École Internationale des Sciences
du Traitement de l'Information

Projet GM1

Reconstruction d'un glacier à partir de photos

Auteurs :
Massilia Ait Gherbi
Cem Hazir
Auxane Manry
Clément Villain

Superviseur :
Rémy Vernay

13 février 2015

Table des matières

Chapitre 1

Travail d'analyse

Introduction

Le but de notre projet est de reconstruire la vue aérienne d'un glacier à partir de photos au sol et de points de contrôle (points sur une image dont on connait la latitude et la longitude).

Nous allons décrire les besoins du projets, les outils que nous avons sélectionnés pour le bon déroulement du projet et enfin le résultat de nos recherches sur les algorithmes que nous allons implémenter pour répondre aux besoins.

1.1 Gestion du projet

Afin d'organiser au mieux le projet, nous allons définir les besoins et mettre en place des outils de gestion de projet.

1.1.1 Analyse des besoins

Les besoins du projet sont les suivants :
- Lire et écrire une image
- Déterminer la longitude et la latitude d'un point d'une image
- Reporter une image sur la vue aérienne à construire

Nous disposons de plusieurs images dont on connait la latitude et la longitude de certains points. La problématique du projet est que le relief de

l'image n'est pas régulié, par conséquent une méthode classique telle que la méthode des moindres carrés [2] ne peut pas être appliquée.

Les algorithmes mis en place pour résoudre cette problématique sont décrits dans la section 1.2.

1.1.2 Outils de gestion de projet

Nous allons maintenant décrire les outils mis en place au bon déroulement du projet.

Gantt Afin de gérer au mieux l'organisation temporelle du projet, nous avons créer un diagramme de Gantt, voir figure 1.1.

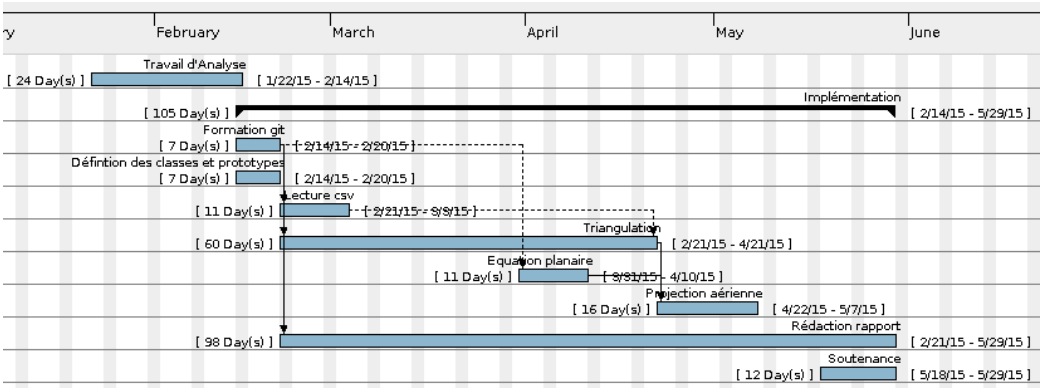

FIGURE 1.1 – Diagramme de Gantt

Gitlab Afin de partager de la manière la plus efficace possible notre code source, nous allons utiliser le logiciel *git* et la plateforme gitlab de l'EISTI. Cependant, un seul membre du groupe est formé à ce logiciel, par conséquent, une formation préalable est nécessaire.

Jenkins et SonarQube Jenkins et SonarQube sont des outils permettant respectivement l'intégration continue et l'évaluation de la qualité du code. Cependant, du fait qu'une partie de l'équipe ne maitrise pas git, nous avons décidé de ne pas mettre en place Jenkins et SonarQube immédiatement afin de pouvoir nous concentrer sur les algorithmes à mettre en place et l'apprentissage de git.

FIGURE 1.2 – Logo de gitlab, jenkins et SonarQube

1.2 Résultats de nos recherches

Nous allons maintenant décrire les résultats de nos recherches et les algorithmes complexes que nous allons implémenter.

1.2.1 État de l'art

Nous avons au préalable fait un état de l'art. Nous avons trouvé un article scientifique [1] présentant exactement notre problème. Dans cet article, les auteurs décrivent comment ils ont reconstruit une image aérienne à partir d'images enregistrées par des caméras. De plus, ces derniers ont également réussi à corriger le dérèglement des caméras dû aux conditions météorologiques au fil des jours (nous ne traitons pas cette partie).

Leur succès nous garantit que leur méthode est efficace.

