RAPPORT INTERMÉDIAIRE

MEso-scale convective System over OCEAN

Équipe du Projet : KADJO MAMI Claude Yrvine-Axel, SEDDIK Emna, SATTARI Montassar, MARTIN Pierre-Jean, AGUDELO Santiago

Encadrants Techniques : BILLOT Romain, REBAI Issam

Clients Partenaires : DUPONT Paco et MESSAGER Christophe de chez EXWEXs

Rapport rédigé par l'équipe du projet n°14

À l'intention de la société EXWEXs

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Version 1.0
Le 7 mai 2019

Table des matières

I. Introduction

I.1 Contexte

Le projet *MEso-scale convective System over OCEAN* se place dans un contexte scolaire, durant le 4ème semestre d'école d'ingénieur (ici l'IMT-Atlantique). D'un point de vue purement scolaire, ce projet a pour but de sensibiliser les étudiants aux projets professionnel futurs, et de leurs apporter les compétences indispensables à la réalisation de ceux-ci (Team Work, maîtrise des délais, apprentissages de nouveaux outils).

En lui-même, le projet consiste en la réalisation d'une application permettant de détecter les amas d'orages (dis Systèmes Convection, SCs) à partir de données contenant la vitesse des vents, et inversement (c'est à dire prévoir la vitesse des vents en fonction de données contenant la présence de SCs). Se référer au *Plan De Management* (*PDM*) pour plus d'informations relatives au projet (*Cahier des charges* par exemple).

Ce document, appelé *Rapport Intermédiaire*, à pour but de présenter à la société cliente *Extreme Weather Expertises* (*EXWEXs*) l'avancée du projet et les techniques utilisées jusque là. Nous verrons en détail comment les données ont été extraites, comment elles ont été traitées et mises en relation.

I.2 Outils et Librairies

Afin de mener à bien ce projet, les étudiants possèdent une base de données contenant des photos prises par deux satellites différents :

- un satellite à orbite géostationnaire, dont les images captées ont une résolution d'environ 3 kilomètres et sont fournis toutes les 15 minutes. Ces images représentent les systèmes convectifs, et leur intensité (ils sont classés par catégories). Les images captées par ce satellite seront appelées i*mages géostationnaires* ;
- un satellite à orbite défilant, qui nous fournit des images de plus haute résolution (100 à 300 mètres) et permet de suivre la vitesse et la direction des vents à un instant donné. La période de prélèvement de ce type d'images est un peu aléatoire. Les images captées par ce satellite seront appelées i*mages snowbird.*

Ces images sont rangées par groupe de trois (ou quatre) comprenant chacun :

- 2 ou 3 images géostationnaires, espacées chacune de 15 minutes ;
- 1 image snowbird dont la date de prise se situe entre les dates des autres images géostationnaires.

Chaque image d'un même dossier présente donc la même situation, avec au maximum un décalage de 15 minutes.

Afin de traiter ces images et de réaliser des opérations dessus, nous utiliserons le langage de programmation *PYTHON* (v3.6). Les modules utilisés sont les suivants : *numpy*, *matplotlib.pyplot*, *sklearn*, *snappy*, *sys* et *os*.

II. Extraction des données

II.1 Types de données

Le format des images fournies est le NetCDF. Ce format est usuel pour la visualisation des images satellitaires. Ce format permet de superposer des données différentes sur un même pixel : sur les images snowbirds par exemple, un pixel comprends la force des vents, la position des géographiques (latitude & longitude) et l'angle d'azymut du vent associé à la position. Afin de visualiser ces données simultanément, on peut utiliser le logiciel *Sentinel Application Platform* (*SNAP*).

