VISA – TP Segmentation

Arnaud Cojez

mercredi 16 novembre 2016

Table des matières

1 Introduction **3**

2 Segmentation des images par K-Means et discrimination linéaire **4**
2.1 Explication 4
2.2 Résultats 4
2.2.1 Segmentation par K-means 4
2.2.2 Classes obtenues 5
2.2.3 Segmentation par distance euclidienne 5

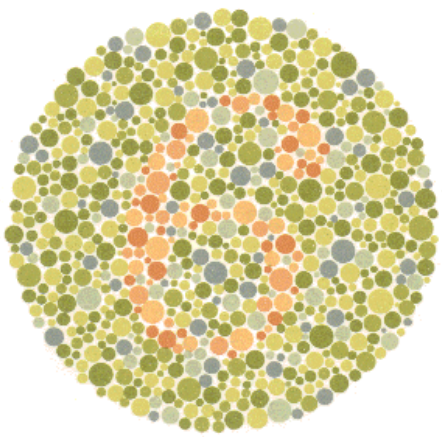
3 Application aux images *cas_4* **6**
3.1 Explication 6
3.2 Résultats 6
3.2.1 Image *cas_4_dalton6* dans l'espace RGB 6
3.2.2 Image *cas_4_dalton6* dans l'espace HSB 7
3.2.3 Image *cas_4_dalton8* dans l'espace RGB 8
3.2.4 Image *cas_4_dalton8* dans l'espace HSB 8
3.2.5 Image *cas_4_dalton29* dans l'espace RGB 9
3.2.6 Image *cas_4_dalton29* dans l'espace HSB 9
3.2.7 Image *cas_4_dalton45* dans l'espace RGB 10
3.2.8 Image *cas_4_dalton45* dans l'espace HSB 10
3.2.9 Image *cas_4_dalton_ligne* dans l'espace RGB 11
3.2.10 Image *cas_4_dalton_ligne* dans l'espace HSB 11

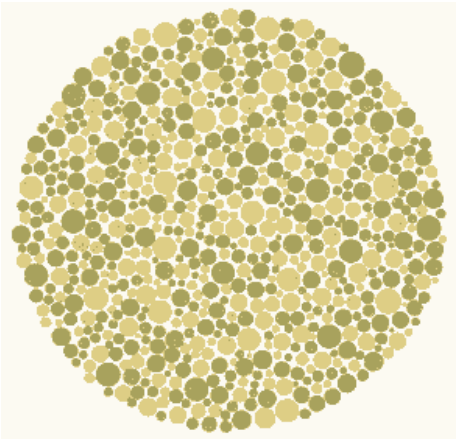
4 Conclusion **12**

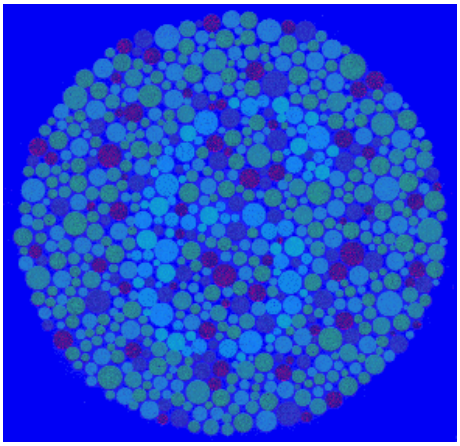
1 Introduction

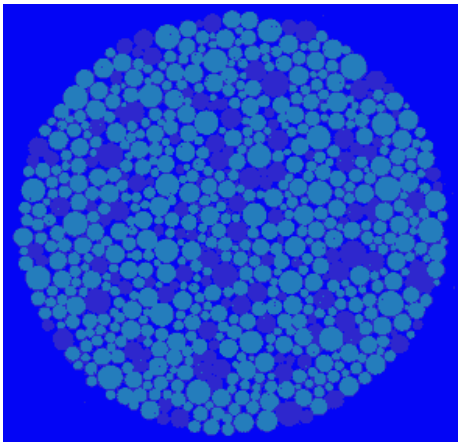
Nous avons vu dans le précédent TP comment étaient représentées les couleurs en informatique. Dans ce TP, nous allons tenter de segmenter des images par classification de pixels.