1.2.2 Triangulation de Delaunay

Nous avons donc choisi d'implémenter une triangulation de Delaunay. Nous avons trouvé 2 algorithmes [3] pour la triangulation de Delaunay, une récursive, une itérative. Le premier est en $\Omega(n \log n)$ et le second est en $O(n^2)$. L'avantage du second est qu'il est plus simple et plus facile à implémenter. De plus, comme dans notre application $n_{max} = 640$ un algorithme en $O(n^2)$ reste envisageable. Nous avons cependant choisi d'implémenter le 1er, par soucis de performance.

L'algorithme récursive repose sur une approche *divide and conquer*. En effet, il sépare l'ensemble de points en deux, calcule la triangulation des

deux ensembles et les fusionne. La principale difficulté est le calcul récursif simultané de l'enveloppe convexe [4] ainsi que du *backtracking* de données, comme par exemple le point le plus à gauche de l'enveloppe convexe.

Il n'est pas pertinent d'écrire ici le pseudo code des algorithmes, car ils sont disponibles dans notre bibliographie.

En revanche, nous allons décrire ce que permet l'algorithme et pourquoi nous l'utilisons. L'algorithme permet à partir d'un ensemble de point P, qui sont nos points de contrôles, de former une triangulation $DT(P)$ tels qu'aucun des points P n'est à l'intérieur du cercle circonscrit d'un des triangles de $DT(P)$. Voir figure 1.3.

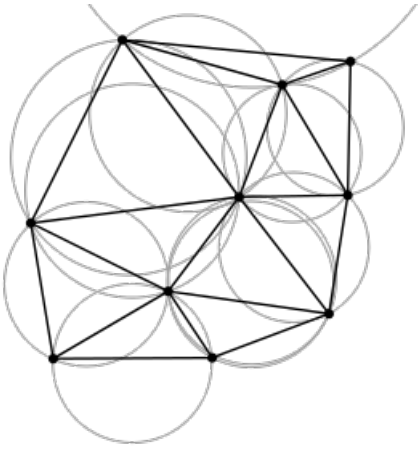

FIGURE 1.3 – Une triangulation de Delaunay

Appliqué à notre problème, l'algorithme donne le résultat suivant : nous pouvons voir les points de contrôle sur la figure 1.4 et le résultat de l'algorithme sur la figure 1.5.

Nous pouvons remarquer qu'à l'intérieur d'un triangle, le relief est relativement régulié. C'est pourquoi, connaissant la latitude et la longitude des 3 points qui forment le triangle, nous pouvons approximer la latitude et la longitude des points à l'intérieur du triangle. Cette méthode est décrite dans la sous partie suivante 1.2.3.

Plus de détails sur les structures de données utilisées seront décrites lors de l'implémentation.

FIGURE 1.4 – Les points de contrôles sur une image

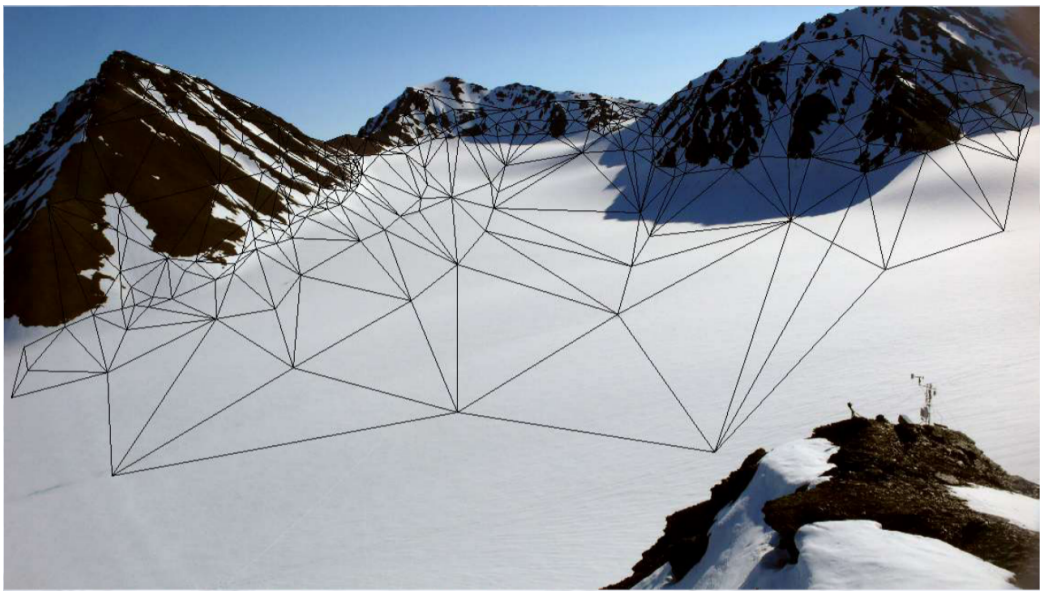

FIGURE 1.5 – Triangulation de Delaunay appliquée à notre image

1.2.3 Résolution d'équation planaire

A l'intérieur d'un triangle, nous supposons que la lattitude et la longitude varient de façon linéaire par rapport aux coordonnées x et y. De ce fait nous cherchons deux équations planaires :