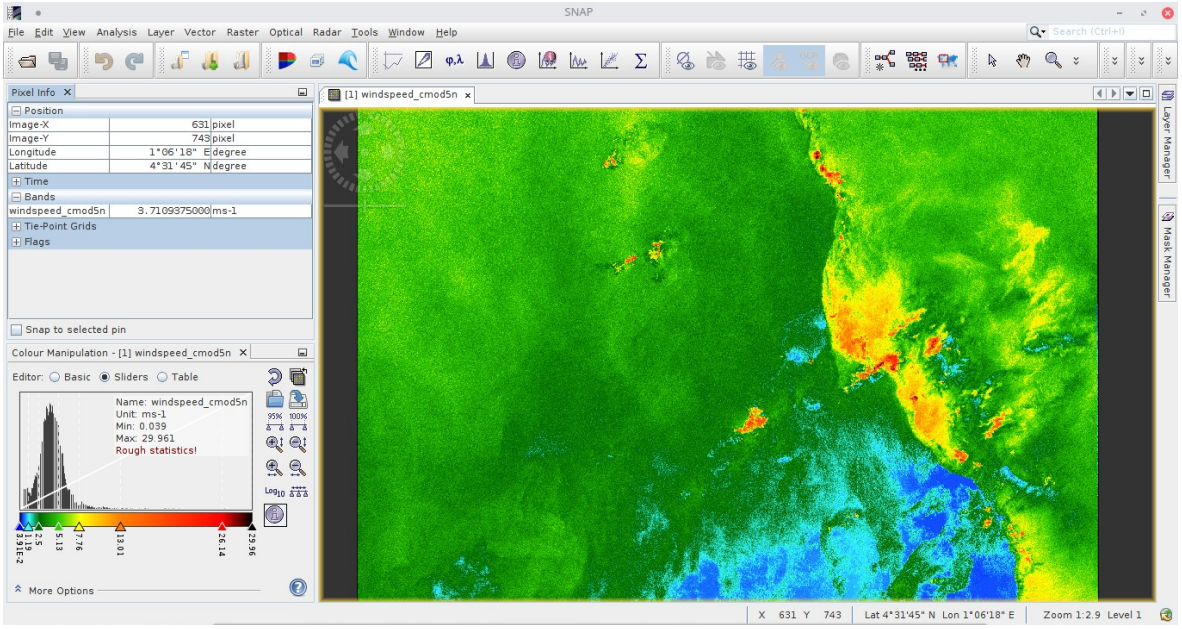


Illustration 1: Capture d'écran du logiciel SNAP. On peut lire les différentes données conservées dans un pixel dans l'encadré supérieur gauche.

Le module *snappy* de *PYTHON* permet de manipuler ce type de fichier et notamment d'extraire chaque type de données conservées dans les images, et donc de les sauvegarder sous d'autres formats.

II.2 Méthode d'extraction et de stockage

La méthode *getBand(str)* permet de récupérer le label des données, tandis que la méthode *readPixel()* permet de parcourir les pixel de l'image et d'en récupérer les données dont le label est celui choisit. Elles peuvent alors être stockées dans un tableau à deux dimensions, dont la position (i, j) correspond au pixel correspondant sur l'image (ligne, colonne). Il est alors très simple d'accéder aux valeurs contenu dans chaque pixel de l'image.

Nous avons choisi d'extraire les champs *vitesse des vents*, *longitude* et *latitude* pour les images snowbirds, et *catégories de Sc*, *latitude* et *longitude* pour les images géostationnaires. Afin de limiter les temps de calculs, ces données extraites sont stockées sous forme de fichiers txt (contenant une liste *PYTHON* exploitable grâce à la fonction *loadtxt(nomFichier)* de la bibliothèque *Numpy*).

Ces fichiers sont stockées dans des dossiers précis et nommés explicitement comme suit

