

ECOLE POLYTECHNIQUE DE L'UNIVERSITÉ FRANÇOIS RABELAIS DE TOURS

Département Informatique

64 avenue Jean Portalis

37200 Tours, France

Tél. +33 (0)2 47 36 14 14

polytech.univ-tours.fr

Projet Recherche & Développement

2017-2018

Développement d'un outil d'analyse de dunes fluviales

Tuteurs académiques

Stéphane RODRIGUES

Pascal MAKRIS

Jean-Yves RAMEL

Étudiant

Kévin ECALLE (DI4)



Liste des intervenants

Nom	Email	Qualité
Kévin ECALLE	kevin.ecalle@etu.univ-tours.fr	Étudiant DI4
Stéphane RODRIGUES	stephane.rodriques@univ-tours.f	Tuteur académique, Département Aménagement et Environnement
Pascal MAKRIS	makris.pascal@univ-tours.fr	Tuteur académique, Département Informatique
Jean-Yves RAMEL	ramel.jean-yves@univ-tours.fr	Tuteur académique, Département Informatique



Avertissement

Ce document a été rédigé par Kévin Ecalte susnommé l'auteur.

L'Ecole Polytechnique de l'Université François Rabelais de Tours est représentée par Stéphane Rodrigues, Pascal Makris et Jean-Yves Ramel susnommés les tuteurs académiques.

Par l'utilisation de ce modèle de document, l'ensemble des intervenants du projet acceptent les conditions définies ci-après.

L'auteur reconnaît assumer l'entière responsabilité du contenu du document ainsi que toutes suites judiciaires qui pourraient en découler du fait du non respect des lois ou des droits d'auteur.

L'auteur atteste que les propos du document sont sincères et assument l'entière responsabilité de la véracité des propos.

L'auteur atteste ne pas s'approprier le travail d'autrui et que le document ne contient aucun plagiat.

L'auteur atteste que le document ne contient aucun propos diffamatoire ou condamnable devant la loi.

L'auteur reconnaît qu'il ne peut diffuser ce document en partie ou en intégralité sous quelque forme que ce soit sans l'accord préalable des tuteurs académiques et de l'entreprise.

L'auteur autorise l'école polytechnique de l'université François Rabelais de Tours à diffuser tout ou partie de ce document, sous quelque forme que ce soit, y compris après transformation en citant la source. Cette diffusion devra se faire gracieusement et être accompagnée du présent avertissement.



Pour citer ce document

Kévin Ecalte, *Développement d'un outil d'analyse de dunes fluviales*, Projet Recherche & Développement, Ecole Polytechnique de l'Université François Rabelais de Tours, Tours, France, 2017-2018.

```
@mastersthesis{
  author={Ecalte, Kévin},
  title={Développement d'un outil d'analyse de dunes fluviales},
  type={Projet Recherche \& Développement},
  school={Ecole Polytechnique de l'Université François Rabelais de Tours},
  address={Tours, France},
  year={2017-2018}
}
```


Table des matières

Liste des intervenants	a
Avertissement	b
Pour citer ce document	c
Table des matières	i
Table des figures	iv
Introduction	1
1 Cadre du projet	2
1 Contexte	2
2 Morphologie d'un litage	3
3 Méthodes de relevés utilisées	3
2 Présentation du projet et ses contraintes	5
1 Objectifs du projet	5
2 Fonctionnalités de l'application	6
3 Fichier en sortie du programme	7
4 Environnement	8
5 Profils des utilisateurs	8
6 Caractéristiques des images à traiter	8
7 Bases méthodologiques	11

I	Recherche	14
3	État de l'art : analyse d'image d'un litage	15
1	Reconnaissance par textures	15
1.1	Types de texture.....	15
1.2	Matrice de co-occurrence.....	16
2	Usage de la transformée de Fourier	17
3	Ligne de partage des eaux	19
4	Reconnaissance par apprentissage.....	21
4.1	Apprentissage supervisé.....	21
4.2	Apprentissage non supervisé	21
4.3	Apprentissage par renforcement.....	21
4	Les technologies utilisables	22
1	JAVA	22
2	C++	22
3	Python.....	22
4	Autres bibliothèques	23
5	Blocs applicatifs en Python	24
1	Calculer la transformée de Fourier 2D d'une image	24
2	Chargement d'une image et son affichage sous Tkinter.....	25
3	Découpage d'une image en fonction d'une taille indiquée	25
4	Tracer un segment sur une image	26
5	Récupérer les caractéristiques de l'image	27
II	Développement	28
6	Création des interfaces	29
1	Menu principal	29
2	Traitement des axes	30
3	Visualisation du profil d'un axe.....	31
7	Gestion des axes	32
1	Placer les axes	32
2	Gestion des axes.....	33
8	Détection des dunes sur un axe	35
1	Transposer les axes sur l'image d'origine.....	35
2	Déterminer les pixels de l'image à prendre	36
3	Reconnaître des dunes.....	38

4	Export texte des résultats	39
5	Le traitement sur l'image complète	41
9	Programme de génération d'image TIF 16 bits	42
10	Gestion de projet	44
	Conclusion	47
	Annexes	48
A	Guide développeur	49
B	Installation	51
	Webographie	53

Table des figures

1 Cadre du projet

1	Image à traiter	2
2	Variation d'un litage en fonction du courant.....	3
3	Fonctionnement de l'Écho-sondeur	4

2 Présentation du projet et ses contraintes

1	Croquis de l'interface de l'applicatif.....	5
2	Diagramme de cas d'utilisation	6
3	Exemples de la structure du fichier de sortie	7
4	Correspondance entre une dune et les informations du fichier de sortie.....	7
5	Générer une image pour le programme 1	8
6	Générer une image pour le programme 2.....	9
7	Générer une image pour le programme 3.....	9
8	Générer une image pour le programme 4.....	9
9	Générer une image pour le programme 5.....	10
10	Générer une image pour le programme 6.....	10
11	Générer une image pour le programme 7.....	11
12	Diagramme de Gant S9	12
13	Diagramme de Gant S10	13

3 État de l'art : analyse d'image d'un litage

1	Exemples de textures.....	15
2	Textures de dunes d'un litages.....	16
3	Texture d'exemple	16

4	Texture découpée par paires de pixels.....	17
5	exemple de motif et leur transformée de Fourier	18
6	Comment déterminer hauteur et longueur d'onde d'une dune	19
7	Ligne de partage des eaux 1	19
8	Ligne de partage des eaux 2	20
9	Ligne de partage des eaux 3	20
10	Ligne de partage des eaux - les mesures	20
5	Blocs applicatifs en Python	
1	Fonctionnalité 1 - calcul et affichage d'une transformée 2D de Fourier.....	24
2	Fonctionnalité 2 - chargement d'une image sous Tkinter	25
3	Fonctionnalité 3 - interface pour découper une image.....	25
4	Fonctionnalité 3 - les sous-images produites.....	26
5	Fonctionnalité 4 - Tracer un segment sur une image	26
6	Fonctionnalité 5 - Les caractéristiques de l'image.....	27
6	Création des interfaces	
1	Interface - Menu principal	29
2	Interface - Traitement des axes	30
3	Interface - Traitement de l'image complète	31
4	Interface - Visualisation des profils	31
5	Interface - Visualisation des profils étirés	31
7	Gestion des axes	
1	Ajout d'un axe	32
2	Dupliquer un axe.....	33
3	Gestion des actions sur les axes.....	33
4	Gestion de l'erreur de l'axe avec à longueur nulle.....	34
5	Gestion de l'erreur des axes en dehors de l'image.....	34
6	Gestion du changement d'image/résolution	34
8	Détection des dunes sur un axe	
1	Transposer un axe de la miniature vers l'image originale.....	36
2	Déterminer les pixels à prendre 1	36
3	Déterminer les pixels à prendre 2.....	37
4	Les types de tracés	37
5	Influence du sens du courant sur le choix du pixel.....	38
6	Export des résultats au format texte	39

7	Export des résultats pour un axe.....	40
8	Export des résultats pour tous les axes	40
9	Fiji / MorphoLibJ pour de la segmentation d'image	41
9	Programme de génération d'image TIF 16 bits	
1	ArcGIS - bug de l'outil "Copy Raster"	43
2	Génération d'une image TIF 16 bits.....	43
10	Gestion de projet	
1	Diagramme de Gant réel du S10	44
A	Guide développeur	
1	Contenu du dépôt GitHub	49
2	Diagramme de classe	50
3	Tests unitaires via un terminal.....	50
4	Tests unitaires via un l'IDE Eclipse.....	50
B	Installation	
1	Installation 1	51
2	Installation 2.....	52
3	Installation 3.....	52



Introduction

Dans le cadre du Projet de Recherche et Développement (PRD) en cinquième année, j'ai choisi d'aborder le sujet d'analyse d'images de la Loire afin d'en repérer automatiquement les dunes et leurs caractéristiques. Ce sujet a été proposé par Stéphane Rodrigues, dont le but est de déterminer les caractéristiques des dunes de sable se situant au fond du fleuve à partir d'images. Les images sont générées à partir de relevés 3D du lit du fleuve et qui, via les dénivelés et un logiciel de traitement dédié, génère une image en niveau de gris. Ce rapport traite de la partie recherche, il concerne ainsi les points suivants :

- ☐ La présentation du cadre du projet.
- ☐ Les spécifications attendues de la future application par Stéphane.
- ☐ Un état de l'art des techniques employées pour faire de la recherche sur une image.
- ☐ Une recherche quant au langage et aux librairies à utiliser pour le programme.
- ☐ La création de quelques blocs applicatifs pour tester la pertinence des choix de programmation.

Par la suite, ce même rapport traitera de la partie développement, et donc sera présenté :

- ☐ Les différentes interfaces de l'application et les choix d'ergonomie choisis.
- ☐ Comment a été faite la fonction des axes tracés par l'utilisateur.
- ☐ La procédure pour détecter des dunes sur un axe.
- ☐ Le sous-programme pour générer des images au format 16 bits.
- ☐ La gestion du projet durant sa phase de développement.

1

Cadre du projet

Pour comprendre le cadre du projet, nous allons expliquer le contexte du PRD, présenter brièvement comment se forme un litage, les objectifs du projet à atteindre, puis comment sont générées les images qui seront en entrées du futur programme.

1 Contexte

Afin de connaître le déplacement des sédiments le long de la Loire, l'étude du litage du fleuve peut apporter moult informations quant à la puissance du flux le parcourant. Ainsi le sujet de ce Projet de Recherche et Développement consiste à partir d'une image du fleuve, d'automatiser le calcul de caractéristiques des dunes présentes dans son lit. L'image est générée à partir d'un fichier .las et du logiciel ArcGIS, sur laquelle chaque teinte de gris d'un pixel correspond à un niveau d'altitude du lit. Voici un exemple d'image à traiter :

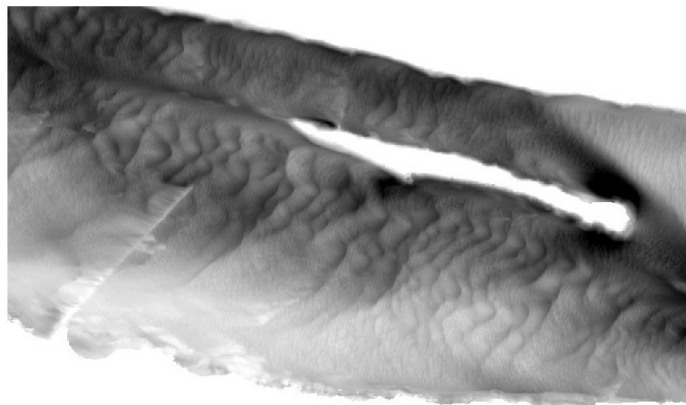


Figure 1 – Image à traiter

Une fois les résultats du traitement des images obtenus et son croisement avec des connaissances en sédimentologie, il est possible d'en déduire les flux sédimentaires et déterminer si le fleuve possède une quantité suffisante ou non de sédiments. Mais aussi, l'analyse du lit d'un fleuve permet aussi de faire de la prévention aux inondations. L'analyse des dunes en milieu aquatique n'étant pas un sujet courant, expliquons brièvement leurs différentes formes suivant la puissance du courant fluvial.

2 Morphologie d'un litage

À la suite de nombreuses analyses théoriques appuyées d'expériences en laboratoire, il a été trouvé une correspondance entre la forme du lit et la puissance de écoulement qui le traverse [WWW2] [WWW4]), de ces études plusieurs motifs ont été répertoriées :

- ☐ Le plus simple à comprendre est le cas d'une surface plane pour le lit, celle-ci est présent quand le courant n'est pas assez fort pour y déplacer les sédiments.
- ☐ Si la puissance du flux atteint un certain seuil, les grains formant le lit se déplacent et forment de nombreuses et petites ondulations (*ripples*).
- ☐ En augmentant encore le flux, les ondulations se rassemblent pour former des dunes 2D, leur crête étant rectiligne.
- ☐ Avec un courant encore plus puissant, les dunes se déforment et se nomment ainsi dunes 3D.
- ☐ Si le courant augmente toujours, les dunes sont balayées par ce dernier, et leurs sédiments se répartissent uniformément sur le lit.
- ☐ Néanmoins si la puissance du courant s'intensifie, il y a formation d'antidunes pouvant ainsi générer des vagues en surface (l'écoulement est dit supercritique).

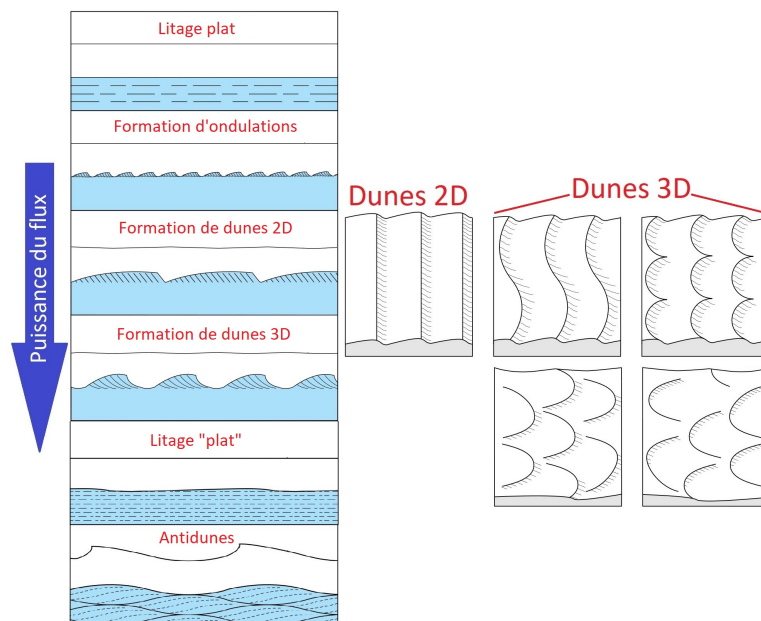


Figure 2 – Variation d'un litage en fonction du courant

Voyons maintenant comment sont opérés les relevés de mesure sur le parcours de la Loire.

3 Méthodes de relevés utilisées

Avant tout, les premiers relevés des caractéristiques des dunes se faisaient à la main lorsque le niveau de la Loire était au plus bas (en été). Les résultats étaient ensuite regroupés par zone et associés à une image de la Loire. Afin de rendre cette tâche beaucoup plus rapide et pouvoir le faire aussi en toutes situations, il a été question que les relevés se fassent par un dispositif.

Actuellement, les relevés de la Loire se font par l'usage d'un écho-sondeur multi-faisceau (*Multibeam echosounder*). Le dispositif est un sondeur acoustique associé à un GPS qui utilise des ondes sonores pour localiser la profondeur des fonds, celles-ci étant peu absorbées par l'eau et

faiblement sensibles à l'obturation du milieu aquatique. De nos jours les meilleurs dispositifs peuvent faire des mesures de profondeur allant jusqu'à 11 km (*EM 122 multibeam echosounder*). Pour obtenir les relevés, un faisceau sonore est dirigé vers le fond du bassin, connaissant la vitesse de propagation des ondes dans l'eau (1500m/s), il ne reste plus qu'à mesurer le temps que met la pulsion sonore pour faire le trajet aller-retour pour connaître la profondeur, et donc l'altitude au fond du cours d'eau.

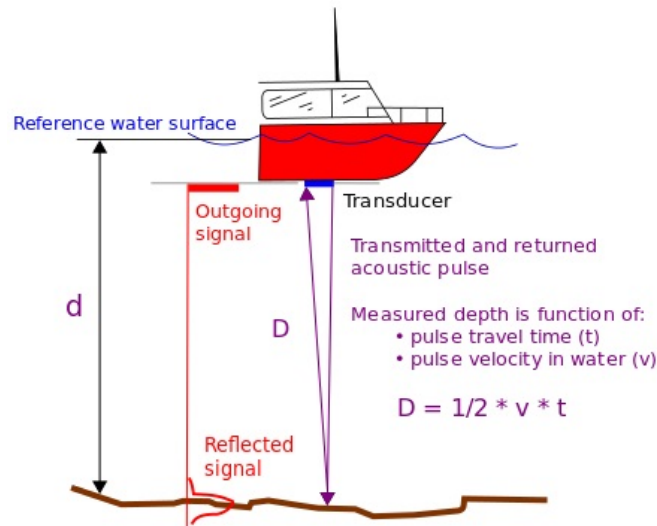


Figure 3 – Fonctionnement de l'Écho-sondeur

Afin d'accélérer le scan des fonds marins, les écho-sondeurs emploient plusieurs flux sonores en même temps, traitant chacun sur des zones différentes, ils deviennent ainsi multi-faisceaux.

Maintenant que le cadre est présenté, étudions le projet en lui-même et ses différents objectifs.

2

Présentation du projet et ses contraintes

1 Objectifs du projet

Le but premier du PRD est de retourner un fichier texte contenant les caractéristiques des dunes (hauteur et longueur d'onde) présentes sur une image indiquée par l'utilisateur. Le choix du format texte a été décidé pour garantir une correspondance avec les autres traitements que le client en fera, dont son utilisation dans un tableau Excel par exemple.

De plus, l'utilisateur doit pouvoir dessiner un axe sur l'image qu'il vient de choisir, et le programme doit en ressortir l'ensemble des caractéristiques des dunes étant coupées par le tracé, mais aussi en tracer une courbe de la hauteur de la dune en fonction de son positionnement sur l'axe. L'intérêt du graphique est de visualiser (comme si c'était une vue en coupe) la hauteur du chenal à une zone précise, cela pour rapidement savoir si les dunes se déplacent en comparant des images d'un même secteur à des instants différents.

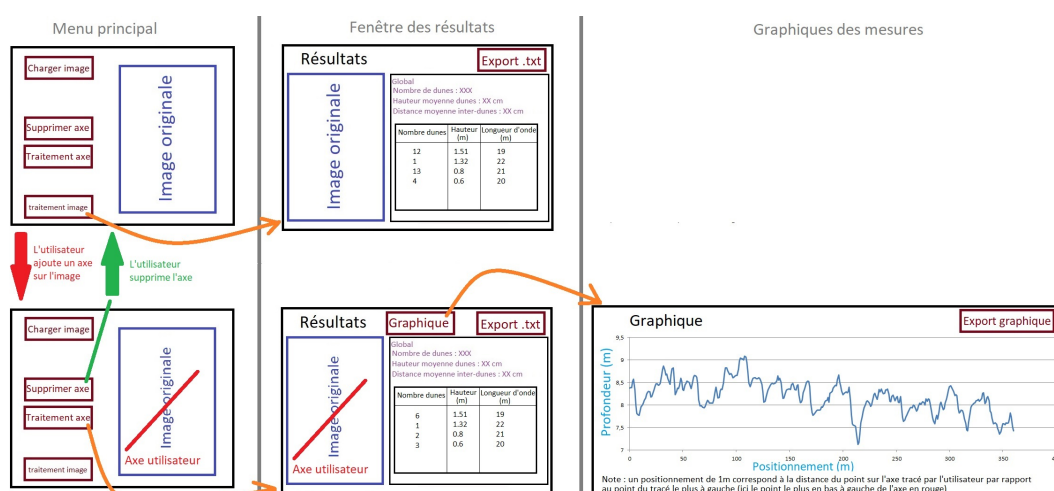


Figure 1 – Croquis de l'interface de l'appli

Plusieurs axes pourront être placés sitôt le premier créée, les suivants adopteront la même taille et la même direction afin que l'utilisateur n'ait plus qu'à les placer sur l'image, et que les résultats de chaque axe puissent être comparés.

Aussi, afin de pouvoir régler quand l'on estime si l'on a détecté une dune ou non, il faut définir un seuillage. Si notre "dune" a été mesurée comme faisant 7 cm de hauteur alors que l'on définit une dune à partir d'une hauteur de 10 cm, alors notre "dune" n'en est pas une. Ce paramètre doit ainsi être réglable depuis l'interface utilisateur, à la valeur 10 par défaut (valeur donnée par le client), et si besoin est, être changé avant d'effectuer le traitement de l'image.

