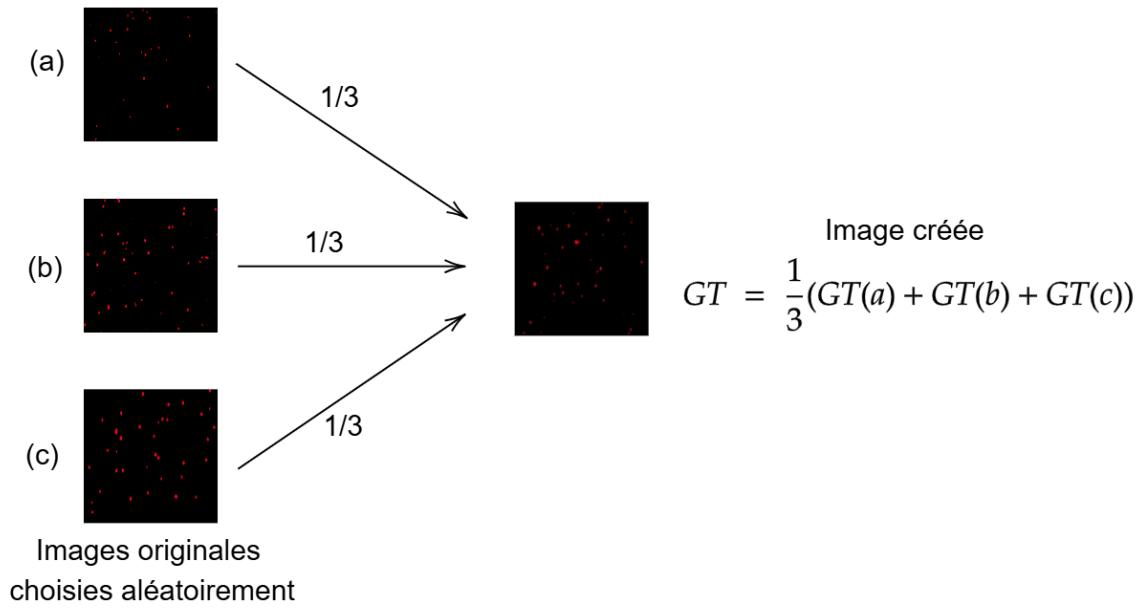


FIGURE 13 – Schéma explicatif du copy\_paste

On effectue cette augmentation sur la partie **train** du dataset en créant 120 nouvelles images (on double le nombre). On effectue ensuite les mêmes transformations via Roboflow que pour le deuxième dataset portant le total d'images annotées à 720 pour le **train**.

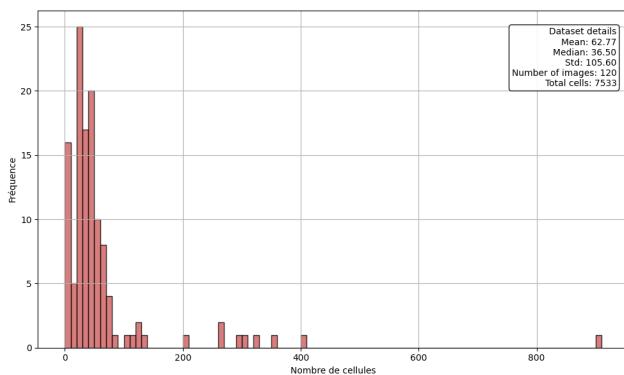


FIGURE 14 – Distribution du nombre de cellules par image (avant augmentation) de **train**

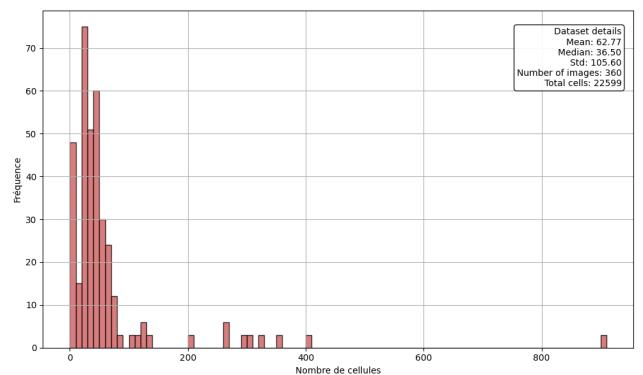


FIGURE 15 – Après augmentation Roboflow

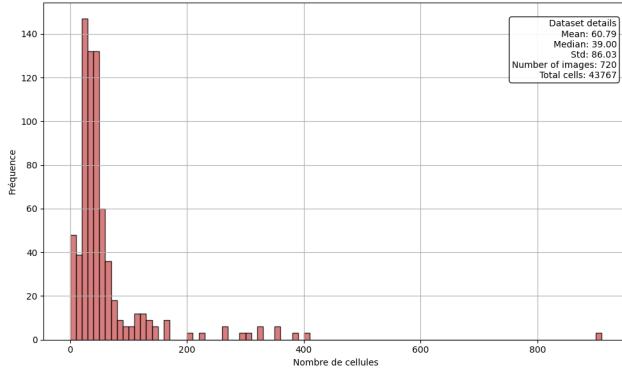


FIGURE 16 – Après copy\_paste et augmentation Roboflow

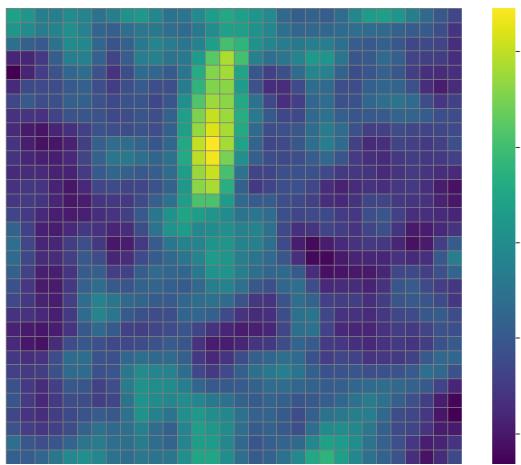


FIGURE 17 – Heatmap (avant augmentation)

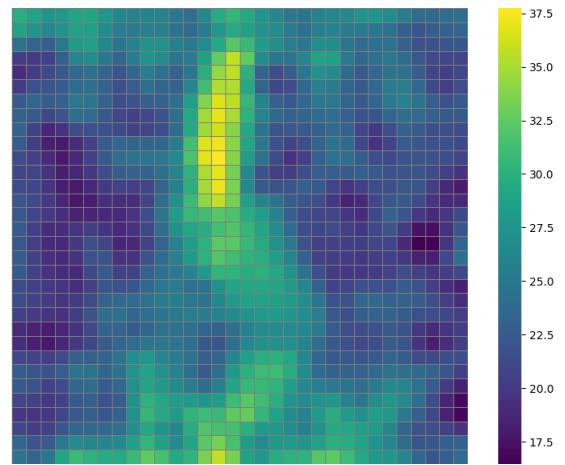


FIGURE 18 – Heatmap après augmentation et statistiques associées.

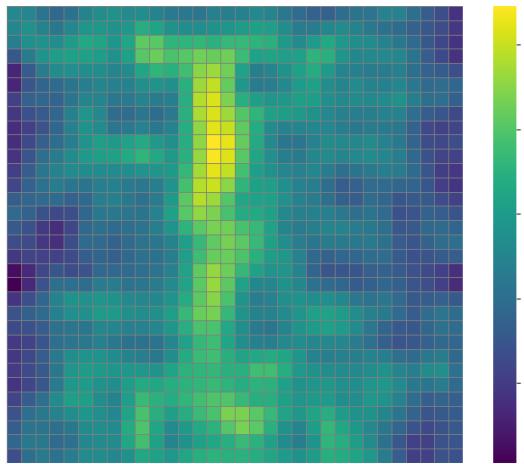


FIGURE 19 – Après copy\_paste et augmentation Roboflow

L'augmentation de données off-policy via Roboflow ne modifie pas l'histogramme. En effet aucune des opérations listées ci-dessus ne modifie le nombre de cellules par images. Nous pouvons noter que l'augmentation de données off-policy ne se fait que sur la partie du dataset dédiée à l'entraînement (donc histogramme total légèrement modifié mais pas celui de la partie **train**). En revanche on peut voir que les zones de fortes densité se plus reparties sur l'ensemble de la heatmap de la Figure 19. Cela apporte plus de variété dans le dataset. La colonne centrale reste la principale zone de forte densité, une piste d'amélioration serait de mettre en place des transformations pour mieux égaliser cette heatmap. La méthode

de copy\_past diversifie légèrement l'histogramme (voir Figure 16). De plus elle égalise la heatmap grâce au positionnement aléatoire des cellules dans les images créées. En plus des augmentations off-policy les différents algorithmes utilisés mettent en place une augmentation on-policy détaillées par la suite.

Nom du dataset	Taille	Méthode d'augmentation
dataset_original	170	Aucune
dataset_augm410	410	Roboflow (flip, saturation, blur)
dataset_copy_past_augm770	770	Copy-Paste + Roboflow

TABLE 1 – Résumé des datasets utilisés pour l'entraînement

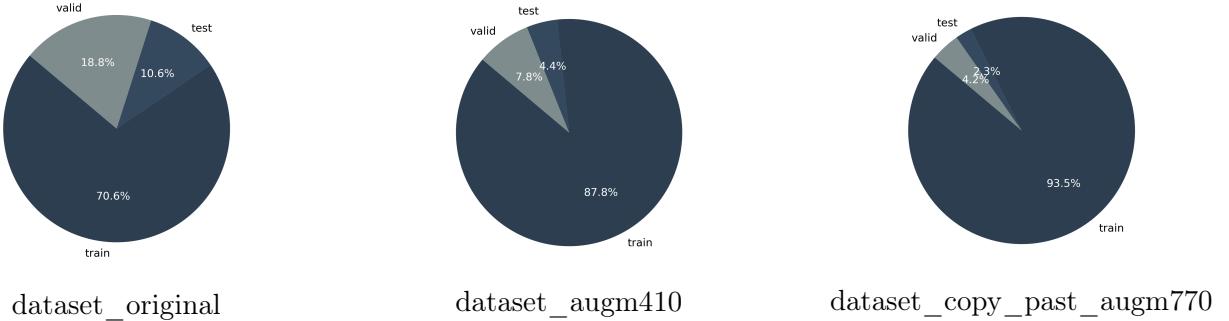


FIGURE 20 – Répartition des ensembles *train*, *test* et *valid* pour les trois datasets

L'ensemble de données comprenait 360 images pour l'entraînement, 32 pour la validation et 18 pour le test, soit une répartition de 88%, 8% et 4%. Au vue de la taille limitée du dataset nous privilégions la partie **train**.

### II.5.2 You Only Look Once (YOLO)

Le fonctionnement de YOLO peut être détaillé à travers plusieurs étapes clés.

**Division de l'image :** YOLO divise l'image en une grille. Chaque cellule de cette grille est responsable de la prédiction de plusieurs informations, comme les boîtes englobantes (bounding boxes) et la classe de l'objet détecté.

**Prédiction des bounding boxes :** Pour chaque cellule de la grille, YOLO prédit plusieurs boîtes englobantes (bounding boxes), contenant les éléments suivants :

- La position du centre de l'objet ( $x, y$ ) dans l'image.
- La largeur et la hauteur de la boîte ( $w, h$ ).
- Une mesure de confiance indiquant la probabilité qu'un objet soit présent dans cette boîte.

**Classification :** YOLO associe à chaque bounding box une classe d'objet. Dans le cadre de ce projet, on peut imaginer une classe unique : microalgue.

**Suppression des prédictions redondantes :** Une fois les objets détectés, YOLO applique une méthode appelée *Non-Maximum Suppression (NMS)* pour éliminer les prédictions redondantes ou celles qui se chevauchent plusieurs fois.

On entraîne deux architectures différentes : YOLO v5 et YOLO v11. Ces deux architectures existent en différentes tailles (*nano*, *small*, *medium*, *large* et *extra large*). Les tailles d'architecture utilisées sont détaillées dans la Table 2. Un résumé des principaux hyperparamètres d'entraînement est donné dans la Table 3. Le modèle a été entraîné pendant 100 epochs sur des images redimensionnées à une taille de  $640 \times 640$  pixels ( $\text{imgsz}=640$ ). La taille de lot (batch=-1) a été laissée en détection automatique, permettant

à l'algorithme d'ajuster dynamiquement le nombre d'images traitées simultanément afin d'utiliser  $\approx 60\%$  de la mémoire du GPU sans basculer sur le CPU. On peut voir lors de l'entraînement que pour YOLOv5su le batchsize était de 11 pour une utilisation à 64% du GPU et de 8 pour YOLO11.

L'optimiseur utilisé est SGD (Stochastic Gradient Descent), avec un momentum fixé à 0.937 pour stabiliser et accélérer la convergence. Le taux d'apprentissage initial était de 0.01 (lr0), et a diminué progressivement jusqu'à une valeur finale de 0.0001 (lrf), selon la stratégie OneCycleLR, qui permet une exploration efficace de l'espace des paramètres en début d'apprentissage suivie d'une stabilisation en fin de phase. L'augmentation de données on-policy Mosaic était activée (mosaic=1.0), ce qui permet de combiner plusieurs images en une seule pour renforcer la variabilité spatiale des objets d'intérêt et améliorer la robustesse du modèle face à des scènes complexes ou très peuplées.

Les modèles utilisés lors de la première phase de travail sont pré-entraînés sur COCO (Common Objects in Context) ce qui entraîne YOLO à reconnaître des formes, contour et identifier de nombreux objets variés. Ce pré-entraînement est nécessaire mais peu optimisé ici car il pourrait être réalisé sur un jeu de données plus proche de nos données (images de microscope, de cellule...).

Nom du modèle	Nombre de couches	Paramètres	GFLOPs
YOLOv5su	153	9.15 M	24.2
YOLOv5mu	197	25.1 M	64.6
YOLOv5lu	128	53.1	137.7
YOLO11s	181	9.5 M	21.7
YOLO11m	231	20.1 M	68.5

TABLE 2 – Comparaison des architectures YOLO utilisées

TABLE 3 – Résumé des principaux hyperparamètres d’entraînement

Paramètre	Valeur	Description
<code>epochs</code>	100	Nombre total de passages sur l’ensemble d’entraînement.
<code>imgsz</code>	640	Taille des images redimensionnées pour l’entrée du réseau.
<code>batch</code>	-1	Taille de batch automatique, déterminée selon la mémoire disponible sur le GPU.
<code>device</code>	0	Entraînement réalisé sur GPU (indice 0).
<code>lr0</code>	0.01	Taux d’apprentissage initial, contrôlant la vitesse d’actualisation des poids.
<code>lrf</code>	0.01	Taux d’apprentissage final. Ici, pas de décroissance.
<code>momentum</code>	0.937	Inertie appliquée au gradient pour stabiliser l’optimisation (SGD).
<code>weight_decay</code>	0.0005	Régularisation L2 pour éviter le surapprentissage.
<code>optimizer</code>	auto	L’optimiseur est automatiquement sélectionné selon les paramètres du modèle.
<code>mosaic</code>	1.0	Augmentation combinant aléatoirement 4 images (on-policy).
<code>fliplr</code>	0.5	Retournement horizontal aléatoire avec une probabilité de 50 %.
<code>hsv_h, s, v</code>	0.015 / 0.7 / 0.4	Perturbations de teinte, saturation et valeur de l’image.
<code>scale</code>	0.5	Redimensionnement aléatoire jusqu’à $\pm 50\%$ .
<code>translate</code>	0.1	Déplacement horizontal ou vertical jusqu’à 10 %.
<code>erasing</code>	0.4	Masquage aléatoire d’une région de l’image.
<code>auto_augment</code>	randaugment	Politique d’augmentation automatique combinant plusieurs transformations.

### III Résultats & Discussion

#### III.1 Métriques utilisées

##### Erreur relative

Afin d’évaluer la pertinence des segmentations obtenues par les différentes méthodes, une métrique simple a été définie afin d’estimer la qualité du comptage cellulaire associé à chaque approche. Pour cela, un comptage manuel du nombre de cellules a été effectué sur un sous-ensemble d’images, constituant ainsi la vérité terrain (ground truth).

La métrique d’erreur utilisée dans ce projet est donnée par la relation suivante :

$$e_{rr} = \frac{n_{\text{algo}} - n_{\text{gt}}}{n_{\text{gt}}} \quad (1)$$

où  $n_{\text{algo}}$  représente le nombre de cellules détectées automatiquement par la méthode de segmentation considérée, et  $n_{\text{gt}}$  correspond au nombre de cellules issues de la vérité terrain.

Par construction, cette erreur est positive en cas de sur-segmentation (plus de cellules détectées que réellement présentes) et négative en cas de sous-segmentation. Cette métrique présente l’avantage d’être simple à calculer et est directement interprétable, notamment pour des applications centrées sur le comptage cellulaire. Cependant, elle comporte certaines limites. Tout d’abord, elle ne fournit aucune information

spatiale sur la qualité de la segmentation, et ne permet donc pas de distinguer si l'écart observé résulte de la présence de faux positifs ou, au contraire, d'une mauvaise détection de véritables cellules (faux négatifs ou vrais positifs mal localisés). C'est pourquoi une évaluation qualitative a également été réalisée par contrôle visuel, permettant d'identifier précisément la nature des erreurs lors de la phase d'optimisation des différentes méthodes de segmentation. Enfin, cette métrique est conçue uniquement pour évaluer la précision du comptage : elle ne permet pas d'évaluer la forme, la taille ou la surface des objets détectés. En particulier, elle ne peut pas quantifier les erreurs de sur ou sous-segmentation de l'aire des cellules, ce qui la rend inadaptée à des applications nécessitant une caractérisation morphologique des objets.

## Circularité

La circularité d'un objet est défini de la manière suivante :

$$\mathcal{C} = \frac{4\pi \times \text{Aire}}{(\text{Périmètre})^2} \quad (2)$$

Une valeur de  $\mathcal{C}$  proche de 1 indique une forme proche d'un cercle parfait, tandis qu'une valeur plus faible correspond à une forme irrégulière, allongée ou fragmentée.