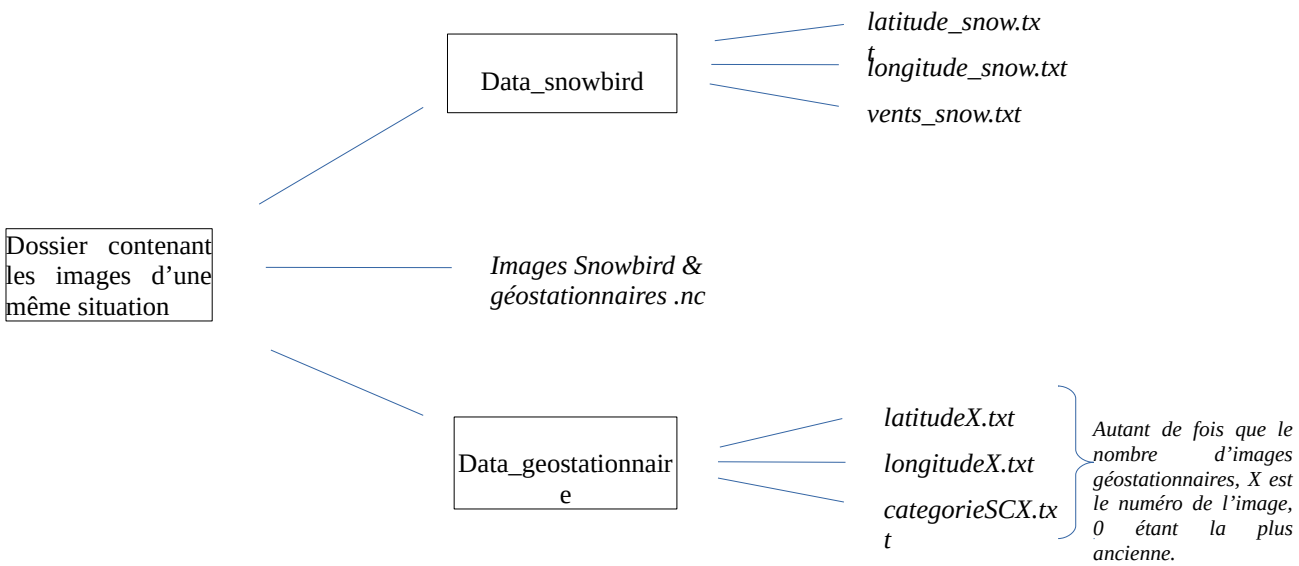


Figure 1: Organisation des dossiers

Ainsi, lorsque nous effectuerons des opérations sur les images, nous utiliserons les fichiers *.txt* relatifs aux données de l'image.

Notons que lors de ce processus, les images géostationnaires sont redimensionnées, et sont centrées sur une fenêtre montrant les même points géographiques que les image snowbird. Les images géostationnaires montrant une zone plus vaste, le temps de calcul est ainsi réduit.

II.3 Représentation via PYTHON

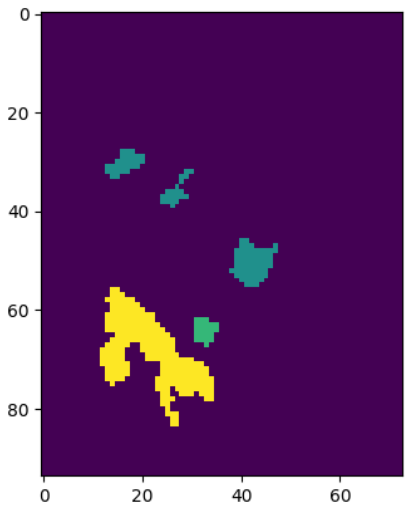


Illustration 2: Image géostationnaire représentée avec PYTHON

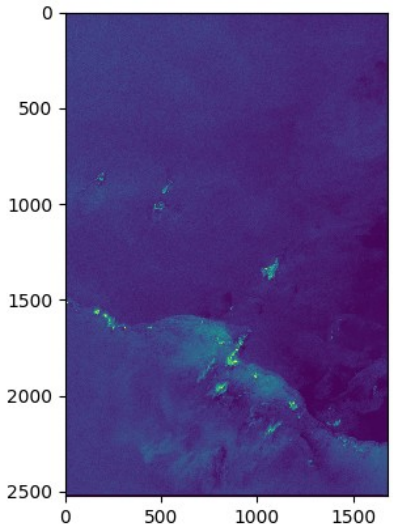


Illustration 3: Image Snowbird représentée avec PYTHON

Après l'extraction des données reçues, et en utilisant la fonction *plot* de la librairie *matplotlib.pyplot* de *PYTHON*, nous obtenons l'image de *l'illustration 2* pour les images provenant du satellite géostationnaires.

Les différentes couleurs représentent les différents catégories de systèmes convectifs (SCs) détectés autour de la localisation photographiée.

L'illustration 2 montre une image provenant du snowbird avec les différents vents observés.

L'objectif est maintenant de déterminer les vents présents sous ces différents SCs, et les SCs présent au dessus de ces vents.

III. Traitement des Images du satellite géostationnaire

III.1 Clustering

Par *clustering*, nous entendons ici une méthode de classification des éléments à notre disposition. Un *clustering* des SCs est par exemple la classification des SCs en fonction de leur catégorie et leur étendue. Pour les vents, on parlera de classification en fonction de la vitesse de ceux-ci.