Et en voici un diagramme de cas d'utilisation du programme, ce dernier nécessitant d'être simple d'usage, peu d'actions sont nécessaires par l'utilisateur.

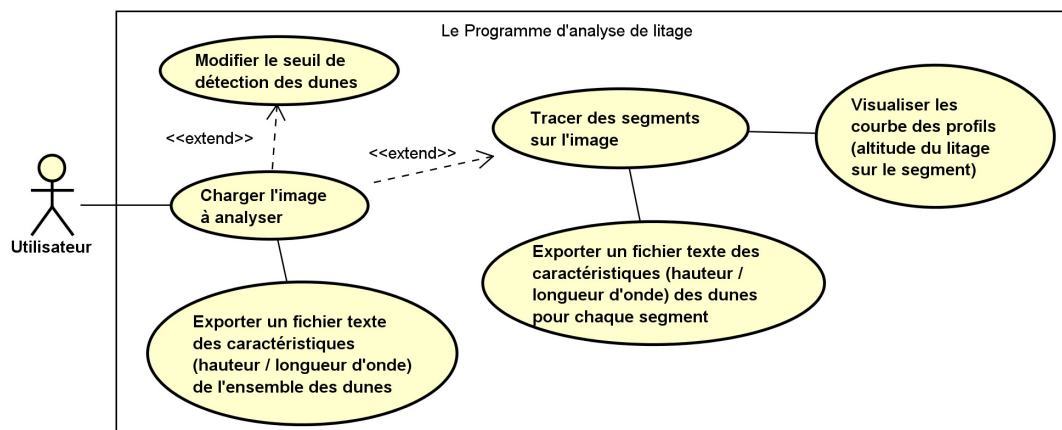


Figure 2 – Diagramme de cas d'utilisation

2 Fonctionnalités de l'application

Pour répondre à l'ensemble des demandes du client, l'application devra répondre aux fonctionnalités suivantes :

- ☐ Charger une image et en afficher une miniature sur l'interface graphique [Section 2](#) (Chapitre 5).
- ☐ Vérifier que l'image fournie par l'utilisateur puisse être traitée [Section 5](#) (Chapitre 5).
- ☐ L'utilisateur peut effectuer des tracés sur l'image pour faire des recherches de dunes à des zones ciblées [Section 4](#) (Chapitre 5).
- ☐ Repérer les dunes présentes sur une image ou une sélection de l'image (via les tracés effectués).
- ☐ Déterminer les caractéristiques des dunes (hauteur et longueur d'onde) découvertes.
- ☐ Exporter les informations extraites des dunes de l'image pour les sauvegarder dans un fichier texte.
- ☐ Établir un graphique d'une vue de profil de l'altitude des dunes se situant sur un tracé défini par l'utilisateur.
- ☐ Une navigation dans les interfaces simples d'utilisation et ergonomiques.
- ☐ Mise en place d'un installateur pour faciliter le déploiement de l'application.

3 Fichier en sortie du programme

Pour ce qui est du fichier en sortie du programme, celui-ci est juste un simple fichier texte permettant d'en extraire le tableau de caractéristiques des dunes depuis un tableur (Excel, LibreOffice Calc). Bien évidemment, ce fichier texte pourra très bien adopter le format .txt, .csv ou .xml, ces trois formats étant compréhensibles avec les logiciels de tableurs existants. Voici des exemple de ce que pourrais contenir le fichier de sortie, si un utilisateur a ou non décidé de faire des tracés :

```

AnalyseDunes 0T Exemple_8_508_5_029 7-12-2017.csv
1 Extraction des caractéristiques des dunes V1.0;;
2 A partir de l'image : Exemple_8_508_5_029.tif;;
3 Fait le 7/12/2017;;
4 Nombre de dunes total : 8;;
5 Nombre de tracé utilisateur : 0;;
6 ;;
7 IdentifiantDune;HauteurDune (m);LongueurOndeDune (m)
8 1;0.32;3
9 2;0.64;9.22
10 3;0.4;5.40
11 4;0.9;10.78
12 5;0.54;7.38
13 6;1.21;5.24
14 7;0.32;6.50
15 8;0.46;7.14

AnalyseDunes 2T Exemple_8_508_5_029 7-12-2017.csv
1 Extraction des caractéristiques des dunes V1.0;;;;;;;;;
2 A partir de l'image : Exemple_8_508_5_029.tif;;;;;;;;;
3 Fait le 7/12/2017;;;;;;;;;
4 Nombre de dunes total : 8;;;;;;;;;
5 Nombre de tracé utilisateur : 2;;;;;;;;;
6 :::::
7 IdentifiantTracé;IdentifiantDune;HauteurDune (m);LongueurOndeDune (m);AltitudeCreux1 (m);Distance1 (m);AltitudePic (m);Distance2 (m);AltitudeCreux2 (m)
8 1;1;0.32;3;-2.2;-1.48;1;-1.8
9 1;2;0.64;9.22;-1.8;7.22;-1.66;2;-2.3
10 1;3;0.4;5.40;-2.3;2.9;-1.8;2.5;-2.2
11 2;1;0.9;10.78;-2.2;5.78;-1.6;5;-2.5
12 2;2;0.54;7.38;-2.5;4.38;-2.1;3;-2.64
13 2;3;1.21;5.24;-2.64;3.24;-1.59;2;-2.80
14 2;4;0.32;6.50;-2.80;3.5;-2.48;3;-2.8
15 2;5;0.46;7.14;-2.8;4.14;-2.46;3;-2.9

```

Figure 3 – Exemples de la structure du fichier de sortie

Tout d'abord, le fichier de sortie devra de par son nom savoir de quelle image il est issu, combien de tracés ont été faits et quand il a été fait, cela pour permettre à Stéphane Rodrigues d'archiver facilement les fichiers générés. Ensuite dans le fichier lui-même, les premières lignes servent d'en-tête et retracent les informations propres à l'analyse faite : la version de l'application de recherche des dunes utilisée, le nom du fichier de l'image analysée, quand elle a eu lieu, le nombre de dunes total repéré, ainsi que le nombre de tracés fait par l'utilisateur.

Ensuite, des informations propres à chacune des dunes sont listées telles que la longueur d'onde et la hauteur des dunes. Bien évidemment, afin de pouvoir générer une courbe d'une vue en profil des dunes, des informations supplémentaires sont générés si l'utilisateur a fait des tracés sur l'image. Voici une image indiquant à quoi correspond chaque valeur de mesure du fichier texte par rapport à une dune (ici un tracé en noir) :

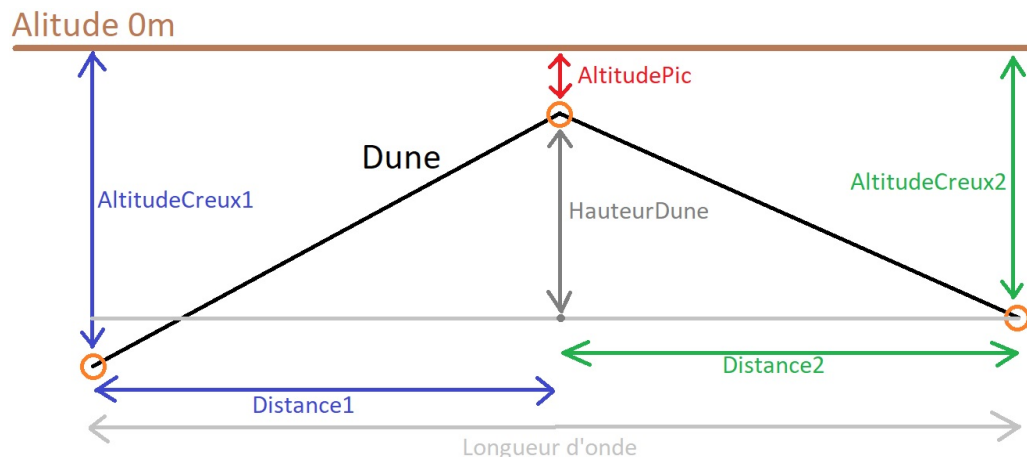


Figure 4 – Correspondance entre une dune et les informations du fichier de sortie

4 Environnement

L'environnement sur lequel le programme sera utilisé sera un Windows 7 (et suivant) 64bits, et si possible il devra aussi être fonctionnel sous Linux.

5 Profils des utilisateurs

Pour ce qui est des profils utilisateurs du programme, ceux-ci sont des personnes n'ayant pas de connaissance particulière en informatique, le design et l'ergonomie du programme sont donc des critères importants pour son utilisation.

Il en va de même pour le déploiement du futur applicatif, n'ayant pas de langage de programmation défini, ce dernier peut grandement influencer le déploiement du programme, un installateur doit donc être prévu.

Si reprise du programme il y a, le code doit être compréhensible, documenté et s'appuyer sur des bibliothèques utilisées et non-obsolètes afin qu'une personne ayant le niveau d'un étudiant Polytech puisse le reprendre en main.

6 Caractéristiques des images à traiter

Les images fournies au programme sont des images en nuances de gris 8 bits (256 teintes) au format .tif, à chaque pixel correspond à un espace de 1m², le noir absolu (0) correspondant au niveau le plus profond de l'image, et le blanc (255) aux secteurs les plus élevés.

Mais comment générer les images pouvant être traitées par le programme ?

1/ Ouvrir ArcMap (logiciel de la suite ArcGIS) et y charger le Modèle Numérique de Terrain (MNT) à utiliser, ici le modèle est nommé "bat26".

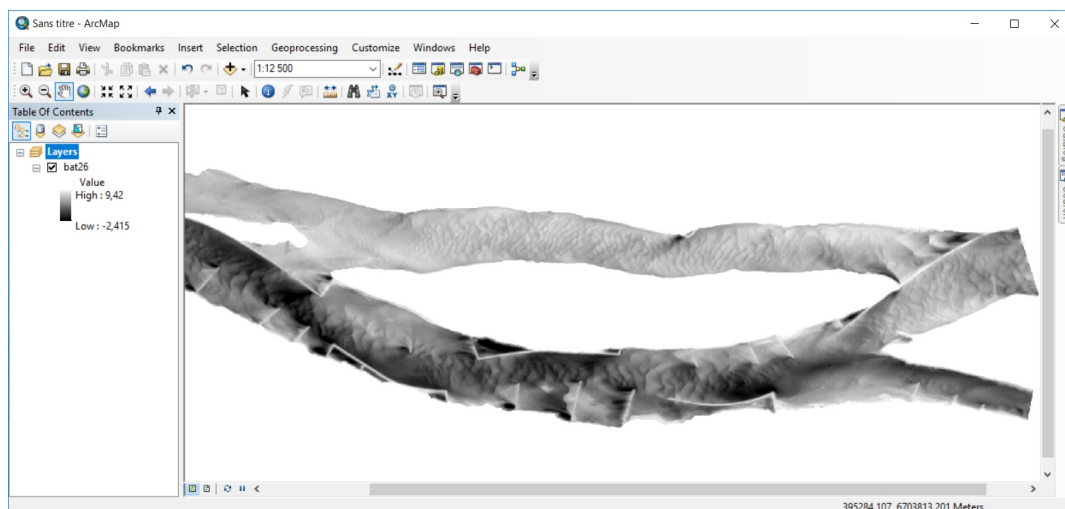


Figure 5 – Générer une image pour le programme 1

2/ Zoomer sur la zone à analyser, puis changer le mode de visualisation pour "Layout View". Remarquer qu'il y a peu de teintes de gris foncées dans la sélection choisie.

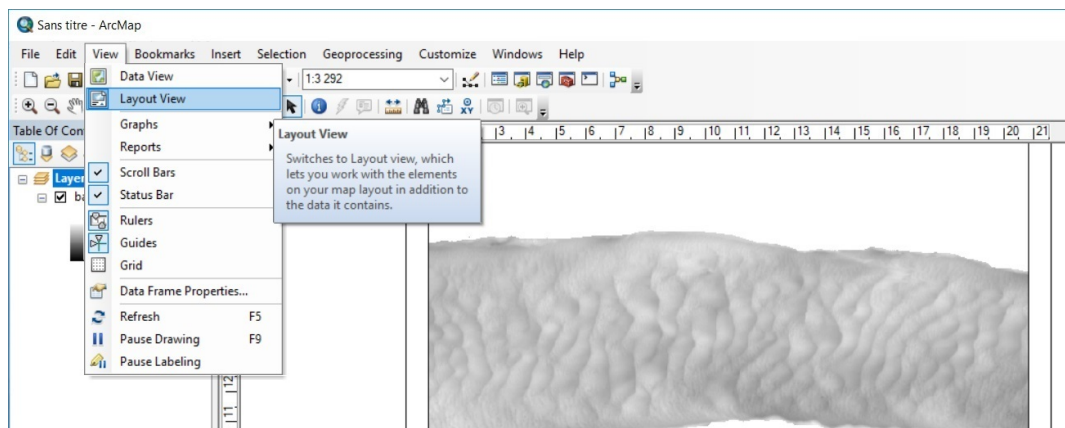


Figure 6 – Générer une image pour le programme 2

3/ Demander un export des données du MNT dont fait partie la zone sélectionnée (ici "bat26").

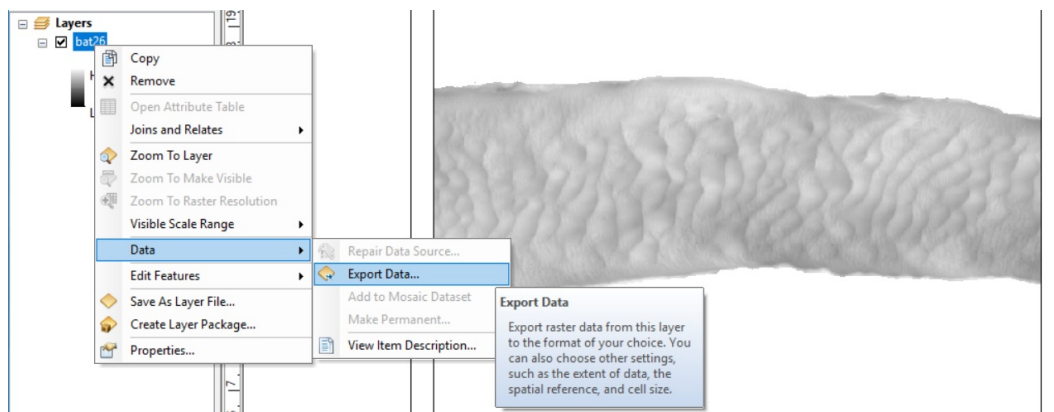


Figure 7 – Générer une image pour le programme 3

4/ Lors du choix des options, demander à ce que les données exportées ne concernent que la zone sélectionnée, et renseigner un nom pour le MNT exporté, dans l'exemple "bat262". Au moment d'enregistrer, accepter de charger les données nouvellement créées.

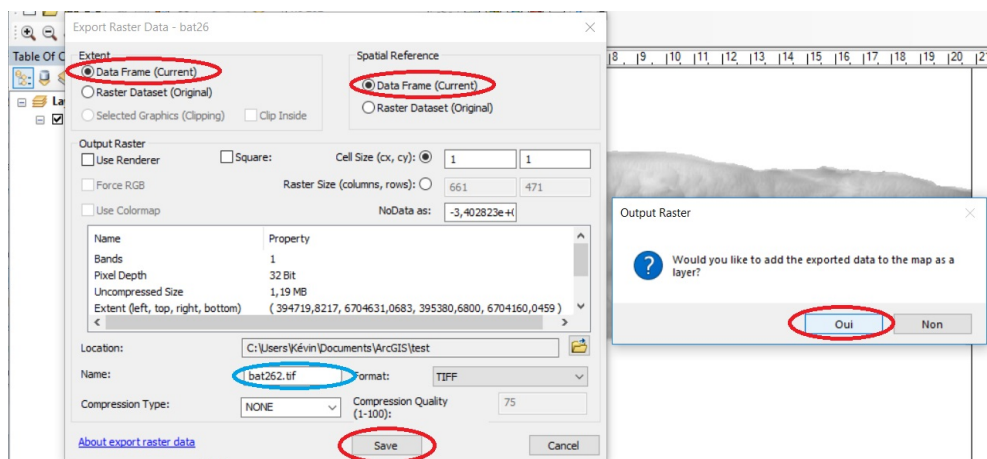


Figure 8 – Générer une image pour le programme 4

5/ Désactiver maintenant l'affichage du MNT de départ ("bat26"), plusieurs choses sont à remarquer :

- Depuis l'arborescence des MNT chargés (sur la gauche), on peut connaître l'altitude maximum et minimum de la zone choisie ("bat26"), ici respectivement 8.508m et 5.029m.
- L'affichage du MNT a étendu sa plage de teintes de gris sur l'ensemble des valeurs de la zone, nous avons augmenté le contraste de l'image.

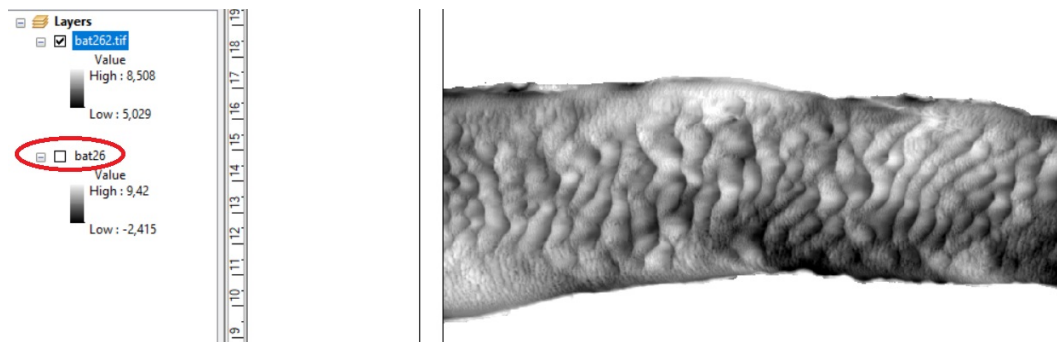


Figure 9 – Générer une image pour le programme 5

6/ Avant d'effectuer un export de l'image, il reste à enlever le cadre noir situé sur les bords de l'image, ce dernier peut considérablement fausser les résultats de par la présence de blancs autour de lui. Pour l'enlever, accéder aux propriétés de l'affichage, sur l'onglet "Fenêtre", et demander une bordure d'épaisseur 0.

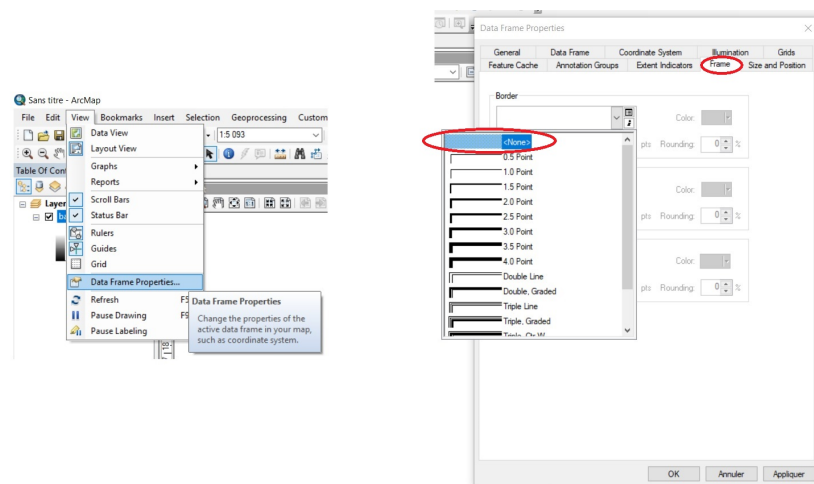


Figure 10 – Générer une image pour le programme 6

7/ Il ne reste plus qu'à exporter sous forme d'image la zone que nous avons choisie, sous format .TIF et comme mode de couleur niveau de gris 8 bits. Ainsi on obtient une image de la zone choisie sous le bon format, et on connaît son altitude minimum (soit sa profondeur maximum, quand la teinte de gris est noire) / son altitude maximum (le blanc).

Dans le nom de l'image on n'oublie pas de renseigner les valeurs maximum et minimum d'altitude, ces deux valeurs seront à ajouter à la fin du nom de l'image séparées par un '_' , pour adopter par exemple le nom *Loire24_-3,16_-7,68.tif*.