## Intersection over Union (IoU)

L'Intersection over Union (IoU) [5] est une métrique courante en détection d'objets. Elle mesure le recouvrement entre la boîte prédite par le modèle et la vérité terrain (ground truth), et est définie comme :

$$\text{IoU}(A, B) = \frac{A \cap B}{A \cup B} \quad (3)$$

où  $A$  et  $B$  représentent respectivement les aires de la boîte englobante prédite et de la vérité terrain. Une détection est considérée comme correcte si l'IoU dépasse un certain seuil (par exemple, 0.5).

Les termes suivants sont utilisés :

- TP (True Positives) : objets correctement détectés.
- FP (False Positives) : objets détectés à tort.
- FN (False Negatives) : objets manqués par le modèle.

À partir de cela, on définit la **précision** et le **rappel** comme suit :

$$\text{Précision} = \frac{TP}{TP + FP}, \quad \text{Rappel} = \frac{TP}{TP + FN} \quad (4)$$

La précision moyenne (AP) correspond à l'aire sous la courbe précision-rappel. Enfin, la précision moyenne moyenne (mAP) est obtenue en faisant la moyenne des AP calculées pour chaque classe :

$$\text{mAP} = \frac{1}{N} \sum_{i=1}^N \text{AP}_i \quad (5)$$

où  $N$  est le nombre total de classes ( $N = 1$  ici).

## III.2 Méthode classique

### III.2.1 Remarque générale

Au-delà des performances numériques, il est important de souligner que la difficulté principale du sujet ne réside pas dans l'optimisation locale de la segmentation d'une unique image, ce qui peut souvent être réalisé manuellement ou par ajustement fin des paramètres. Le véritable enjeu réside dans la capacité à automatiser la segmentation de manière robuste et reproductible sur un grand nombre d'images, acquises dans des conditions variées : lumière, contraste, netteté, bruit, saturation...

### III.2.2 Segmentation par composantes connexes

La segmentation par composantes connexes constitue une première approche dans la prise en main des fonctions de base du traitement de l'image sous *Python*. Sa simplicité d'utilisation et sa rapidité d'exécution se payent par son manque de robustesse lorsque des cellules se touchent. Dans ce cas-ci, une telle approche conduira nécessairement à une sous-segmentation de l'image et entraînera donc une erreur importante, comme on peut le voir sur la Figure 21.

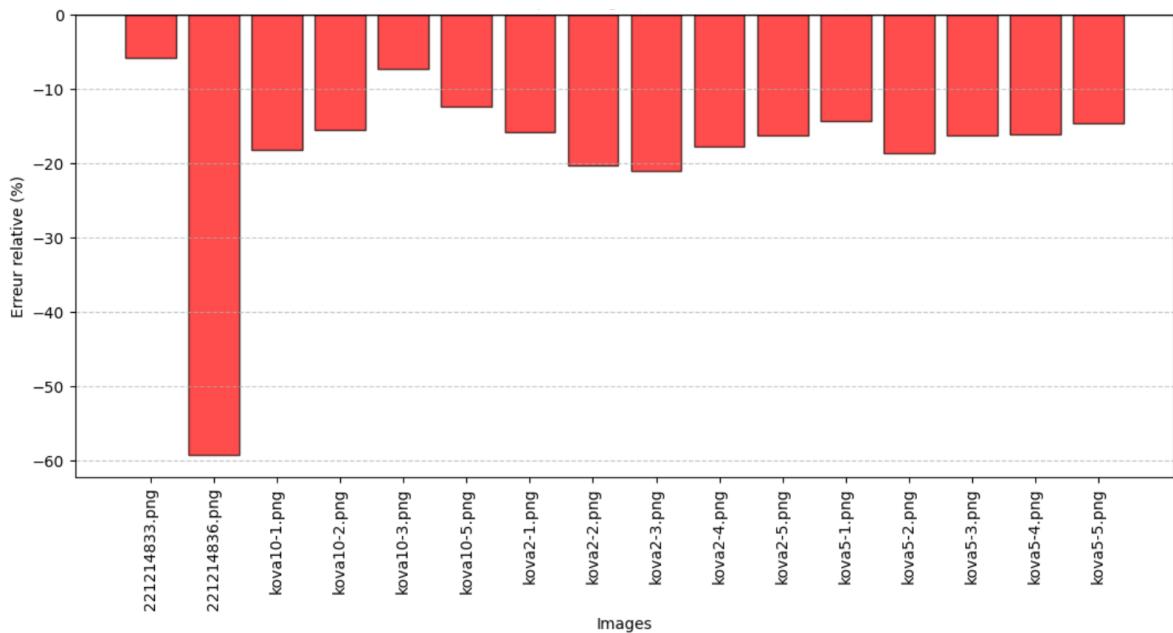


FIGURE 21 – Erreur relative de la méthode par composantes connexes pour des images de plus de 100 cellules

Sur les images de plus d'une centaine de cellules, l'erreur absolue moyenne est de 19%, ce qui témoigne de la limitation de cette méthode dans le cadre d'échantillons à forte densité cellulaire. Cette méthode peut être cependant pertinente dans le cas où l'image présente moins d'une dizaine de cellules et où la possibilité de contact entre cellules est minimale.

Ces résultats mettent en évidence les limites de la segmentation par composantes connexes lorsqu'elle est appliquée à des images à forte densité cellulaire. Afin de surmonter ces difficultés et de mieux séparer les cellules agglomérées, des techniques de segmentation plus élaborées ont été explorées dans la suite de ce travail.

### III.2.3 Segmentation par érosion

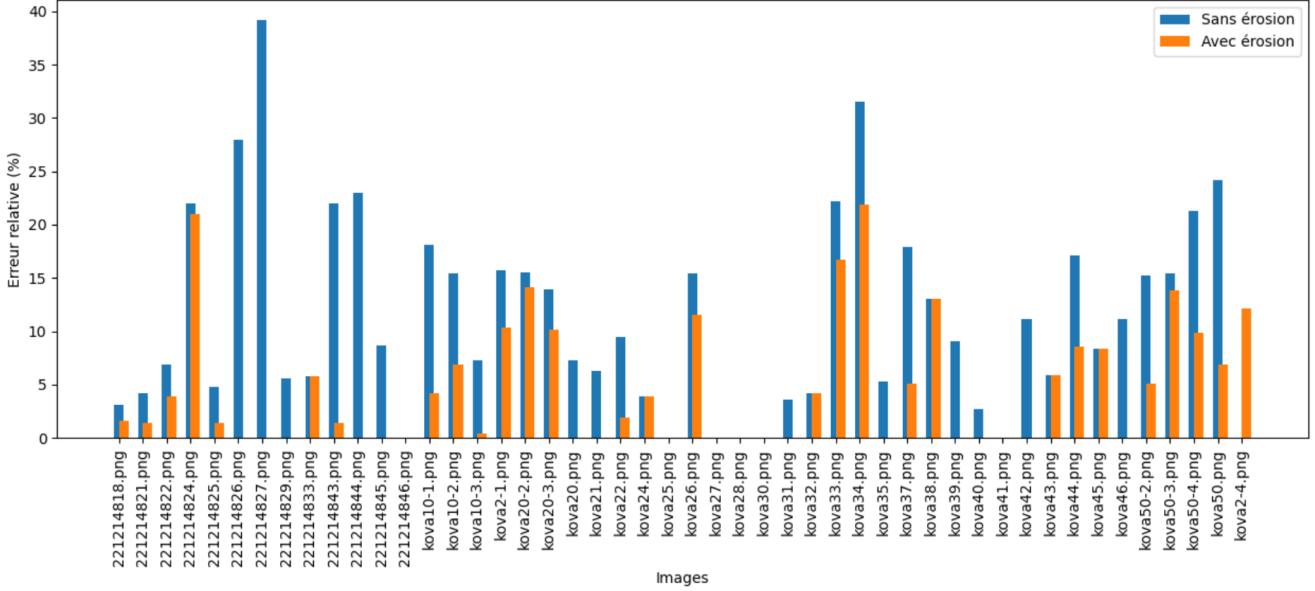


FIGURE 22 – Comparaison de l’erreur relative (en valeur absolue) entre la méthode par composantes connexes et celle par érosion pour un groupe d’images

Cette approche constitue une première tentative pour appliquer un contrôle dans la segmentation réalisée. L’idée naïve est de forcer la séparation des cellules pour éviter le phénomène de sous-segmentation qui occure dans la méthode précédente. À cette fin, nous avons utilisé la fonction *erode* avec les arguments suivants :

- l’image source (en niveaux de gris ou binaire)
- le kernel (noyau), matrice (de type np.uint8) utilisée pour l’opération d’érosion, de forme carré, rectangulaire, en croix, en disque, elliptique, etc.
- iterations : (optionnel, par défaut = 1). Nombre de fois que l’érosion est appliquée. Utile pour renforcer l’effet de l’érosion.

Par la suite, une optimisation des arguments précédemment évoquées a été implémentée pour chaque image dans le but de minimiser la métrique 1. Nous avons obtenu les résultats de la Figure 22. La méthode par érosion permet a priori de réduire l’erreur commise lors du comptage. Nous passons en effet d’une erreur moyenne sans érosion de 11.26 % à une erreur moyenne avec érosion de 4.82 %. Cependant, bien que satisfaisante du point de vu de la métrique employée, cela ne l’est pas lorsque qu’on effectue un contrôle visuel des images segmentée (voir Figure 23).

L’approche a permis, dans le cas (1), de séparer correctement les cellules. Cependant, de nombreuses cellules (2) sont restées attachées malgré l’optimisation, et d’autres cellules (3) ont complètement disparu. L’optimisation a été effectuée pour chaque image, mais les résultats se sont révélés mitigés. Cette méthode soulève ainsi une problématique d’applicabilité sur un ensemble d’images, et montre une dépendance excessive à la taille des cellules. En effet, pour un certain choix de paramètres lié à leurs optimisations sur un jeu d’image donné, si les cellules d’une nouvelle image sont trop grandes, l’érosion ne permettra pas de les séparer correctement. Inversement, si les cellules sont trop petites, une proportion significative de celles-ci sera éliminée.

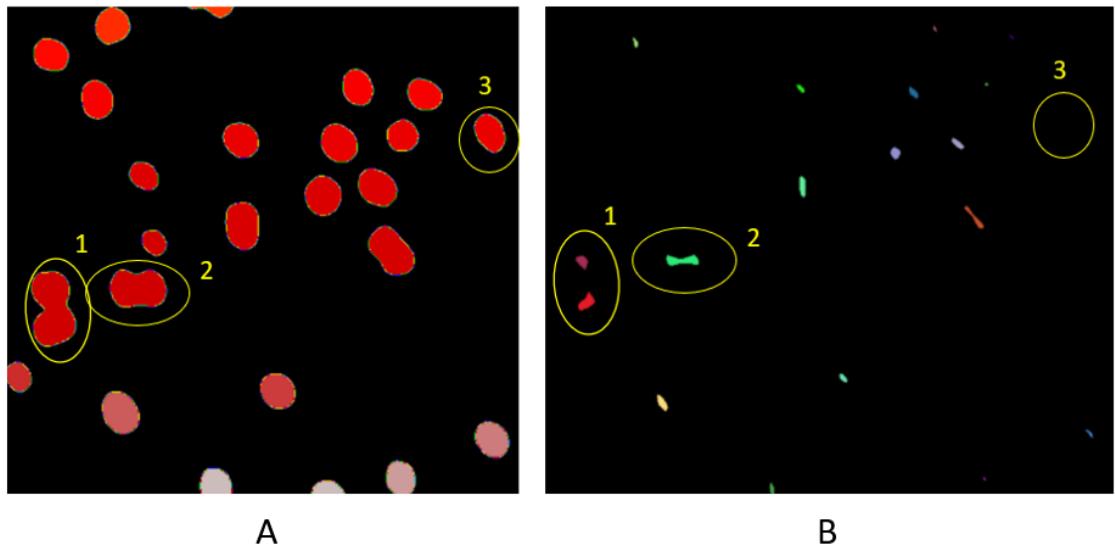


FIGURE 23 – Visualisation de la segmentation par érosion. A : après seuillage. B : après érosion.

### III.2.4 Segmentation par détection de contours

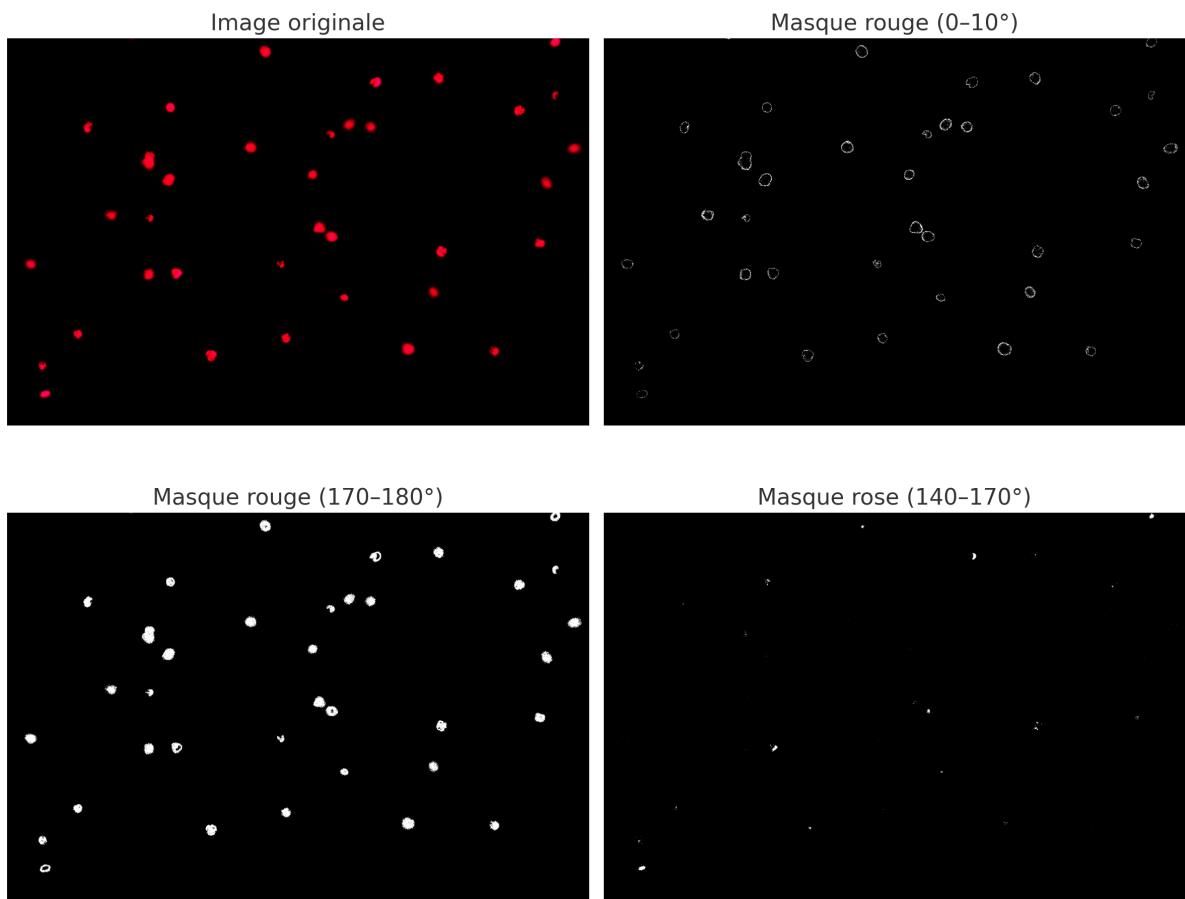


FIGURE 24 – Exemple de masques

Ici la majorité des cellules fluo se trouvent dans la plage [170, 180], comme indiqué sur la Figure 24. Le choix du masques s'avèrent très important et il est nécessaire de bien l'adapter aux données. Sur certaines images, comme celle analysée ici, la segmentation en HSV peut être trop stricte. Lorsque les seuils de teinte sont trop sélectifs, une même cellule peut être partiellement détectée en plusieurs morceaux distincts. Chaque fragment étant ensuite entouré et compté individuellement, cela entraîne une surestimation du nombre de

cellules. Ce type de comportement peut-être visualisé en Figure 25.

Dans l'exemple de la Figure 25, 83 contours ont été détectés alors que le ground truth indique 62 cellules. Cette erreur vient d'une sur-segmentation causée par une séparation trop fine des teintes internes à une même cellule (par exemple des zones rouge foncées et rouge claires séparées par des gradients).

On pourra remarquer que la teinte rouge est centrée autour de  $0^\circ$  dans l'espace HSV classique. Cependant, comme la composante Hue est cyclique, les rouges apparaissent également en fin de spectre (proche de  $360^\circ$ ). Dans OpenCV, où la composante Hue est codée de 0 à 179, cela correspond à deux plages distinctes : environ [0, 10] et [170, 180].

