Essaim de drones pour la cartographie de grandes zones
comment optimiser le plan de vol

Laurent JOSPIN - Marthe ORTOLO

L'utilisation de drones est une solution innovante dans le cadre de la cartographie. Avec un drone volant à environ 100 mètres au-dessus du sol et qui reste dans un périmètre permettant de le contrôler à distance, il est possible de générer des orthophotographies et des modèles 3D de l'ordre du centimètre. Cette alternative à moindre coût à la télédétection spatiale promet des résultats de très bonne qualité, de part la résolution de l'orthophotographie. Cependant, un problème majeur et d'actualité se pose : que faire quand les zones à cartographier sont trop grandes pour être couvertes par un seul vol ? Il faut alors pouvoir organiser le plan de plusieurs vols de manière à couvrir efficacement la zone avec un ou plusieurs appareils. Dans cette étude, réalisée en partenariat avec l'entreprise senseFly, deux étudiants en Master de l'EPFL ont compilé et étudié une série d'algorithmes existants qui peuvent être employés pour obtenir, pour un ou plusieurs drones sur une telle zone, un plan de vol minimisant le temps de travail pour les opérateurs. L'algorithme complet peut fonctionner avec un essaim de drones de types potentiellement différents, embarquant des caméras de résolutions différentes. Il est parallélisable lors des principales étapes de la recherche de solution et sa mise en œuvre est simple.

MOTS-CLÉS

Optimisation, NP-complétude, Algorithme, Heuristique, Drone, Plan de vol, Division du travail, Essaim

Introduction

L'optimisation de plan de vol pour la couverture de larges zones est un sujet qui a déjà été étudié [Lak13] [API15] [Meh12]. Des solutions efficaces pour traiter le problème existent déjà. Mais la recherche dans ce domaine reste très prisée du fait de l'impossibilité de trouver la solution optimale d'un problème en temps polynomial, les problèmes rencontrés étant souvent de type NP-complet* [Bie15,CLRS09].

L'optimisation des plans de vol est soumise à diverses contraintes, notamment du point de vue de la géométrie des blocs, de l'autonomie des drones et des contraintes légales : la visibilité et les points de départ des drones.
L'approche choisie a été de diviser le problème principal en plusieurs sous-problèmes offrant une bonne approximation du problème original. Un algorithme a ensuite été proposé pour traiter chaque sous-problème.

La sélection des points de départ : Le but est de choisir, parmi une liste de points de départ proposés, un minimum d'entre eux de telle sorte que l'opérateur puisse couvrir toute(s) la(les) zone(s) à cartographier sans déplacer son matériel trop souvent. L'hypothèse sous-jacente est que déplacer hommes et matériel est de toute façon moins pratique que de faire voler le drone quelques centaines de mètres de plus.

La définition des zones d'influence : Ensuite, il faut attribuer une zone d'influence fixe aux points de départ. En effet, il reste en général plusieurs zones de la carte qui peuvent être rejointes depuis plusieurs points de départ.

Le découpage des parcelles : Une fois les zones d'influence connues, il faut les découper en parcelles que l'on attribue aux différents drones. Le but étant d'avoir assez de flexibilité pour attribuer des zones aux drones tout en garantissant que la géométrie des blocs soit assez régulière pour faciliter le travail d'assemblage de l'orthophotographie ou l'extraction d'un modèle 3D par la suite. S'il est tout à fait possible d'assembler un ensemble hétérogène de photos, avoir des blocs homogènes pris à la même heure par le même capteur photo permet d'améliorer la solution ou au moins d'en faciliter l'obtention.

L'attribution des parcelles aux drones : Une fois les parcelles prêtes, il reste à les attribuer aux drones, réalisant ainsi l'esquisse du plan de vol final. Une parcelle contiendra ensuite plusieurs lignes de vol. Considérant que les algorithmes de constructions des lignes de vol existent déjà, ils n'ont pas été pris en compte dans cette étude.