Nous devrons écrire une Macro ImageJ qui permet de segmenter une image en un nombre donné de classes. Nous segmenterons ensuite d'autres images en utilisant ces classes et observerons les résultats, à la fois dans les espaces RGB et HSB.









2 Segmentation des images par K-Means et discrimination linéaire

2.1 Explication

Le but ici est d'écrire une macro qui permet dans un premier temps de segmenter une image en un nombre donné de classes. Pour ce faire nous utilisons l'algorithme supervisé K-Means grâce au plugin Segmentation > K-Means Clustering fourni.

Nous parcourons ensuite l'image Cluster Centroid Values afin de stocker les valeurs RGB des différentes classes.

Puis nous demandons à l'utilisateur d'ouvrir une nouvelle image.

Ensuite, nous segmentons la nouvelle image en sélectionnant pour chaque pixel la classe dont la couleur est la plus proche de la couleur courante (dans l'espace RGB).

La Macro a été fournie en pièce jointe.

2.2 Résultats

2.2.1 Segmentation par K-means

Nous testons dans un premier temps la macro sur l'image *cas_3_dalton15*, en demandant 3 classes. Nous obtenons l'image segmentée ci-dessous (à droite) ainsi que les valeurs des différentes classes.

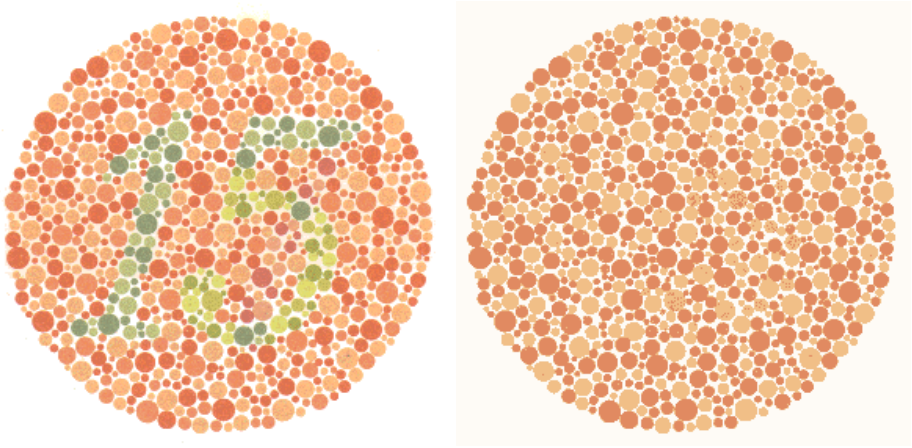


FIGURE 1 – À gauche : Image initiale | À droite : Image segmentée

2.2.2 Classes obtenues

Après analyse de l'image, nous retrouvons les 3 classes de couleurs suivantes :

1. R=254, G=251, B=246
2. R=241, G=191, B=135
3. R=225, G=138, B=97

2.2.3 Segmentation par distance euclidienne

Nous devons maintenant ouvrir une nouvelle image. Nous sélectionnons *cas_3_dalton74* pour la segmenter selon la méthode citée plus haut. Voici le résultat obtenu.