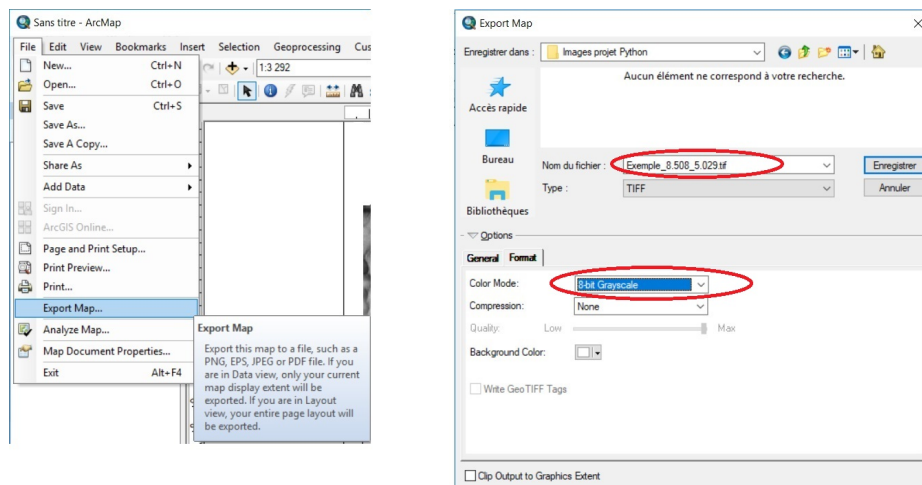


Figure 11 – Générer une image pour le programme 7

Pour connaître quelle est la hauteur d'une dune, en se basant sur les valeurs extrêmes indiquées dans le nom de l'image, il suffit de connaître la variation de nivellement entre chaque teinte de teinte de gris, puis connaissant la teinte du pixel que l'on étudie, on lui ajoute X (son niveau de gris) fois le nivellement inter-teinte de gris ainsi que l'altitude minimum de l'image.

Si on suppose notre image de départ avec comme altitude minimum 1,95m, altitude maximum 12,4m et le niveau de gris de chaque pixel codé sur 8 bits, il y a donc 256 teintes de gris différents (2^8).

Nous voulons connaître l'altitude d'un pixel dont le niveau de gris est de 80, comment le déterminer ?

- Nivellement inter-teinte de gris = $(12,4 - 1,95) / (2^8 - 1) = 10,45 / 255 \approx 0,041\text{m}$
- Mais pourquoi division par $(2^8 - 1)$ et non 2^8 ? Sur les 256 teintes de pixels, la première correspond à l'altitude minimum (la dernière à l'altitude maximum), il ne reste donc que 255 intervalles.
- Ainsi nous pouvons connaître l'altitude correspond au pixel qui est de $1,95 + 80 * 0,041 \approx 5,23\text{m}$

7 Bases méthodologiques

Pour garantir un suivi de projet régulier pour la partie recherche, un Trello a été utilisé ([Ici](#)) et des comptes rendus-hebdomadaires ont été envoyés aux différents tuteurs du projet.

Étant dans un projet de recherches expérimentales, le sujet initial m'étant inconnu (l'analyse de dunes fluviales) et le client ayant de nombreux déplacements, savoir et comprendre ce qu'il faut faire peut mettre du temps, ainsi planifier à l'avance ce que devra faire l'application et comment résoudre le problème de reconnaissance des dunes, n'est pas une tâche réalisable précisément. On peut néanmoins faire un bilan de ce qui a été fait via le diagramme de Gant suivant :

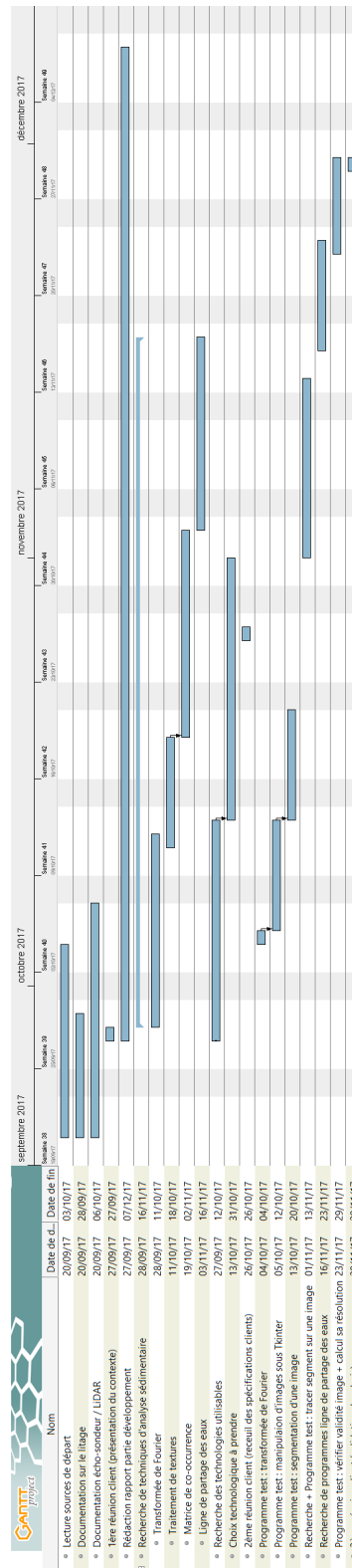


Figure 12 – Diagramme de Gant S9

La partie recherche étant terminée avec une piste pour reconnaître les dunes à partir d'une image, on peut maintenant faire un diagramme de Gant prévisionnel de ce qui sera fait pour le semestre prochain.

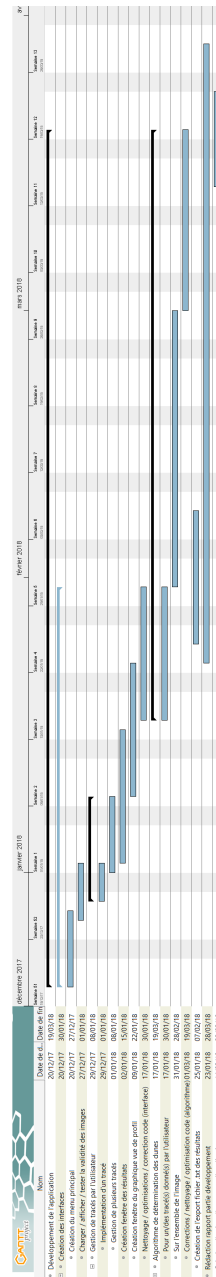


Figure 13 – Diagramme de Gant S10

Le projet sera développé en langage Python 3.6 (pour fonctionner sous Windows et Linux), utilisera les bibliothèques OpenCV 3.3, Pillow 4.3. l'EDI (Environnement de Développement Intégré) choisi sera **Eclipse** avec le plugin Python **PyDev**. L'installateur sera certainement un dossier .zip contenant l'installateur Python pour Windows, les bibliothèques additionnelles utilisées, deux scripts (un pour Windows, l'autre pour Linux) permettant d'installer les librairies, ainsi qu'un dossier contenant le programme lui-même.

Pour ce qui sera propre au développement, l'usage de la méthodologie SCRUM semble la plus adaptée. En effet, le programme final n'étant pas spécialement lourd, et l'interface/ergonomie étant un critère important, garantir régulièrement des livrables permet de limiter les risques de rebuter l'utilisateur par les choix de design choisis via la prise en compte de ses commentaires.

Connaissant maintenant le contexte du projet, ce que souhaite le client et comment générer les images en entrée du futur programme, voyons quelles peuvent être les solutions envisageables pour répondre aux besoins demandés.

Première partie

Recherche

3

État de l'art : analyse d'image d'un litage

Faisons un état de l'art des techniques actuellement utilisées et utilisables pour la reconnaissance de dunes, et ainsi y récupérer leurs caractéristiques (hauteur et longueur d'onde). Pour ce faire, nous devons définir les différents descripteurs (caractéristiques) qui permettent de reconnaître une dune, en commençant par une technique d'analyse de texture.

1 Reconnaissance par textures

1.1 Types de texture

Avant de faire rechercher une texture, il convient d'en expliquer que pour le traitement d'images, il en existe deux types :

- Les macrotextures (ou textures régulières) représentent des motifs qui se répètent et sont liés par une règle précise, par exemple un mur de briques, du tissu ou même du grillage.
- Les microtextures (textures aléatoires) sont quant à elles plus désorganisées donnant ainsi un ensemble plus chaotique mais qui visuellement reste homogène, du feuillage, des nuages ou l'image d'un bois en sont des exemples.

Macrotextures



Microtextures

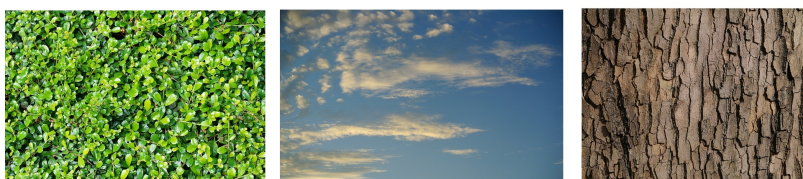


Figure 1 – Exemples de textures

Les dunes d'un litage n'ayant pas une texture homogène, de par les différentes formes adoptées suivant la puissance du courant, on la range ainsi dans la catégorie des macrotextures. Dans le but d'obtenir des exemples de textures d'un lit, on peut aussi s'inspirer des dunes de sable dans le désert. En effet ces dernières adoptent une structure similaire, où la place de l'eau, c'est le vent qui forme les motifs sur le sable.



Figure 2 – Textures de dunes d'un litages

1.2 Matrice de co-occurrence

La technique la plus utilisée pour effectuer de l'analyse de texture procède par usage de matrices de co-occurrences, cette dernière étant relativement simple à mettre en place et offre des performances convaincantes.

Elle consiste à créer une matrice carrée de taille L , correspondant au nombre de teintes de gris différents, dont chaque teinte est associée à un indice, et suivant une loi géométrique choisie composée d'une distance interpixel δ et d'une orientation θ (0° , 45° , 90° ...), permet d'indiquer le nombre de pixels qui respectent les contraintes de paires de teintes (i et j) celle de la loi géométrique. La matrice de co-occurrence permet ainsi de connaître le calcul des probabilités $P(i, j, \delta, \theta)$.

Avec un exemple cela serait plus simple à comprendre, prenons ainsi la texture sur l'image suivante, et mettons un nombre sur chacune des teintes de gris rencontrées :

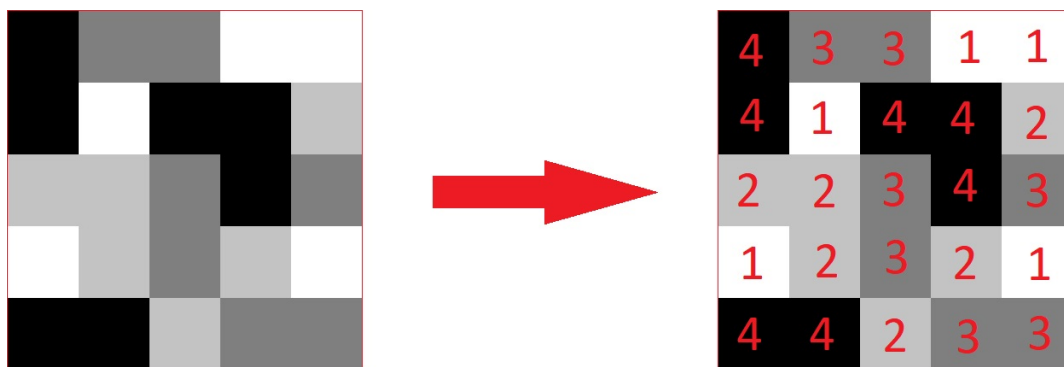


Figure 3 – Texture d'exemple

Choisissons maintenant la loi géométrique que nous allons utiliser, prenons une distance $\delta = 1$ et une orientation $\theta = 0^\circ$, on répertorie ainsi le nombre de paires de pixels suivant leur teinte, pixels qui sont côte à côte (horizontalement seulement). L'image de départ ayant 4 teintes de gris différents, la matrice de co-occurrence sera de taille 4×4 . Calculons maintenant le nombre de paires de pixels que nous avons respectant la loi définie.

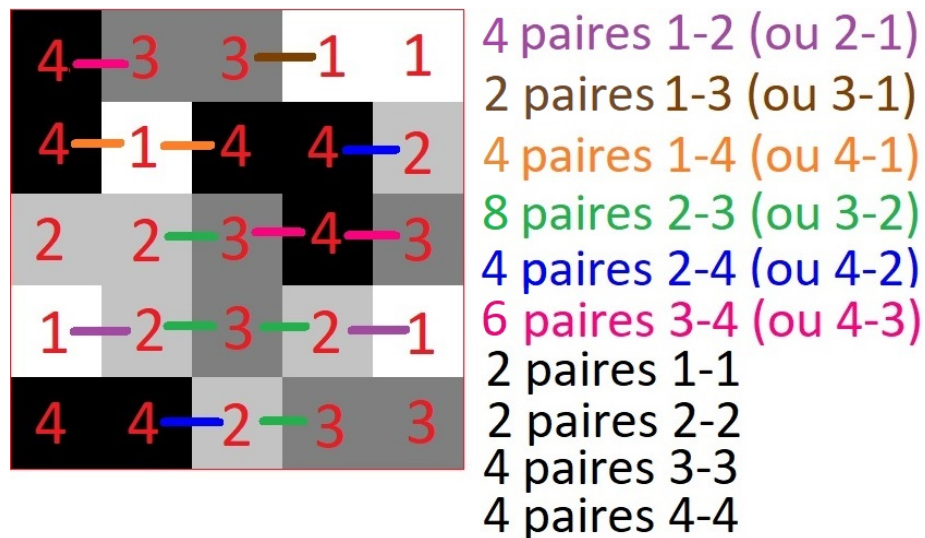


Figure 4 – Texture découpée par paires de pixels

Avec les valeurs déterminées, on peut maintenant compléter le tableau du nombre de couple de teintes de gris rencontré.

Teinte j \ Teinte i	1	2	3	4
1	2	4	2	4
2	4	2	8	4
3	2	8	4	6
4	4	4	6	4

Et de cela en déduire la matrice de co-occurrence :

$$\begin{bmatrix} 2 & 4 & 2 & 4 \\ 4 & 2 & 8 & 4 \\ 2 & 8 & 4 & 6 \\ 4 & 4 & 6 & 4 \end{bmatrix}$$

Une fois la matrice obtenue, il est possible d'en déduire des descripteurs mettant en évidence des caractéristiques propres à la texture, tels le Second Moment Angulaire (ASM), le Contraste, l'Entropie... Il reste ainsi plus qu'à retrouver dans l'image de départ des motifs qui correspondent aux descripteurs calculés.

Dans le cas du projet de PRD, l'usage de matrices de co-occurrence sur une image d'une dune permettrait de récupérer différents descripteurs pour les reconnaître, descripteurs qui pourront servir à retrouver les dunes sur les images fournies par l'utilisateur.

L'un des principaux défauts de l'usage des matrices de co-occurrences est la réduction des niveaux de gris par rapport à l'image originale, cela afin que la matrice soit la plus rapide à déterminer. Néanmoins, en lissant teintes de gris, cela revient aussi à ne plus pouvoir mesurer les caractéristiques des dunes sans une approximation convenable.

2 Usage de la transformée de Fourier

La transformée de Fourier est un procédé mathématique utilisé dans le cadre du traitement d'image pour faire de l'échantillonnage, compression et filtrage. Dans le cadre du PRD, parmi

les différentes sources sur internet, elle est la méthode la plus employée pour ma reconnaissance de dunes ([WWW5] [WWW1] [WWW3]).

L'analyse d'image par traitement de Fourier implique que l'image de départ soit en nuances de gris (pas en couleurs), que l'on possède au moins une représentation spectrale par Fourier d'une dune, et d'indiquer une taille / un taux de recouvrement comme paramètres pour découper l'image de départ en imagerie, pour ces dernières on applique la transformée de Fourier, et on compare les résultats obtenus avec la représentation spectrale d'une dune.

Ainsi, il suffira pour chacune des sous images obtenues, de déterminer sa transformée de Fourier et de la comparer avec celle d'une dune. Si les images sont similaires (notion de distance entre deux images), alors il y a une dune de déterminée.

Voici quelques exemples de motifs simples et leur allure après transformée 2D de Fourier :

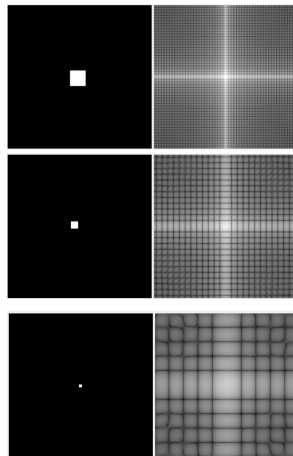


Figure 5 – exemple de motif et leur transformée de Fourier

Ainsi on remarque que plus le carré est petit, plus sa transformée de Fourier est constituée de gros carré, signalant ainsi que la fréquence de changement de couleur (du noir au blanc et inversement) est plus faible.

Pour connaître la hauteur de la dune, il suffira de récupérer la valeur de la hauteur la plus haute (le pixel le plus blanc, le pic de la dune), puis de déterminer le pixel ayant la valeur la plus basse à son creux le plus proche (pixel le plus noir), en les soustrayant et utilisant l'échelle de l'altitude, on obtient la donnée souhaitée. Par exemple : On a repéré une dune, la valeur du pixel de son pic est 100, la valeur la plus basse des pixels alentour 30, la légende varie de 0m (pixel à 255) à -2.55m (pixel à 0) soit 0,01 mètre par teinte de gris. $100 - 30 = 70$, soit 70 teintes de gris de variance, la dune mesure donc $0.01 \times 70 = 70\text{cm}$.

La longueur d'onde elle, se détermine par la distance moyenne entre chaque point de la crête de la dune est le point le plus proche faisant partie d'un creux.

Prenons le schéma ci-dessus, les deux tracés en blanc au milieu sont les crêtes des dunes A et B, plus la couleur tend vers le noir, plus on est à un niveau profond. Pour déterminer la hauteur de la dune A, on prend la valeur de sa crête soit du blanc, et celui du creux voisin le plus profond, du noir, sa hauteur correspond donc à la différence de profondeur entre ce qui est représenté par la couleur blanche et noire. La longueur d'onde de la dune A correspond à la distance moyenne entre les points de sa crête, et les points des creux voisins, soit entre les points de la crête A et les points noirs du creux à gauche, ou gris foncé pour le creux de droite.

La dune B étant entre deux creux gris foncé, sa hauteur est donc inférieure à celle de la dune A (qui possède un creux plus profond en noir). La longueur d'onde de la dune B est déterminée de la même manière, la distance moyenne entre le point de sa crête et ceux de ses creux les plus proches (en gris foncés).

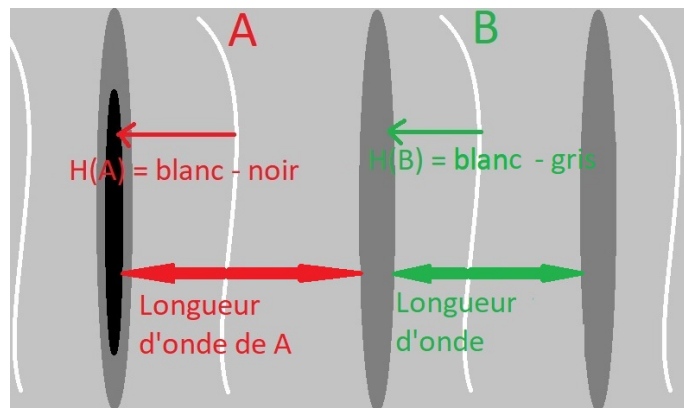


Figure 6 – Comment déterminer hauteur et longueur d'onde d'une dune

3 Ligne de partage des eaux

L'algorithme de ligne de partage des eaux (*watershed*) permet de considérer l'image comme si elle était un grand bassin avec de nombreuses cuvettes. À chaque pixel de l'image correspond une profondeur suivant son niveau de gris. Ainsi, l'algorithme correspond au procédé inverse de la génération des images (en niveau de gris) à partir des relevés de profondeur.

Comment l'algorithme procède :

Nous allons procéder avec l'exemple d'une image constituée d'une ligne de pixel pour simplifier la démarche. Pour chaque pixel de l'image, suivant leur teinte de gris, nous allons leur attribuer un nombre qui l'associe à un niveau de profondeur. Puis on réalise un graphique de l'altitude de la zone en fonction de la position du pixel.

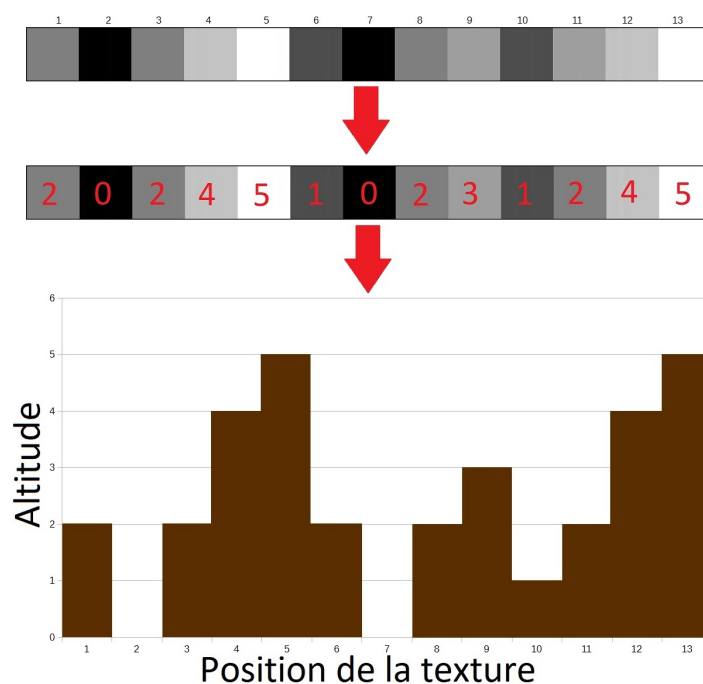


Figure 7 – Ligne de partage des eaux 1

Maintenant, nous allons verser de l'eau de sorte qu'à chaque itération, l'altitude minimum du bassin est augmentée de 1.