Effectuer un clustering sur toute *l'illustration 2* ne pourrait nous donner ni des informations exactes sur les vents induits des différents SCs, ni le nombre de classe nécessaire pour cette classification, d'autant plus qu'elle varie en fonction de l'image. Ainsi la solution trouvée est de déterminer les coordonnées exactes de chaque SC sur *l'illustration 2* et de trouver la zone correspondantes sur *l'illustration 3* afin de déterminer les vents correspondant au SC.

La détection de la position de chacun des SCs est basées sur une détection proche en proche de chaque catégorie et une sauvegarde des données de localisation (*latitude*, *longitude*). Nous découpons ainsi chacun de nos SCs afin de les obtenir séparément, et les superposer aux images

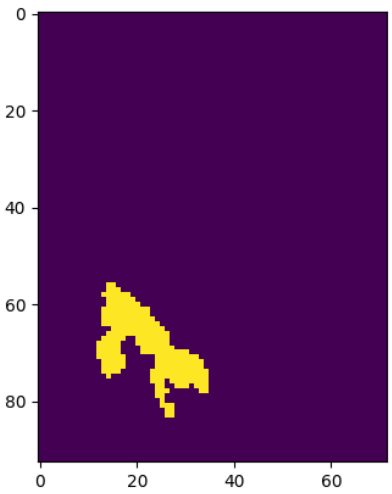


Illustration 5: Exemple de clustering sur les SCs

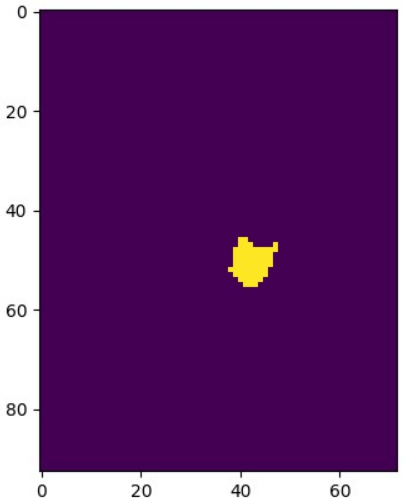
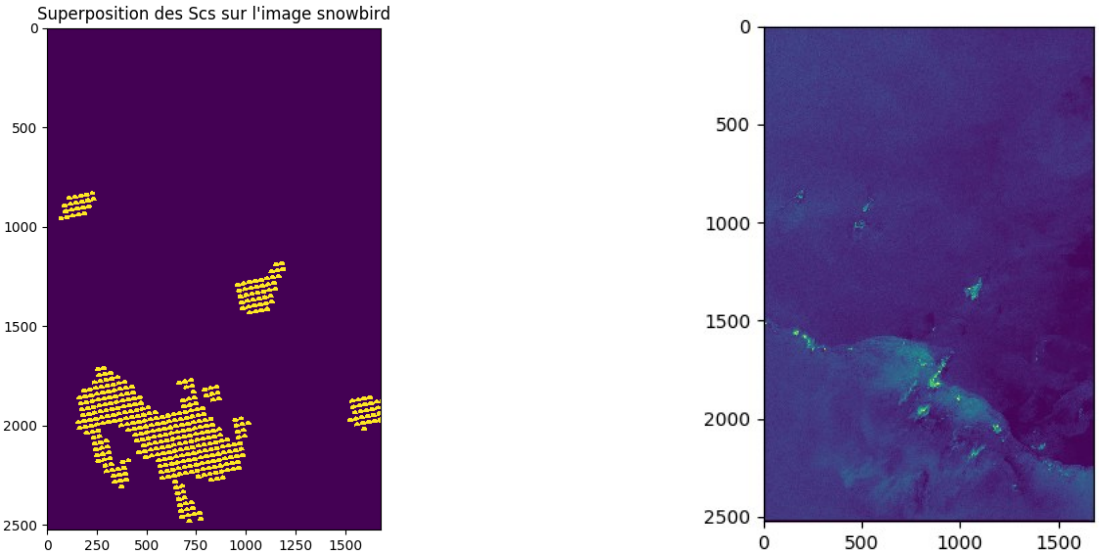


Illustration 4: Exemple de clustering sur les SCs

Chaque image représente un et un seul SC comme s'il était le seul à exister. La superposition consiste alors en ne représentant que les pixels où se situent un SCs sur les images Snowbird (en matchant les coordonnées *latitude & longitude*).