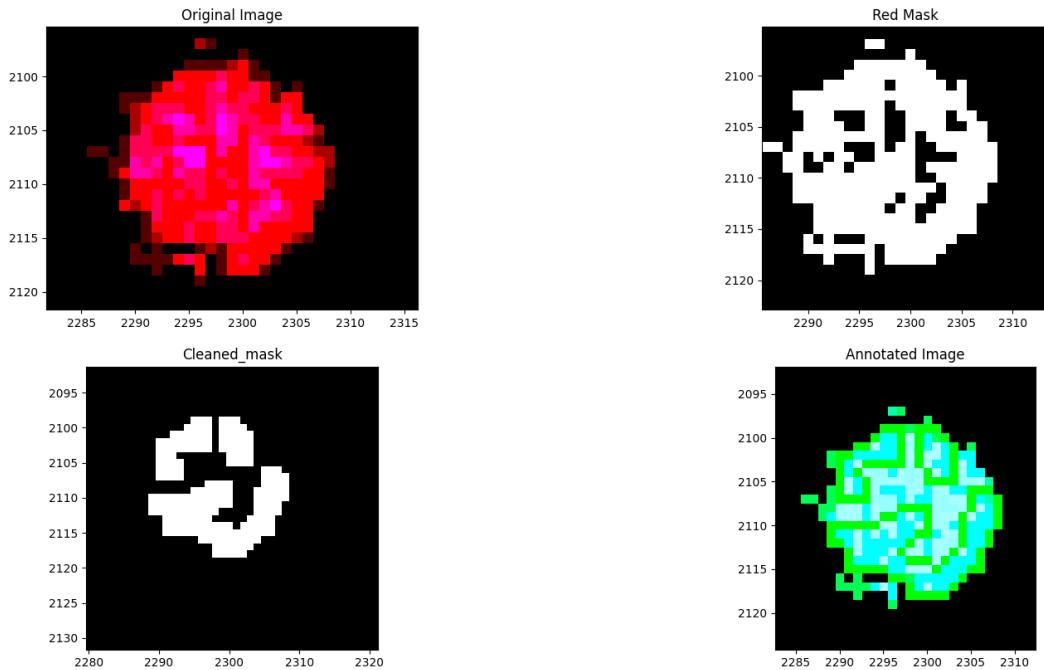


FIGURE 25 – Exemple de mauvaise red mask

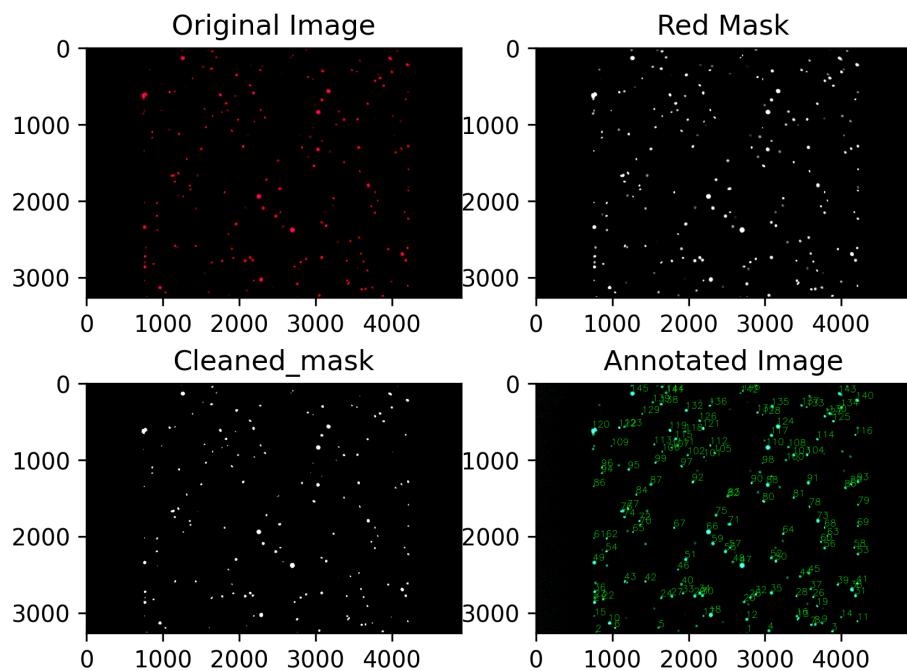


FIGURE 26 – Exemple de résultat sur une image

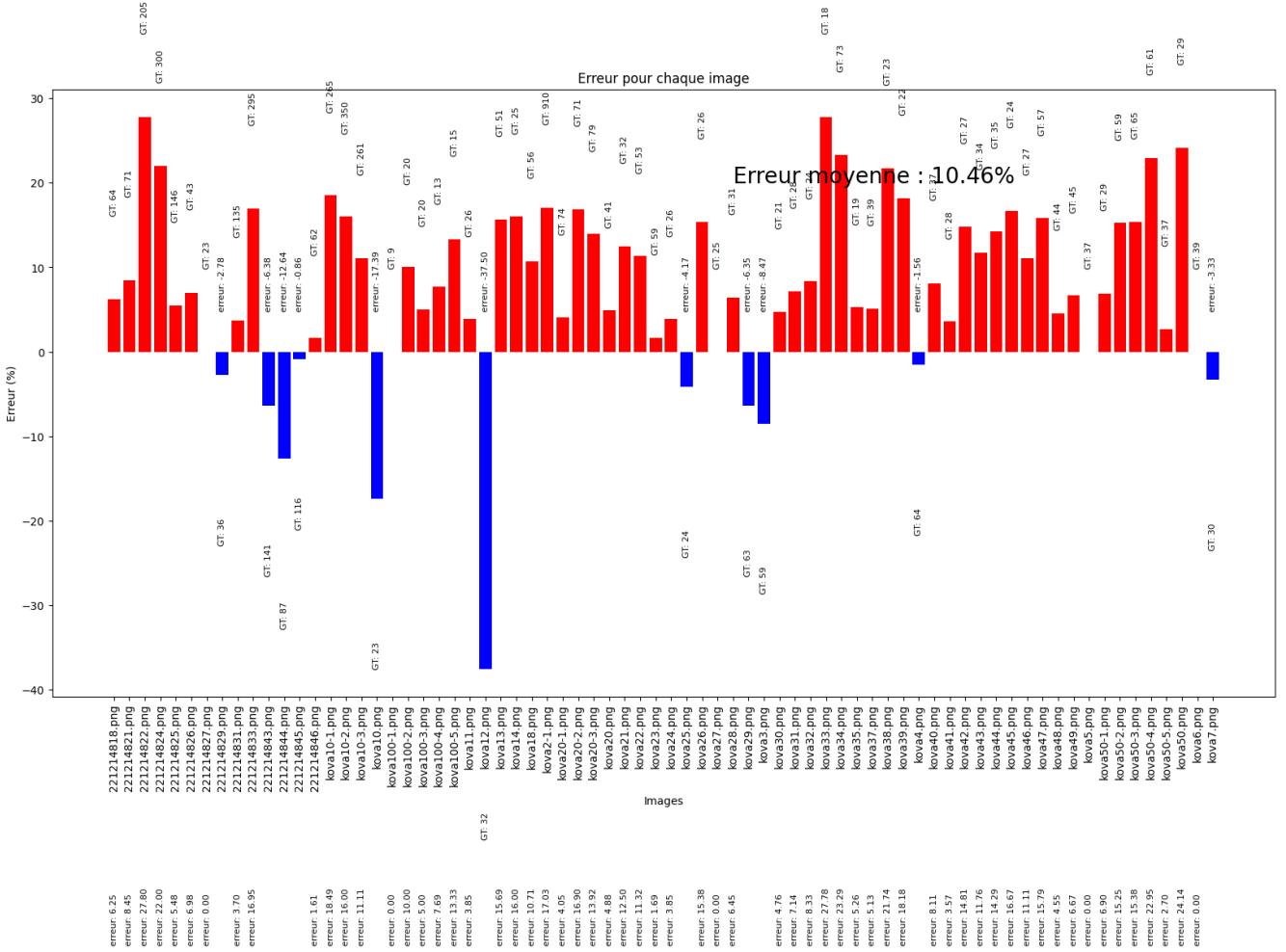


FIGURE 27 – Résultats de l’algorithme

### III.2.5 Segmentation par *Watershed* avec contrôle des "seeds"

L’un des principaux inconvénients de la méthode du *Watershed* est le risque de sur-segmentation qu’elle peut entraîner. Celui-ci est illustré sur la Figure 28. Pour limiter le phénomène de sur-segmentation, il est nécessaire d’adapter la détection des maxima locaux après l’application de la transformée de distance. Une solution consiste à ajuster la taille du noyau utilisé dans la fonction *peak\_local\_max*, laquelle opère directement sur la carte des distances. Cette fonction permet d’identifier les points les plus éloignés des contours, qui serviront ensuite de germes (ou seeds) pour l’algorithme de *Watershed*. En modulant la taille du noyau (paramètre *footprint*), on contrôle la distance minimale entre deux maxima locaux détectés. Un noyau plus grand favorise une fusion de régions proches, ce qui peut réduire la sur-segmentation. À l’inverse, un noyau plus petit permet de détecter plus de maxima, au risque de sur-segmenter la structure. La taille des cellules étant généralement comprise entre 10 et 50 pixels (pour la plupart autour de 30 pixels), nous choisissons une taille du noyau de 11 pixels × 11 pixels.

Cependant, le réglage de la taille du noyau ne permet pas à lui seul d’éliminer entièrement la sur-segmentation. Pour affiner ce processus, nous avons introduit une étape de discrimination supplémentaire : si une région segmentée ne satisfait ni le critère d’aire minimale, ni celui de circularité minimale 2, elle est considérée comme une segmentation erronée et est supprimée. Une fois l’ensemble des régions segmentées évaluées selon ces critères morphologiques, nous relançons une dernière application de l’algorithme de *Watershed*, en utilisant uniquement les seeds validées. Cette démarche permet de renforcer la robustesse de la segmentation finale, en réduisant le nombre de faux positifs tout en préservant les structures cellulaires conformes aux caractéristiques attendues.

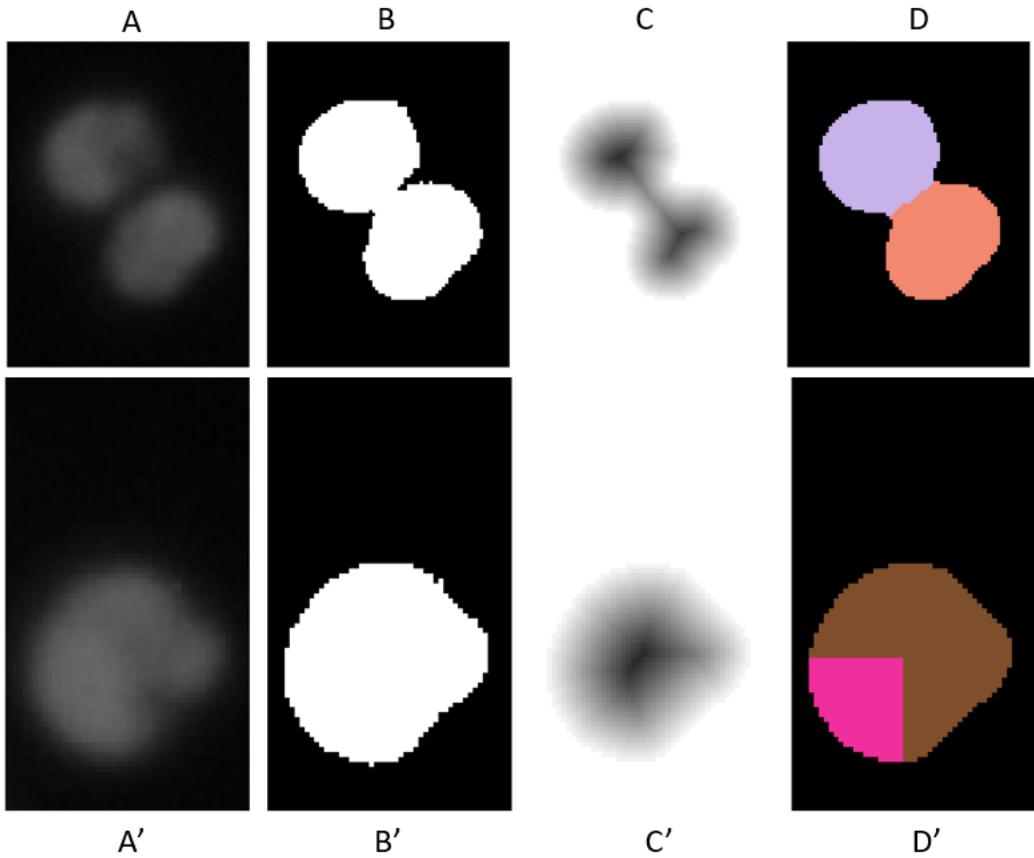


FIGURE 28 – Exemple de bonne segmentation et de sur-segmentation avec l’algorithme du *Watershed* ; Image originelle : A, A' ; Seuillage : B, B' ; Carte des distances : C et C' ; Segmentation : D et D'

**Etude sur les images comportant plus de 100 cellules** Afin d’optimiser le choix des paramètres de détection, nous avons mené une analyse de l’erreur relative en fonction des seuils d’aire minimale et de circularité, spécifiquement pour les images à haute densité cellulaire (voir Figure 30).

Nous obtenons une aire minimale de 60 pixels  $\times$  pixels et une circularité minimale de 0.7 à appliquer pour obtenir l’erreur relative moyenne la plus faible sur l’ensemble des images de plus de 100 cellules. Ces seuils sont cohérents de la part les caractéristiques des cellules. En effet, les plus petites cellules peuvent avoir un diamètre de 10 pixels, donnant une aire de 78.5 pixels  $\times$  pixels. Les cellules ne sont pas par ailleurs quasiment circulaire et peuvent davantage présenter un profil elliptique (voir Figure 29).

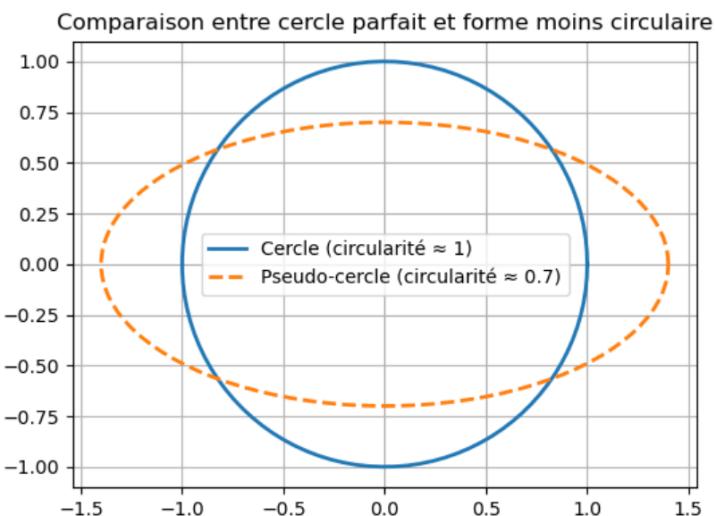


FIGURE 29 – Comparaison visuelle entre un cercle parfait et un pseudo-cercle de circularité 0.7

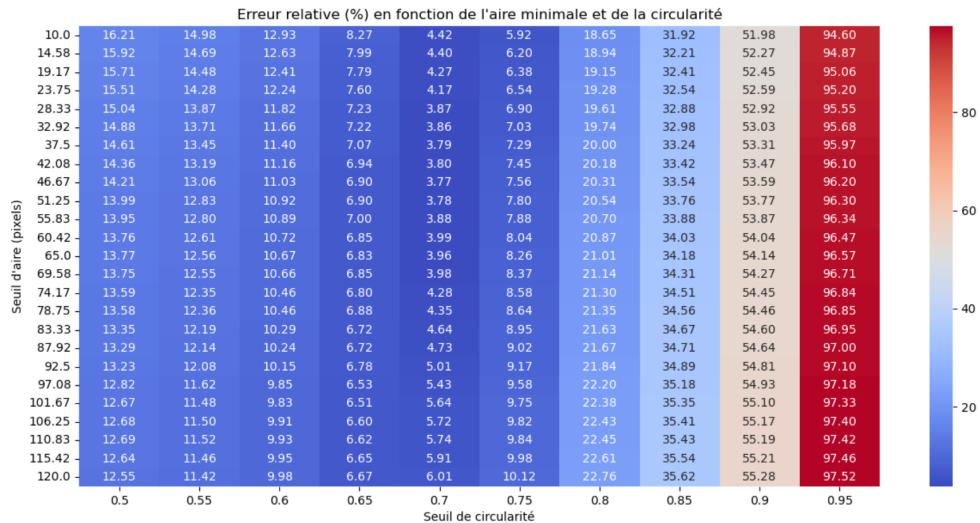


FIGURE 30 – Évolution de l'erreur relative moyenne (en valeur absolue) pour l'ensemble des images de plus de 100 cellules en fonction de l'aire minimale ainsi que de la circularité minimale. L'échelle de couleur à droite renvoie à l'erreur relative commise après comptage

## Étude sur les images comportant moins de 100 cellules

L'étude de l'évolution de l'erreur relative en fonction de l'aire et de la circularité minimale pour les images à faible densité nous donne le résultat de la Figure 31. Les seuils minimisant l'erreur relative moyenne sur les cellules à faible densité sont similaires aux images à haute densité. Cette observation est cohérente car les caractéristiques des cellules sont les mêmes entre les images à faible et haute densité. Seul le nombre des cellules présentes se retrouve modifié. Nous retiendrons donc comme valeurs seuils :

$$A_s = 60 \quad \text{et} \quad C_s = 0.7$$

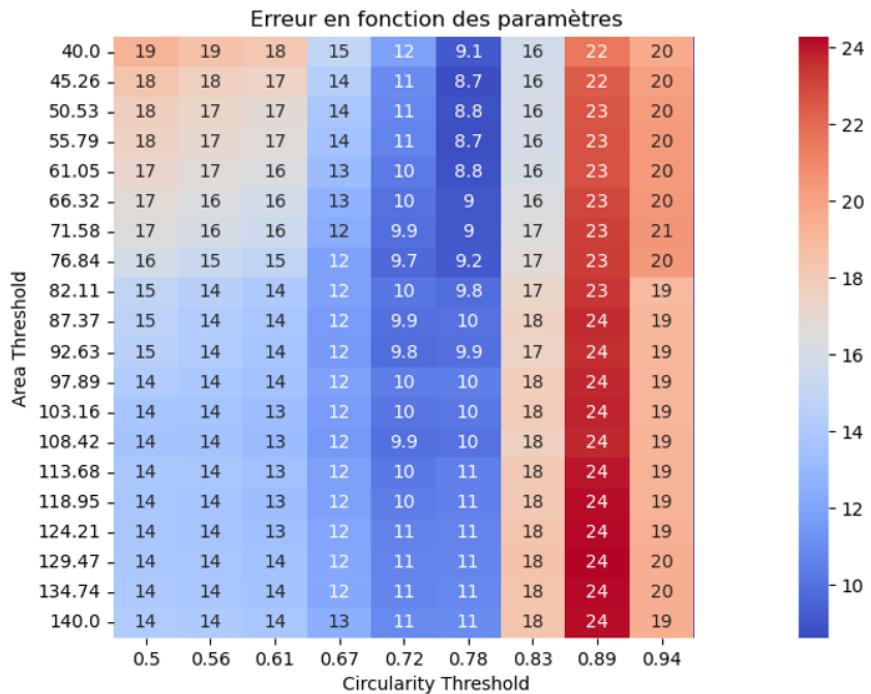


FIGURE 31 – Evolution de l'erreur relative moyenne (en valeur absolue) pour l'ensemble des images de moins de 100 cellules en fonction de l'aire minimale ainsi que de la circularité minimale. L'échelle de couleur à droite renvoie à l'erreur relative commise après comptage

## Analyse des résultats

Pour les images de faible densité, nous obtenons une erreur moyenne de 12.85 % avec un écart type de 22.33%. Pour les images de haute densité, nous obtenons une erreur moyenne de 3.34 % avec un écart type de 2.32 %. Des erreurs plus importantes peuvent être observées sur les images à faible densité. Il convient toutefois de souligner que ce phénomène est étroitement lié à l'utilisation de la métrique relative (1), dont l'ordre de grandeur dépend du nombre total de cellules présentes sur l'image. Nous remarquons par ailleurs que dans les deux cas, l'algorithme a tendance à davantage sur-segmenter (voir Figure 32, 33). Ceci est dû aux valeurs des seuils ; il suffirait d'augmenter leur valeur pour diminuer cette tendance.