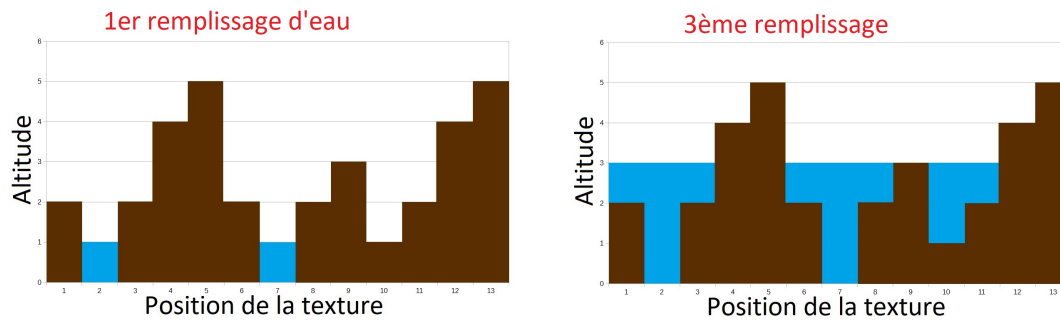


Figure 8 – Ligne de partage des eaux 2

Lors du 4ème et 6ème remplissages, deux bassins se rassemblent en un seul, une frontière entre les bassins a donc été franchie. C'est ainsi que l'algorithme de partage des eaux procède pour détecter quand il y a des pics présents. Ce même procédé peut aussi être appliqué sur des images 3D.

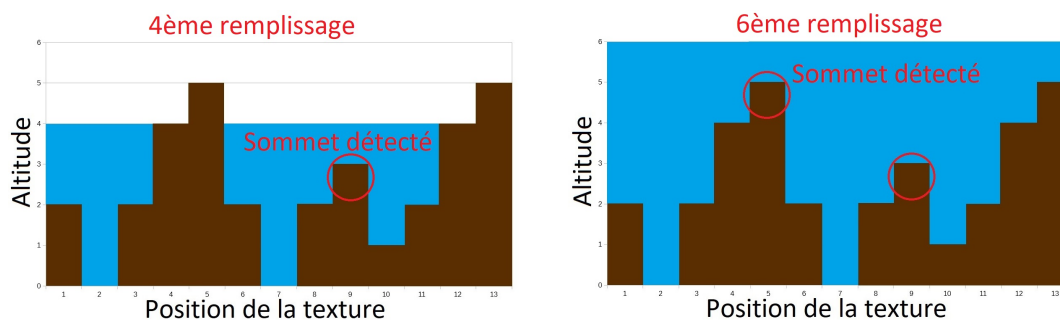


Figure 9 – Ligne de partage des eaux 3

Dans le cadre du projet de PRD, la hauteur et longueur d'onde d'une dune peuvent être déterminées suite à la récupération des points de sommet.

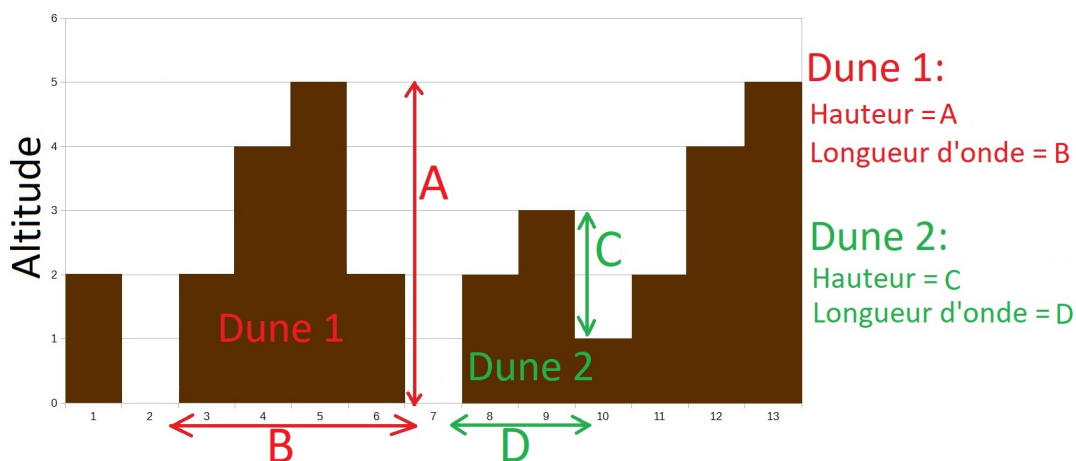


Figure 10 – Ligne de partage des eaux - les mesures

4 Reconnaissance par apprentissage

Dans le cadre d'une poursuite du programme, le développement d'un algorithme d'apprentissage pour repérer les dunes à partir d'images est aussi une piste d'amélioration envisageable. L'usage d'un algorithme d'apprentissage permet au programme de reconnaître des formes qui lui ont été demandées, puis d'y classer l'image suivant ce que l'algorithme a pu repérer. Trois types d'algorithmes d'apprentissage existent, ceux étant supervisés par un utilisateur, ceux sans et ceux par renforcement.

4.1 Apprentissage supervisé

Un algorithme d'apprentissage supervisé comme son nom l'indique, requiert au préalable que l'on définisse les différentes classes d'objets à reconnaître, et ces derniers possèdent de plusieurs images d'exemples étiquetées de leur classe associée.

L'algorithme va alors construire un tableau de corrélation entre les images fournies, et déterminer les descripteurs à utiliser (Contraste, l'Entropie...) permettant de caractériser les objets. Ce tableau sera utilisé par l'algorithme pour déterminer si oui ou non, un objet de telle ou telle classe est identifiable sur les images auxquelles l'utilisateur veut effectuer une analyse.

Dans le but de répertorier différents objets différents à repérer (par exemple différentes images de dunes), il est possible d'employer un apprentissage supervisé où toutes images d'un même objet sont regroupées dans un classifieur.

4.2 Apprentissage non supervisé

Dans le cas où l'apprentissage n'est pas appuyé par une aide extérieure, l'utilisateur fourni à l'ordinateur plusieurs images traitant des différents objets à reconnaître (pas d'étiquetage), et c'est à l'ordinateur de trier les images par classes de similitudes (dont l'utilisateur doit définir le nombre). Chacune de ses classes est composée d'images ayant le plus de ressemblance entre elles, la similarité entre images étant déterminée suivant une notion de distance.

Néanmoins, c'est à l'utilisateur de donner du sens aux classes créées. Une fois les classes établies, le tableau de corrélation associée à l'ensemble des images de départ est créé, et sera utilisé pour identifier les objets associés à chacune des classes trouvables sur les images envoyées par l'utilisateur.

4.3 Apprentissage par renforcement

Pour les algorithmes d'apprentissage par renforcement, ces derniers sont inspirés de phénomènes comme la sélection naturelle ou l'apprentissage chez un humain, ils effectuent diverses actions et suivant le résultat obtenu, s'adapte afin de faire mieux la prochaine fois. L'apprentissage se fait de façon empirique, cela via une notion de cause à effet. Ainsi l'algorithme adopte un comportement, qui à chaque fois qu'il trouve ou non des résultats s'adaptera afin d'être de plus en plus précis / rigoureux dans ses prochaines analyses. Dans le cas d'une analyse d'image, il affinera son modèle à chaque fois qu'il trouvera l'élément qu'il est censé trouver, le programme évolue ainsi après chaque utilisation. Un des algorithmes d'apprentissage par renforcement connu est AlphaGO Zero, intelligence artificielle développée par Google qui a appris à devenir de plus en plus doué au jeu de Go, battant plusieurs fois le champion du monde en titre, tout cela en apprenant des stratégies à force de s'affronter contre lui-même (plusieurs millions de parties).

4

Les technologies utilisables

Pour la réalisation du projet, trois langages ont été proposés, le JAVA, C++ et le Python. Nous allons commencer par en présenter les avantages, inconvénients et librairies envisageables pour chacun, puis par lister et d'expliquer pourquoi certaines librairies ont été mises de côté.

1 JAVA

Le Java a pour principal avantage d'être un langage multiplate-forme, à condition d'installer la machine virtuelle permettant d'y exécuter les programmes .JAR. N'étant pas particulièrement à l'aise avec le langage JAVA et n'étant pas non plus le mieux fourni en librairie pour le traitement d'images, celui-ci n'a pas été choisi.

Bibliothèques de traitement d'images : [OpenCV](#) / [JMagick](#) / [ImageJ](#)

2 C++

Le C++ est le langage pouvant donner les meilleures performances possibles, le programme doit cependant être compilé suivant le système d'exploitation (Windows / Linux / MacOS / Android...) pour être exécuté. Pour le développement du programme, l'IDE choisi serait Qtcreator afin de permettre une création aisée de l'interface graphique. Même si le C++ semble une solution intéressante, elle n'a pas été celle privilégiée, la possible reprise du programme étant un critère important.

Bibliothèques de traitement d'images : [OpenCV](#) / [VIGRA](#) / [Magick++](#)

3 Python

Le Python quant à lui est un langage interprété, pour fonctionner il n'a besoin que l'interpréteur Python soit installé sur les machines (fournie de base avec Linux / MacOS), ainsi tout comme le JAVA le code est multiplate-forme. N'étant pas un langage compilé, il n'a pas forcément besoin d'un IDE pour reprendre son développement, un simple traitement de texte peut suffire. Étant plutôt compact, l'apprentissage du Python est la plus simple des trois proposées, de plus, de nombreuses bibliothèques sont disponibles sur internet, dont un bon nombre pour le traitement

d'images. Néanmoins, toutes les bibliothèques utilisées par les programmes Python doivent être installées sur le poste, qui plus est, certaines librairies python ont besoin de leur équivalent C++ en plus pour fonctionner.

Bibliothèques de traitement d'images : **OpenCV** / Scipy + Numpy + Matplotlib / **scikit-image** / **VIGRA**

Ainsi le choix du langage s'est porté sur Python avec comme bibliothèques OpenCV (traitement d'images), Pillow (pour supporter l'affichage dans la fenêtre de divers formats d'images dont le .tif) et numpy (librairie optimisée pour les opérations sur les matrices).

Le logiciel ArcGIS utilisant aussi le Python, mais une version 2.7 datant de 2010, le programme final du PRD se basera sur Python 3.6 en prévision des futures versions de ArcGIS (ArcGIS Pro utilisant déjà Python 3.5) et ainsi rendre pérenne l'utilisation du programme.

4 Autres bibliothèques

De nombreuses autres bibliothèques ont aussi été envisagées, mais ces dernières sont inadaptées (trop lourdes / complexes / manquent des fonctionnalités) pour l'envergure du projet, en voici un tableau pour chaque langage de programmation envisagé :

Bibliothèques JAVA	JAI	Marvin	Visualization Toolkit
Fonctionnalités	-	- - -	+
Documentation	+	++	+
Poids	+	+++	- - -
Payant ?	Gratuit	Gratuit	Gratuit
Pourquoi à oublier	Abandonné en 2013	Inutilisable	Usine à gaz / complexe

Bibliothèques C++	Boost	Visualization Toolkit	IPSDK	VXL
Fonctionnalités	++	+	+	+
Documentation	++	+	+	0
Poids	--	- - -	--	++
Payant ?	Gratuit	Gratuit	Payant	Gratuit
Pourquoi à oublier	Trop Lourde	Usine à gaz / complexe	Lourd et payant	Supplanté par OpenCV

Bibliothèques Python	Boost	PythonMagick	IPSDK
Fonctionnalités	++	++	+
Documentation	++	++	+
Poids	--	- - -	--
Payant ?	Gratuit	Gratuit	Payant
Pourquoi à oublier	Trop Lourde	Lourd (Boost, Python.Boost et Magick++)	Lourd et payant

5

Blocs applicatifs en Python

Ici figurent diverses briques fonctionnelles en Python qui pourront être reprises pour la conception du programme final, leur conception fut juste pour tester la pertinence des bibliothèques à utiliser, ainsi que trouver comment répondre techniquement à des problématiques / demandes du client.

1 Calculer la transformée de Fourier 2D d'une image

Ce prototype consiste à calculer et afficher la transformée de Fourier d'une image, il est issu de plusieurs codes disponibles sur le site d'OpenCV sur [l'utilisation de la transformée de Fourier](#).

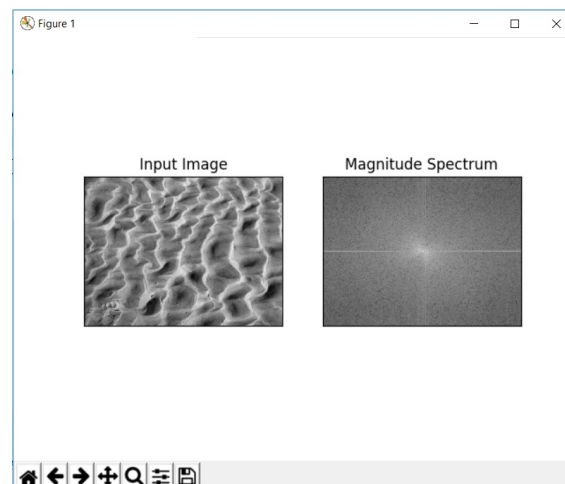


Figure 1 – Fonctionnalité 1 - calcul et affichage d'une transformée 2D de Fourier

Le but de ce programme est de charger en mémoire une image, l'afficher dans la fenêtre, la transformer en tant que matrice, y faire les calculs nécessaires pour la transformée 2D de Fourier, puis d'y afficher le résultat. L'image qui est utilisée est définie dans le programme et non par l'utilisateur, de plus l'interface utilisant Matplotlib, l'ajout de boutons ou divers éléments de contrôles supplémentaires n'est pas possible. Ainsi, le second prototype aura pour but de permettre à l'utilisateur de choisir l'image, mais aussi de lancer la transformée de Fourier quand il le souhaite.

2 Chargement d'une image et son affichage sous Tkinter

Ce deuxième prototype est une application codée en Python permettant à une image renseignée (via le bouton "Charger image"), de pouvoir l'afficher dans la même fenêtre (usage de Pillow), et suite à l'appui du bouton "Traitement image" de pouvoir calculer et d'afficher sa transformée 2D de Fourier (usage de Numpy et OpenCV). Comparée au prototype Précédent, l'image apparaît dans une fenêtre utilisant Tkinter plutôt que Matplotlib, cela permet d'y ajouter divers éléments graphiques comme des boutons et de leur attribuer des fonctionnalités.

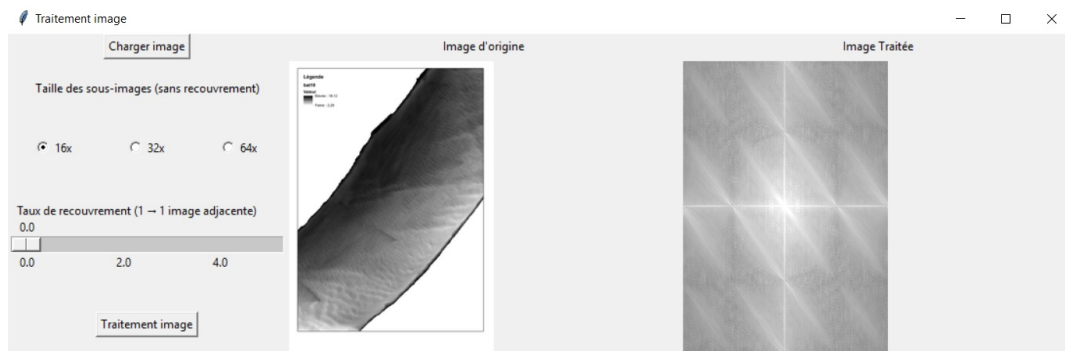


Figure 2 – Fonctionnalité 2 - chargement d'une image sous Tkinter

Les radios boutons et la slide bar à gauche de l'interface vont par la suite indiquer comment définir le découpage de l'image de départ (taille des sous images et taux de recouvrement), pour y faire sur chaque morceau, une comparaison entre leur transformée de Fourier et celle d'une ondulation / dune et ainsi déterminer à quel élément il se rapproche le plus.

3 Découpage d'une image en fonction d'une taille indiquée

Pour utiliser la transformée de Fourier sur une image de départ, on procède par un découpage de cette dernière en plusieurs sous images. Pour chacune d'entre elles, il sera fait une transformée de Fourier qui sera comparée à une transformée de Fourier d'une ondulation / dunes / etc...

Le but de ce prototype est de savoir comment segmenter une image en imageries dont la taille est définie par l'utilisateur (ici le choix se fait via des radios boutons). Sur l'interface nous pouvons ainsi remarquer que la première sous image est constituée des pixels (carré de 256 pixels) en haut à gauche de ceux de l'image de départ.

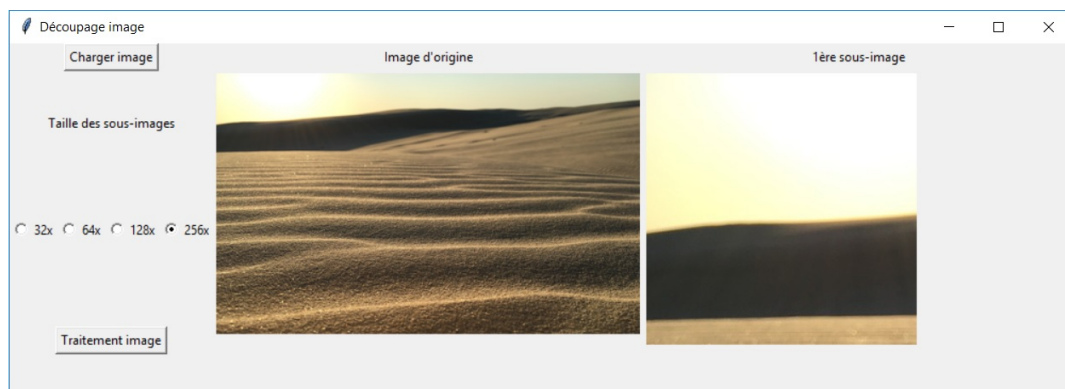


Figure 3 – Fonctionnalité 3 - interface pour découper une image

Et voici les différentes sous-images produites par l'application :

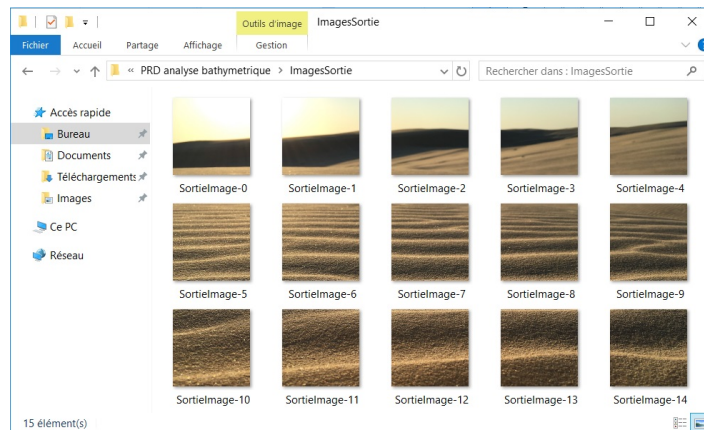


Figure 4 – Fonctionnalité 3 - les sous-images produites

4 Tracer un segment sur une image

Afin de répondre au besoin d'effectuer une recherche sur une partie des dunes de l'image, celles parcourant un tracé défini par l'utilisateur, il faut d'abord savoir comment faire ce tracé.

Pour rendre cette fonctionnalité simple à implémenter, il sera demandé à l'utilisateur de cliquer deux fois sur la miniature de l'image affichée, le programme se chargera de créer le segment les reliant. Un bouton permettra la suppression du tracé s'il ne convient pas à l'utilisateur, et ce dernier pourra de nouveau en faire un nouveau.