Considérations légales

Pour profiter pleinement d'un essaim de drones, un opérateur doit avoir le droit de confier le pilotage des machines à un superviseur informatique. Idéalement, il devrait aussi pouvoir laisser les drones évoluer hors-vue.
Pour des raisons de sécurité, le vol hors-vue et/ou supervisé est encore interdit ou très largement restreint dans les pays occidentaux. Voici la situation en Suisse, en France et un petit point sur la situation internationale :
En Suisse La principale source de droit est l'ordonnance sur les aéronefs de

* Les problèmes NP-complet sont une classe de problèmes qu'il est, pour l'instant, impossible de résoudre simplement avec un algorithme. Comme tous les algorithmes connus sont de complexité au moins exponentielle il devient impossible pour un ordinateur de trouver une solution pour des jeux de données même de taille moyenne.

catégorie spéciale [fdlc17]. Pour les aéronefs sans occupant d'un poids allant jusqu'à 30 kg, l'ordonnance précise à l'article 17 que le pilote doit avoir en permanence le contact visuel avec l'aéronef. Ce qui disqualifie directement tout usage supervisé et/ou hors-vue. Notons que des exceptions peuvent être accordées dans certains cas.
Une autre spécificité suisse est qu'il est de la compétence des cantons de légiférer sur le sujet, et donc d'ajouter des contraintes supplémentaires.

En France L'usage des drones est essentiellement réglementé par les arrêtés du 17 décembre 2015 relatifs à l'utilisation de l'espace aérien par les aéronefs qui circulent sans personne à bord et relatifs à la conception des aéronefs civils qui de même circulent sans personne à bord, aux conditions de leur emploi et aux capacités requises des personnes qui les utilisent. Le vol hors-vue y est autorisé dans certain cas, hors des zones peuplées. Par contre, les pilotes de drones doivent obtenir une licence presque équivalente à celle d'un pilote civil usuel. Et donc, il n'est pas envisageable, dans ce cas, de confier le pilotage à un superviseur robot.

A l'international La situation varie grandement d'un pays à l'autre. Si les lois et règlements sont encore très restrictifs en Europe et en Amérique du nord, d'autres états peuvent se montrer un brin plus laxistes.

Notons que toutes ces législations évoluent rapidement. Actuellement des grands groupes tels Amazon et/ou Google font pression pour autoriser les livraisons par drones autonomes (ce qui est encore interdit aux États-Unis et en Europe).
Concernant les applications possibles des algorithmes présentés ici, il faut garder à l'esprit que pour la plupart des cas d'utilisation, il faudra encore prévoir un opérateur par drone. Ce qui veut dire que le partage du travail ne permet une économie de temps qu'à la condition d'avoir plus de personnel.

Méthodes proposées

Sélection des points de départ

Les contraintes de visibilité dessinent des cercles autour des points de départ. Si la zone est trop large pour être couverte par un seul vol, il est probable qu'elle ne puisse pas l'être depuis un seul point de départ.
L'utilisateur devra donc identifier plusieurs points de départ potentiels. Il ne sera sans doute pas rentable de tous les utiliser car cela demande de déplacer des hommes et du matériel au sol. Il faut donc sélectionner un minimum de points de départ qui puissent couvrir l'ensemble des zones à cartographier.
Ce problème est assimilable à un problème de couverture par ensemble *(Set Cover)*. Le problème s'énonce comme suit : soit U l'ensemble de la zone à cartographier (représenté en orange sur la *figure 1*) que nous appellerons l'univers, et S_i, une série de sous-ensembles de U, qui sont les intersections des cercles d'influence avec la zone à cartographier.

$$\bigcup_{i=1}^{n} S_i = U \qquad (1)$$

On cherche un ensemble $J \in 1, ..., n$ tel que $|J|$, le nombre d'éléments dans J soit minimal et que l'univers soit toujours couvert.

$$\bigcup_{i \in J} S_i = U \qquad (2)$$

Ce problème, comme de nombreux problèmes combinatoires, est NP-complet. Il existe des heuristiques qui donnent de bonnes approximations de la solution optimale mais, à ce jour, aucune ne peut trouver la solution exacte du problème général en temps polynomial.

Pour le résoudre, l'algorithme proposé est un algorithme dit glouton (https://fr.wikipedia.org/wiki/Algorithme_glouton), la procédure est la suivante :

Algorithme 1. sélection des points de départ.

for all point de départ **do**
  Vérifier que son cercle d'influence soustrait de ceux des autres points soit non vide.
  if non vide **then**
    Sélectionner le point.
  end if
end for
while La zone n'est pas entièrement couverte **do**
  for all Point de départ non-sélectionné **do**
    Calculer l'aire de l'intersection de sa zone d'influence avec le reste de la zone à cartographier.
  end for
  Sélectionner le point couvrant la plus grande aire sur la zone restant à cartographier.
end while

Cet algorithme a une complexité de $O(n^2)$, étant le nombre de points de départ. Dans le cas où les éléments constituent un ensemble discret, on peut prouver que l'algorithme glouton offre la meilleure approximation en temps polynomial de la solution [Sla97].