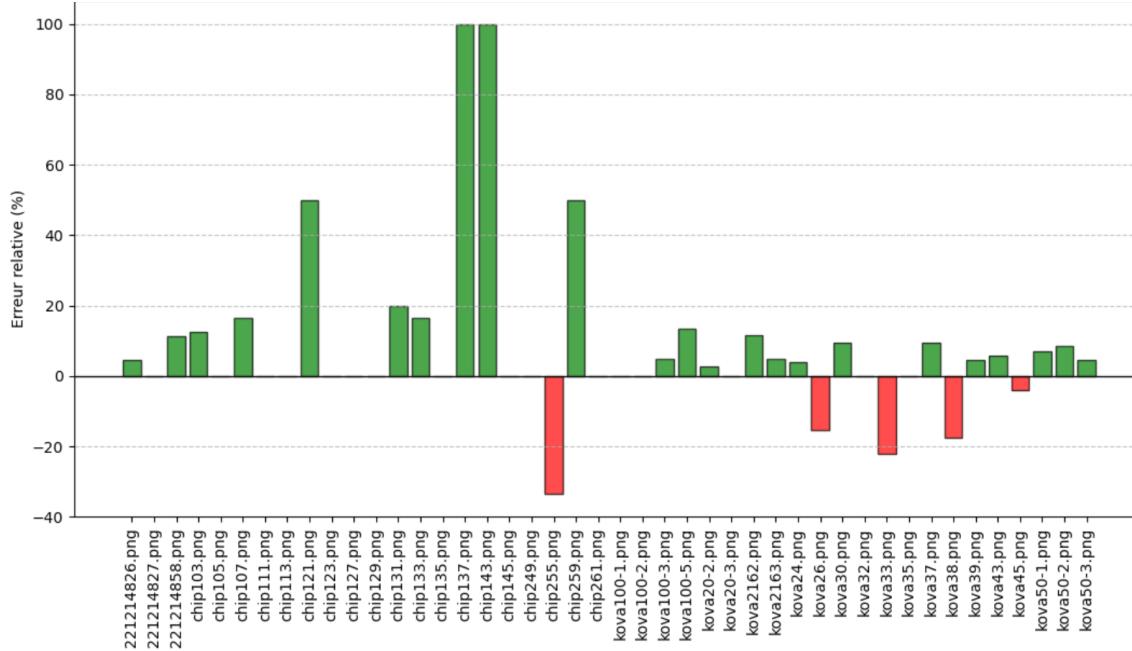


FIGURE 32 – Erreur relative pour les images de moins de 100 cellules

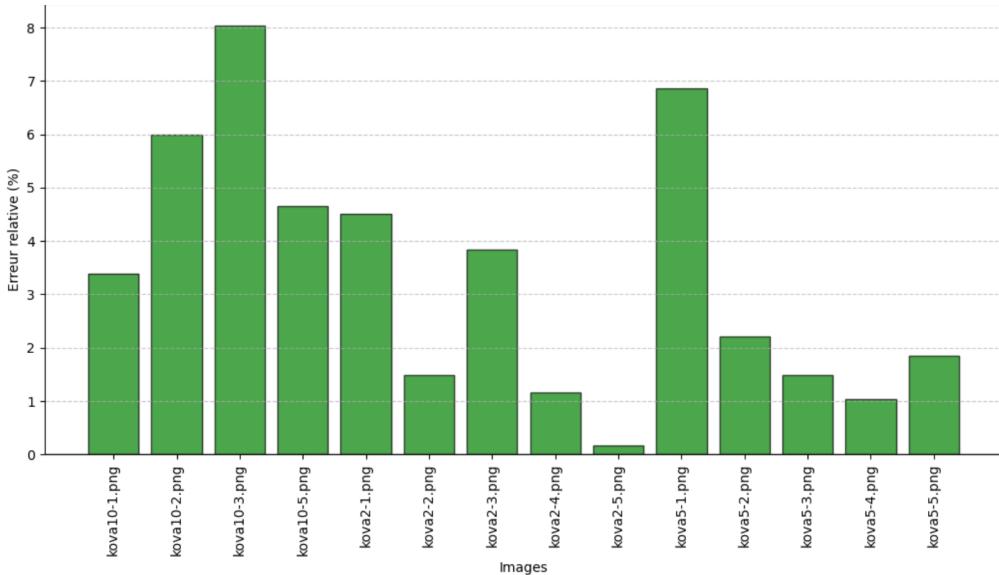


FIGURE 33 – Erreur relative pour les images de plus de 100 cellules

Nous pouvons observer la segmentation réalisée sur une image type (voir Figure 34).

- (1) : bonne segmentation réalisée, les quatres cellules ont été correctement identifiées et séparées
- (2) : filtrage opéré, l'algorithme compte bien deux cellules ; sans, trois cellules au lieu de deux auraient été détectées

- (3) sur-segmentation malgré l'étape de seuillage : circularité suffisante
- (4) forme de cellule similaire à (3) mais filtrage opéré

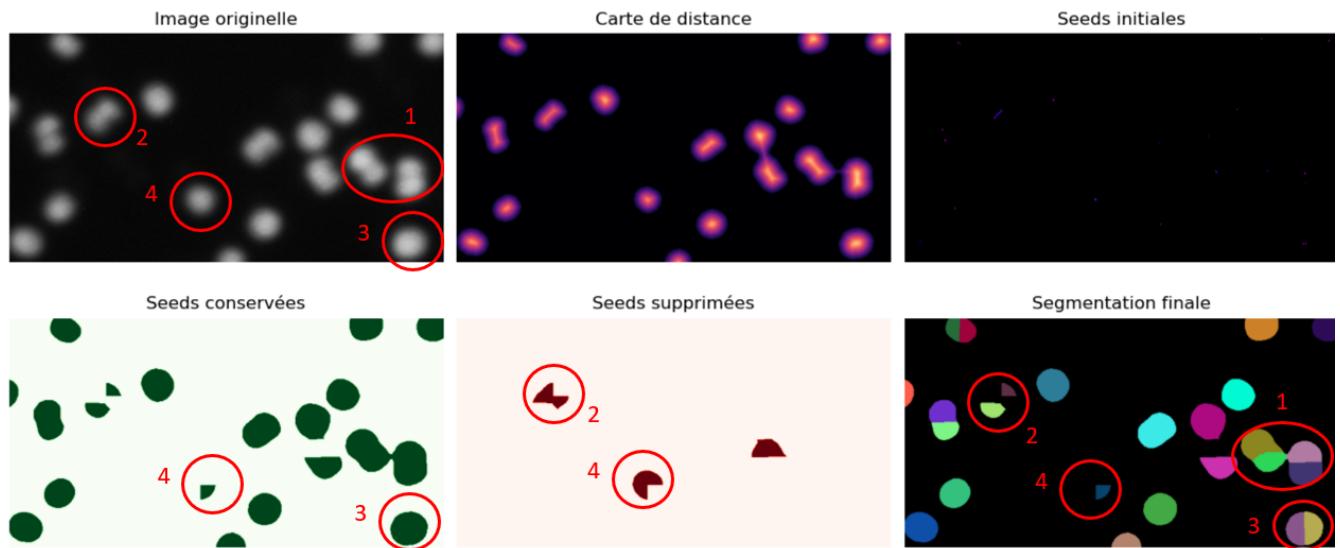


FIGURE 34 – Exemple de segmentation réalisée par la méthode

Nous comprenons donc la persistance de la sur-segmentation, certaines seeds conduisent à des subdivisions d'une unique cellule en deux. Les deux zones ont une circularité suffisante pour passer l'étape de filtrage.

Ce phénomène se retrouve sur les images à faible densité et induit les erreurs relatives importantes constatées par exemple sur la Figure 35.

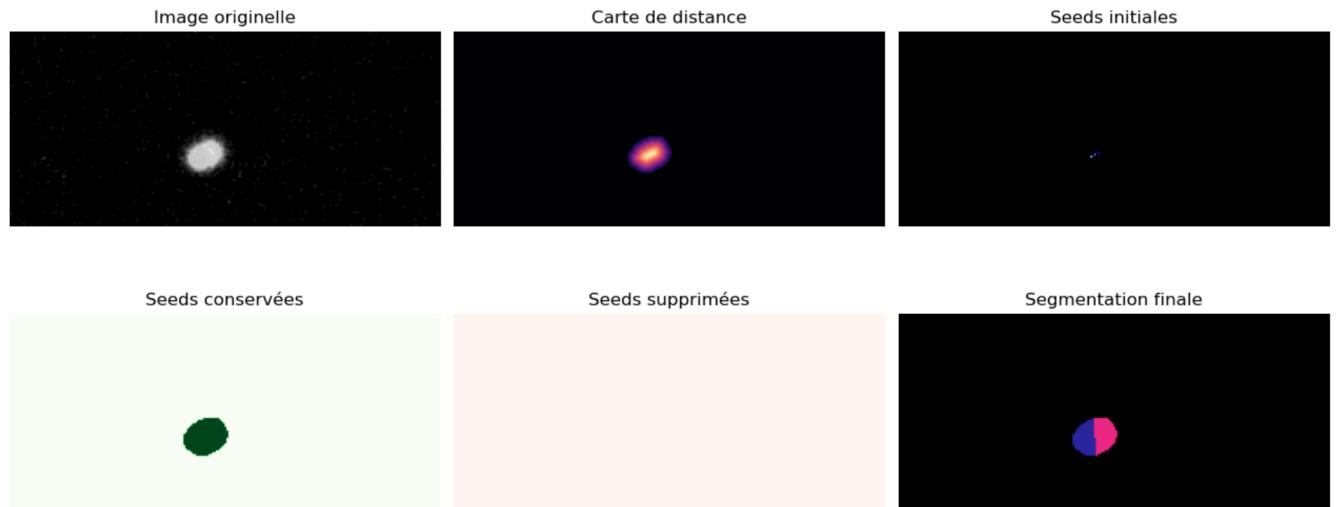


FIGURE 35 – Exemple de sur-segmentation sur l'image chip137.png ne contenant qu'une unique cellule

Toutefois, une augmentation des seuils pour supprimer ce type d'artefacts conduit également à l'élimination de cellules correctement segmentées, ce qui altère la qualité globale du comptage sur l'image.

### III.2.6 Segmentation via le logiciel *CellProfiler*

L’arborescence des fonctions choisies (encadrées en orange sur la figure 11) pour réaliser la segmentation ainsi que le comptage du nombre de cellules a été choisie via l’étude [15]. Deux types de paramètres (encadrés en bleu sur la figure 11) entrent en jeu lors de la segmentation :

- Diamètre minimal : en dessous de cette valeur, un objet détecté est considéré comme du bruit et n'est pas pris en compte dans le comptage.
- Borne inférieure de seuillage : valeur définie entre 0 et 1 (sur l'intensité normalisée de l'image), en dessous de laquelle les pixels sont considérés comme appartenant au fond.

Ces deux paramètres permettent de lutter contre les phénomènes de sur-segmentation inhérents à la méthode de segmentation choisie. Au regard du temps de calcul du logiciel sur un grand jeu d'image, ces paramètres ont été optimisées sur les images présentant la plus forte densité de cellules (images avec plus de 100 cellules). Par ailleurs, il paraît cohérent qu'une optimisation de la segmentation sur des images à haute densité devrait amener à une bonne segmentation sur des images de faible densité si nous parvenons à contrôler le phénomène de sur-segmentation.

#### Réglage de la borne inférieure de seuillage

Dans le cas où par exemple, nous ne souhaitons pas discriminer d'objets selon la borne inférieure de seuillage, on obtient le résultat de la Figure 36.

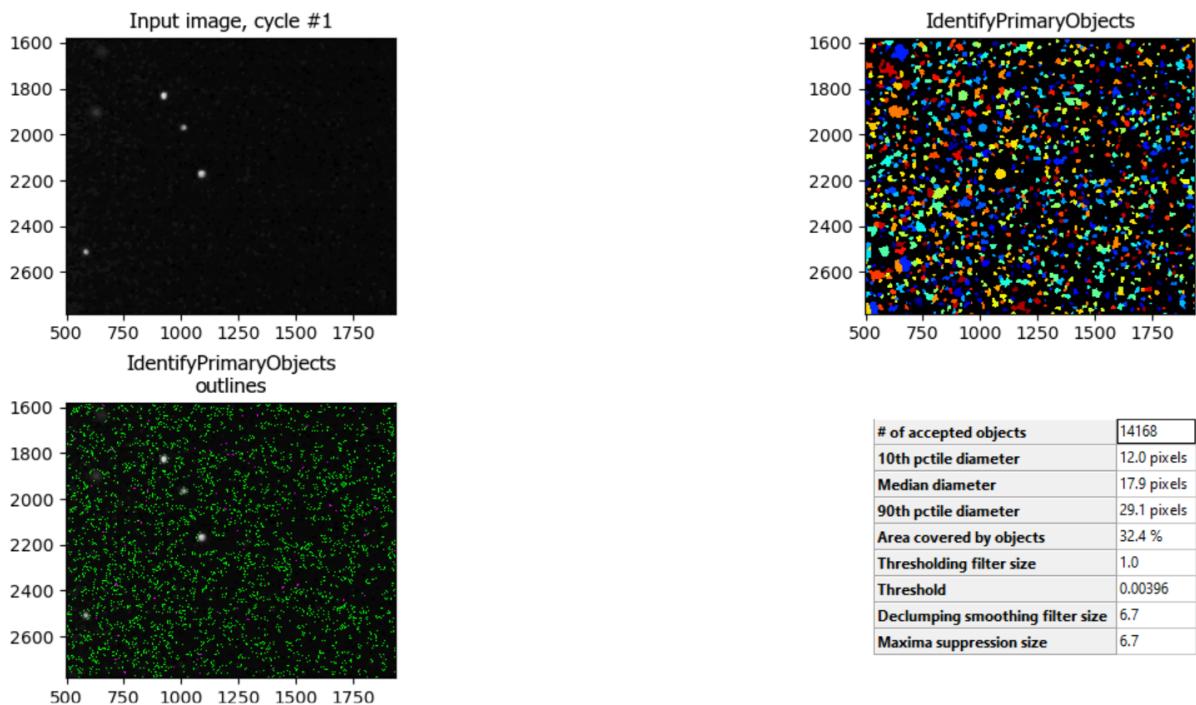


FIGURE 36 – Exemple de sur-segmentation pour un seuil d'intensité minimal trop faible (ici nul) - Image retournée par le logiciel

Malgré les étapes de pré-traitement (filtre médian etc.), la méthode de segmentation conduit à détecter de nombreux objets parasites, d'où la nécessite de correctement régler cette valeur seuil de l'intensité.

