Matias Amaglio	Alec Basset	Maëlle Bitsindou	Inès Jung	Célie Ponroy

Rapport SAE 4.1

Détection de biomes sur des exoplanètes

1. Prétraitement de l'image

Nous avons tout d'abord créé une interface InterfaceFlou qui possède la méthode *flou()* qui implémentera les classes FlouMoyenne et FlouGaussien.

1.1. Flou par moyenne

Initialisation :

- Attribut int taille qui permet de choisir la taille de la matrice de pixels

Méthode *flou()* :

Pour chaque pixel, on crée une matrice de la taille choisie, qui contiendra les pixels voisins du pixel courant. On récupère ensuite les couleurs de chaque pixel (la valeur rouge, vert, bleu). On fait la moyenne de ces valeurs pour créer un nouveau pixel qui sera de la couleur moyenne de ses voisins.

Dans la classe FlouMoyenne, chaque pixel voisin a la même importance.

1.2. Flou Gaussien

Initialisation :

- Attribut int taille qui permet de choisir la taille de la matrice de pixels
- Attribut double sigma qui sera utile pour le calcul
- Attribut final matrice3x3 et final matrice5x5 avec les valeurs d'importance de chaque pixel de la matrice

Méthode *flou()* :

On calcule les distances entre le pixel du centre et l'axe horizontal et entre le pixel du centre et l'axe vertical. On peut ensuite appliquer la formule de Gauss. Après cela, on récupère les valeurs rouge, verte et bleu pour calculer la moyenne de la couleur du nouveau pixel.

2. Détection et visualisation des biomes

2.1 Détéction des groupes de pixels de couleur similaire

Algorithme KMeans:

Nous avons choisi de commencer par implémenter l'algorithme de **KMeans** car c'est un algorithme rapide pour un grand nombre de données. De plus, le fait que KMeans ne soit pas résistant au bruit ne pose pas de réel problème avec les couleurs qui serait plus présent avec les coordonnées.

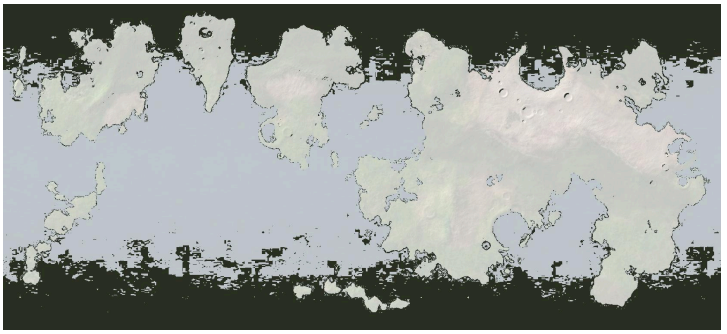
La métrique choisie que nous avons choisie a été CIELAB pour son efficacité, même si c'est un calcul un peu lourd. Nous avons pu nous le permettre car l'algorithme KMeans est rapide. Nous avons tout de même testé les trois normes proposées.

Voici nos résultats des différentes métriques:

Image de base:

Métrique RedMean:

RedMean s'est avéré être la plus rapide mais les résultats n'ont pas été assez précis. Les bordures vertes foncées devraient être bleu clair.



Métrique CIELAB:

CIELAB a été un peu plus lente mais les résultats ont été beaucoup plus satisfaisants.

Métrique Norme 94:

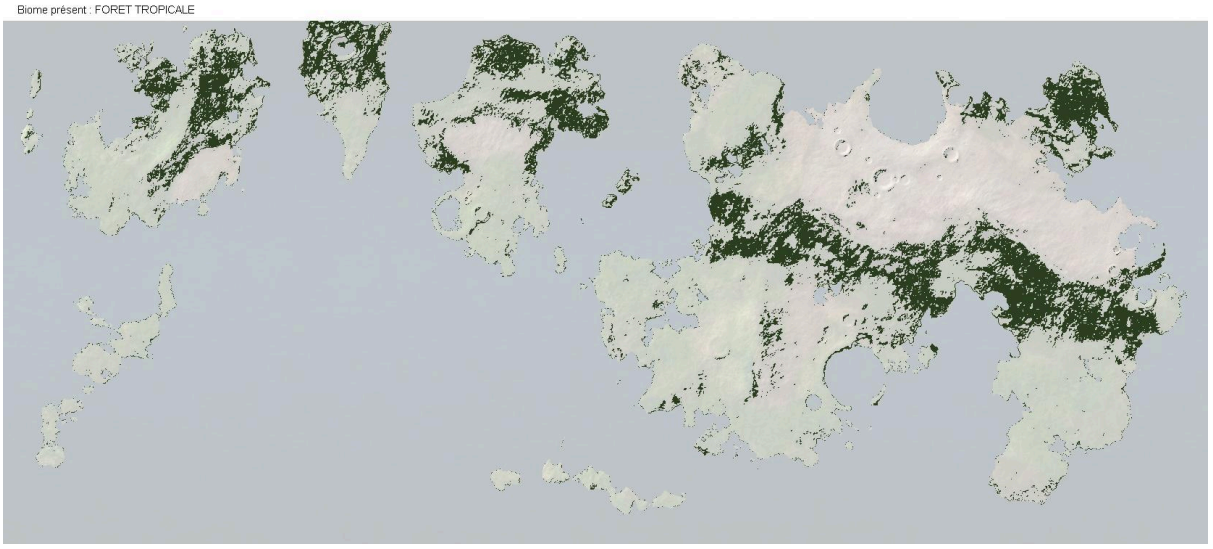
La norme 94 a été beaucoup plus lente que CIELAB (environ 3 min sur un ordinateur portable) et la différence de biome ne se voit pas à l'œil nu.