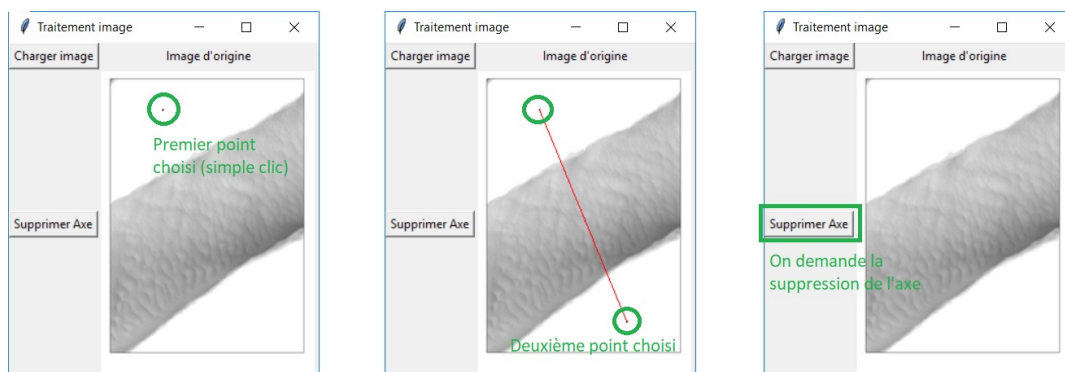


Figure 5 – Fonctionnalité 4 - Tracer un segment sur une image

5 Récupérer les caractéristiques de l'image

Afin de tester si l'image fournie par l'utilisateur respecte les conventions de nommages définies (sonNom_AltitudeMaximum_AltitudeMinimum.tif), ce bloc applicatif a pour but de tester cela et d'en informer l'utilisateur si l'image n'est pas correctement nommée, mais aussi, de déterminer la résolution de l'image. En connaissant sa résolution, différence d'altitude entre 2 niveaux de gris, on peut ainsi résoudre si une "dune" est assez haute ou non pour être qualifiée comme telle.

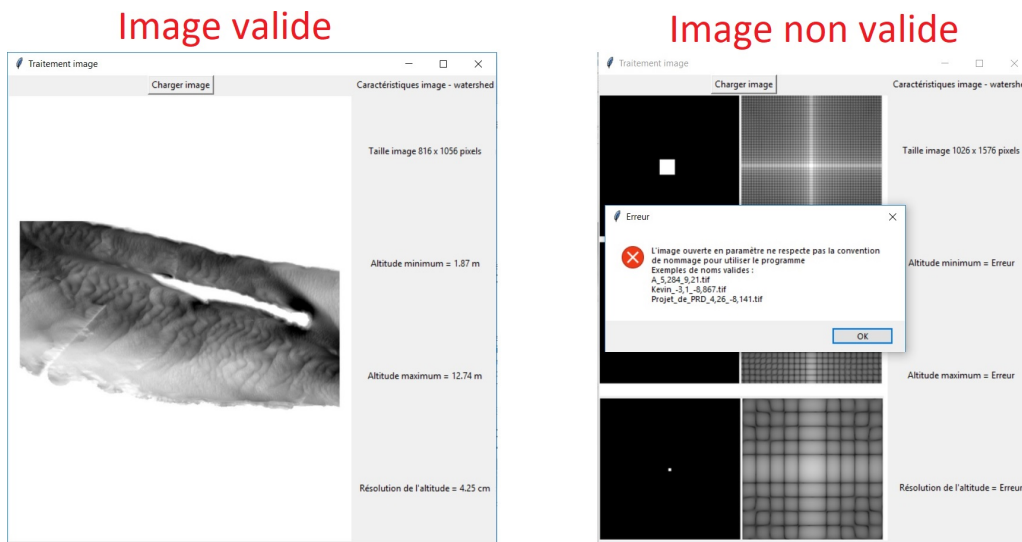


Figure 6 – Fonctionnalité 5 - Les caractéristiques de l'image

Deuxième partie

Développement

6

Création des interfaces

Pour la création des interfaces, ces dernières sont toutes codées avec le module graphique Python de base Tkinter. Son utilisation est pour rendre le plus léger possible et supporter au mieux l'ensemble des systèmes d'exploitation possibles les fenêtres créées.

1 Menu principal

Au lancement du programme, en exécutant le fichier "main.pyw" à la racine du projet, l'interface suivante va apparaître (image de gauche) :

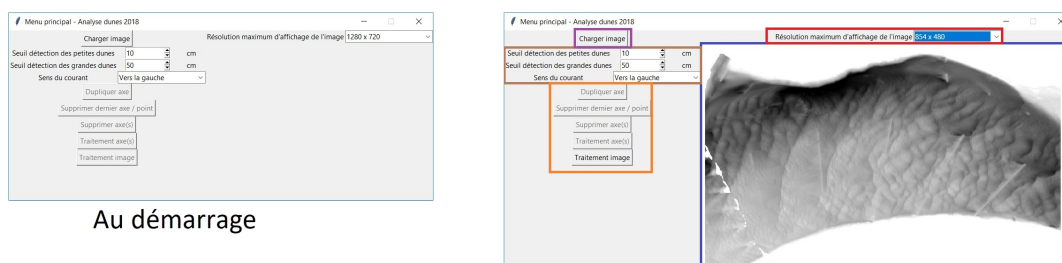


Figure 1 – Interface - Menu principal

Suite à un clique sur le bouton "Charger une image", cela ouvre une boîte de dialogue où l'on peut chercher l'image que l'on va renseigner au programme, cette dernière sera ensuite chargée dans la moitié droite de la fenêtre. Afin de présenter brièvement les fonctionnalités implémentées, nous utiliserons un code couleur

- ☐ En violet nous avons le bouton pour indiquer et charger l'image.
- ☐ Dans le cadre marron nous renseignons les informations sur les critères de détection des dunes.
- ☐ Le orange est pour manipuler les axes que l'utilisateur aura tracé, ainsi que pour passer aux analyses de l'image (par les axes, ou l'image entière).
- ☐ en rouge nous pouvons renseigner une taille (en pixels) pour laquelle l'image sera affichée à l'écran (les images à traiter pouvant dépasser la résolution de l'écran).
- ☐ Enfin, le cadre bleu indique l'emplacement où l'image est affichée, et où l'utilisateur devra effectuer ses clics de souris pour y placer ses axes.

Depuis le semestre précédent, il a été mentionné que le sens du courant, pour l'analyse d'image pour le cas de la Loire, de la droite vers la gauche. Hors, n'ayant pas eu connaissance de cette information, il était question de traiter l'image de la gauche vers la droite (sens de lecture), pour ajuster cela, il a donc été implémenté une boîte de dialogue pour définir le sens du courant, qui par défaut va vers la gauche.

Dans le même ordre, il fut aussi question de pouvoir discerner 2 types de dunes. En effet dans un cas pratique, il peut y avoir des dunes pouvant être sur d'autres dunes beaucoup plus grosses, pouvoir en reconnaître les 2 types et les classer sera donc un plus. Ainsi, il est donc demandé à l'utilisateur à partir de quelle taille l'on estime que nous sommes sur une petite dune, et à partir de quelle autre nous sommes sur une grosse dune (servant aussi de taille maximum pour les petites dunes). Au sein du programme, cette gestion des petites et grandes dunes n'est néanmoins pas implémentée.

2 Traitement des axes

Après que l'utilisateur est tracé ses différents axes, une fenêtre similaire au menu précédent s'ouvre et affiche les résultats des dunes repérées, triés par axe.

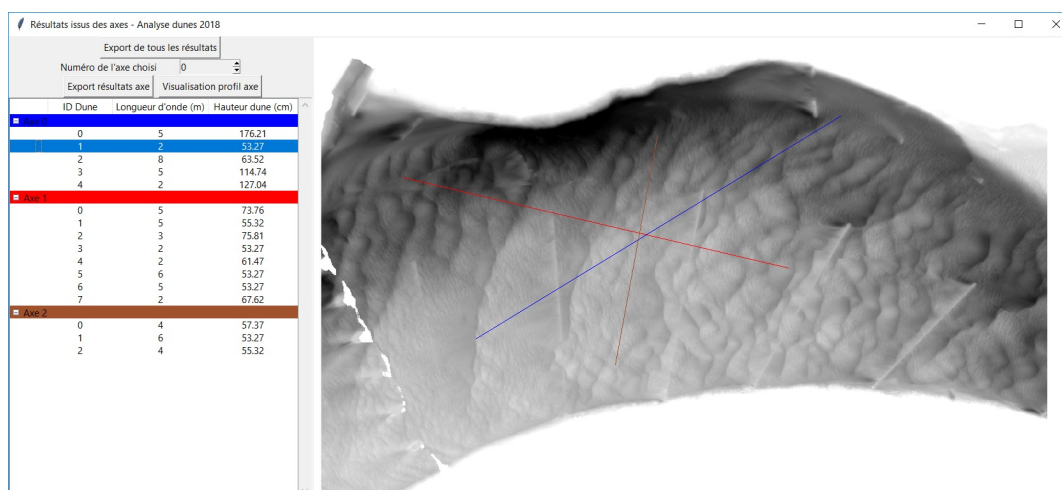


Figure 2 – Interface - Traitement des axes

Depuis cette fenêtre, avec l'association d'une couleur unique à chacun des axes tracés, il devient possible de les distinguer visuellement, et donc rapidement les comparer. Dans l'exemple ci-dessus où trois axes ont été tracés, le tableau (utilisant une arborescence) sur la droite est utilisé pour aisément naviguer dans les résultats obtenus et de connaître leur axe associé.

Les divers boutons en haut sont pour visualiser la courbe de profil d'un axe particulier (suivant le numéro choisi), d'en faire l'exportation des résultats de cette dernière, ou d'exporter l'intégralité des résultats pour l'ensemble des axes.

D'une façon assez similaire, la fenêtre où sont renseignés les résultats de la reconnaissance de dunes adopte l'allure voir page suivante

Comme, la partie du projet traitant sur la reconnaissance de dunes à partir d'une image n'ayant pas atteint un stade suffisant, elle sera non traité pour la suite du rapport en tant que développement, juste des pistes seront mentionnées.

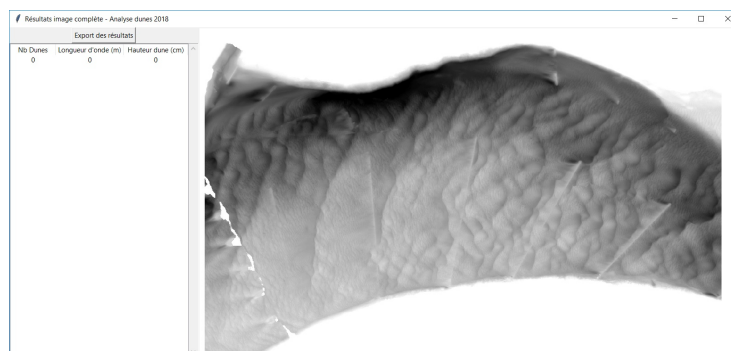


Figure 3 – Interface - Traitement de l'image complète

3 Visualisation du profil d'un axe

Parce qu'afficher simplement une courbe de vue de profil n'est pas des plus intéressant, il a été question de lui rajouter un curseur permettant d'afficher les données de la courbe suivant où se positionne la souris sur le graphique.

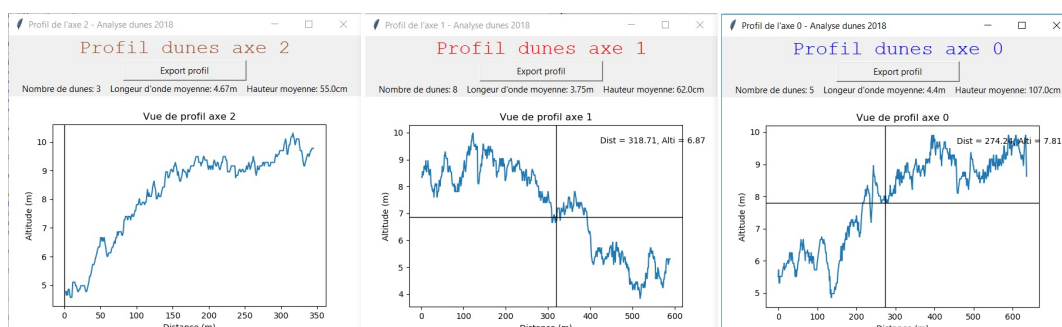


Figure 4 – Interface - Visualisation des profils

Cette fenêtre permet donc ainsi de naviguer sur la courbe et d'en connaître directement les valeurs obtenues (affichées en haut à droite), d'exporter le graphique et de lister les informations globales obtenues sur les dunes repérées (le nombre de dunes ainsi que leur longueur d'onde/hauteur moyenne).

Bien évidemment, la fenêtre est étirable et son graphique s'adapte s'il y a besoin d'être plus précis quant au parcours de la courbe.

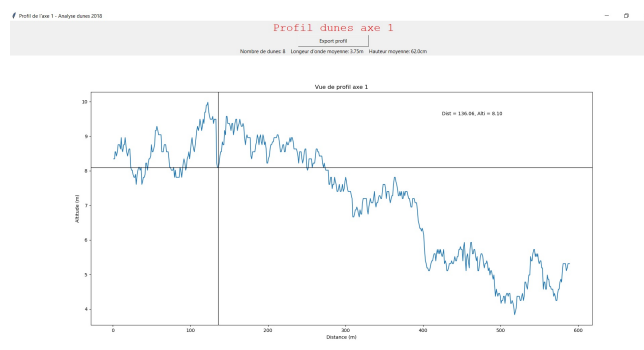


Figure 5 – Interface - Visualisation des profils étirés

Maintenant que nous avons présenté les diverses interfaces graphiques, voyons comment nous pouvons à partir d'une image et de quelques axes déterminer la présence ou non de dune sur ces dernières.

7

Gestion des axes

Avant toute chose, cette fonctionnalité de prime abord plutôt simple à implémenter, tracer des axes sur une image, n'est pas si évidente à concevoir.

- ☐ Tous les tracés effectués par l'utilisateur ne doivent en aucun cas modifier l'image elle-même, elles doivent être en sur-impression.
- ☐ Les tracés des axes en tant que tels sont inutiles pour la recherche de dunes (ils ne sont que des indicateurs visuels pour l'utilisateur), seuls les coordonnées des points de départs et d'arrivées sont utilisées.
- ☐ L'image affichée n'est possiblement qu'une miniature (les images d'origines pouvant être trop grande pour la résolution de l'écran), il faut donc prendre compte de cela.
- ☐ Des cas particuliers d'axes ne doivent pas être autorisés.

Commençons d'abord sur comment placer un axe et la gestion en mémoire de ses derniers.

1 Placer les axes

La méthode choisie pour placer un axe sur l'image est on ne peut plus simple.

- ☐ On clique sur l'image à l'emplacement voulu, un petit point rouge est placé.
- ☐ On clique sur le deuxième emplacement, un autre point est dessiné et le segment les reliant se fait automatiquement, nous obtenons ainsi notre axe.

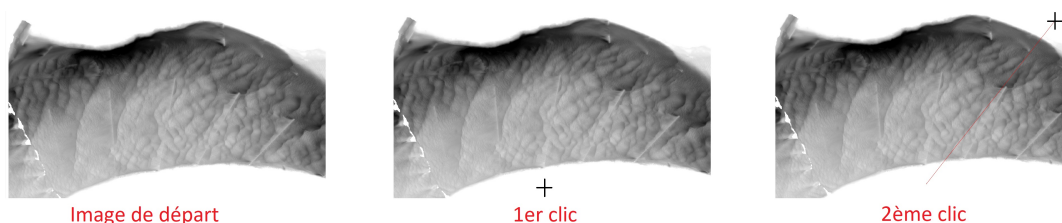


Figure 1 – Ajout d'un axe

Visuellement le résultat semble identique tel que vu dans le 4^{ème} bloc applicatif, néanmoins son fonctionnement s'est vu ajouter plusieurs subtilités pour améliorer l'usage du programme.

- ☐ Plusieurs axes peuvent être tracés (maximum 10).
- ☐ Le dernier axe tracé est en rouge, les autres sont en bleus.

- ☐ Il est possible de supprimer le dernier axe effectué (ou le dernier point placé).
- ☐ Afin d'avoir des axes équivalents, il est possible de recopier le vecteur du dernier axe (celui en rouge) à partir d'un premier point placé.

La capture ci-dessous présente comment on procède à la duplication d'un axe :

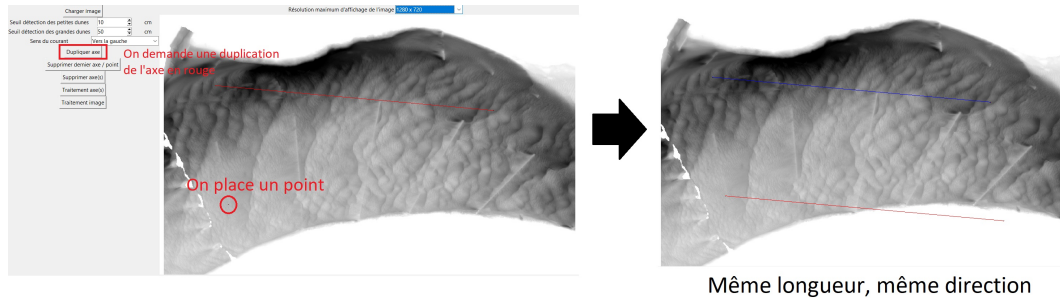


Figure 2 – Dupliquer un axe

2 Gestion des axes

Permettre d'effectuer des tracés sur une image c'est bien, mais gérer les possibles cas où des erreurs peuvent subvenir en avertissant l'utilisateur c'est mieux. Quatre thèmes seront donc abordés.

- Désactiver les boutons au besoin

Suivant si l'utilisateur a placé un seul point, au moins 1 axe ou rien du tout sur l'image, le comportement des boutons permettant la duplication ou suppression des éléments graphiques (axes comme points) ou même l'exécution du traitement de l'image sur les axes, sont actifs ou non. Cela permet ainsi de présenter les actions réalisables, et empêcher les combinaisons impossibles (comme effectuer un traitement d'image sur les axes, alors qu'il n'y en a aucun).

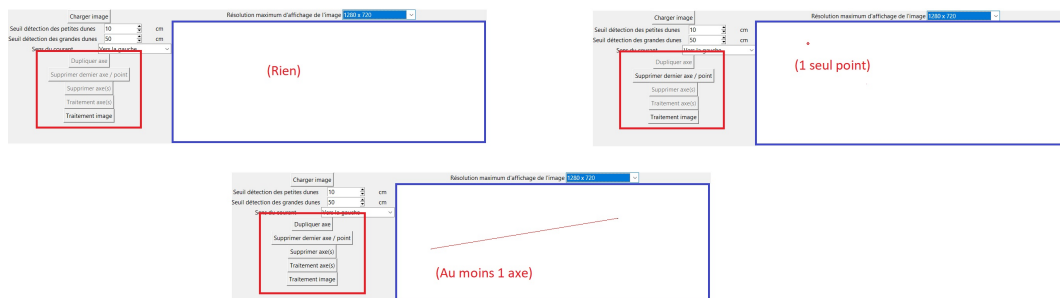


Figure 3 – Gestion des actions sur les axes

- Empêcher les axes à une taille nulle

Comme la recherche des dunes s'effectue en parcourant les pixels entre les deux points définissant l'axe, ce dernier ne peut se permettre d'avoir ses deux extrémités au même endroit sur l'image. Si l'utilisateur tente tout de même de placer les deux points d'un axe au même endroit, un message d'avertissement lui préviendra et annulera le deuxième point d'être placé.

- Empêcher d'obtenir des axes en dehors de l'image

Autant la duplication d'un axe est un procédé utile pour comparer des profils de longueur similaire et avec la même direction, autant cela implique aussi de vérifier que la copie de l'axe ne soit pas positionné en dehors de la taille de l'image. Dans l'image ci-dessus, nous avons voulu dupliquer un même axe à partir de deux points différents. Dans le premier cas (celui du haut),

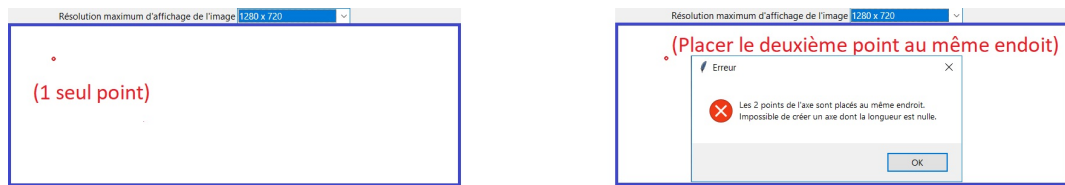


Figure 4 – Gestion de l'erreur de l'axe avec à longueur nulle

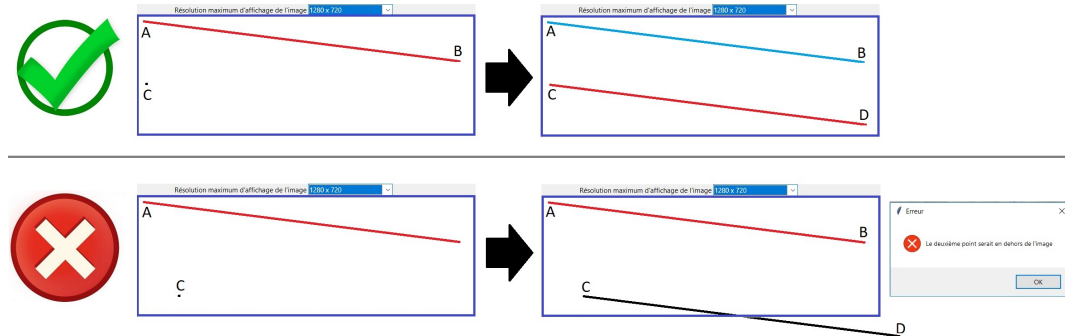


Figure 5 – Gestion de l'erreur des axes en dehors de l'image

une fois que le vecteur \vec{AB} appliqué depuis le point C, nous obtenons donc le point d'arrivée D qui est dans le cadre de l'image, l'axe peut donc être dupliquée. À l'inverse, pour l'exemple du bas, le point D est déterminé comme étant en dehors de l'image, il n'est donc pas possible de dupliquer le segment [AB] avec comme point de départ C.