Le réglage de ce paramètre se retrouve être par ailleurs complexifié par la différence d'intensité que peuvent présenter certaines cellules. Tandis que certaines auront une forte intensité (par exemple 0.3) d'autres en présenteront une faible (0.15) :

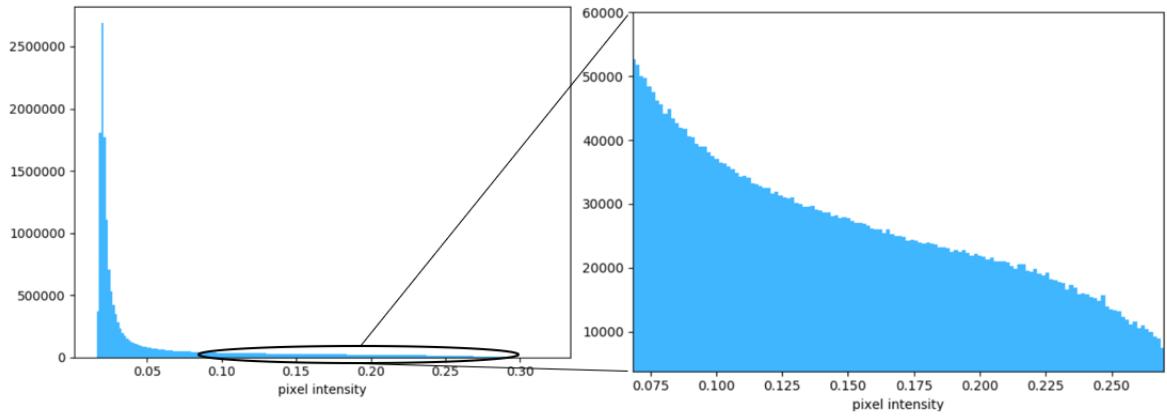


FIGURE 37 – Exemple d’histogramme pour la distribution d’intensité des pixels

Si la valeur du seuil minimal est fixée trop élevée dans le but de rejeter efficacement le bruit (certains résidus pouvant présenter des intensités proches de celles de certaines cellules), l’algorithme conduit alors à une sous-segmentation, certaines cellules n’étant plus détectées. À l’inverse, un seuil trop bas laisse subsister des éléments parasites dans l’image, ce qui peut fausser le comptage en introduisant des faux positifs. Le choix de cette valeur de seuillage a été effectué à partir d’images considérées comme exemptes de défauts. Les limitations de cette approche en présence d’artefacts ou de défauts sont discutées en partie *Annexe*, où des exemples d’images comportant des anomalies typiques sont présentés. Le réglage de cette valeur seuil peut par exemple induire les résultats de la Figure 38. Après plusieurs essais, la valeur finale de seuil retenue est :

$$I_{\text{seuil}}^{\min} = 0.12$$

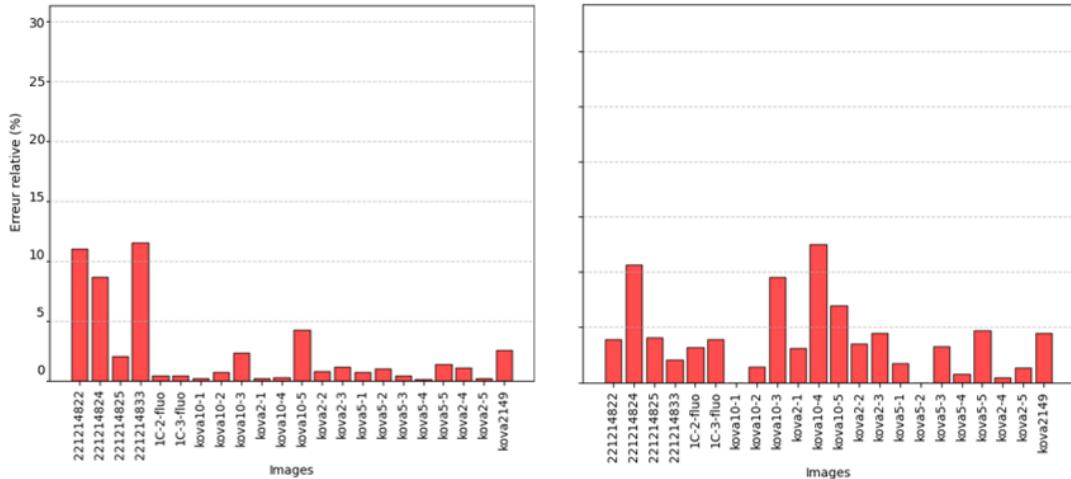


FIGURE 38 – Erreur relative absolue pour une valeur minimale de seuil de 0.12 et de 0.2

### Réglage du diamètre minimal

Le réglage de ce paramètre a un impact direct sur les phénomènes de sur-segmentation ou de sous-segmentation, selon l’approche employée. Dans les images à forte densité cellulaire (plus de 100 cellules), la taille d’une cellule peut varier de manière significative, allant de 10 à 50 pixels de diamètre environ. Dans un premier temps, nous avons tenté de régler ce paramètre de manière empirique, en nous appuyant sur un échantillon représentatif d’images. Cependant, cette méthode s’est révélée peu fiable : l’erreur relative associée à la détection variait fortement selon les cas. Nous avons donc choisi de représenter l’évolution de cette erreur en fonction du diamètre minimal considéré, afin d’identifier plus objectivement une valeur optimale. Il a donc par la suite été décidé de fixer la valeur de ce seuil à :

$$d_{\text{seuil}}^{\min} = 10 \text{ pixels}$$

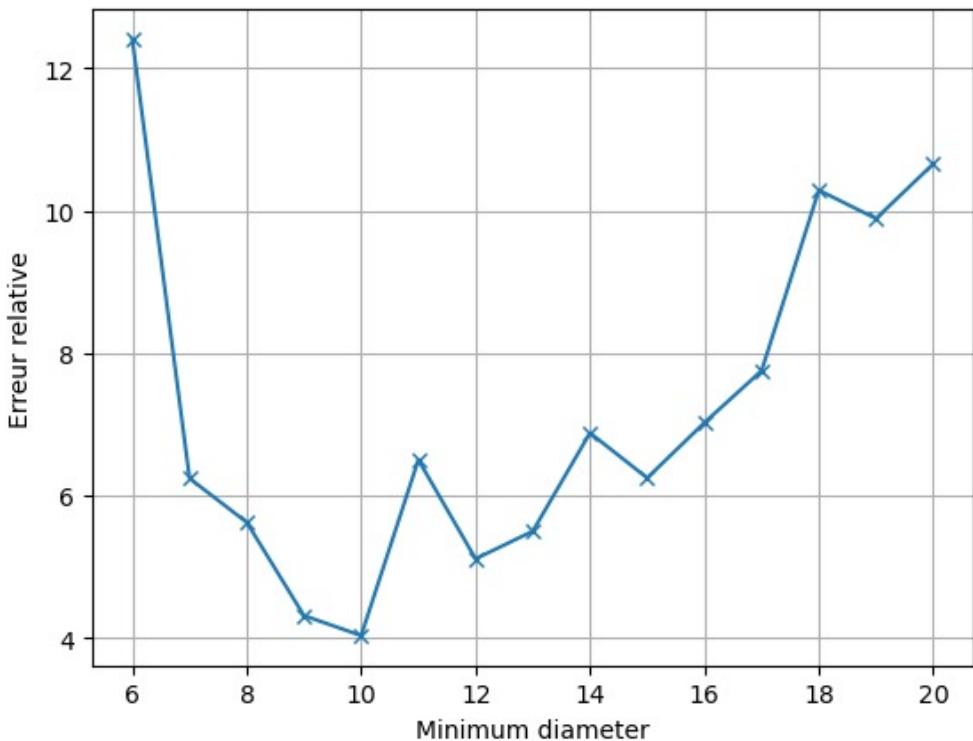


FIGURE 39 – Evolution de l'erreur relative (en valeur absolue) pour l'ensemble du jeu d'images (plus de 100 cellules)

### Ajout d'un filtre gaussien

Afin d'améliorer la qualité de la segmentation et de réduire les erreurs liées à la sur-segmentation, un filtre gaussien a été intégré en amont du module IdentifyPrimaryObjects dans CellProfiler. Ce prétraitement permet de lisser l'image en atténuant les variations d'intensité à petite échelle, souvent causées par le bruit ou des structures subcellulaires non pertinentes pour l'analyse. En réduisant la présence de minima locaux artificiels, le filtrage gaussien limite la détection excessive d'objets et améliore la robustesse du seuillage adaptatif appliqué par la suite. Cette étape de lissage favorise ainsi une détection plus cohérente et plus fidèle des objets biologiques d'intérêt [18].

Le filtre gaussien (dans le cas d'une distribution centré) est défini par :

$$G(x, y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right)$$

où  $\sigma$  est l'écart-type de la distribution (contrôle la largeur du lissage).

Le choix de l'écart type s'est fait en déterminant l'évolution de l'erreur relative en fonction de ce paramètre pour le jeu d'image de plus de 100 cellules (voir Figure 41). Il a donc été retenu comme valeur :

$$\boxed{\sigma = 5}$$

Au final, nous obtenons les résultats de la Figure 42 pour le jeu d'images comportant plus de 100 cellules.

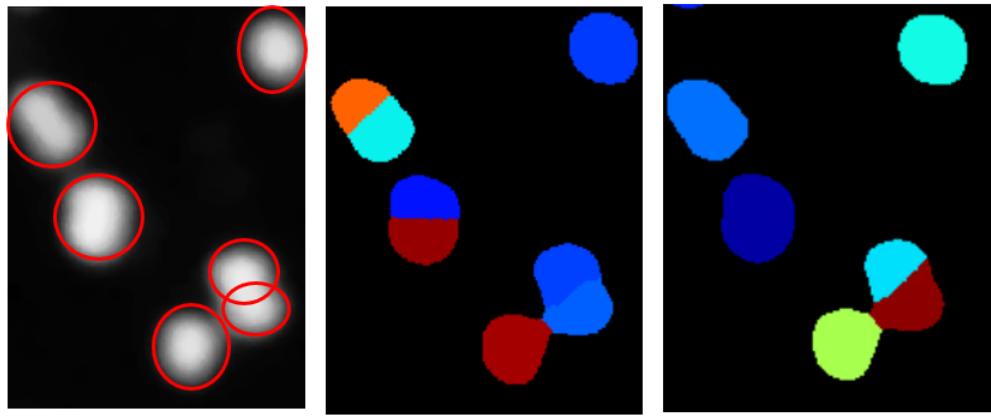


FIGURE 40 – Exemple d'effet d'ajout d'un filtre gaussien. La vérité terrain est entourée en rouge

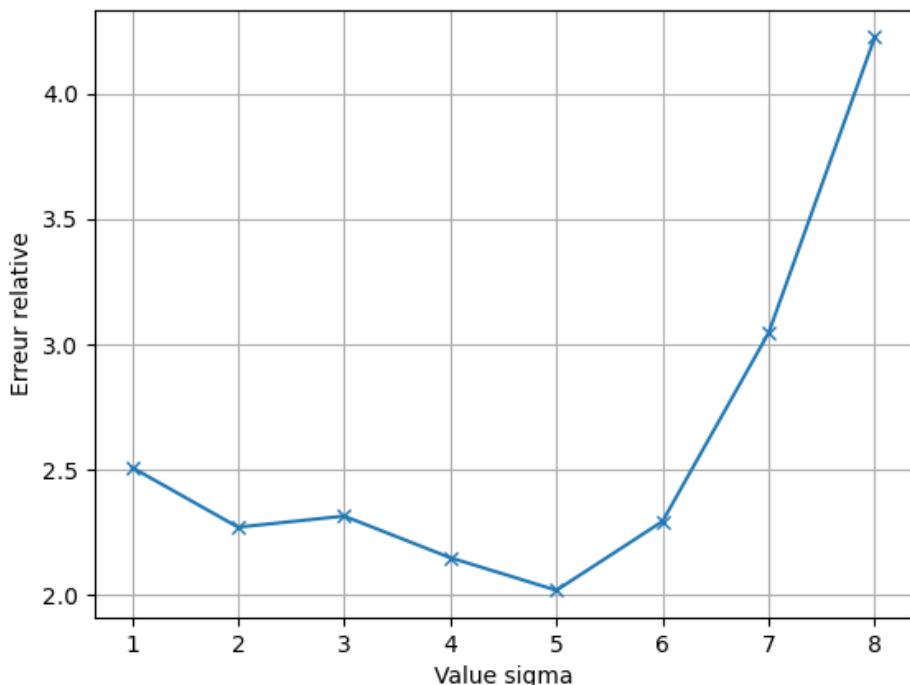


FIGURE 41 – Evolution de l'erreur relative (en %) en fonction de l'écart type du filtre gaussien

La méthode ainsi décrite amène à de bons résultats. Nous constatons que l'approche amène à avoir davantage de sous-segmentation que de sur-segmentation. Cette observation est cohérente au regard du contrôle que nous avons apporté via l'implémentation des deux seuils précédemment évoqués. Selon la métrique utilisée, nous obtenons une erreur moyenne de : 2.17 % ainsi qu'un écart type de 1.18 %.

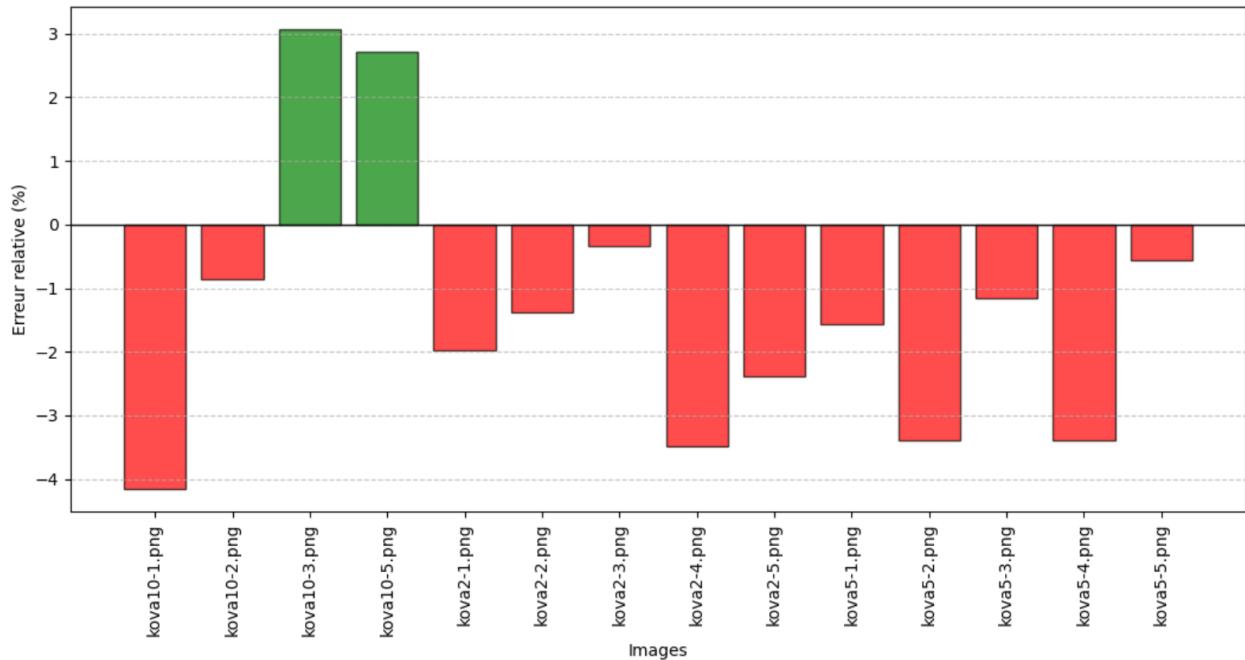


FIGURE 42 – Erreur relative pour les images de plus de 100 cellules

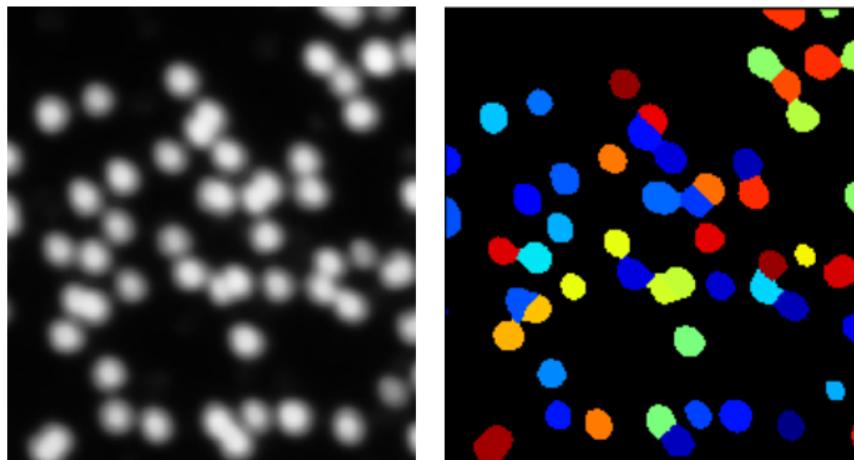


FIGURE 43 – Exemple de segmentation réalisées via la méthode

### Etude sur les images comportant moins de 100 cellules

En utilisant l'approche que nous avons développé sur les images à faible densité (comportant moins de 100 cellules), nous obtenons les résultats de la Figure 44. Des erreurs plus importantes peuvent être observées sur les images à faible densité. Il convient toutefois de souligner que ce phénomène est étroitement lié à l'utilisation de la métrique relative (1), dont l'ordre de grandeur dépend du nombre total de cellules présentes sur l'image.

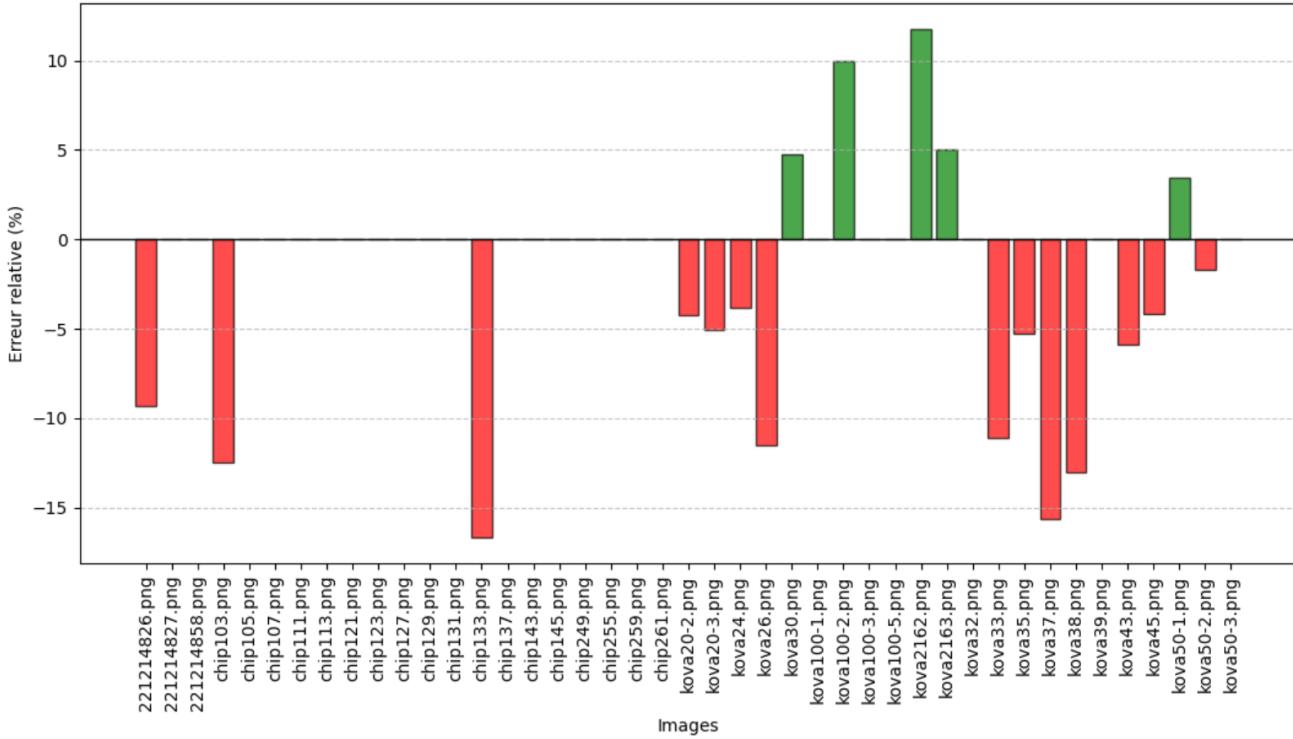


FIGURE 44 – Erreur relative pour les images de moins de 100 cellules

En particulier, lorsque l'image contient un faible nombre de cellules (par exemple moins d'une dizaine), une erreur absolue modeste — de l'ordre d'une à deux cellules — peut conduire à une erreur relative élevée, ce qui amplifie artificiellement l'écart entre les méthodes comparées. Cette sensibilité met en évidence une limite de la métrique utilisée lorsqu'elle est appliquée à des images à faible densité cellulaire. Par exemple, dans le cas de l'image *chip133*, l'algorithme a conduit au comptage de 5 cellules tandis que l'image en comportait 6. Nous obtenons au final une erreur relative moyenne de 3.60 % ainsi qu'un écart type de 4.99 %.