Définition de zones d'influence

Une fois les points de départ définis, il s'agit de diviser la zone à cartogra-

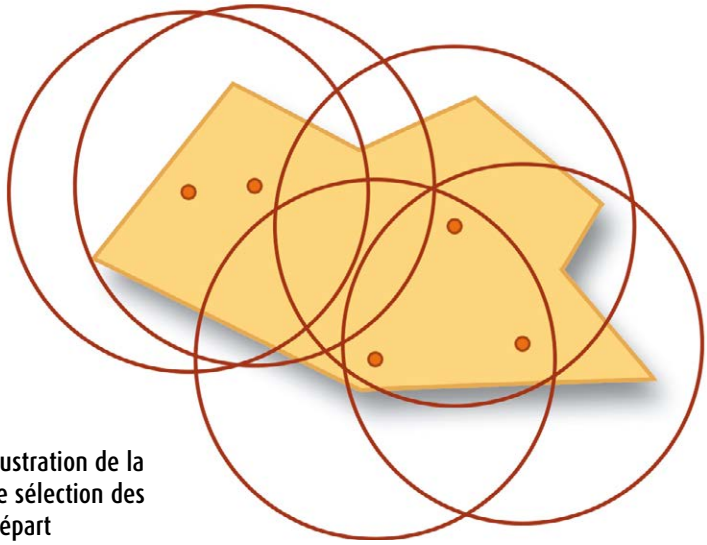

Figure 1. Illustration de la méthode de sélection des points de départ

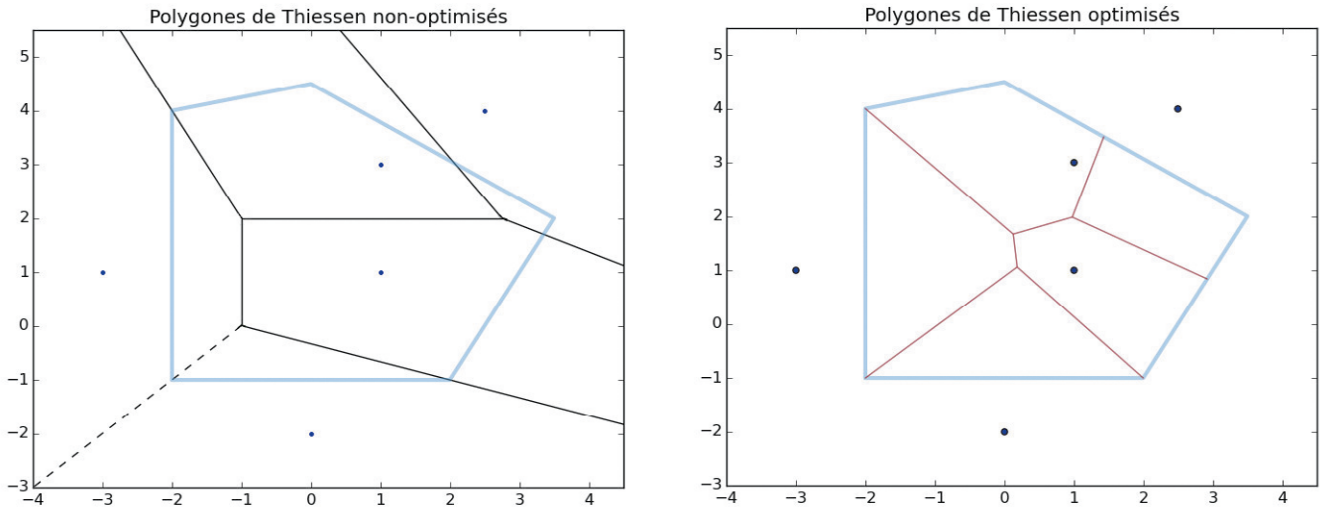

Figure 2. Optimisation des zones d'influence

phier en plusieurs zones, une par point de départ, qui puissent être cartographiées par les drones lancés depuis ce même point.

Une solution facile à implémenter est celle des polygones de Thiessen, ou une généralisation comme les polygones de Thiessen pondérés [Meh12], mais elle peut poser certains problèmes, surtout s'il existe des points de départ à l'extérieur de la zone à cartographier. Dans le cas pondéré, il pourrait aussi arriver que les zones découpées soient en dehors des cercles d'influence. Il est néanmoins possible d'optimiser le découpage des zones d'influence en recourant aux outils de l'optimisation convexe [BV07]. Le but est de formuler une fonction convexe à minimiser avec une série de contraintes convexes. Ce problème peut ensuite être résolu par un solveur standardisé, par exemple cvxpy [DB16], qui fera lui-même usage d'algorithmes tel que les méthodes des points intérieurs.