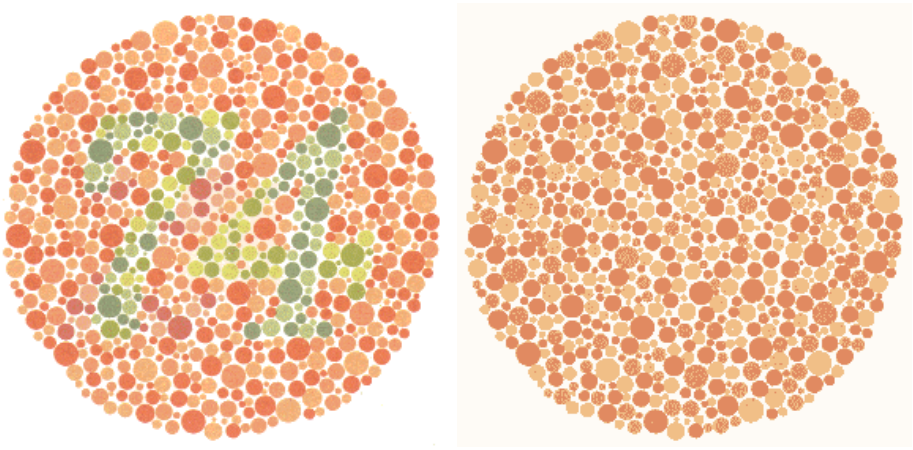


FIGURE 2 – À gauche : Image initiale | À droite : Image segmentée

3 Application aux images *cas_4*

3.1 Explication

Le but de cette partie est d'appliquer la macro codée dans la partie précédente et d'observer les résultats sur le lot d'images *cas_4*, à la fois dans l'espace RGB et HSB.

La première image sélectionnée et segmentée est l'image *cas_4_dalton6*. Nous utiliserons les 3 classes extraites de cette image afin de segmenter les images suivantes.

3.2 Résultats

3.2.1 Image *cas_4_dalton6* dans l'espace RGB

Après classification et segmentation par la méthode K-Means, nous obtenons l'image ci-dessous (à droite).

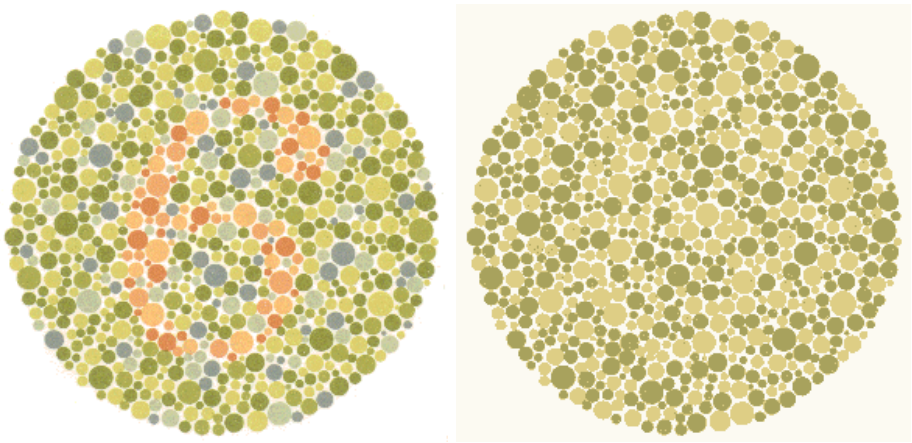


FIGURE 3 – À gauche : Image initiale | À droite : Image segmentée

Les classes obtenues sont les suivantes :

1. R=252, G=250, B=242
2. R=222, G=206, B=133
3. R=168, G=162, B= 93

Nous voyons que les points de couleurs sont pour la plupart bien segmentés, mais nous ne pouvons pas distinguer les chiffres présents initialement.
Nous allons donc tester cet algorithme dans l'espace HSB afin de voir si la segmentation nous donne des résultats plus utiles.

3.2.2 Image *cas_4_dalton6* dans l'espace HSB

Nous analysons les résultats de cette macro dans l'espace HSB.
Ainsi, nous convertissons l'image choisie dans l'espace HSB, nous obtenons l'image ci-dessous à gauche, puis nous utilisons la même méthode que dans la partie précédente. Nous obtenons l'image ci-dessous à droite.