Pour ce qui concerne le nombre de clusters, nous avons défini une palette de couleurs qui prend en compte tous les biomes, puis le nombre de clusters qui est le nombre de couleurs contenus dans cette palette afin de pouvoir toutes les détecter. Lors de l'utilisation de notre application une palette de couleurs par défaut de 10 couleurs est proposée, mais selon l'image l'utilisateur peut prendre une autre palette plus adaptée (changement du nombre de couleurs ou des teintes).

2.2. Etiquetage des biomes

Nous avons ensuite ajouté une méthode *afficherBiomeImage()* qui permet d'afficher la légende des différents biomes présents sur l'image passée en paramètre. Nous avons également une liste d'entiers, généré par l'algorithme **KMeans** qui associe chaque pixel de l'image à une couleur.

Voici le résultat obtenu avec la légende:



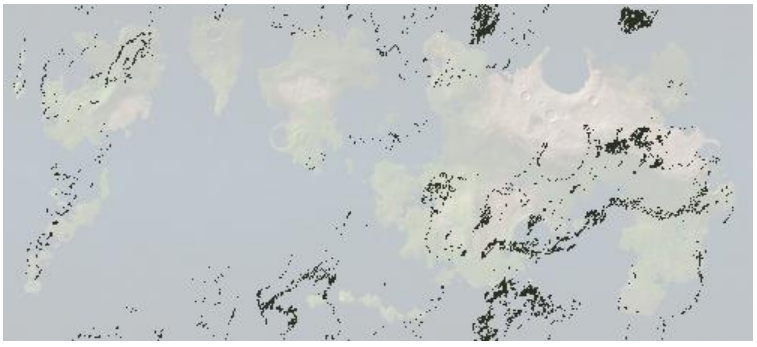
3. Détection et affichage des écosystèmes pour chaque biome

Dans un deuxième temps, nous avons choisi d'implémenter l'algorithme DBScan car il est plus rapide d'exécution que l'algorithme HAC.

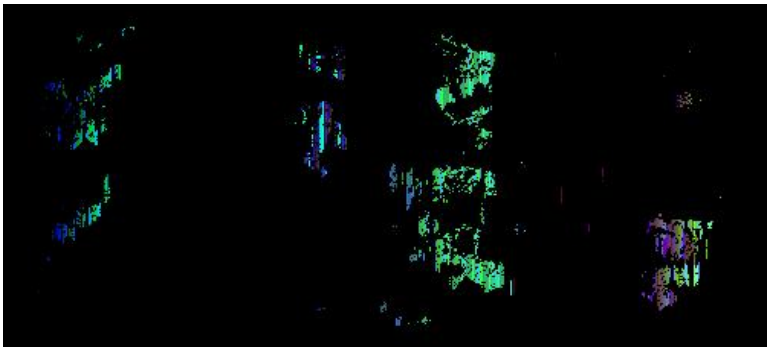
Néanmoins, nous avons rencontré quelques difficultés lors de son implémentation. L'algorithme fonctionne sur des très petites données (avec peu de pixels) mais avec de plus grandes données, il n'est pas fonctionnel. De plus, cela crée des clusters en ligne et non les résultats attendus.

Voici la première version de DBScan :

Tous les points (en vert foncé) sont dans le même cluster.Nous nous sommes alors demandés si c'est un problème de paramètre ou d'algorithme.



Sur cette deuxième version, on constate que les clusters sont en ligne. Il y a un nombre important de clusters (sur cet exemple plus de 200). Cela n'est pas dû à un mauvais paramétrage car la forme des clusters serait pas si inhabituelle. Le problème rencontré serait soit un problème dans l'algorithme soit un problème dans le traitement des données.



Pour cette étape, nous avons créé une méthode permettant d'avoir automatiquement une liste de couleurs assez grande pour pouvoir afficher tous les clusters calculés. Grâce à cela nous pouvons afficher jusqu'à 250 clusters si besoin. Cette méthode est modulable et peut être facilement modifiée pour avoir plus ou moins de couleurs.

Conclusion de cette SAE:

Les résultats obtenus montrent que l'algorithme KMeans est performant pour la détection des biomes sur des exoplanètes. Cet algorithme, simple à implémenter et rapide à exécuter, s'est avéré efficace pour segmenter les zones de la carte en clusters de couleur distinctes, permettant une identification claire des biomes.

Toutefois, des difficultés ont été rencontrées avec l'algorithme DBSCAN. L'implémentation de cet algorithme a posé des défis, notamment dans le réglage des paramètres, ce qui a complexifié son utilisation.

En conclusion, pour des raisons de temps nous n'avons pas pu tester plusieurs algorithmes pour chaque type de clusterisation, nous aurions aimé pouvoir tester et comparer nos deux algorithmes sur les mêmes tâches.