III.2 Superposition & Comparaison



Les résultats ne sont pas très significatifs et peuvent être améliorés. La difficulté réside dans le fait de passer d'une grande distance (c'est-à-dire peu de précision) à une plus petite échelle. Le résultat donne quelque chose de peu précis.

IV. Traitement des images du satellite Snowbird

IV.1 Clustering

On cherche à regrouper les données de vent à travers les algorithmes de clustering suivants. Pour chaque méthode, le principe est présenté suivit des résultats obtenus pour une image de test. Les algorithmes sont tous disponibles dans la librairie *sklearn* de Python, que nous avons utilisée pour les tests présentés dans ce document.

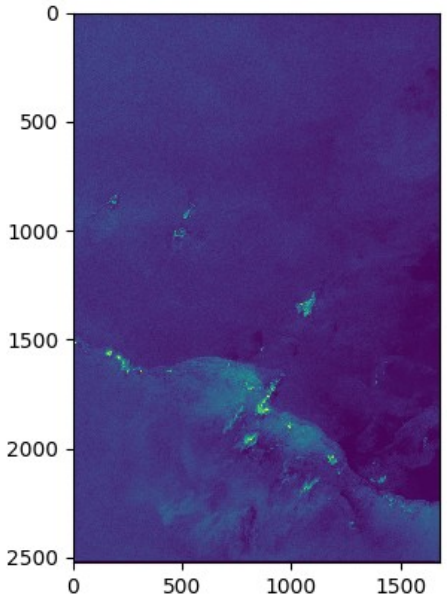


Illustration 6: Image de test. Les zones en jaune correspondent aux vitesses du vent les plus élevées.

IV.1.1 K- means

Un des algorithmes les plus utilisés pour effectuer un *clustering* est le *K-means* (*K-moyennes*). Cet algorithme divise un ensemble de N échantillons *X* en K clusters. Chacun de ces clusters est défini par la moyenne μ_j des échantillons lui appartenant. Cette moyenne est également appelée le *centroid* du cluster, et elle est choisie de façon à ce qu'elle minimise le critère quadratique suivant:

$$\sum_{i=0}^{n} min_{\mu_j \in C} \left\| x_i - \mu_j \right\|^2$$

Ce critère constitue une mesure de la cohérence interne des clusters. Son utilisation fait cependant l'hypothèse de clusters isotropes et convexes, et sa performance devient faible lorsque cette hypothèse n'est pas satisfaite.

Les étapes de l'algorithme des K-moyennes sont les suivantes :

- Initialisation des centroids ;
- Associer chaque échantillon au centroid le plus proche ;
- Les clusters sont formés par les échantillons ayant le même centroid. Une fois les clusters formés, on calcule à nouveau le centroid et on itère jusqu'à convergence.

L'algorithme converge vers un minimum local du critère et il est sensible au choix initial des centres. Des méthodes existent pour s'assurer que les centres initiaux sont suffisamment lointains entre eux, ce qui donne de meilleurs résultats qu'une initialisation aléatoire.

Après exécution de l'algorithme K-means pour 4 clusters on obtient le résultat affiché ci-dessous. On constate que les zones dont la vitesse associée est élevée ont été regroupées dans le même cluster.

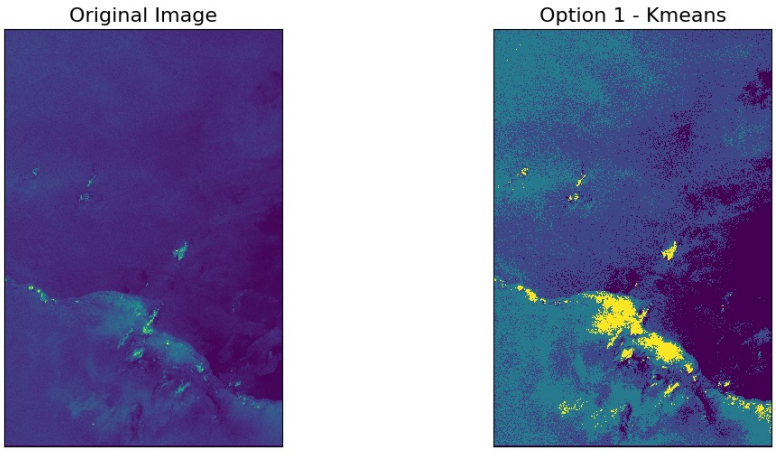


Figure 2. Clustering en utilisant l'algorithme K-means.