### Limits de la méthode

L'ensemble de cette méthode a été optimisée sur un type d'images précises que nous considérons sans défauts :

- géométrie quasi-circulaire des cellules pour ne pas biaiser le filtrage lié au diamètre minimal ainsi que la détection basée sur la forme de l'objet (voir Figure 45) ;
- taille modérée pour ne pas présenter une trop forte luminosité (voir Figure 46) ;
- un fond sans "résidu" (voir Figure 47) ;
- présentant des cellules en "fond" avec un problème de focus (voir Figure 48) ;

La Figure 49 est un dernier exemple d'un ensemble d'images ne respectant pas les critères précédemment évoqués. Au contraire, les images de la Figure 50 respectent les critères.

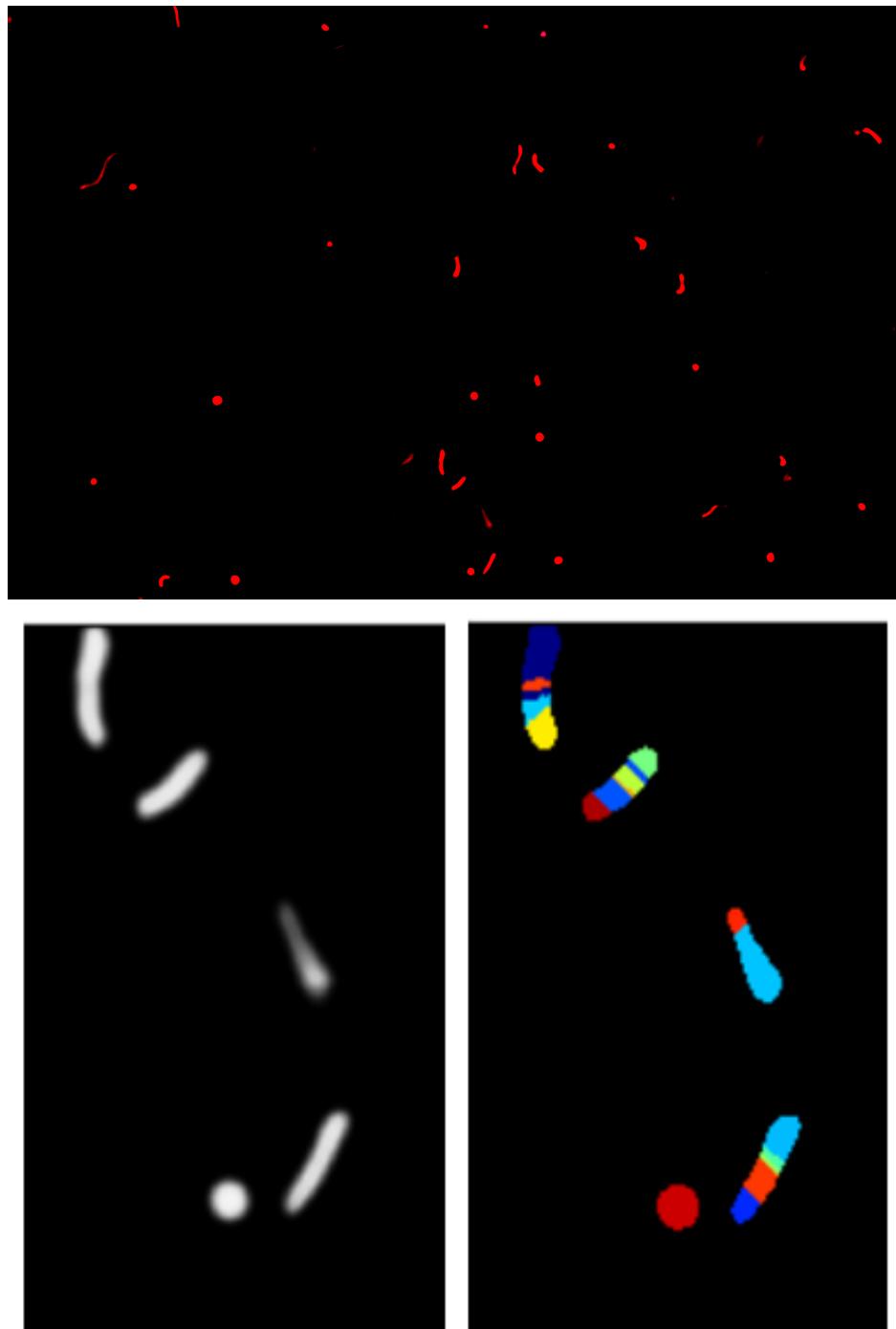


FIGURE 45 – Exemple d’images où les cellules n’ont pas une géométrie quasi-circulaire

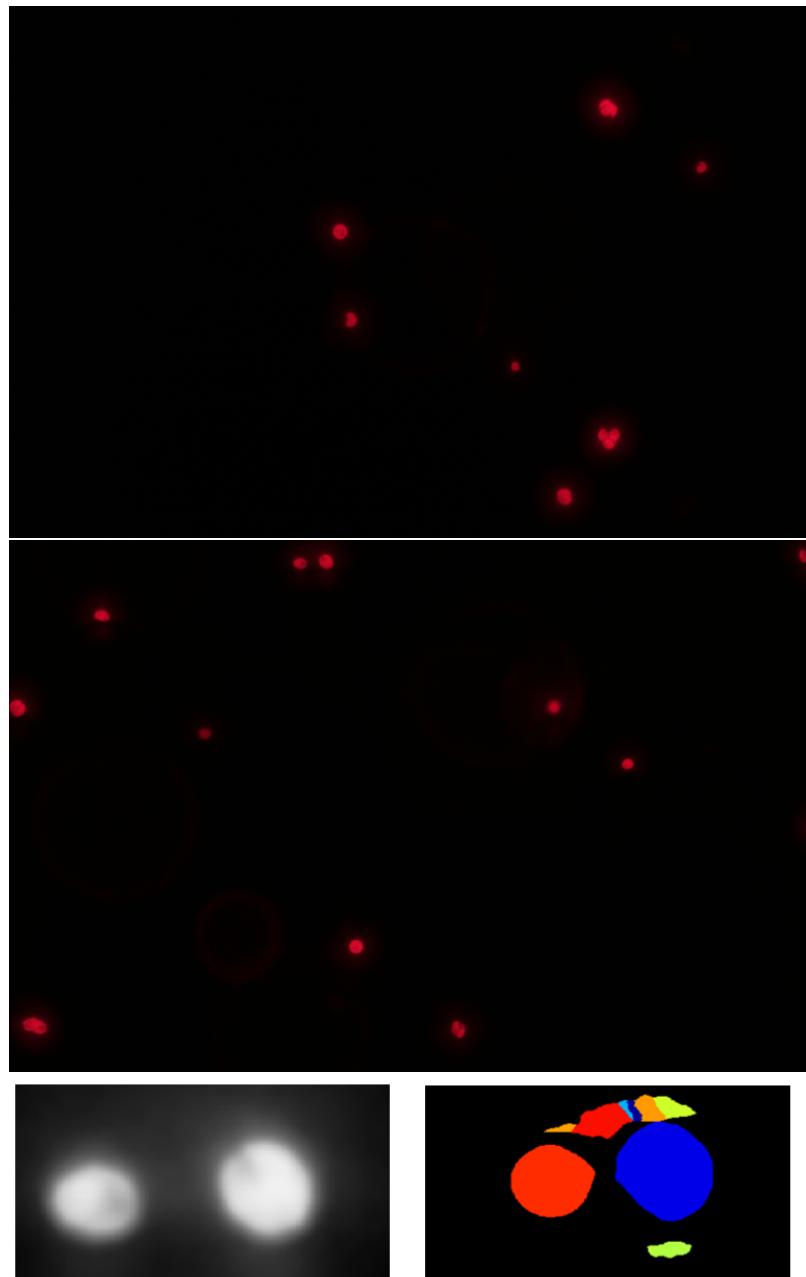


FIGURE 46 – Exemple d'images où les cellules sont trop grandes et présentent une luminosité trop importante