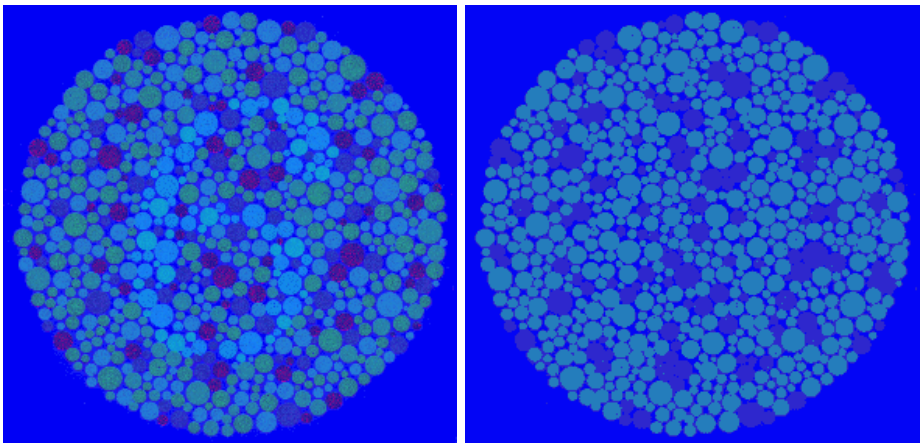


FIGURE 4 – À gauche : Image initiale (HSB) | À droite : Image segmentée

Les classes obtenues sont les suivantes :

1. R= 3, G= 5, B=245
2. R= 46, G= 39, B=205
3. R= 36, G=125, B=188

De même ici, nous ne pouvons pas distinguer les points formants le chiffre 6 des points formants le disque.

3.2.3 Image *cas_4_dalton8* dans l'espace RGB

Après segmentation, nous obtenons l'image suivante :

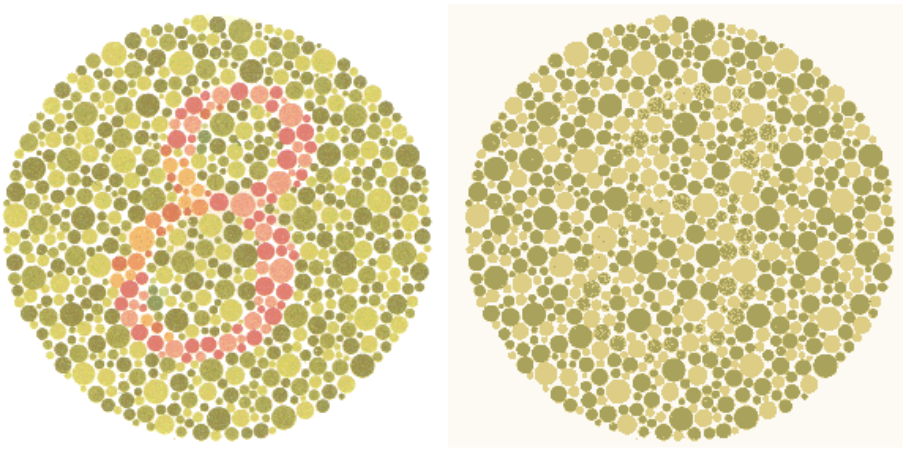


FIGURE 5 – À gauche : Image initiale | À droite : Image segmentée

Nous ne pouvons pas distinguer les points formants le chiffre 8 des points formants le disque.

3.2.4 Image *cas_4_dalton8* dans l'espace HSB

Après conversion de l'image et segmentation, nous obtenons l'image suivante :