- Supprimer les axes au changement d'image/résolution

Enfin dernier cas recensé, si l'utilisateur du programme change d'image ou souhaite l'afficher sous une autre résolution, alors il est impératif de supprimer tous les axes précédemment. Ces derniers étant adaptés pour l'ancienne image, un avertissement est alors levé pour informer l'utilisateur que de poursuivre (changer d'image/résolution) entraînera la suppression des axes.

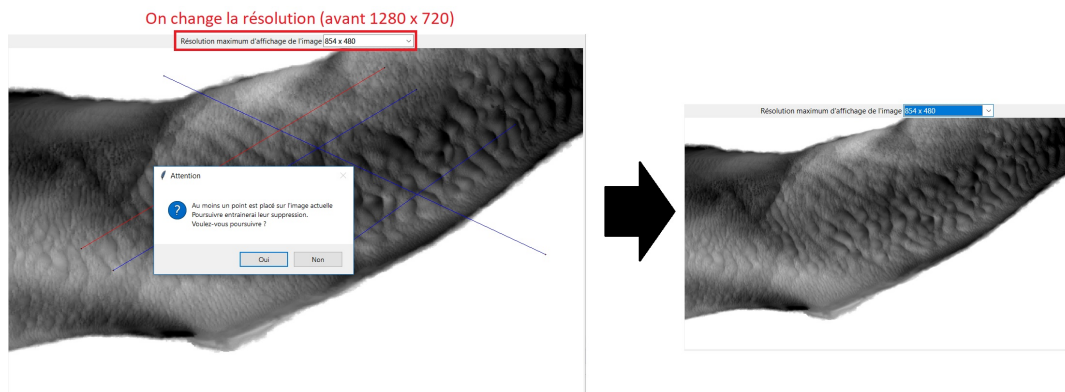


Figure 6 – Gestion du changement d'image/résolution

Maintenant que nous avons vu plus en détails comment son gérer les axes, et comme quoi c'est surtout les extrémités de ses derniers qui nous intéresse, voyons comment à partir de tracés fait par un utilisateur nous pouvons en repérer des dunes.

8

Détection des dunes sur un axe

Après que l'utilisateur est tracé ces différents axes sur la miniature de l'image, il y a plusieurs étapes avant qu'il puisse récupérer les caractéristiques des dunes passant par l'axe, à commencer par transposer l'axe sur l'image originale.

1 Transposer les axes sur l'image d'origine

Comme les images où le traitement d'image se fait peuvent faire une taille très importante, l'affiché en taille réelle sur l'interface utilisateur rendrai la fenêtre beaucoup trop grande par rapport à la taille de l'écran, c'est pourquoi c'est une miniature qui y est affichée. Cependant, cela implique donc que les axes tracé sur les miniatures ne peuvent pas correspondre directement au même axe sur l'image originale, une adaptation est donc à faire.

Comme nous l'avons vu lors des explications sur la gestion des axes, nous nous contentons de sauvegarder les coordonnées des points de chaque extrémité des axes. Pour les adapter à l'image originale, il suffit de multiplier les coordonnées (x, y) de chacun des points par le coefficient inverse qui a fait "réduire" la copie de l'image pour en faire sa miniature.

Par exemple :

- ☐ La miniature de l'image a ses dimensions égales à 0,4 fois celles de l'originale.
- ☐ Si sur la miniature nous traçons un axe partant du pixel en position (8, 8) vers celui en position (96, 56).
- ☐ Les coordonnées des points aux extrémités de l'axe sur la miniature sont donc pour le pixel de départ $(8 * (1/0,4), 8 * (1/0,4)) = (20, 20)$.
- ☐ Et les coordonnées du pixel d'arrivé $(96 * (1/0,4), 56 * (1/0,4)) = (240, 140)$.

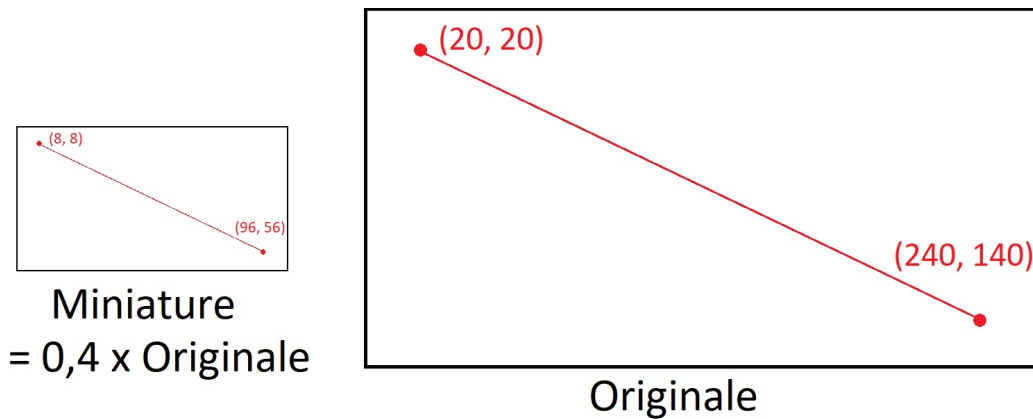


Figure 1 – Transposer un axe de la miniature vers l'image originale

2 Déterminer les pixels de l'image à prendre

Maintenant que nous avons placé nos deux points extrêmes de l'axe sur l'image originale, nous devons choisir les pixels les plus appropriés que l'on doit choisir pour par la suite, déterminer s'il y avait une dune ou non.

Reprenons l'exemple précédent avec notre axe qui part du point aux coordonnées (20, 20) vers celui en (240, 140), pour savoir quels sont les pixels à prendre, on va déjà savoir combien de pixels nous allons devoir récupérer. Pour ce faire, on détermine quel est le déplacement horizontal ou vertical le plus grand pour passer du point de départ à celui d'arrivée.

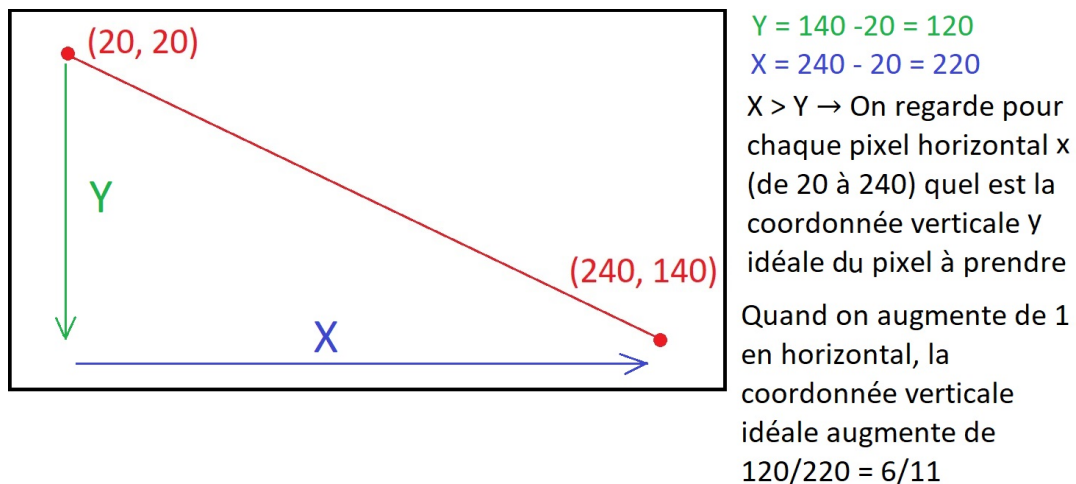


Figure 2 – Déterminer les pixels à prendre 1

Nous avons donc un vecteur (220, 120) pour aller du point de départ à celui d'arrivée, la coordonnée horizontale étant la plus grande, nous avons donc besoin de prendre 221 pixels (220 + 1). Pour chacune des valeurs horizontales de l'image de 20 à 220, nous avons donc besoin de connaître la valeur idéale de la coordonnée verticale pour ensuite savoir ou prendre le pixel dans l'image. Sachant aussi qu'à chaque fois que l'on augmente de 1 en horizontalité, l'on augmente de $220/120 = 6/11$ en verticalité

Le premier pixel que l'on prend et bien évidemment celui positionné aux coordonnées (20, 20), mais comment savoir quel est le pixel à prendre quand nous sommes à la colonne 80 de l'image (horizontalité = 80)?

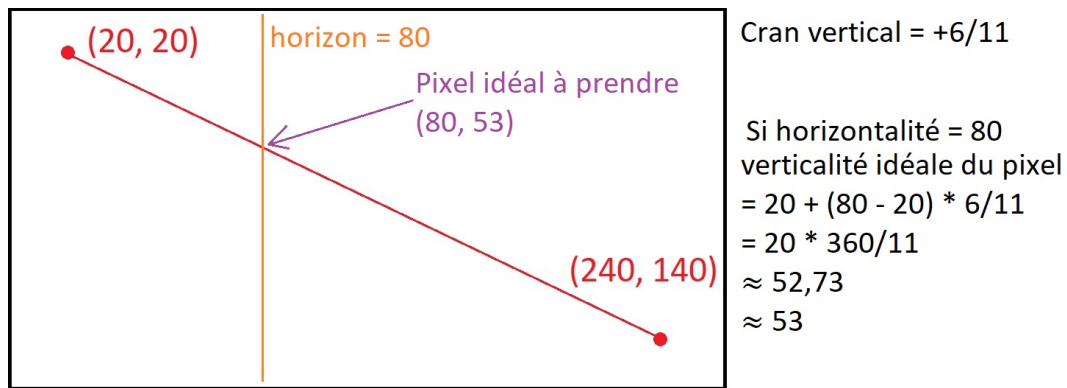


Figure 3 – Déterminer les pixels à prendre 2

Il suffit donc de procéder ainsi pour toutes les valeurs horizontales de 21 à 240 (le pixel de départ à 20 est déjà connu) et ainsi nous avons la liste de tous les pixels à prendre pour ensuite passer à la détection de dunes.

À noter que l'exemple ci-dessus concerne le cas pour un axe, dont son vecteur associé a une composante plus grande en verticale qu'en horizontale, que les ses composantes X et Y du vecteur sont toutes les deux positives, et que l'on suppose que le sens du courant et de la gauche vers la droite. Mais dans le cas de la récupération des pixels en fonction de la manière dont l'axe est tracé (et le sens du courant) suivant les critères déjà énoncés, la formule pour récupérer les pixels change quelque peu.

Il existe ainsi 8 types de tracés différents que peut effectuer l'utilisateur, auquel on peut encore associé le sens du courant ce qui fait 16 combinaison pour lesquelles l'algorithme de choix du pixel doit prendre en compte.

	Y positif	Y négatif	
X positif			Verticalité dominante
Y négatif			Horizontalité dominante

Figure 4 – Les types de tracés

Influence du sens du courant dans tout cela ?

Cela est encore une fois plutôt simple à comprendre, tel que son sauvegardé les axes, le premier point placé par l'utilisateur parmi les deux que comporte l'axe est qualifié de point de départ. Hors, si l'utilisateur choisi que le sens du courant est de la droite vers la gauche, alors que le premier point de l'axe est placé plus à gauche que l'autre, il faut alors intervertir le point de départ et celui d'arrivée. Mais attention : cet intervertissement se fait seulement pour la détermination des pixels, on n'intervertit pas les coordonnées des 2 points qui forme l'axe (dans la classe Axe), sinon cela changerait le comportement pour une duplication de l'axe.

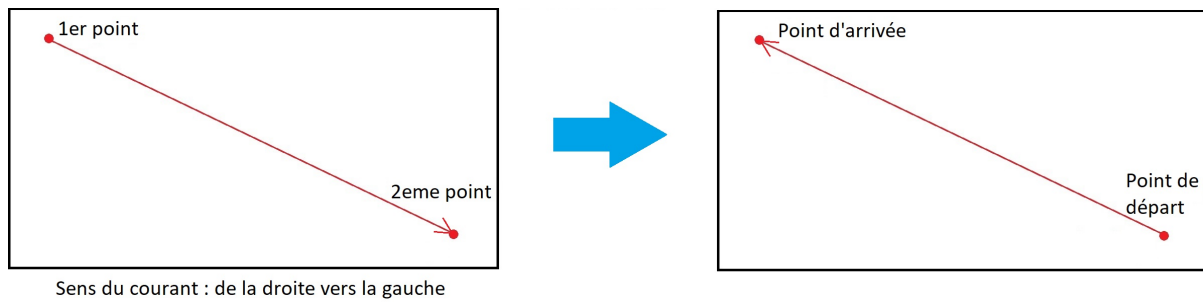


Figure 5 – Influence du sens du courant sur le choix du pixel

Ainsi on peut se retrouver à changer qui est le point de départ et qui est le point d'arrivée, et cela entraîne bien évidemment aussi un changement du type de tracé.

Au sein de l'algorithme de retour de la liste des valeurs des pixels à prendre, nous retournons deux listes de données :

- ☐ La liste ordonnée des valeurs (des niveaux de gris) pour tous les pixels que nous avons choisis.
- ☐ Une autre liste ordonnée de la distance entre le point de départ et le pixel choisi. Le calcul de la distance se faisant via la formule du théorème de Pythagore.

Cette deuxième liste permettra de connaître la distance entre chaque creux d'une dune et sa crête, pour ainsi obtenir sa longueur d'onde.

3 Reconnaître des dunes

Maintenant que nous avons la liste des valeurs de niveau de gris ordonnées des pixels choisis suivant un axe établi, que nous avons de plus la liste des distances entre les pixels, nous pouvons tout simplement déterminer s'il y a des dunes à détecter ou non.

Tout d'abord, une dune commence forcément par un creux, nous parcourons donc notre liste des valeurs de gris tant que la valeur obtenue ne fait que descendre (ou stagner). Une fois atteinte, on garde en mémoire son niveau d'altitude, et sa distance par rapport au point de départ.

Puis, nous allons chercher la crête de la dune, nous parcourons donc la liste des niveaux de gris, et cela tant que la valeur lue ne fait qu'augmenter (ou stagner). Nous regardons alors la valeur de niveau de gris :

- ☐ Si la valeur mesurée est 255 (pour une image 8 bits), c'est que nous avons atteint la surface, nous avons non pas détecté une dune mais une côte, on fera en sorte par la suite de forcer que nous sommes pas sur une dune (par exemple une distance par rapport au point de départ négative).
- ☐ Sinon, on garde en mémoire le niveau d'altitude de la crête, ainsi que sa distance avec le point de départ.

Il ne reste plus qu'à déterminer le deuxième creux de la dune, et pour se faire, on continue de parcourir la liste des niveaux de gris, et cela tant que la valeur diminue (ou stagne), on récupère une fois de plus le niveau d'altitude et la distance par rapport au point de départ.

Maintenant que nous avons récupéré l'ensemble des données d'une possible dune, encore faut-il qu'elle respecte la hauteur de seuil minimum qui a été défini par l'utilisateur (10 cm par défaut).

Pour cela il faut déterminer quel creux est le plus proche de la crête (d'où l'intérêt de garder en mémoire les distances par rapport au point de départ), puis de soustraire l'altitude de ce dernier à l'altitude de la crête pour enfin connaître la hauteur de la dune. Quant à la longueur

d'onde, il suffit de soustraire la distance point de "départ/premier creux" à la distance "point de départ/deuxième creux". Voir le schéma **Figure 4** (Chapitre 2) pour se rappeler de comment est déterminé les caractéristiques d'une dune.

Bien évidemment l'axe tracé par l'utilisateur n'étant pas forcément composé que d'une seule dune, la procédure énoncée ci-dessous est à refaire tant que nous n'avons pas fini de parcourir l'intégralité des niveaux de gris de la liste (la valeur des pixels choisis).

4 Export texte des résultats

Une fois que les dunes ont donc été reconnues et que les résultats de leurs caractéristiques sont disponibles, ils conviendrait de pouvoir exporter ses résultats sous un format texte.

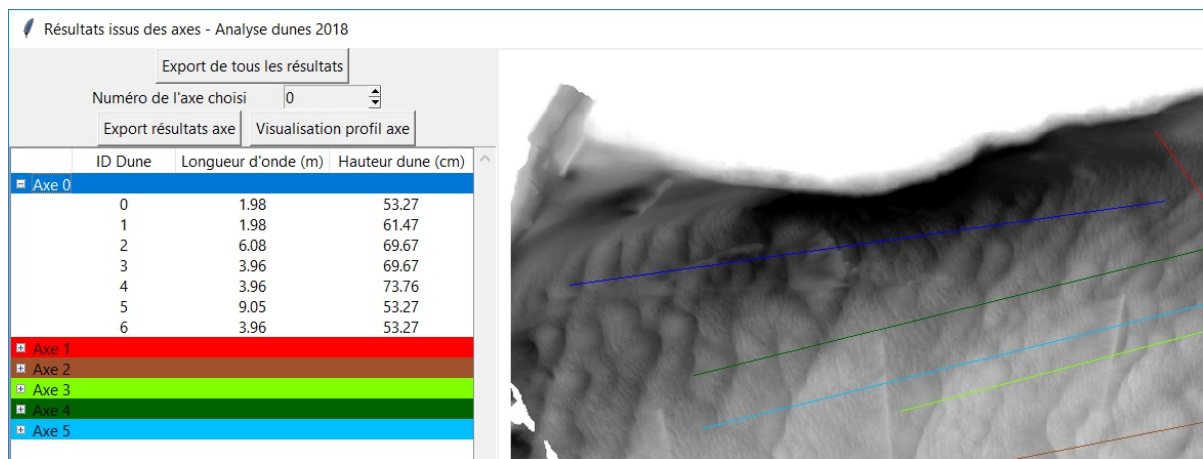


Figure 6 – Export des résultats au format texte

Ainsi, via le menu de l'interface il est possible de demander l'export des résultats pour un axe dédié, ou pour l'ensemble des axes tracés. Bien évidemment suivant le choix opéré l'entête du fichier texte est plus ou moins conséquent pour prendre en compte l'ensemble des informations à y inclure.

Dans tous les cas l'en-tête global du fichier texte de sortie retrace les informations suivantes :

- ☐ De quelle image provient l'analyse.
- ☐ Quant a été fait l'exportation des résultats.
- ☐ Le seuil de détection des petites dunes (quand cela sera implémenté, inclure aussi le seuil des détection des grande dunes).
- ☐ Dans quel sens le courant a été décidé de couler.
- ☐ Les coordonnées des points constituant l'axe étudié, ainsi que des données générales sur les dunes repérées dessus.
- ☐ la légende du tableau pour le listing des dunes.
- ☐ Les informations dunes par dunes.