IV.1.2 Mini Batch K-Means

Le *Mini Batch K-Means* est une variante de l'algorithme des K-means qui vise à réduire le temps de simulation requis. Un Mini Batch est un sous-ensemble de données d'entrées qui est aléatoirement échantillonné lors de chaque itération. Les étapes clés de cet algorithme sont les suivantes :

- Prise aléatoire de *b* échantillons de l'ensemble des données d'entrées pour constituer un Mini Batch ;
- Assigner chaque échantillon du Mini Batch au centroid le plus proche ;
- Pour mettre à jour le centroid, on ne considère que la moyenne des échantillons du Mini Batch ayant été assignées à ce centroid. Toutes les données ne sont donc pas prisent en compte, ce qui accélère ainsi le calcul.

Bien que cette méthode aboutit à des résultats de moins bonne qualité que la méthode des K-moyennes, cette différence est en pratique négligeable.

Le résultat obtenu après exécution de cet algorithme est affiché ci-dessous (on reste sur le même nombre de clusters que dans le cas précédent: 4). On constate que le clustering est similaire à celui obtenu en utilisant l'algorithme K-means, ce qui était attendu d'après ce qui a été mentionné antérieurement.

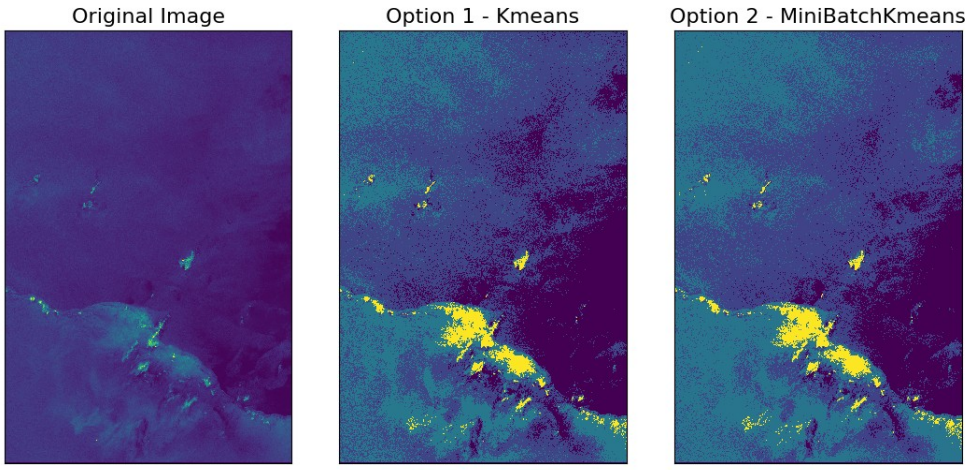


Figure 3. Clustering en utilisant l'algorithme Mini Batch K-means et comparaison avec l'algorithme K-means..

IV.1.3 Birch

Cet algorithme construit un arbre appelé CFT (Characteristic Feature Tree). Dans un tel arbre, les données sont comprimées dans un ensemble de nœuds (nœuds CF) qui contiennent un certain nombre de sous-clusters (sous-clusters CF). Ces sous-clusters contiennent l'information nécessaire au clustering des données, à savoir :

- Nombre d'échantillons dans le sous-cluster ;
- Somme linéaire des échantillons ;
- Somme de la norme L2 des échantillons ;
- Centroid du sous-cluster ;

- Norme au carré des centroids ;

Le seuil et le facteur de ramage sont les paramètres à régler dans cet algorithme.

Le seuil limite la distance entre l'échantillon entrante et les sous-clusters existants, et le facteur de ramage limite le nombre de sous-clusters pouvant exister dans un nœud. Si on cherche à réduire le nombre d'instances des données ou si on souhaite un nombre élevé de sous-clusters, l'algorithme Birch devient particulièrement adapté.