Après quelques recherches, une proposition a été de minimiser la variance de la distance des coins des zones d'influence aux centroïdes de ces mêmes zones.

$$V = \frac{1}{m}\sum_{j=1}^{m}\left(\frac{1}{n_i}\sum_{i=1}^{n_i}(d_{ji}-\bar{d})^2\right) \quad (3)$$

$$d_{ji} = \sqrt{\left(x_{ji}-\frac{1}{n_i}\sum_{k=1}^{n_i}x_{jk}\right)^2+\left(y_{ji}-\frac{1}{n_i}\sum_{k=1}^{n_i}y_{jk}\right)^2} \quad (4)$$

$$\bar{d} = \frac{1}{m}\sum_{j=1}^{m}\left(\frac{1}{n_i}\sum_{i=1}^{n_i}d_{ji}\right) \quad (5)$$

Minimiser la variance de la distance au centroïde revient à équilibrer la taille des zones d'influence. Il faut toutefois être un peu attentif, le centroïde a parfois un comportement capricieux, notamment si la forme de la zone a un grand nombre de points concentrés sur une petite zone.

Si la fonction $d_{ji}(x,y)$ dans l'équation 4 est bien convexe, ce n'est plus vrai pour la fonction V dans l'équation 3.

Pour résoudre ce problème, on choisit de se restreindre à la somme des distances au carré. Cela revient à minimiser le deuxième moment non-centré et donc correspond à minimiser la variance. De plus, comme c'est une somme de fonctions convexes, elle l'est également.

$$V_{nc} = \sum_{j=1}^{m}\left(\sum_{i=1}^{n_i}d_{ji}^2\right) \quad (6)$$

Il reste ensuite un certain nombre de contraintes à écrire, notamment le fait que la distance des coins des zones d'influence aux points de départ (et non pas aux centroïdes de la zone d'influence) soit plus petite que le rayon maximal autorisé et que ces mêmes coins soient toujours dans la zone à cartographier.

$$\begin{aligned}(x_{ji}-x_{start,j})^2+(y_{ji}-y_{start,j})^2 &\leq r_{max}^2 \;\forall i,j\\ a_k x_{ji}+b_k y_{ji}+c_k &\leq 0 \;\forall j,i,k\end{aligned} \quad (7)$$

Avec s_k le k-ème côté de la zone à cartographier et $a_k x + b_k y + c_k = 0$ l'équation caractéristique de la droite passant par ce côté et dont le vecteur directeur pointe vers l'extérieur.
Il faut aussi prendre en compte le fait que les points qui appartiennent à la limite de la zone à cartographier appartiennent toujours à la frontière. Il est nécessaire de s'assurer que l'équation 4 prenne aussi en compte les coins de la zone à cartographier (même si les contraintes les empêcheront de bouger) pour que tous les points de la frontière aient une influence. Sinon, certaines zones d'influence risquent d'être plus grandes que leur taille estimée par la fonction objectif et donc de pousser toutes les autres zones vers le centre.

$$a_k x_{ji} + b_k y_{ji} + c_k = 0 \;\forall (x_{ji}, y_{ji}) \in s_k \quad (8)$$

Minimiser V_{nc} sous les contraintes 7 et 8 est un problème convexe. Il est à noter que l'équation 8 impose au point de se trouver sur un segment précis de la frontière de la zone à cartographier (voir même, deux contraintes combinées obligent les points qui étaient sur un coin à rester sur ce coin). Une extension de l'algorithme serait de désactiver et réactiver certaines de ces conditions pour laisser plus de liberté aux points sur la frontière mais au prix de devoir appeler plusieurs fois le solveur.

Tout cela est vrai si et seulement si la zone à cartographier est elle-même convexe, autrement les contraintes 7 n'auraient pas de sens. Pour le projet, les zones non-convexes ont simplement été refusées par le programme. Pour retirer cette contrainte, il faudrait décomposer nos zones à cartographier en plusieurs zones convexes. Une approche est proposée dans [CD85]. Une autre approche consiste à utiliser

une décomposition quasi-convexe, telle que présentée dans [LA05], puis à se servir de l'enveloppe comme domaine pour optimiser. L'avantage de la décomposition quasi-convexe est qu'elle générera moins de polygones, et ceux-ci seront presque convexes, donc assimilables à leur enveloppe convexe.
Un exemple de résultat est présenté dans la *figure 2*.