Là où les deux types de fichiers diffèrent, c'est que celui traitant de l'ensemble des dunes associe un numéro à chacun des axes (celui provenant de l'application), et donc à chacune des dunes listées, à quel axe il est rattaché.

```
Traitement analyse dunes de l'image " Exemple_1,95_12,4.tif " fait le 02-04-2018 15:25:38
Altitude minimum = 1.95 Résolution altitude = 0.04098
Detection des dunes (petites) dès 0.5m
Le sens du courant choisi est vers la gauche
Numéro axe choisi 0 constitué des pixels aux extrémités ((71, 231), (652, 149))
Nombre de dunes = 7 | Longueur d'onde moyenne = 4.42m | Hauteur moyenne 62.0cm
IdDune;Longueur d'onde (m);Hauteur (cm);AltitudeCreux1 (m);AltitudePic (m);AltitudeCreux2 (m)
0;1.98;53.27;4.24;5.19;4.65
1;1.98;61.47;3.63;4.65;4.04
2;6.08;69.67;3.22;3.92;3.22
3;3.96;69.67;3.02;3.71;3.02
4;3.96;73.76;3.63;4.37;3.63
5;9.05;53.27;3.3;3.84;2.89
6;3.96;53.27;2.89;3.43;2.89
```

Figure 7 – Export des résultats pour un axe

```
Traitement analyse dunes de l'image " Exemple_1,95_12,4.tif " fait le 02-04-2018 15:20:20
Altitude minimum = 1.95 Résolution altitude = 0.04098
Detection des dunes (petites) dès 0.5m
Le sens du courant choisi est vers la gauche
Numéro axe 0 constitué des pixels aux extrémités ((71, 231), (652, 149))
Nombre de dunes = 7 | Longueur d'onde moyenne = 4.42m | Hauteur moyenne 62.0cm
Numéro axe 1 constitué des pixels aux extrémités ((642, 81), (773, 261))
Nombre de dunes = 3 | Longueur d'onde moyenne = 3.13m | Hauteur moyenne 94.0cm
Numéro axe 2 constitué des pixels aux extrémités ((265, 448), (877, 328))
Nombre de dunes = 8 | Longueur d'onde moyenne = 5.54m | Hauteur moyenne 94.0cm
Numéro axe 3 constitué des pixels aux extrémités ((964, 204), (394, 354))
Nombre de dunes = 5 | Longueur d'onde moyenne = 3.3m | Hauteur moyenne 55.0cm
Numéro axe 4 constitué des pixels aux extrémités ((192, 319), (908, 142))
Nombre de dunes = 8 | Longueur d'onde moyenne = 3.06m | Hauteur moyenne 71.0cm
Numéro axe 5 constitué des pixels aux extrémités ((200, 371), (916, 194))
Nombre de dunes = 9 | Longueur d'onde moyenne = 3.77m | Hauteur moyenne 82.0cm
IdAxe;IdDune;Longueur
d'onde (m);Hauteur (cm);AltitudeCreux1 (m);AltitudePic (m);AltitudeCreux2 (m)
0;0;1.98;53.27;4.24;5.19;4.65
0;1;1.98;61.47;3.63;4.65;4.04
0;2;6.08;69.67;3.22;3.92;3.22
0;3;3.96;69.67;3.02;3.71;3.02
0;4;3.96;73.76;3.63;4.37;3.63
0;5;9.05;53.27;3.3;3.84;2.89
0;6;3.96;53.27;2.89;3.43;2.89
1;0;5.0;127.04;4.98;6.46;5.19
1;1;2.2;81.96;6.13;6.54;5.72
1;2;2.2;73.76;5.72;5.93;5.19
2;0;3.93;106.55;8.1;9.16;8.34
2;1;1.96;53.27;8.42;8.75;8.22
2;2;1.96;53.27;9.16;9.7;9.16
2;3;3.93;73.76;8.75;9.49;9.37
```

Figure 8 – Export des résultats pour tous les axes

5 Le traitement sur l'image complète

Le traitement de l'image dans son intégralité n'ayant pas eu le temps d'être implémenté, la conception pour la création d'une variante de la ligne de partage des eaux adaptée à notre problème (celle de détecter des dunes au fond d'un bassin) a donc été mise de côté pour le développement de l'application.

Néanmoins afin de ne pas devoir repartir sur des bases vides pour une possible poursuite de ce projet, il est possible facilement de faire du traitement de la ligne de partage des eaux avec l'application **Fiji**, auquel on y rajoute le plugin MorphoLibJ dont on y trouve la procédure d'installation de ce dernier **ici**.

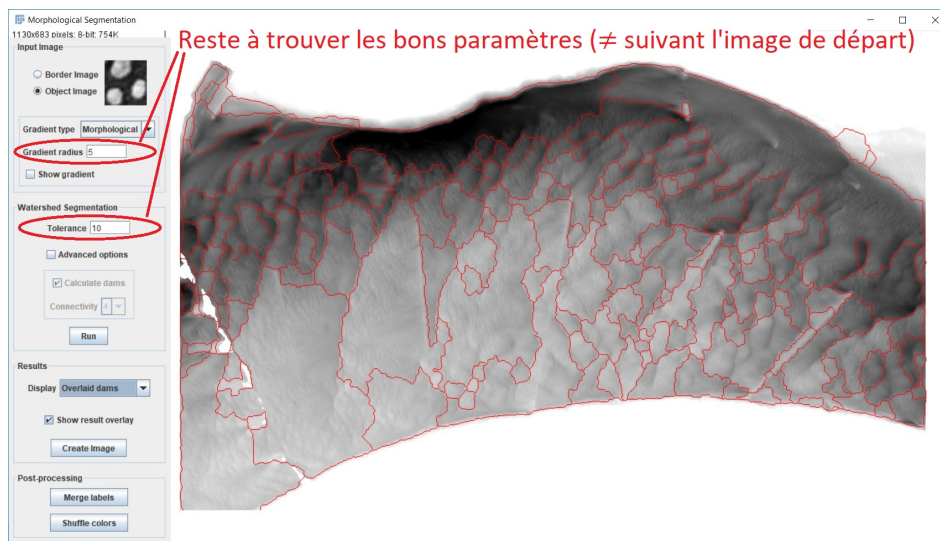


Figure 9 – Fiji / MorphoLibJ pour de la segmentation d'image

N'ayant cependant aucune idée des paramètres à définir pour qu'il puisse délimiter les dunes, je ne peux non plus poursuivre sur cette voie. Le traitement de l'image entière reste donc un pan encore à définir et incorporer.

9

Programme de génération d'image TIF 16 bits

L'un des possibles problèmes de ce PRD est le manque concret de précision des images pour effectuer une reconnaissance de dunes. En effet, si l'on prend l'exemple d'une image codée sur 8 bits, avec comme altitude minimum 1,95m et comme maximum 12,4m, cela nous laisse donc une précision entre deux niveaux de gris de :

$$(12,4 - 1,95)/(2^8 - 1) = 0,04098m \approx 4,1cm \quad (1)$$

Avec une précision aussi faible, et une détection des dunes dès 10cm, l'usage d'images dont chaque pixel est codé sous 8 bits, rendrai les résultats assez peu fiable, il suffirait de 3 variantes de gris pour détecter la dune. Ainsi a été donc envisagée l'idée de rechercher comment à partir des données ArcGIS, pouvoir générer des images noir et blanc 16 bits. En créant des images au format 16 bits, la précision serai ainsi de :

$$(12,4 - 1,95)/(2^{16} - 1) \approx 0,000159m \approx 0,016cm \quad (2)$$

Les images originales étant effectuées avec un écho-sondeur centimétrique, la réelle précise serai donc de 1cm, soit une valeur bien plus exploitable pour détecter des dunes à partir de 10cm.

Pour générer des images sous divers formats, il est utilisé l'outil "Raster Copy", néanmoins il n'a jamais été possible de générer une quelconque image de cette manière, la même erreur étant toujours levée, et aucun moyen d'y résoudre.

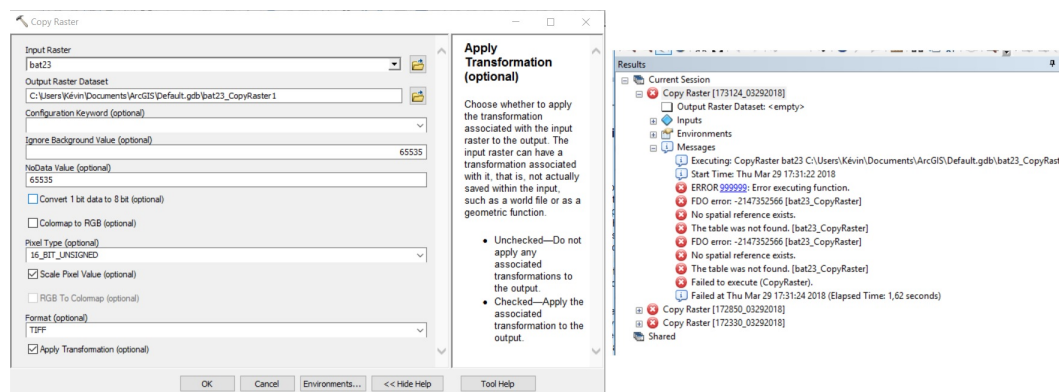


Figure 1 – ArcGIS - bug de l'outil "Copy Raster"

En cherchant plus loin, il a donc été question de voir directement les fonctions utilisées pour la génération de l'image, et ce fut celle-ci, et avec elle de créer un programme Python (à la version 2.7 pour gérer la librairie ArcGIS) permettant de générer les images.

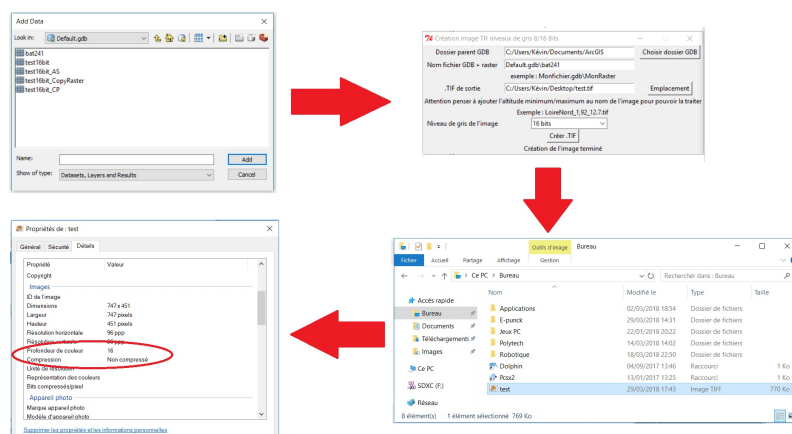


Figure 2 – Génération d'une image TIF 16 bits

Malheureusement, toutes les images générées par le programme sont illisibles par la librairie Pillow (celle de référence en Python pour le chargement / l'affichage d'image) utilisée par l'application, et ce peu importe le format de compression choisie.

10

Gestion de projet

Pour ce qui touche à la gestion du PRD durant cette partie développement, celui-ci est bien différent entre le théorique **Figure 13** (Chapitre 2) et le pratique avec le diagramme de Gant ci-dessous.

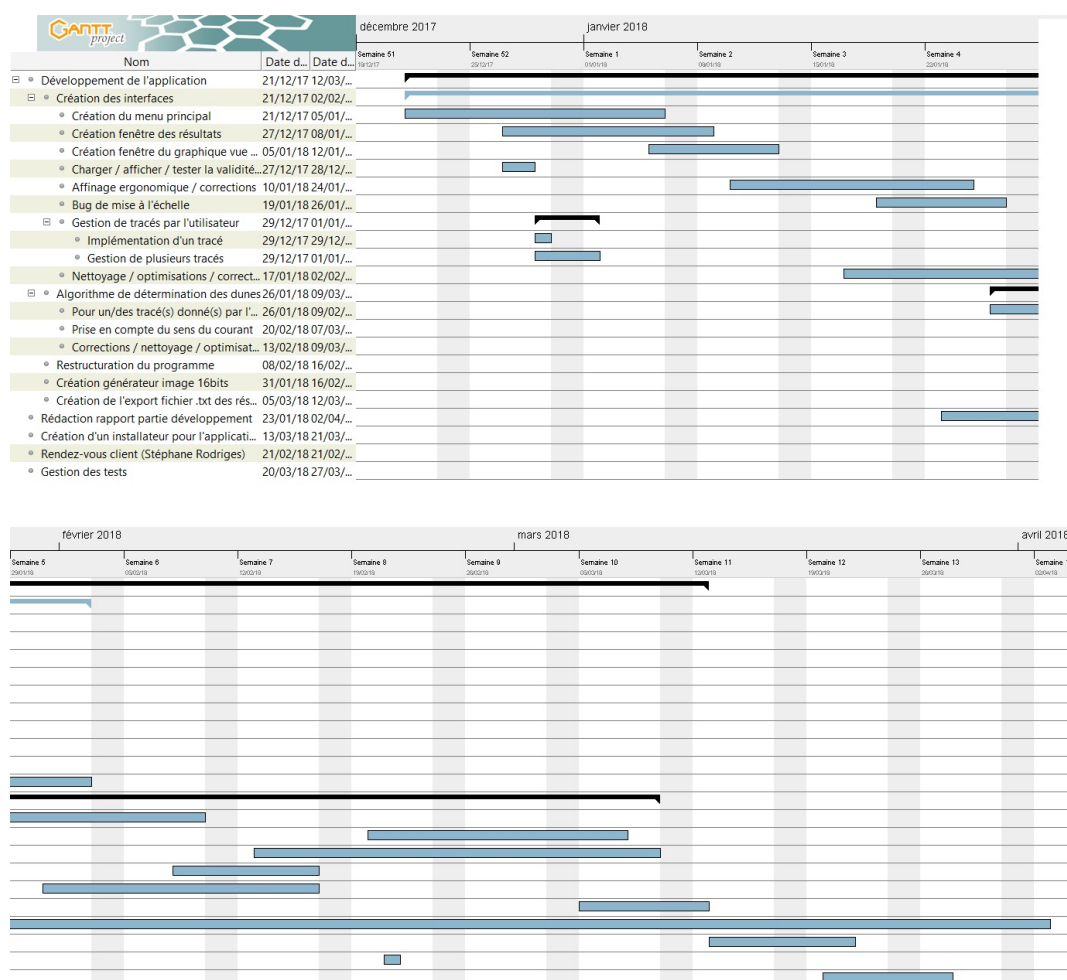


Figure 1 – Diagramme de Gant réel du S10

Pour commencer, il avait été prévu de créer les différentes interfaces durant les mois de décembre

et janvier, cependant à cause d'un bug d'affichage, l'image affichée ne respectait pas la résolution définie. C'est après plusieurs tests, qu'il s'est avéré que l'image affichée avait toujours sa résolution environ 1,25 fois plus grande que celle indiquée. La résolution de ce bug était primordiale, sans valeur d'échelle exacte, la transposition des coordonnées des points d'un axe de la miniature (image affichée à l'utilisateur) à l'image originale entraînerait des problèmes de correspondance.

C'est donc après plusieurs jours de recherche, à laquelle je procédé aussi aux nettoyages et corrections des interfaces, qu'il s'est avéré que le bug provenait d'une non prise en charge du facteur de mise à l'échelle des ordinateurs sous Windows. Mon ordinateur ayant par défaut un niveau à 125%, on retrouve ainsi le facteur d'échelle d'agrandissement des images. Ce bug est toujours d'actualité est a été remonté depuis un an déjà, plus d'informations [ici](#), ce rapport de bug a aussi fourni un code permettant de corriger le problème.

Pendant la phase d'ajustement des interfaces, il a été question de prendre à bras-le-corps le problème d'impossibilité de générer des images en noir et blanc sur 16 bits avec ArcGIS, prérequis pour avoir une précision suffisante pour faire une analyse des dunes. Au commencement par la recherche sur la documentation en ligne du logiciel comment cela pourrait être possible, ainsi est venu l'idée d'utiliser l'outil "Copy Raster", mais ce dernier n'a jamais pu générer une simple image. il a donc été question de voir directement les fonctions utilisées pour la génération de l'image, il a été possible de générer une image mais les résultats obtenus ne sont pour autant pas utilisable. La génération d'images en noir et blanc 16 bits a donc été repoussée pour une date ultérieure.

Une fois les interfaces créées, la conception de l'algorithme de détection des dunes sur un axe a été la prochaine étape. Cette dernière est découpée en deux parties distincts, déterminer les pixels à prendre sur l'image originale en fonction de l'axe, et déterminer les dunes suivant le parcours de la liste des valeurs des pixels choisis. Les difficultés ont été surtout présentes pour la première partie, en effet comment déterminer si tel ou tel pixel doit-être sélectionné ou non, et qui plus est, il faut que l'algorithme puisse s'adapter suivant l'axe de l'utilisateur (sa longueur, sa direction et son sens).

Le 21 février il a été question de se réunir avec le client (une avait été prévue le 19 janvier, mais à cause de multiples projets et d'un entretien pour un stage, il n'a pas pu aboutir), lors de celle-ci une remarque au sujet de la prise du sens du courant fut posée, et donc l'algorithme de sélection des pixels à prendre à du être modifié pour le prendre en compte.

Pendant la mi-février, l'architecture complète du programme a été revu afin d'isoler les interfaces des objets manipulées (images des dunes, les axes tracés par l'utilisateur) et des algorithmes pour déterminer les dunes. Cette restructuration était nécessaire pour rendre plus souple et lisible l'ensemble du projet.

Ensuite est venue la création d'un module permettant d'exporter les résultats obtenus des dunes trouvées par le traitement d'un axe ou de l'image entière (le traitement de la ligne de partage des eaux n'étant pour autant pas implémenté), pas de problème rencontrée.

Enfin, le temps restant a été consacré à la mise en place des installateurs (Linux et Windows), la rédaction des livrables, et la création des tests unitaires aux différentes classes existantes, ces tests ont ainsi permis de consolider le programme en retournant quelques erreurs lors de cas spécifiques, erreurs qui ne sont plus d'actualité.

Pour faire un bilan de ce qu'il reste à faire :

- Il n'y a pas de moyen sûr de générer des images noir et blanc sur 16 bits.
- Le traitement de l'image entière pour y détecter les dunes par un algorithme modifié de la ligne de partage des eaux reste à faire.
- Il n'y a pas de récupération des dunes jugées grosses, seules les petites sont actuellement repérées.

Il y a encore de nombreuses choses à implémenter pour l'application de ce PRD, néanmoins le manque de documentations (ArcGIS) et informations sont des critères impératifs pour voir poursuivre le développement de ce projet, critères qui sont à ce jour toujours pas remplis.



Conclusion

Pour répondre au problème de détermination des caractéristiques de dunes fluviales présentes depuis une image, la première interrogation fut tout d'abord de les reconnaître. Trois pistes ont été envisagées pour retrouver les dunes sur l'image, mais aucune d'entre elles n'est parfaite.

La première, est de procéder par la reconnaissance de textures en utilisant une matrice de co-occurrence, néanmoins l'usage de ce procédé implique une réduction significative de la résolution de l'image, rendant ainsi plus possible de différencier les dunes des simples ondulations.

La deuxième, consiste à l'utilisation d'une transformée de Fourier, il faut tout d'abord obtenir un nombre d'images types de dunes, obtenir leur transformée de Fourier, puis comparer la similarité de ces dernières avec des parcelles de l'image choisies par l'utilisateur.

La dernière, usant de l'algorithme de ligne de partage des eaux a pour but de connaître les frontières séparant chacune des dunes. Néanmoins, les implémentations existantes ne récupèrent que les frontières qui sont proches de la surface, impossible donc de retrouver les dunes en zones profondes. Une variante sera donc à développer pour traiter sur l'ensemble de la profondeur, et ne récupérer que les zones dont le dénivelé soit jugé suffisant pour être qualifié comme une dune.

Durant le S10, le développement s'est consacré au départ par les interfaces, afin d'obtenir un retour actif du client sur les choix opérés. Sitôt ces dernières créées, il a été question de créer un utilitaire pour produire des images au bon format (16 bits). Ensuite d'implémenter le moyen de reconnaître les dunes sur des axes que l'utilisateur aura préalablement tracés. Mais à cause de nombreux soucis rencontrés issus pour la plupart de bugs provenant de bibliothèques reconnues (Pillow) ou mal par une mauvaise documentation (ArcGIS), le retard c'est accumulé et il reste encore beaucoup de fonctionnalités à intégrer dans l'application.

Annexes

A

Guide développeur

Dans le but de récupérer le projet afin de la poursuivre, celui-ci est disponible sous un dépôt Github ouvert [Dépôt GitHub](#), ce dernier contient toutes les informations nécessaires pour bien démarrer.

Afin de ne pas à surcharger inutilement le dépôt Github de librairies pouvant atteindre des dizaines de MO, le fichier README associé au projet contient tous les liens utiles sur le [site pypi.python.org](#) pour les trouver à la version utilisée pour le projet.

Ainsi, une fois une copie du dépôt obtenue et l'installation effectuée (Python et ses librairies), il ne suffira plus qu'à modifier directement le code soit à partir d'un simple éditeur de texte, soit d'un IDE comme PyCharm ou Eclipse (avec le plugin PyDev).

Interfaces	28/03/2018 16:22	Dossier de fichiers	
Tests	28/03/2018 15:16	Dossier de fichiers	
TraitementImage	28/03/2018 16:22	Dossier de fichiers	
Exemple_1,95_12,4	15/02/2018 14:29	Image TIFF	755 Ko
Exemple2_5,548_11,20	08/03/2018 15:29	Image TIFF	269 Ko
Installeur Analyse Dunes Unix	21/03/2018 18:35	Shell Script	1 Ko
Installeur Analyse Dunes Windows (en m...	22/03/2018 16:38	Fichier de comma...	4 Ko
Main	08/03/2018 15:43	Python File (no co...	1 Ko
README.md	21/03/2018 18:39	Fichier MD	4 Ko

Figure 1 – Contenu du dépôt GitHub

Les différents fichiers du projet sont découpés en plusieurs dossiers

- ☐ "Interfaces" contient les fichiers python associés au code pour créer chacune des interfaces.
- ☐ "TraitementImage" est le répertoire contenant l'ensemble des classes et modules Python utilisés par le programme.
- ☐ "Test" contient les tests unitaires pour effectuer quelques tests pour vérifier le bon comportement des classes implémentées.