On a fait tourner l'algorithme Birch pour 4 clusters en obtenant les résultats ci-dessous pour *seuil=0.05* et *facteur de ramage=10.* On remarque qu'un grand clusters correspondant aux zones à faible vitesse a été formé et les zones à grande vitesse ont été regroupées dans les trois clusters restants. Le niveau de détail que cet algorithme permet d'atteindre dans ces dernières zones est significativement plus élevé que celui obtenu avec les autres algorithmes considérés. Il faut constater que les résultats s'avèrent très sensibles aux variations des paramètres, d'où l'importance de bien les choisir.

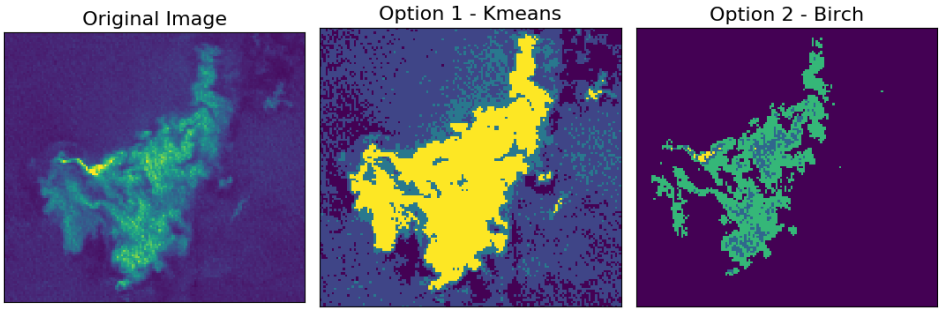
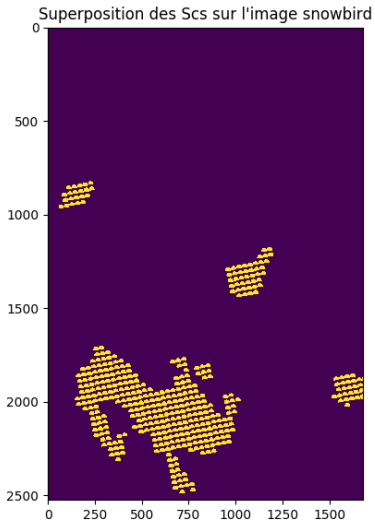


Figure 4. Clustering en utilisant l'algorithme Birch.

IV.2 Superposition & Comparaison

L'utilisation de K-Means n'a pas encore été utilisé pour superposer les vents et systèmes convectifs. À la place, nous avons simplement itéré sur chaque pixel des images du snowbird, repéré la position géographique ainsi que le la vitesse du vent. Pour les vents d'une vitesse supérieure à 10 m.s-1, nous avons regardé si un système convectif se situait au dessus. Voici le résultat :



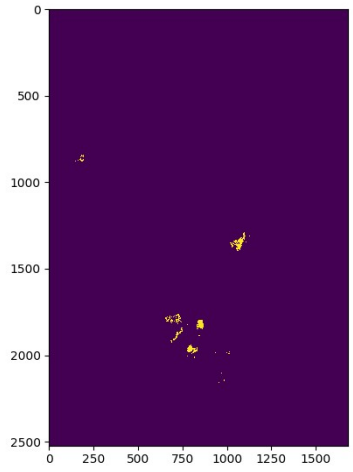


Illustration 7: Superposition SCs/ vents pour des vents > 10 m.s-1

Les résultats montrent d'une part qu'un fort vent ne veut pas forcément dire qu'un SC est présent au dessus, et d'autre part que le décalage spatio-temporel va jouer un rôle important dans nos déterminations.

V. Conclusion

En conclusion, le projet avance comme prévu, nous sommes désormais capable de matcher des positions sur les deux types d'images, de prévoir si oui ou non des vents de fortes intensités et des SCs se superposent. Nous sommes également capable de catégoriser et de découper nos données afin de mieux les exploiter.

Nous allons désormais nous concentrer sur les conditions d'apparition des Systèmes Convectifs et des vents de fortes intensités afin de corréler ces deux données, et de prévoir l'apparition de l'un en fonction de la présence (ou non) de l'autre.