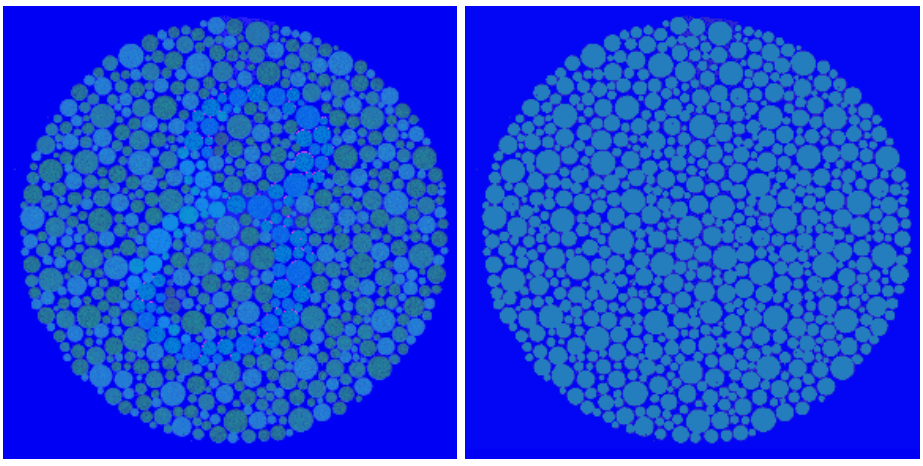


FIGURE 6 – À gauche : Image initiale (HSB) | À droite : Image segmentée

Nous ne pouvons pas distinguer les points formants le chiffre 8 des points formants le disque.

3.2.5 Image *cas_4_dalton29* dans l'espace RGB

Après segmentation, nous obtenons l'image suivante :

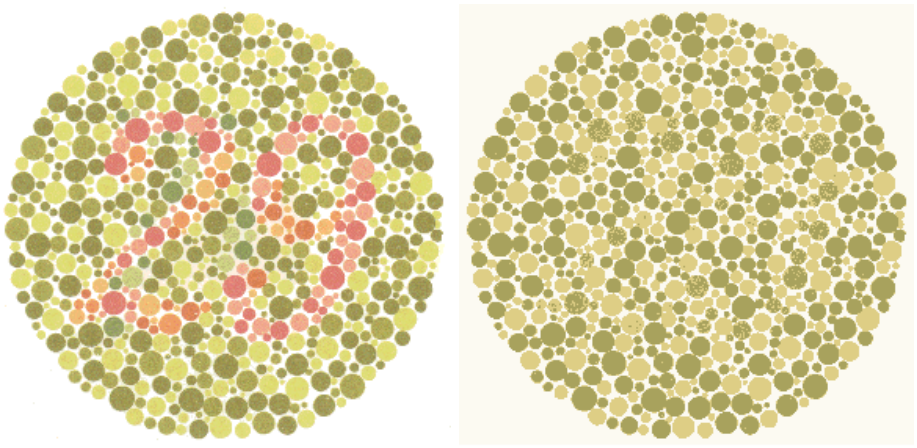


FIGURE 7 – À gauche : Image initiale | À droite : Image segmentée

Nous ne pouvons pas distinguer les points formants le nombre 29 des points formants le disque.

3.2.6 Image *cas_4_dalton29* dans l'espace HSB

Après conversion de l'image et segmentation, nous obtenons l'image suivante :

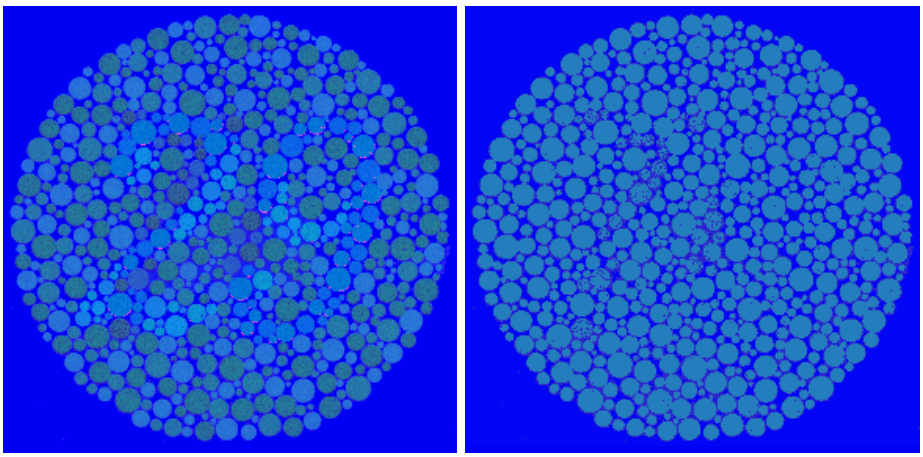


FIGURE 8 – À gauche : Image initiale (HSB) | À droite : Image segmentée

Nous ne pouvons pas distinguer les points formants le nombre 29 des points formants le disque.