Les différentes classes implémentées dans le projet sont "GestionAxes", "Axe", "Point" et "ImageDune"; "AlgorithmmeAxe", "AlgorithmeImageComplete" et "ExportTXT" quant à eux sont des modules. En voici le diagramme de classe de la structure :

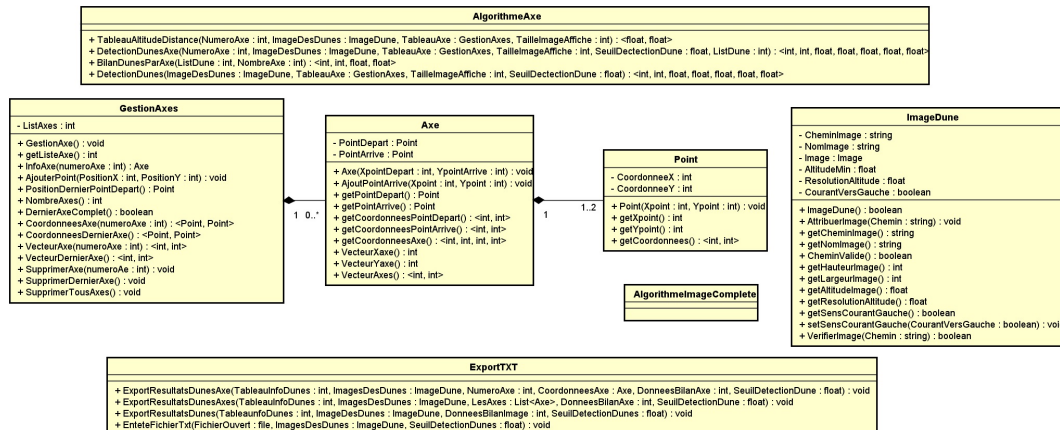


Figure 2 – Diagramme de classe

Pour effectuer les tests unitaires, il est conseillé d'utiliser l'IDE Python choisi plutôt que d'utiliser les lignes de commande (via la commande "python -m unittest discover"), ces dernières renvoient des faux positifs pour certains tests, rendant ainsi non-fiable leur exécution via un terminal.

```

C:\Windows\System32\cmd.exe
C:\Users\Kévin>python -m unittest discover
EE.....F.....EEE.FFFFE.....
ERROR: testInitialisationAxe (test_Axe.test_Axe)
-----
Traceback (most recent call last):
  File "C:\Users\Kévin\workspace\AnalyseDunesFluviales\DossierGithub\AnalyseDunesFluviales\Tests\test_Axe.py", line
  9, in testInitialisationAxe
    PointDepart = MonAxe.getCoordonneesPointDepart()
AttributeError: 'Axe' object has no attribute 'getCoordonneesPointDepart'

ERROR: testPointArriveAxe (test_Axe.test_Axe)
-----
Traceback (most recent call last):
  File "C:\Users\Kévin\workspace\AnalyseDunesFluviales\DossierGithub\AnalyseDunesFluviales\Tests\test_Axe.py", line
  19, in testPointArriveAxe
    PointArrive = MonAxe.getCoordonneesPointArrive()
AttributeError: 'Axe' object has no attribute 'getCoordonneesPointArrive'

ERROR: testCheminImage (test_ImageDune.test_ImageDune)
-----
Traceback (most recent call last):
  File "C:\Users\Kévin\workspace\AnalyseDunesFluviales\DossierGithub\AnalyseDunesFluviales\Tests\test_ImageDune.py",
  line 62, in testCheminImage
    self.assertEqual(MonImage.getCheminImage(), NomImage, "Echec du nom de l'image")
AttributeError: 'ImageDune' object has no attribute 'getCheminImage'

```

Figure 3 – Tests unitaires via un terminal

À l'inverse, à partir de son IDE Python comme ici avec Eclipse, il suffit d'exécuter les tests unitaires pour savoir combien on réussit et lesquels sont échoués, il n'y a pas de faux positif trouvé. Pour plus d'informations sur la création des tests unitaires, ceux implémentés utilisent

```

1# from PIL import Image
2 from TraitementImage import ImageDune
3 import unittest
4
5 NiveauDeGrisDifferent = 256
6 NomImage1 = "Exemple_1_95_12_4.tif"
7 NomImage2 = "Exemple2_5_548_11_20.tif"
8 NomImageEchec = "Echec test.tif"
9 NomImageIncomplet = "Exemple2_5_548.tif"
10 NomImageExistePas = "Exemple3_5_548_11_20.tif"
11
12 class test_ImageDune(unittest.TestCase):
13
14     # L'image existe
15     def testImageDuneValide1(self):
16         MonImage = ImageDune.ImageDune()
17         ImageValide = MonImage.AttribuerImage(NomImage1)
18         self.assertEqual(ImageValide, True, "Echec Test image")
19

```

Figure 4 – Tests unitaires via un l'IDE Eclipse

le framework unittest (implémenté de base avec l'installation de Python) et la documentation pour les créer se trouve à l'adresse suivante [Documentation unittest](#).

B

Installation

Pour installer le projet, ce dernier peut-être téléchargé librement sur le dépôt GitHub suivant [Dépôt GitHub](#), le fichier README associé contient tous les liens nécessaires pour trouver les fichiers non inclus dans le dépôt mais utile pour procéder à une installation.

Une fois la récupération des fichiers du dépôt Github (en .zip ou à partir d'un git clone), et si Python et les librairies utilisées ne sont pas installées, il suffit d'exécuter le script correspondant à son système d'exploitation (le fichier .bat pour windows et .sh pour les environnements Unix) pour installer le nécessaire.

Prenons par exemple le déroulement du script d'installation pour Windows, celui-ci doit-être exécuté en mode administrateur afin d'obtenir les permissions pour les installations. Si nous essayons d'exécuter le script sans rien mettre en plus dans le dossier du projet, et que nous n'avons pas Python 3.6.X d'installé sur notre machine, on nous prévient qu'il faut télécharger l'exécutable de python (python-3.6.4-amd64.exe) et de la déposer dans le répertoire du projet.

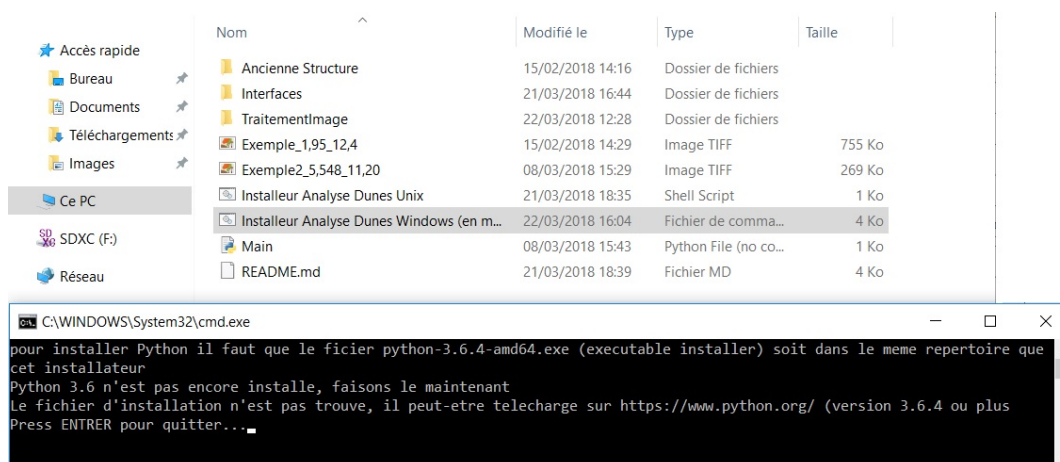


Figure 1 – Installation 1

Si l'on rajoute donc le fichier d'installer de Python dans le répertoire du projet, au lancement de l'installateur, l'exécutable python sera lancé et le script signalera qu'il faut l'installer à l'emplacement C:\Python36\python.exe.

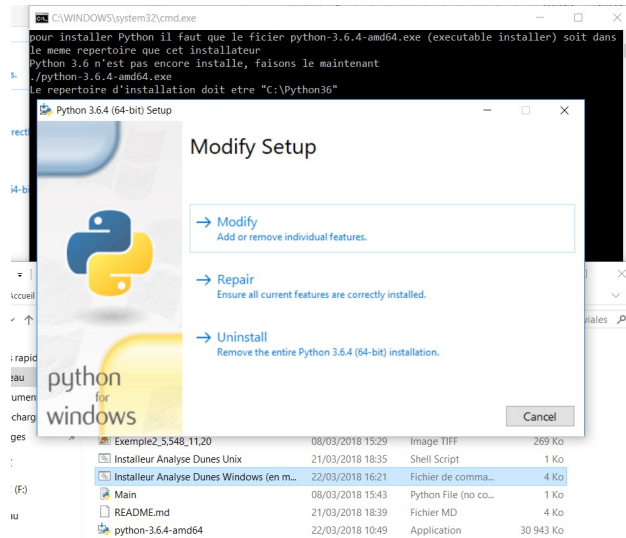


Figure 2 – Installation 2

Une fois Python installé, le programme propose si l'on souhaite installer les différentes bibliothèques utilisées (Matplotlib v2.2.2 / Numpy v1.14.2 / Scipy v1.0.0 / Pillow v5.0.0) en récupérant les installateurs en ligne ou en les recherchant dans le dossier courant. Si l'on procède via la méthode en ligne, le script va donc télécharger et installer les différentes bibliothèques, puis à la fin prévenir l'utilisateur que le poste est maintenant prêt à exécuter le programme d'analyse des dunes.

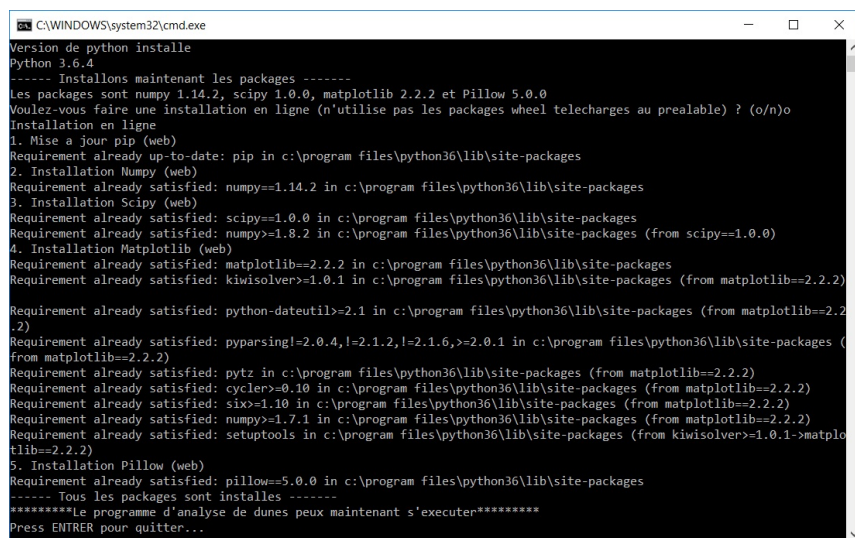


Figure 3 – Installation 3



Webographie

- [WWW1] Pierre W. CAZENAVE, David O. LAMBIN et Justin K. DIX. *Quantitative bedform analysis using decimetre resolution swath bathymetry*. 8 sept. 2008. URL : <http://www.caris.com/conferences/caris2008.dsbld/proceedings/presentations/cazenave/CAZENAVE%20-%20Paper.pdf>.
- [WWW2] Antoine FOURRIÈRE. *Morphodynamique des rivières : Sélection de la largeur, rides et dunes*. 7 déc. 2009. URL : <https://tel.archives-ouvertes.fr/file/index/docid/501313/filename/PhDThesisAntoine.pdf>.
- [WWW3] A. LEFEBVRE, V. B. ERNSTSEN et C. WINTER. *Bedform characterization through 2D spectral analysis*. 11 juil. 2008. URL : https://epic.awi.de/25619/8/SP64_781-785_A.Lefebvre-a.pdf.
- [WWW4] Lutz LESSHAFFT, B. HALL, E. MEIBURG et B. KNELLER. *Deep-water sediment wave formation : Linear stability analysis of coupled flow/bed interaction*. 17 juil. 2014. URL : <https://hal.archives-ouvertes.fr/hal-00997971/document>.
- [WWW5] Roderik C. LINDENBERGH, Thaiënne A.G.P. van DIJK et Paul J.P. EGBERTS. *SEPARATING BEDFORMS OF DIFFERENT SCALES IN ECHO SOUNDING DATA*. 1^{er} jan. 2004. URL : <http://rs.tudelft.nl/~rlindenberg/publications/lindenbergvandijkkegbert.pdf>.

Développement d'un outil d'analyse de dunes fluviales

Kévin Ecalle

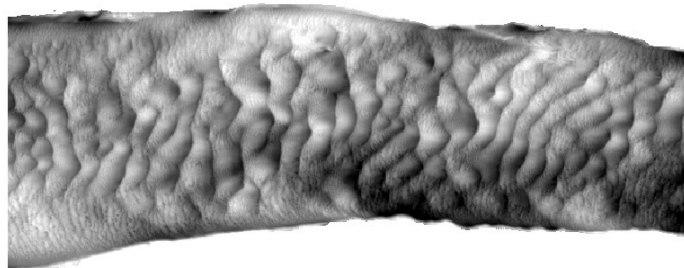
Encadrement : Stéphane Rodrigues, Pascal Makris et Jean-Yves Ramel

Problématique

Connaître les caractéristiques des dunes (hauteur et longueur d'onde) à partir de relevés bathymétriques.

But :

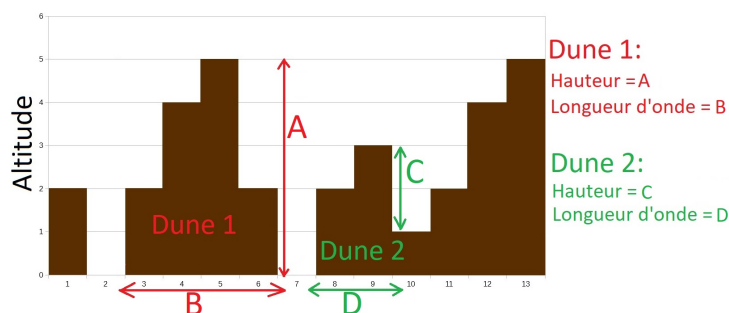
- ☐ Déterminer le déplacement des sédiments
- ☐ Faire de la prévention aux inondations



Solutions envisagées

Il faut savoir reconnaître les dunes, plusieurs méthodes sont envisagées :

- Transformée de Fourier 2D (fréquentiel).
- Utilisation d'une matrice de co-occurrence (texture).
- Algorithme de ligne de partage des eaux.



Bilan partie recherche

La technique de ligne de partage des eaux semble la plus adaptée.

La modifier pour prendre en compte les spécificités de la problématique et ainsi repérer les dunes sur l'image.

L'objectif étant de fournir un fichier listant les informations des dunes et y faire des analyses ultérieures.

```
Traitement analyse dunes de l'image " Exemple_1,95_12,4.tif " fait le 02-04-2018 15:25:38
Altitude minimum = 1.95 Résolution altitude = 0.04098
Detection des dunes (petites) dès 0.5m
Le sens du courant choisi est vers la gauche
Numéro axe choisi 0 constitué des pixels aux extrémités ((71, 231), (652, 149))
Nombre de dunes = 7 | Longueur d'onde moyenne = 4.42m | Hauteur moyenne 62.0cm
IdDune;Longueur d'onde(m);Hauteur(cm);AltitudeCreux1(m);AltitudePic(m);AltitudeCreux2(m)
0;1.98;53.27;4.24;5.19;4.65
1;1.98;61.47;3.63;4.65;4.04
2;6.08;69.67;3.22;3.92;3.22
3;3.96;69.67;3.02;3.71;3.02
4;3.96;73.76;3.63;4.37;3.63
5;9.05;53.27;3.3;3.84;2.89
6;3.96;53.27;2.89;3.43;2.89
```

Développement d'un outil d'analyse de dunes fluviales

Kévin Ecalle

Encadrement : Stéphane Rodrigues, Pascal Makris et Jean-Yves Ramel

Problématique

Connaître les caractéristiques des dunes (hauteur et longueur d'onde) à partir de relevés bathymétriques.

But:

- ☐ Déterminer le déplacement des sédiments
- ☐ Faire de la prévention aux inondations

Solutions envisagées

Il faut savoir reconnaître les dunes, plusieurs méthodes sont envisagées :

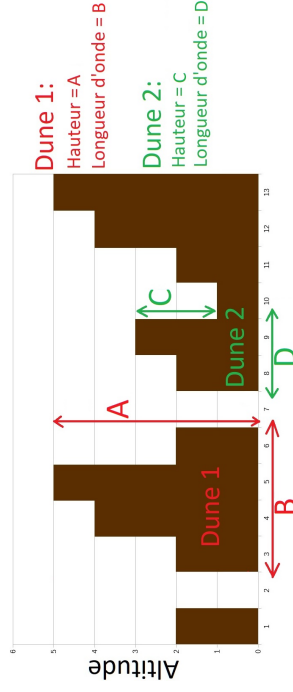
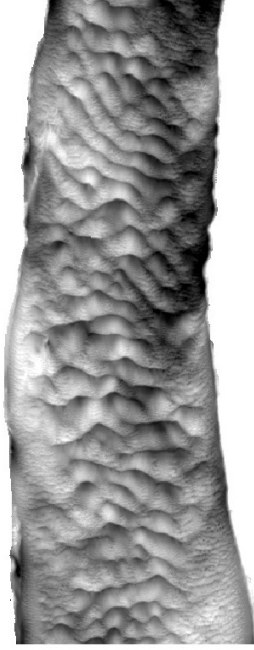
- Transformée de Fourier 2D (fréquentiel).
- Utilisation d'une matrice de co-occurrence (texture).
- Algorithme de ligne de partage des eaux.

Bilan partie recherche

La technique de ligne de partage des eaux semble la plus adaptée.

La modifier pour prendre en compte les spécificités de la problématique et ainsi repérer les dunes sur l'image.

L'objectif étant de fournir un fichier listant les informations des dunes et y faire des analyses ultérieures.



Traitement analyse dunes de l'image
Altitude minimum = 1.95 résolution altitude = 0.04096
Détection des dunes (petites) des 0.3m
Le sens du contour est vers la gauche
Nom des courbes consistant des pixels au extrêmes (71, 231), (62, 149)
Sans les courbes consistant des pixels au extrêmes (71, 231), (62, 149)
dune1 Longueur d'onde m Altitude (cm) Hauteur (cm) AltitudeCm/AltitudeCm2 (m)
0.1 98.53 27.4 2.45 19.4 6.5
1.1 98.61 47.3 63.4 65.4 0.4
2.1 68.69 67.3 22.3 92.3 0.22
3.1 96.69 67.3 02.3 71.3 0.32
4.1 96.73 76.3 63.3 37.3 0.63
5.1 95.53 27.3 3.3 84.2 0.89
6.1 96.53 27.2 89.3 43.2 0.89

Développement d'un outil d'analyse de dunes fluviales

Résumé

Afin de répondre à une demande de Stéphane Rodrigues, un Projet de Recherche et Développement a été proposé pour déterminer les caractéristiques des dunes de la Loire. Le prélèvement des informations se faisant depuis une image créée par des relevés bathymétriques du fleuve, et généré via le logiciel ArcGIS. Pour la phase de recherches 3 méthodes ont été envisagées : l'analyse de textures en exploitant la matrice co-occurrence, la transformée de Fourier pour travailler dans le fréquentiel, et l'usage de l'algorithme de partage des eaux. De plus l'application finale doit permettre l'analyse sur les segments que l'utilisateur a tracés afin de faire de l'analyse recentrer sur une zone précise de l'image. Pour celle de développement, l'application a été faite sous langage Python et permet actuellement de générer des rapports au format texte sur les dunes trouvées sur un axe.

Mots-clés

litage, Analyse d'images, Écho-sondeur multi-faisceaux, Transformée de Fourier, Matrice de co-occurrence, Ligne de partage des eaux, Python, ArcGIS

Abstract

In the context of a research and development project for the fifth year at Polytech Tours, I chose the problem of Stéphane Rodrigues, the automatic prediction of dune characteristics from an image of the Loire. The image is generated in gray level from bathymetric statements of the river and with the use of the software ArcGIS. Three methods have been considered : analysis of texture with co-occurrence matrix, the Fourier transform by usage of the frequency, and the watershed algorithm. Finally, the application must allow the analysis of segments draw by the user, and focus the analysis of the specific areas of the image. For the development one, the application made under language Python, and now can generate textual reports found on an axis

Keywords

Bedform, Image's analysis, Multibeam echosounder, Fourier transform, co-occurrence matrix, Watershed, Python, ArcGIS

Tuteurs académiques

Stéphane RODRIGUES
Pascal MAKRIS
Jean-Yves RAMEL

Étudiant

Kévin ECALLE (DI4)