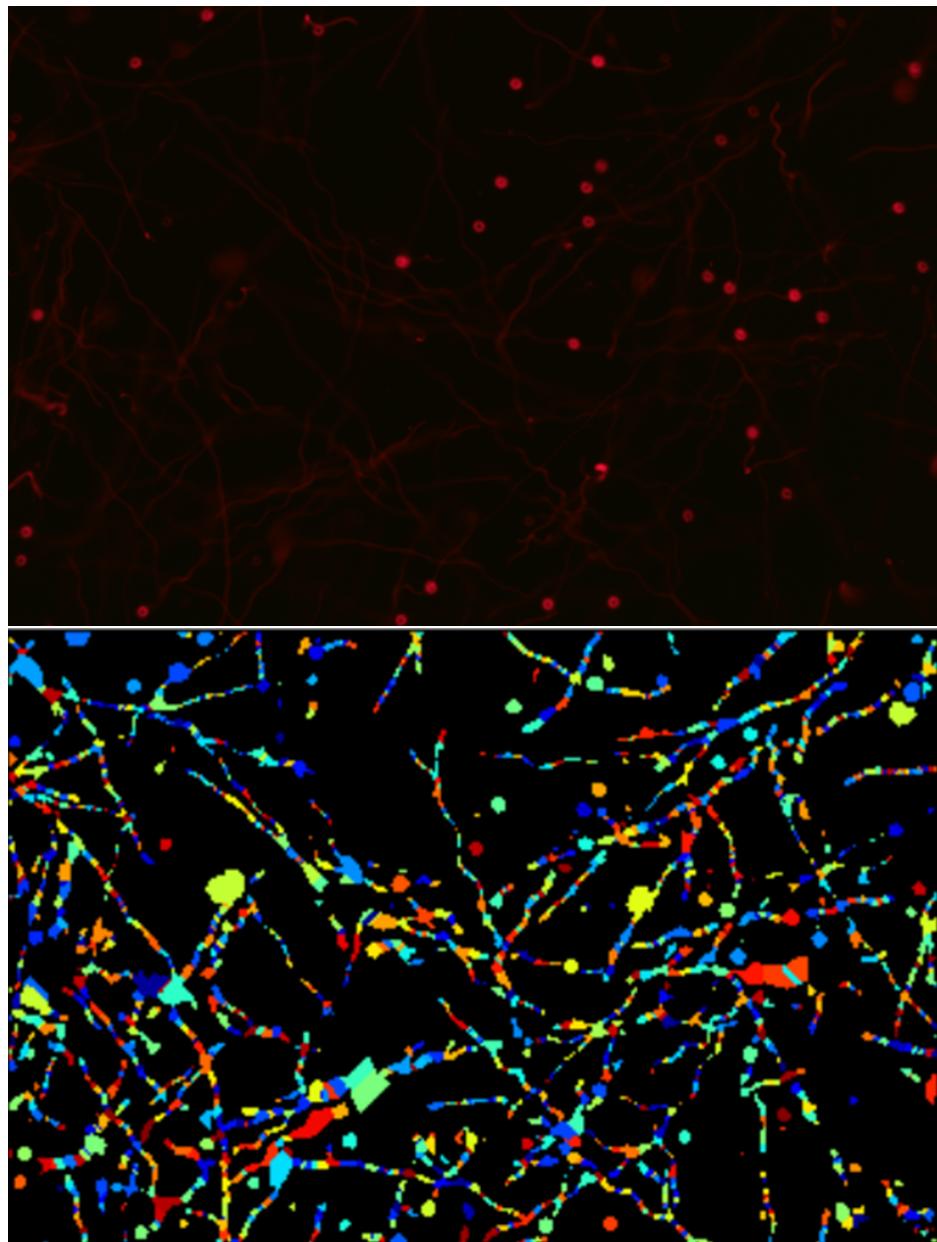


FIGURE 47 – Exemple d’image présentant un fond avec des résidus

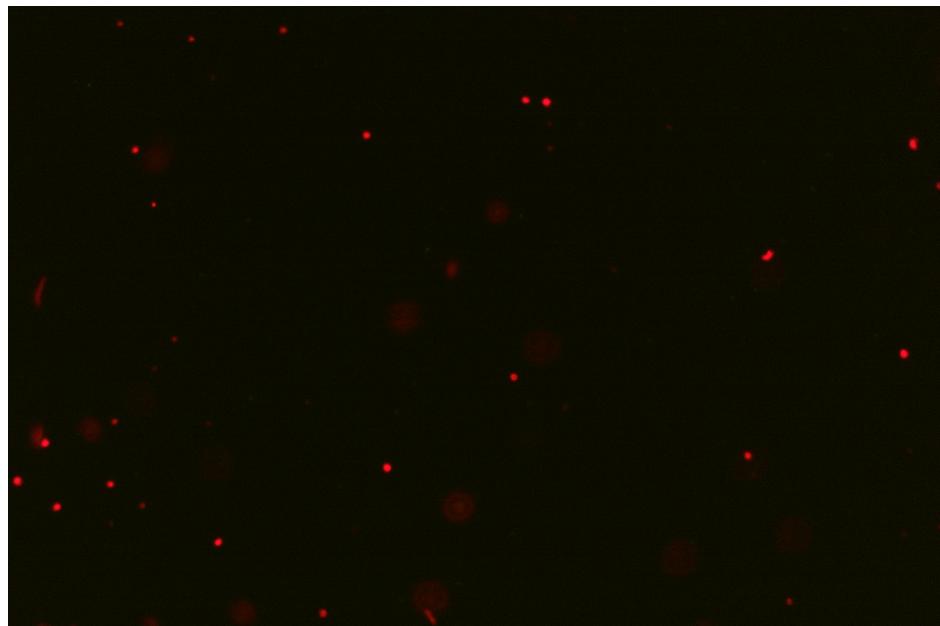


FIGURE 48 – Exemple d'image où certaines cellules apparaissent en fond

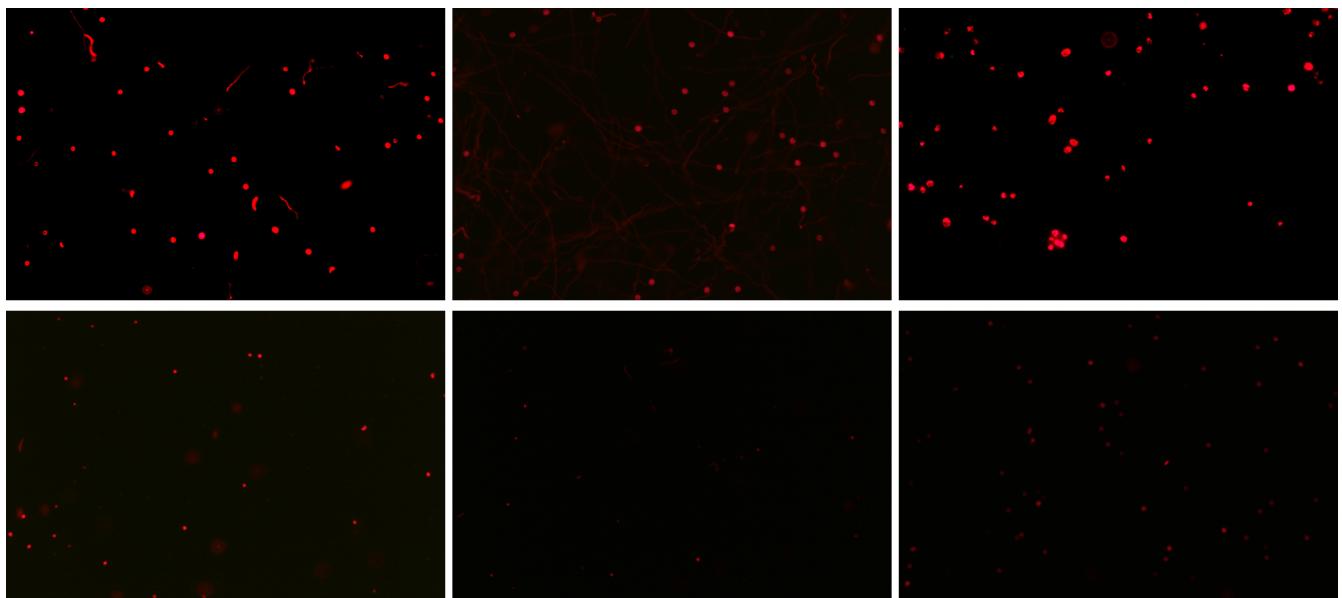


FIGURE 49 – Exemple d'images invalides

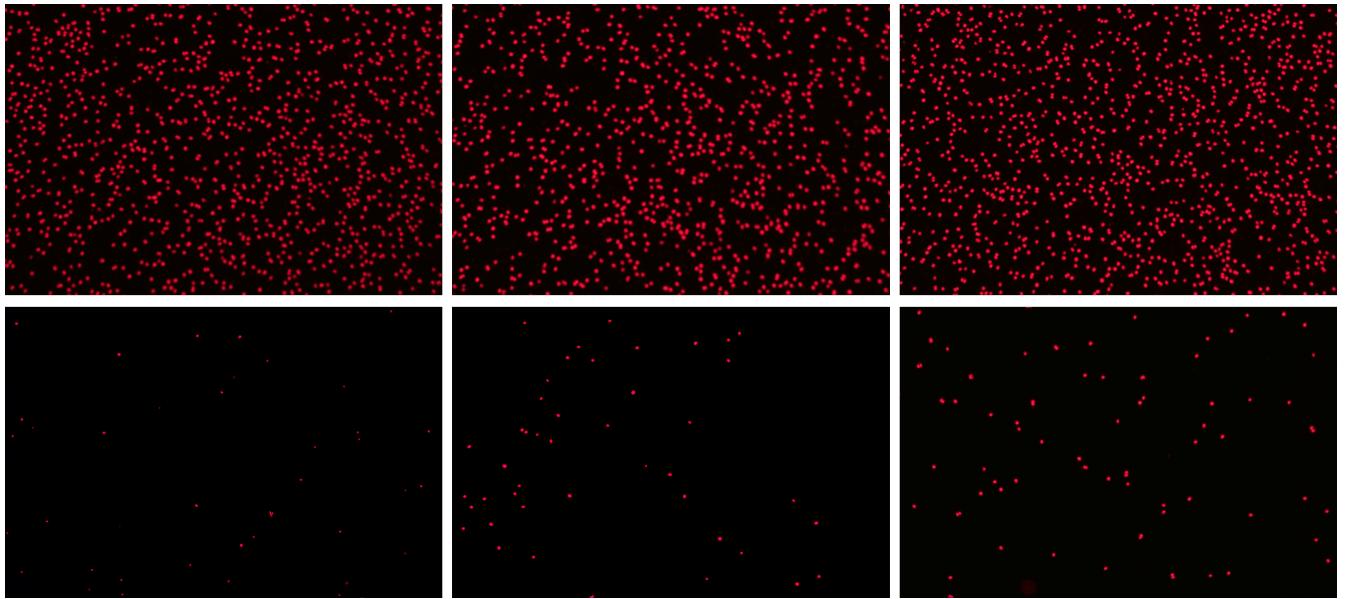


FIGURE 50 – Exemple d’images respectant les critères

### III.2.7 Conclusion méthodes classiques et choix de la méthode

Méthode de segmentation	Erreur relative (%)	Ecart relatif (%)
Composantes connexes	82.80	123.73
Érosion	7.74	16.70
Détection de contours	11.26	19.57
Watershed	12.85	22.33
Cellprofiler	2.17	1.18

TABLE 4 – Comparaison des méthodes classiques de traitement d’images pour les faibles densités cellulaires

Méthode de segmentation	Erreur relative (%)	Ecart relatif (%)
Composantes connexes	18.55	10.78
Erosion	6.12	6.03
Détection de contours	23.65	17.94
Watershed	3.34	2.32
Cellprofiler	3.6	4.99

TABLE 5 – Comparaisons des méthodes classiques sur les images à haute densité

Les différentes études menées au cours de ce travail ont mis en évidence la complexité d’une segmentation à la fois pertinente, robuste et généralisable. La méthode par composantes connexes, bien qu’intuitive et simple à implémenter, montre rapidement ses limites face à des images à forte densité cellulaire ou contenant des cellules agglomérées. À l’inverse, la méthode *Watershed*, plus sophistiquée, permet une meilleure séparation des cellules proches, en particulier lorsque couplée à un contrôle précis des graines de segmentation. Néanmoins, cette méthode reste sensible aux paramètres de seuillage et aux seuils morphologiques définis, et peut engendrer des phénomènes de sur-segmentation, notamment dans les images à faible densité, malgré les filtres mis en place pour les limiter.

À l’issue de ces comparaisons, la méthode développée via le logiciel *CellProfiler* se distingue par sa robustesse et sa performance globale, aussi bien sur les images à faible qu’à forte densité cellulaire. Elle offre une grande souplesse de paramétrage tout en proposant une interface structurée facilitant la reproductibilité. Au vu de l’ensemble des résultats obtenus, nous recommandons l’utilisation de cette approche pour des tâches de segmentation cellulaire classique. Toutefois, il est important de souligner que l’efficacité de cette méthode repose sur certaines hypothèses fortes concernant la morphologie des objets, leur contraste par

rapport au fond, et la qualité de l'image d'entrée. Ces hypothèses doivent donc être prises en compte et vérifiées au cas par cas avant application à de nouvelles images.

Un autre avantage important de la méthode *CellProfiler* réside dans ses bons résultats sur les images à haute densité cellulaire. Cette robustesse est particulièrement pertinente d'un point de vue expérimental : en effet, il est souvent plus simple en laboratoire de produire des échantillons denses, car cela évite les étapes de dilution qui peuvent être fastidieuses, sources d'erreurs ou consommatrices de matériel. La possibilité d'analyser correctement ces images denses sans ajustements majeurs constitue donc un atout considérable de l'approche retenue.

### III.3 Méthode par apprentissage

On choisit l'epoch qui maximise le critère mAP@0.5 :0.95. Nous allons analyser plus en détail les résultats du modèle sortant du lot : yolov5mu entraîné sur notre dataset augmenté à 770 images via notamment la technique copy-past.

Modèle	Dataset	Epoch optimale	Précision	Rappel	mAP@0.5	mAP@0.5 :0.95
<b>yolov5su</b>	dataset_copy_past_augm770	84	0.875	0.834	0.867	0.415
	dataset_augm410	68	0.854	0.714	0.795	0.372
	dataset_original	90	0.867	0.785	0.834	0.396
<b>yolov5mu</b>	dataset_copy_past_augm770	83	0.875	<b>0.854</b>	<b>0.881</b>	<b>0.419</b>
<b>yolov5lu</b>	dataset_copy_past_augm770	89	0.865	0.843	0.873	0.418
<b>yolov11s</b>	dataset_copy_past_augm770	53	0.866	0.797	0.839	0.392
<b>yolov11m</b>	dataset_copy_past_augm410	99	<b>0.888</b>	0.691	0.768	0.365
	dataset_augm410	80	0.872	0.687	0.757	0.358

TABLE 6 – Performances des modèles YOLO selon les jeux de données. En gras : meilleures valeurs par colonne.

La Figure 51 présente l'évolution des principales métriques d'apprentissage et de validation au cours des 100 epochs d'entraînement. On observe une décroissance progressive des pertes (loss) pour les boîtes englobantes (*box\_loss*), les classes (*cls\_loss*) et la régression de distance (*dfl\_loss*), aussi bien sur les ensembles d'entraînement que de validation, signe d'une convergence efficace du modèle. En parallèle, les performances en détection s'améliorent clairement, avec une augmentation régulière de la précision, du rappel, ainsi que des métriques mAP@0.5 et mAP@0.5 :0.95. Ces courbes indiquent que le modèle apprend de manière stable, sans surapprentissage apparent, et atteint une performance satisfaisante au terme des 100 epochs.

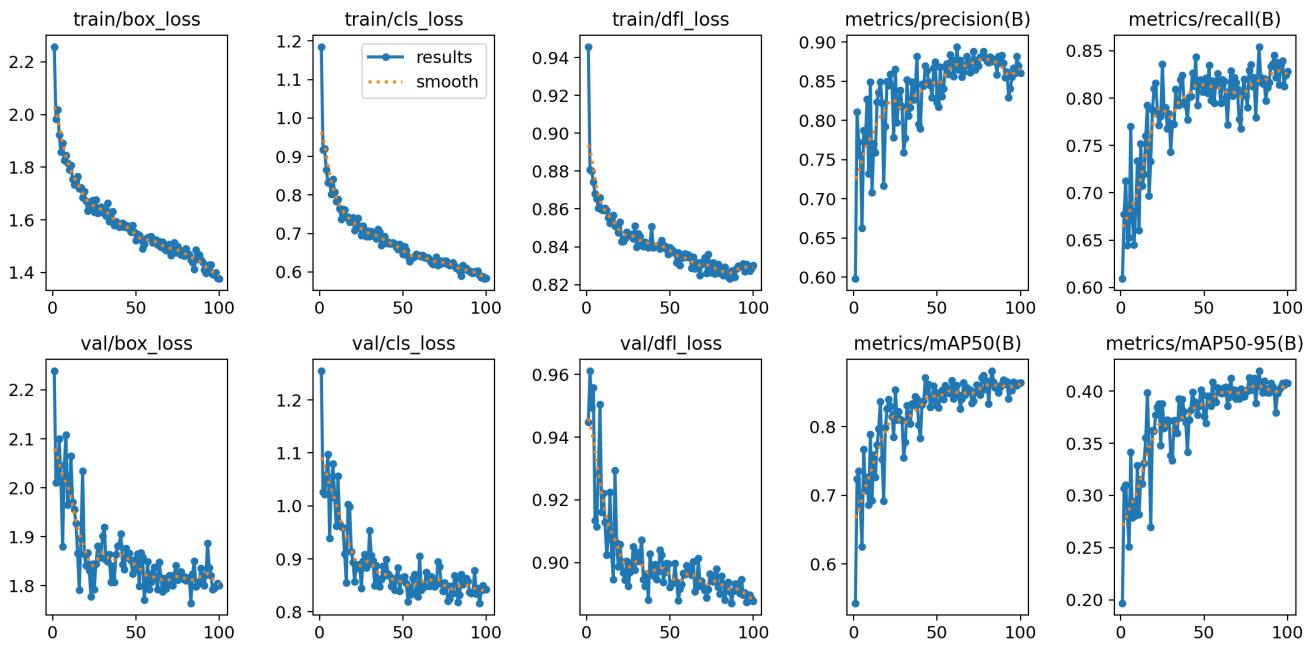


FIGURE 51 – Résultats de YOLOv5mu entraîné sur dataset\_copy\_paste\_augm770

La Figure 52 montre la courbe précision-rappel obtenue à la meilleure epoch d’entraînement du modèle. Cette courbe illustre le compromis entre la précision (taux de bonnes prédictions parmi les détections) et le rappel (taux de détections parmi les objets présents), pour différents seuils de confiance. La forme convexe et élevée de la courbe indique que le modèle parvient à maintenir une \*\*précision élevée même pour des valeurs de rappel proches de 0.9\*\*, ce qui témoigne d’un bon équilibre entre sensibilité et spécificité. Le score mAP@0.5 associé est de 0.867, ce qui reflète une performance robuste sur l’ensemble des prédictions.

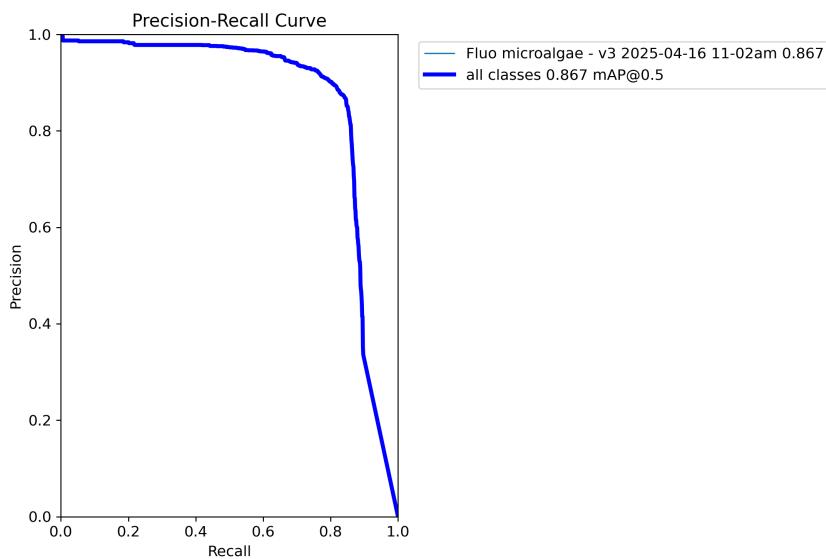


FIGURE 52 – Courbe précision-rappel du modèle pour la détection des microalgues fluorescentes (mAP@0.5 = 0.867)

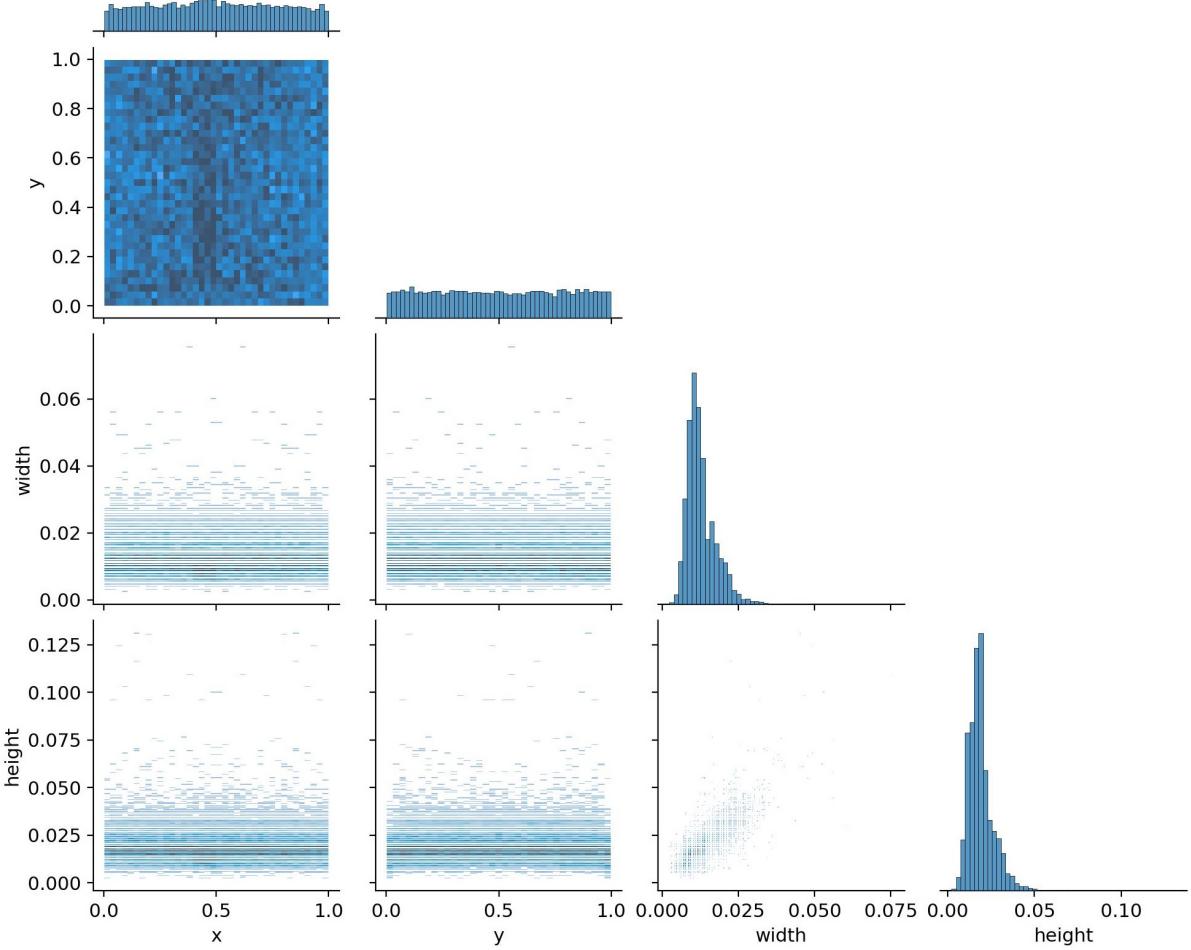


FIGURE 53 – Distribution conjointe des coordonnées ( $x, y$ ) et des tailles (width, height) des objets annotés

La Figure 53 présente un correlogramme montrant la distribution des coordonnées ( $x, y$ ), ainsi que des dimensions normalisées (width, height) des boîtes englobantes annotées dans le dataset. On observe une répartition relativement homogène des centres ( $x, y$ ) des objets dans l'image, ce qui indique une bonne diversité spatiale des instances annotées. Les histogrammes des largeurs et hauteurs révèlent que la majorité des objets sont de petite taille (autour de 0.01 à 0.025 en valeurs normalisées), avec peu de variabilité. Cela suggère que les objets sont de forme relativement constante (des cellules de taille comparable), ce qui peut faciliter l'apprentissage du modèle. La faible diversité de taille pourrait être exploitée pour adapter les ancrées ('anchors') lors de l'entraînement avec YOLO. En effet, la faible variabilité dans les dimensions des boîtes englobantes suggère une forme relativement constante des objets à détecter. Ce constat peut être exploité en adaptant les boîtes d'ancrage (anchors) du modèle à cette distribution restreinte. L'ajustement automatique des ancrées, proposé par YOLO via l'option `autoanchor`, permet de mieux couvrir l'espace des objets présents, et améliore la qualité des prédictions tout en réduisant les erreurs de localisation.

**Conclusion.** L'ensemble des résultats met en évidence l'efficacité des modèles YOLO pour la détection de microalgues à partir d'images en fluorescence. Parmi les architectures testées, YOLOv5mu combiné à un dataset enrichi par la méthode *Copy-Paste* se distingue nettement, atteignant un score mAP@0.5 : 0.95 de 0.419. Ce modèle offre un excellent compromis entre précision, rappel et robustesse. Ces résultats suggèrent qu'un apprentissage supervisé, même sur un dataset limité mais bien préparé, peut aboutir à une détection fine et généralisable. Parmi les pistes d'amélioration envisageables, on peut citer l'exploration de modèles/dataset spécifiquement conçus pour la détection d'objets petits et nombreux, l'optimisation automatique des boîtes d'ancrage (*anchors*) en fonction de la distribution réelle des tailles dans le dataset, ainsi que l'ajout d'un post-traitement permettant d'affiner la segmentation ou la séparation des objets proches.