$$\begin{cases} L = ax + by + c \\ H = a'x + b'y + c' \end{cases} \tag{1.1}$$

Nous disposons des 3 points du triangle qui sont des points de contrôle, nous pouvons donc parfaitement déterminer les inconnus a, b, c, a', b' et c' (pivot de Gauss [5]). Il suffit donc de faire cela pour tous les triangles pour connaître la lattitude et la longitude de chaque point de l'image.

Nous sommes alors capable de réaliser la projection de la latitude et longitude sur notre image (voir figure 1.6). On peut alors reporter l'image sur la vue aérienne.

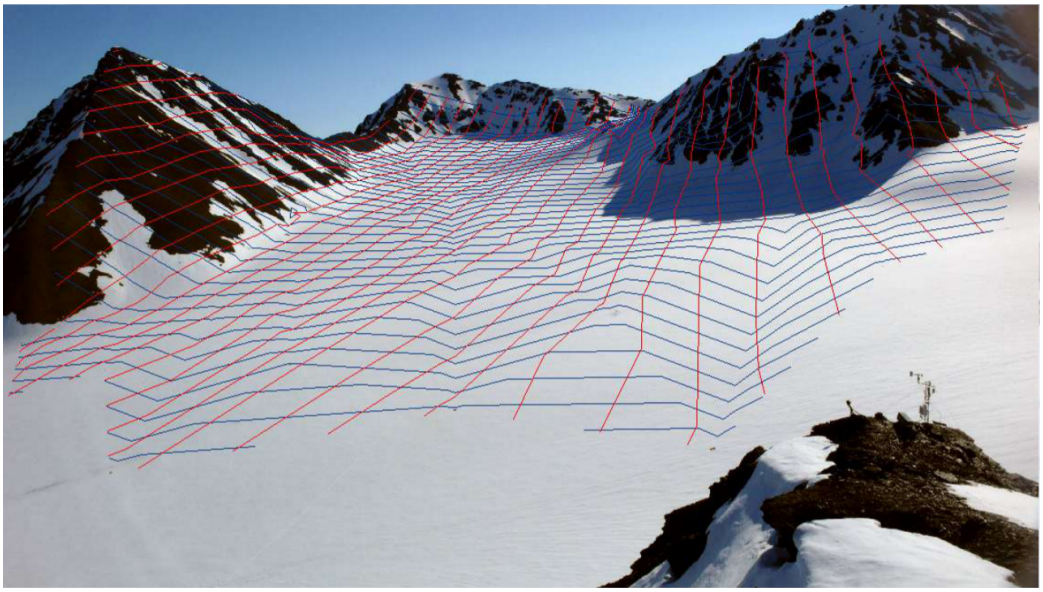

FIGURE 1.6 – Projection de la latitude et de la longitude sur l'image

Conclusion

Finalement, nous savons quels algorithmes utiliser pour créer une image aérienne du glacier. Nous avons réparti notre travail dans le temps.

Nous pouvons donc passer à l'étape suivante, c'est à dire, l'implémentation.

Bibliographie

[1] Eric Bernard, Madeleine Griselin, Jean-Michel Friedt, Florian Tolle, Dominique Laffly, Christelle Marlin, and Emerick Delangle. Monitoring snow cover dynamics on an arctic hydrosystem using field measurements, remote and in situ sensing (austre lovenbreen, spitsberg, 79n). June 2010.

[2] Inc. David Eberly, Magic Software. Least squares fitting of data, 2001.

[3] D. T. Lee and B. J. Schachter. Two algorithms for constructing a Delaunay triangulation. *International Journal of Parallel Programming*, 9(3) :219–242, June 1980.

[4] F. P. Preparata and S. J. Hong. Convex hulls of finite sets of points in two and three dimensions. *Commun. ACM*, 20(2) :87–93, February 1977.

[5] Homer F. Walker. Gaussian elimination. `http://users.wpi.edu/~walker/MA514/HANDOUTS/gaussian_elim.pdf`.