3.2.7 Image *cas_4_dalton45* dans l'espace RGB

Après segmentation, nous obtenons l'image suivante :

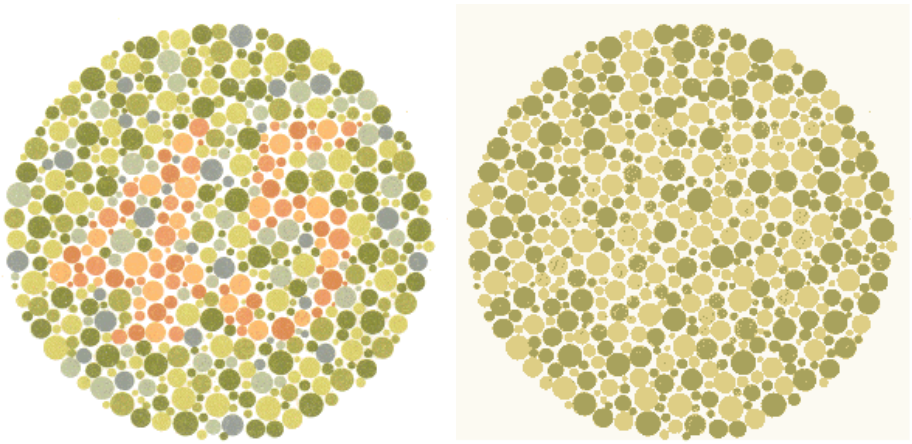


FIGURE 9 – À gauche : Image initiale | À droite : Image segmentée

Nous ne pouvons pas distinguer les points formants le nombre 45 des points formants le disque.

3.2.8 Image *cas_4_dalton45* dans l'espace HSB

Après conversion de l'image et segmentation, nous obtenons l'image suivante :

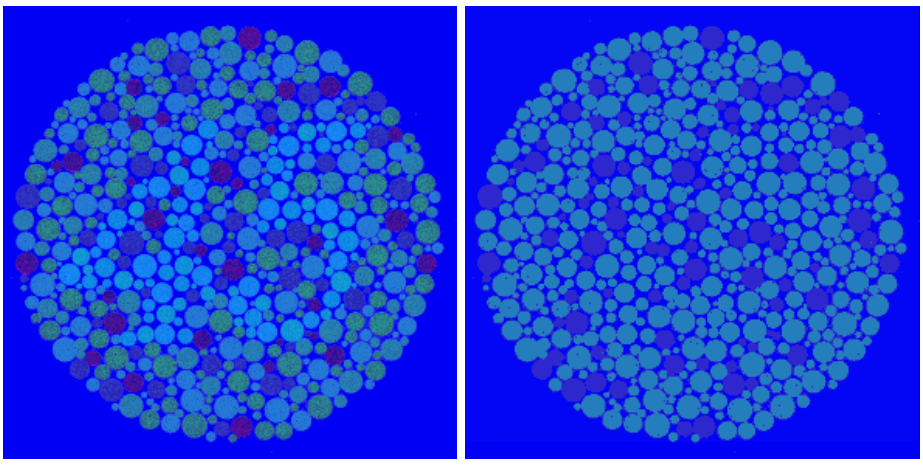


FIGURE 10 – À gauche : Image initiale (HSB) | À droite : Image segmentée

Nous ne pouvons pas distinguer les points formants le nombre 45 des points formants le disque.