## IV Conclusion

Ce travail s'inscrit dans le cadre d'une problématique concrète et récurrente en biologie expérimentale : segmenter automatiquement non pas une unique image, mais un ensemble hétérogène d'images issues de l'imagerie en microscopie, afin d'en extraire des informations quantitatives fiables sur des populations de microalgues. L'enjeu est donc non seulement d'atteindre une bonne performance locale, mais aussi de proposer une approche reproductible, robuste et adaptable aux variations inévitables d'un protocole expérimental (densité cellulaire, qualité optique, intensité du bruit, artefacts).

Deux familles de méthodes ont été comparées dans ce travail : les méthodes classiques de traitement d'image, représentées notamment par la méthode implémentée via le logiciel *CellProfiler*, et les approches récentes par apprentissage supervisé, basées ici sur l'architecture YOLOv5.

Les résultats obtenus montrent que la méthode classique offre d'excellentes performances lorsque les images respectent les hypothèses précédemment évoquées à la page 35. Dans ce contexte, elle présente l'avantage d'être rapide à mettre en œuvre et est facilement interprétable. Elle est donc tout à fait adaptée à un protocole maîtrisé, dans lequel l'expérimentateur est en mesure d'ajuster finement les conditions de prise de vue ou d'appliquer des filtres adaptés.

À l'inverse, la méthode par apprentissage automatique demande davantage d'efforts de mise en œuvre (annotation manuelle, réglage des hyperparamètres, entraînement), mais elle se montre plus robuste face aux imperfections des images : bruit, sur- ou sous-exposition, chevauchement marqué des cellules. Elle constitue ainsi une solution pertinente dans les cas où le contrôle expérimental est plus limité, ou lorsque l'on souhaite segmenter de grands volumes d'images issues de conditions variées.

En définitive, le choix de la méthode à utiliser dépend fortement du contexte expérimental. Si l'utilisateur dispose de moyens pour garantir la qualité des images en amont, les méthodes classiques peuvent suffire et offrir une excellente précision. En revanche, dans un cadre plus contraint, où les images présentent des défauts ou une forte hétérogénéité, l'usage d'une méthode d'apprentissage devient préférable, malgré sa complexité initiale. Les images des figures 54 et 55 sont des exemples des résultats obtenus avec les 2 méthodes développées. Les résultats explicites sont également donnés dans les tableaux 7 et 8

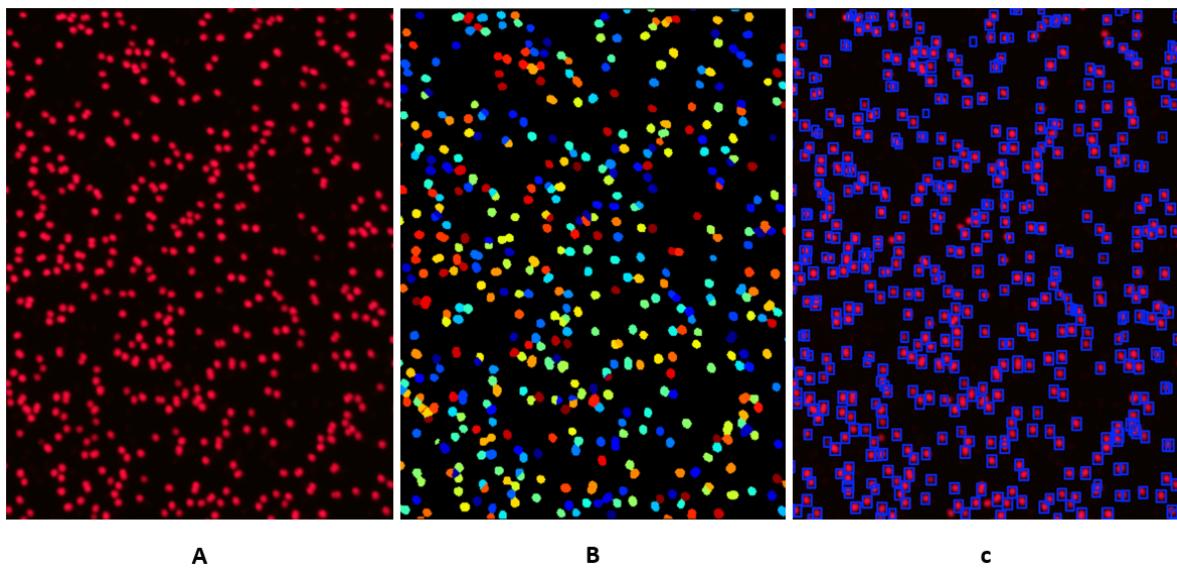


FIGURE 54 – Résultats de segmentation sur des images de haute densité cellulaire ; A : image du microscope, B : segmentation via *CellProfiler*, C : segmentation par la méthode par apprentissage

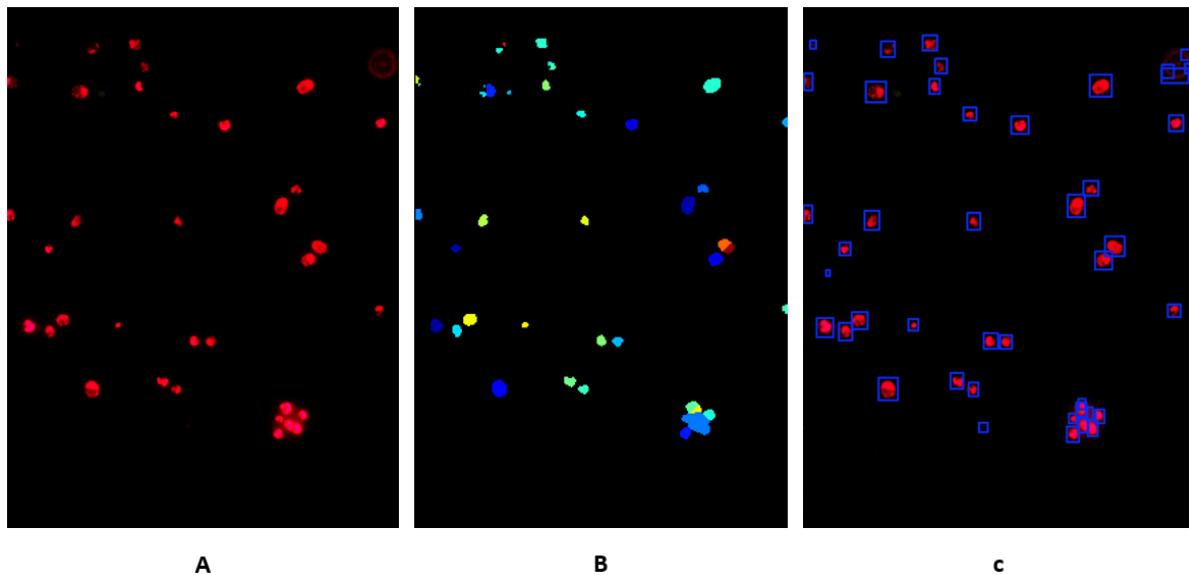


FIGURE 55 – Résultats de segmentation sur les images ne respectant pas les hypothèses de la méthode via *CellProfiler* ; A : image du microscope, B : segmentation via *CellProfiler*, C : segmentation par la méthode par apprentissage

Type d'image	Erreur relative (%)	Ecart relatif (%)
Faible densité	2.17	1.18
Haute densité	3.6	4.99
"Mauvaises"	25.19	73.07

TABLE 7 – Résultats de segmentation par la méthode *CellProfiler* sur différents types d'images

Type d'image	Erreur relative (%)	Ecart relatif (%)
Faible densité	20.1	
Haute densité	10.6	15.6
"Mauvaises"	17.3	14.1

TABLE 8 – Résultats de segmentation par la méthode d'apprentissage sur différents types d'images

La méthode de détection cellulaire par apprentissage automatique a été développée avec pour objectif principal l'étude de l'influence du jeu de données et de son augmentation sur des métriques spécifiques, telles que la moyenne de précision (MAP), plutôt que sur l'erreur relative moyenne. Malgré cela, les résultats montrent que l'algorithme s'adapte remarquablement bien à des cas complexes, en particulier sur les images dites « problématiques » où les méthodes classiques échouent souvent. L'analyse des prédictions révèle que les zones contenant des cellules sont généralement bien localisées, bien qu'il arrive qu'une cellule soit segmentée en plusieurs entités. Ce phénomène, s'il peut dégrader la précision selon certaines métriques comme l'erreur relative moyenne, a un impact moindre sur les critères d'optimisation utilisés. Il pourrait néanmoins être corrigé facilement par un ajustement de l'apprentissage ou un fine-tuning des paramètres de détection, ouvrant la voie à une amélioration ciblée des performances selon les besoins spécifiques de l'application.

## Références

- [1] Zhou S, Jiang J, Hong X, Fu P and Yan H (2023) Vision meets algae : A novel way for microalgae recognition and health monitor. *Front. Mar. Sci.* 10 :1105545.  
<https://doi.org/10.3389/fmars.2023.1105545>
- [2] Orenstein, E. C., Bejbom, O., Sosik, H. M., & Olson, R. J. (2015). WHOI-Plankton : A large-scale fine-grained visual recognition benchmark dataset for plankton classification. *arXiv preprint arXiv* :1510.00745.  
<https://arxiv.org/abs/1510.00745>
- [3] Li, W., Luo, Z., Xu, X., Wang, J., Xu, Y., Liu, M., & Zhang, H. (2021). EMDS-5 : Environmental Microorganism Dataset Fifth Version for Image Classification and Segmentation. *Computational Intelligence and Neuroscience*, 2021, 1–15.  
<https://neurips.cc/virtual/2023/events/datasets-benchmarks-2023>
- [4] INRIA, Sorbonne Université, & CNRS (2024). CIDACC : Cell Image Dataset for Automatic Cell Counting. Available at  
<https://cidacc.inria.fr>.
- [5] Metrics for evaluating 3D medical image segmentation : analysis, selection, and tool ; Abdel Aziz Taha and Allan Hanbury  
<https://bmcmedimaging.biomedcentral.com/articles/10.1186/s12880-015-0068-x>
- [6] Orenstein, E. C., Bejbom, O., Peacock, E. E., & Sosik, H. M. (2015). Whoi-plankton-a large scale fine grained visual recognition benchmark dataset for plankton classification. *arXiv preprint*  
<https://arxiv.org/pdf/1510.00745>
- [7] Image Segmentation Based on 2D Otsu Method with Histogram Analysis ; Jun Zhang, Jun Zhang  
<https://ieeexplore.ieee.org/abstract/document/4723207>
- [8] Robust Cell Image Segmentation Methods, E. Bengtsson, C. Wählby, and J. Lindblad  
[https://www.researchgate.net/publication/240223597\\_Robust\\_cell\\_image\\_segmentation\\_methods](https://www.researchgate.net/publication/240223597_Robust_cell_image_segmentation_methods)
- [9] Red blood cell counting analysis by considering an overlapping constraint ; Razali Tomari, Wan Nurshzwani Wan Zakaria, Rafidah Ngadengon and Mohd Helmy Abd Wahab  
[https://www.researchgate.net/publication/282053141\\_Red\\_blood\\_cell\\_counting\\_analysis\\_by\\_considering\\_an\\_overlapping\\_constraint](https://www.researchgate.net/publication/282053141_Red_blood_cell_counting_analysis_by_considering_an_overlapping_constraint)
- [10] L. Vincent and P. Soille, "Watersheds in digital spaces : An efficient algorithm based on immersion simulations," <https://ieeexplore.ieee.org/document/87344>
- [11] S. Beucher, "The watershed transformation applied to image segmentation,"  
[https://www.researchgate.net/publication/2407235\\_The\\_Watershed\\_Transformation\\_Applied\\_To\\_Image\\_Segmentation](https://www.researchgate.net/publication/2407235_The_Watershed_Transformation_Applied_To_Image_Segmentation)
- [12] Split and merge watershed : A two-step method for cell segmentation in fluorescence microscopy images ; Margarita Gamarraa, Eduardo Zurekb, Hugo Jair Escalantec, Leidy Hurtadod, Homero San-Juan-Vergarad.  
<https://www.sciencedirect.com/science/article/abs/pii/S1746809419301491>
- [13] Cell Image Segmentation Based on an Improved Watershed Algorithm ; Xiaoqiang Ji, Yang Li, Jiezhang Cheng, Yuanhua Yu, Meijiao Wang.  
<https://ieeexplore.ieee.org/document/7407919>
- [14] Evident Scientific, *Contrast Stretching and Histogram Normalization*, consulté en avril 2025.  
<https://evidentscientific.com/en/microscope-resource/tutorials/digital-imaging-processing/histogramstretching>
- [15] Automated Cell Counting using Image Processing ; Dewi Kartini Hassan, Hazwani Suhaimi, Muhammad Roil Bilad, Pg Emeroylariffion Abas.  
<https://computingonline.net/computing/article/view/3224>
- [16] Image Data Augmentation for Deep Learning : A Survey ; Suorong Yang, Weikang Xiao, Mengchen Zhang, Suhan Guo, Jian Zhao, Furao Shen  
<https://arxiv.org/abs/2204.08610>

- [17] What is the Difference Between Online and Offline Data Augmentation ?  
<https://milvus.io/ai-quick-reference/what-is-the-difference-between-online-and-offline-data-augmentation>
- [18] Gonzalez, R. C., & Woods, R. E. (2018). Digital Image Processing (4th ed.). Pearson.  
[https://www.c172.org/090imagePLib/books/Gonzales\\_Woods-Digital.Image.Processing.4th.Edition.pdf](https://www.c172.org/090imagePLib/books/Gonzales_Woods-Digital.Image.Processing.4th.Edition.pdf)
- [19] Terven J and Córdova-Esparza D (2023) A Comprehensive Review of YOLO Architectures in Computer Vision : From YOLOv1 to YOLOv8 and YOLO-NAS. *Mach. Learn. Knowl. Extr.* 5(4) :561–581.  
<https://www.mdpi.com/2504-4990/5/4/83>
- [20] Badithela A., Wongpiromsarn T. and Murray R.  
Evaluation Metrics for Object Detection for Autonomous Systems  
[https://arxiv.org/pdf/2210.10298](https://arxiv.org/pdf/2210.10298.pdf)
- [21] Ghiasi G, Cui Y, Srinivas A, Qian R, Lin T-Y, Cubuk E D, Le Q V and Zoph B (2021)  
Simple Copy-Paste is a Strong Data Augmentation Method for Instance Segmentation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 2918–2928.  
[https://arxiv.org/pdf/2012.07177](https://arxiv.org/pdf/2012.07177.pdf)
- [22] Zoph B, Cubuk E D, Ghiasi G, Lin T-Y, Shlens J and Le Q V (2019) Learning Data Augmentation Strategies for Object Detection. arXiv preprint arXiv :1906.11172.  
[https://arxiv.org/pdf/1906.11172](https://arxiv.org/pdf/1906.11172.pdf)
- [23] Roboflow : Computer vision tools for developers and enterprises  
<https://roboflow.com>
- [24] Local color and morphological image feature based vegetation identification and its application to human environment street view vegetation mapping Istvan G. Lauko and Adam Honts and Jacob Beihoff and Scott Rupprecht  
<https://doi.org/10.1080/10095020.2020.1805367>

## V Annexe

Article	Type de cellules	Méthode	Description brève
[1]	Microalgues marines	Acquisition et annotation des données, segmentation via méthode par apprentissage	Nouveau jeu de données proposé pour la détection des microalgues ainsi que des méthodes de détection (YOLOv5, YOLOv8 et TOOD)
[2]	Phytoplancton marin	Classification automatique	Jeu de données benchmark à large échelle pour la reconnaissance fine de phytoplancton via réseaux neuronaux
[3]	Micro-organismes environnementaux	Méthodes classiques et IA	Jeu de données EMDS-5 utilisé pour la classification et la segmentation avec diverses approches
[4]	Cellules diverses (inclus microalgues)	Détection automatique, comptage	Jeu de données annoté pour le comptage automatique de cellules, proposé par l'INRIA
[5]	Images médicales	Analyse des métriques de segmentation	Définition et comparaison de nombreuses métriques utilisées pour évaluer la qualité des segmentations automatiques
[7]	Objets biomédicaux	Seuillage automatique (Otsu 2D)	Méthode basée sur l'analyse 2D de l'histogramme pour améliorer la segmentation par seuillage binaire
[8]	Cellules animales	Morphologie mathématique, seuillage	Revue détaillée sur la robustesse des méthodes classiques pour segmenter des cellules en microscopie
[9]	Globules rouges	Composantes connexes avec contrainte de recouvrement	Comptage automatique de cellules prenant en compte les cas de chevauchement partiel
[10]	Images biomédicales	Segmentation morphologique (Watershed par immersion)	Méthode de segmentation fondée sur la simulation d'inondation, base de nombreuses variantes ultérieures

<b>Article</b>	<b>Type de cellules</b>	<b>Méthode</b>	<b>Description brève</b>
[11]	Cellules (microscopie)	Watershed avec marqueurs	Approche guidée par des marqueurs pour limiter la sur-segmentation causée par les minima locaux spurious
[12]	Cellules en fluorescence	Watershed amélioré (split and merge)	Algorithme en deux étapes combinant séparation initiale et fusion contrôlée des régions sur-segmentées
[13]	Cellules en microscopie optique	Watershed amélioré	Méthode basée sur une version optimisée de l'algorithme de watershed, prenant en compte les contours flous et les sur-segmentations excessives
[14]	Cellules en général (prétraitement)	Étirement d'histogramme (contrast stretching)	Justifie l'utilisation de l'étirement d'histogramme pour améliorer le contraste global et ainsi faciliter la segmentation automatique des cellules
[15]	Cellules variées	Pipeline modulaire (traitement d'image classique)	Utilisation de CellProfiler pour le comptage automatique via des modules d'analyse adaptables

TABLE 9 – Articles utilisant des méthodes classiques pour la segmentation d'images de cellules