Division en parcelles

Une fois les zones d'influence déterminées pour chaque point de départ, il faut diviser ces dernières de manière à pouvoir partager le travail entre les drones. Gérer ce partage du travail au niveau des photos peut être très compliqué. Il faut garder des blocs cohérents, s'assurer que toute la zone soit couverte et qu'il y a assez de redondance, tout en ayant la possibilité d'avoir des drones embarquant des caméras différentes, avec des résolutions différentes et ainsi des emprises au sol différentes.

Il serait plus judicieux de séparer la zone en un certain nombre de parcelles d'une taille déterminée. Il existe de nombreux algorithmes de pavage, qui vont de la simple intersection d'une grille régulière avec la forme jusqu'à des algorithmes tels que la tessellation de Voronoï [DFG99].
Nous proposons une approche basée sur le paradigme diviser-conquérir qui est à la fois simple mais qui peut également être extrêmement flexible tout en étant directement parallélisable (illustrée dans la *figure 3*).

Algorithme 2. fonction de découpage des parcelles.

```
function decouperParcelles(zone)
    if estAssezPetit(zone) then
        return zone
    else
        subZone1, subZone2 =
        decouper(zone)
        return
        decouperParcelles(subZone1),
        decouperParcelles(subZone2)
    end if
end function
```

Le comportement de l'algorithme sera largement conditionné par la façon dont les fonctions decouper() et estAssezPetit() sont implémentées.

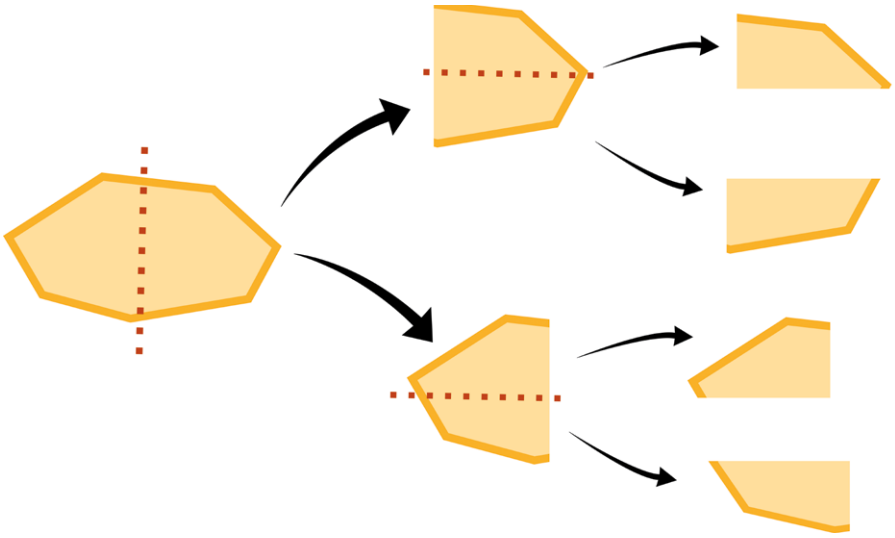

Figure 3. Découpage en parcelles d'une zone à cartographier

Une implémentation basique mais néanmoins suffisante dans la plupart des cas est présentée ci-dessous. Nous assumons que nous traitons des polygones convexes, puisque l'étape précédente en a besoin.

Algorithme 3. fonction découper.

```
function decouper(zone)
    rectangle =
rectangleEnglobant(zone)
    if largeur(rectangle) >
hauteur(rectangle) then
        sousZone1, sousZone2 =
couperEnDeuxParLaHauteur(zone)
        return sousZone1, sousZone2
    else
        sousZone1, sousZone2 =
couperEnDeuxParLaLargeur(zone)
        return sousZone1, sousZone2
    end if
end function
```

Algorithme 4. fonction est assez petit.

```
function estAssezPetit(zone)
    rectangle = rectangleEnglobant(zone)
    grandCote =
    max(largeur(rectangle),
    hauteur(rectangle))
    if grandCote > limite then
        return faux
    else
        return vrai
    end if
end function
```

Cet algorithme marche dans de nombreux cas en pratique, mais il est assez facile de trouver des cas pour lesquels il donne de mauvais résultats. L'exemple le plus trivial serait un triangle rectangle assez allongé, dont les parcelles découpées à la pointe seraient très fines.