3.2.9 Image *cas_4_dalton_ligne* dans l'espace RGB

Après segmentation, nous obtenons l'image suivante :

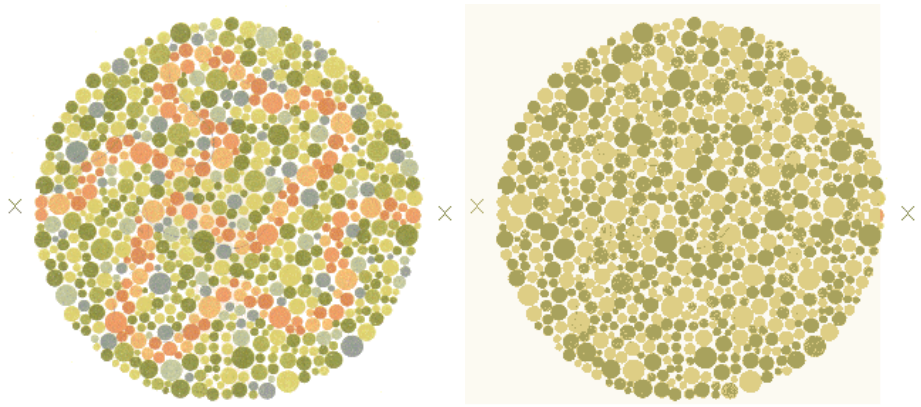


FIGURE 11 – À gauche : Image initiale | À droite : Image segmentée

Nous ne pouvons pas distinguer les points formants la ligne des points formants le disque.

3.2.10 Image *cas_4_dalton_ligne* dans l'espace HSB

Après conversion de l'image et segmentation, nous obtenons l'image suivante :

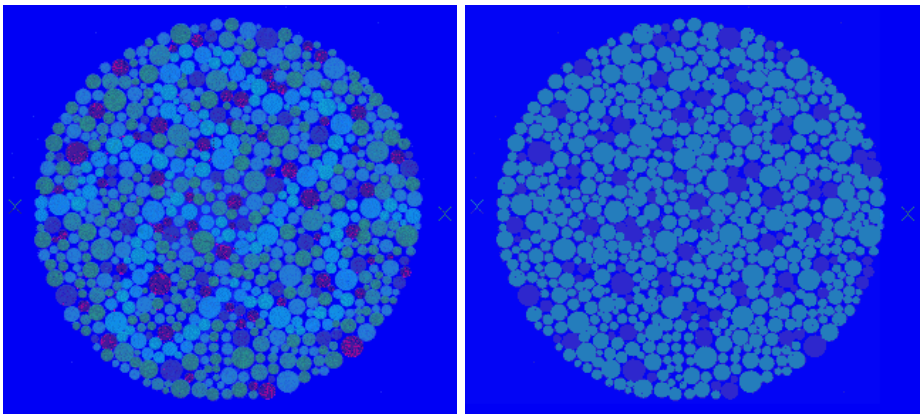


FIGURE 12 – À gauche : Image initiale (HSB) | À droite : Image segmentée

Nous ne pouvons pas distinguer les points formants la ligne des points formants le disque.

4 Conclusion

Après avoir défini ce qu'étaient les couleurs et leurs représentations, nous avons pu segmenter des images en un nombre donné de classes en utilisant l'algorithme K-Means Clustering.

Cet algorithme nous a permis – une fois spécifié le nombre de classes voulues – d'obtenir les couleurs de chaque classes. Une fois ces couleurs stockées, nous avons pu segmenter des images similaires en récupérant pour chaque pixel, la classe la plus proche de sa couleur.

Nous observons ici que les points formant les disques sont bien segmentés. Cependant, nous ne pouvons plus distinguer les nombres à l'intérieur de ces disques. Et ce aussi bien dans l'espace RGB que dans l'espace HSB.

Par conséquent, bien que cette méthode de segmentation soit efficace pour dissocier le fond et le disque, elle ne nous permet pas de segmenter l'image dans sa totalité.