Il serait possible d'implémenter des algorithmes plus intelligents pour le découpage ou des indices spatiaux plus pertinents pour le critère d'arrêt. La structure générale de l'algorithme a été prévue pour adapter les routines. Mais ce projet ne va pas beaucoup plus loin sur ce point.

Attribution des parcelles aux drones

Une fois les différentes zones découpées en parcelles, il devient possible de répartir ces parcelles entre les drones. Nous proposons un algorithme basé sur une file de priorité qui permet de construire une solution relativement efficace.

Le principe général est d'ordonner les parcelles grâce à un indice spatial dans chacune des zones d'influence puis de faire voler les drones le long de l'indice spatial ainsi obtenu.

La file de priorité permet de sélectionner le drone ayant le moins volé, s'assurant ainsi que les temps de vol entre les différents drones sont à peu près égaux.

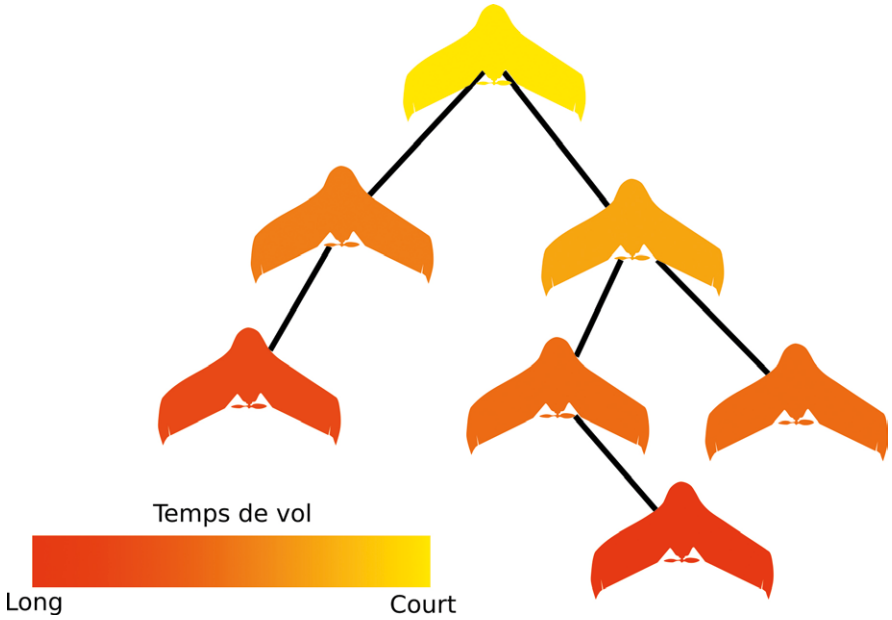

Figure 4. Tas (heap) de drones comme file de priorité.

Algorithme 5. vol des drones dans les parcelles.

```
function vol(parcelles[], filePriorité)
    ptOrder[] =
trierParcelles(parcelles[])
    drone = extraireDrone(filePriorité)
    estEndurant = estEndurant(drone)
    if estEndurant then
        volDepuisLaFin(parcelles[], ptOr-
        der[], drone)
    else
        volDepuisLeDébut(parcelles[],
        ptOrder[], drone)
    end if
    insérerDrone(filePriorité, drone)
end function
```

Une solution pour implémenter efficacement la file de priorité est d'utiliser un tas, par exemple binomial, tel que présenté dans la *figure 4*. S'il est bien implémenté, le tas permet de gagner du temps par rapport à une liste, et son avantage grandit pour les flottes de drones de grande taille. En effet, le tas permet de ne maintenir qu'un tri partiel de la flotte de drones. Ceci permet de sortir et d'entrer des drones dans le tas très rapidement, plus rapidement que s'il fallait trouver l'ordre du drone dans une liste.

L'algorithme proposé différencie les drones "endurants" des autres. Cette optimisation permet, dans le cas d'une flotte de drones de natures différentes, de placer en priorité les drones faiblement endurants proches du point de départ. Elle peut néanmoins poser certains problèmes, notamment dans le cas où un seul drone peut couvrir seul la zone d'influence complète d'un point de départ. L'idée est de faire voler les drones endurants depuis les points éloignés et les autres en priorité sur les points proches.

La fonction de tri qui a été utilisée pour ce projet s'inspire du problème de la tournée de voyageur de commerce. Il s'agit de trier les centroïdes des parcelles, du plus proche au plus éloigné du point de départ, dans l'ordre du plus court chemin passant par tous les autres centroïdes des parcelles.

La tournée du voyageur de commerce étant un problème NP-complet, il n'est pas possible de trouver le chemin exact. Il existe toutefois de nombreuses heuristiques qui permettent d'obtenir une bonne approximation. Celle qui a été utilisée pour le projet est d'insérer itérativement dans le chemin le point qui génère le moindre détour, voir *figure 5*.

Algorithme 6. ordre des parcelles pour les vols.

```
function trierParcelles(parcelles[])
    chemin[] = [extraireMin(parcelles[]),
    extraireMax(parcelles[])]
    while !estVide(parcelles[]) do
        parcelleSelectionnee =
        parcelles[0]
        positionSelectionnee = 0
        scoreSelection = inf
        for all parcelles do
            for all positions possibles do
                if score(parcelle, position) <
                scoreSelection
```

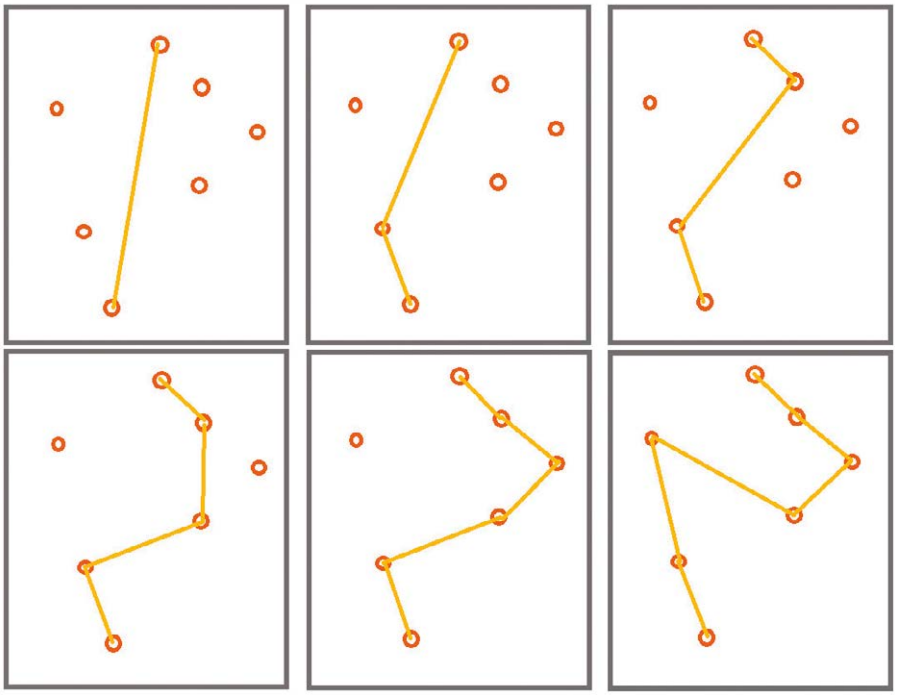

Figure 5. Construction de l'index spatial

```
then
                parcelleSelectionnee =
                parcelle
                positionSelectionnee =
                position
                scoreSelection =
                score(parcelle, position)
            end if
        end for
    end for
    extraire(parcelles[], parcelleSe-
    lectionnee)
    insererDans(chemin[],
    positionSelectionnee, parcelle-
    Selectionnee)
  end while
end function
```

La fonction score() est simplement l'agrandissement du chemin.

$$s = ||p2 - p1||_2 + ||p3 - p2||_2 - ||p3 - p1||_2 \quad (9)$$

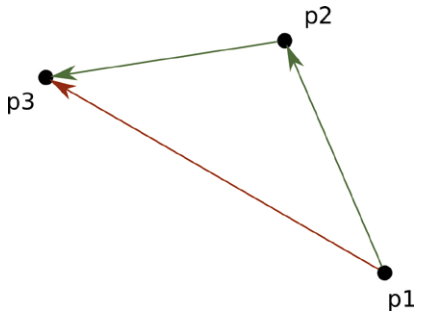

Figure 6. Calcul du score

Avec p1 et p3 des points sur le chemin actuel et p2 le point à insérer entre p1 et p3.

Il est possible de paralléliser le processus, pour peu que la file de priorité sache éviter les collisions lors de l'exécution, en attribuant successivement différentes zones d'influence à traiter à différentes files d'exécution. Dans ce cas, plusieurs drones risquent de sortir simultanément de la file de priorité, ce qui peut être une source d'imprécision supplémentaire pour les petites formations. Le nombre de files d'exécution est limité par le minimum entre le nombre de drones et le nombre de points de départ alors que dans le cas idéal cela devrait être le maximum.

Il serait possible de transformer un peu la file de priorité, de façon à ne plus sortir les drones mais en estimant leur temps de vol futur quand une file d'exécution décide d'accéder à un drone, puis en marquant le temps de vol définitif une fois que la file d'exécution a fini de le faire voler. De même, en estimant préalablement le nombre de vols nécessaire à couvrir toutes les parcelles d'une zone d'influence, il serait possible de faire travailler plusieurs files d'exécution sur la même parcelle.

Résultats

Les méthodes décrites précédemment ont été implémentées, puis testées. Nous avons créé un prototype de logiciel. Une capture d'écran est présentée sur la *figure 7*. Le chemin en jaune indique l'ordre des parcelles à parcourir. Ensuite, les photos devront être positionnées pour obtenir le plan de vol final.

Cet exemple met en évidence le problème que peut poser la contrainte de commencer le vol par la parcelle la plus proche et finir par la plus éloignée. En effet, les chemins parcourus par les drones ont, en général, un ou plusieurs croisements dus à cette contrainte.

La figure 8 présente un autre exemple réalisé avec le logiciel ainsi codé. Il s'agit de la couverture du lac de Morat en Suisse à l'aide d'une flotte de trois drones virtuels. Comme précisé, il ne s'agit pas du plan de vol définitif mais d'une ébauche sans lignes de vol définitives.

Conclusion

La solution qui est proposée ici permet de construire une approximation de la solution, en décomposant le problème en différentes étapes et en utilisant des méthodes de résolution adaptées à chaque étape. Les algorithmes utilisés dans les différents sous-problèmes restent modulables et peuvent être adaptés par les utilisateurs. Ces derniers peuvent même employer une partie des solutions proposées pour les combiner avec les leurs.

Les algorithmes proposés sont simples à mettre en œuvre et produisent une solution possible si le problème n'est pas dégénéré. Voici un bref résumé des méthodes proposées :

La sélection des points de départ : il est possible d'assimiler ce problème à une couverture par set. Un algorithme glouton a été proposé.

La définition des zones d'influence : une approche à base de polygone de Thiessen a été proposée. Une formule permettant d'utiliser les méthodes de l'optimisation convexe pour améliorer la solution en question a aussi été proposée.

Le découpage des parcelles : une méthode basée sur le paradigme diviser-conquérir a été présentée. Elle offre l'avantage d'être parallélisable et flexible, puisque la routine qu'elle appelle peut être adaptée.

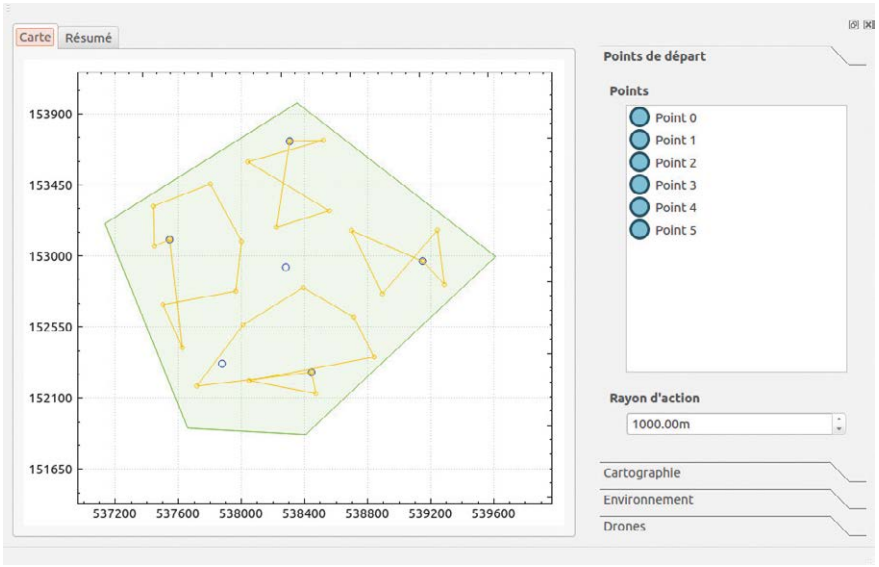

Figure 7. Fenêtre de résultat de l'implémentation réalisée dans Qt (un modèle virtuel de drone a été employé)

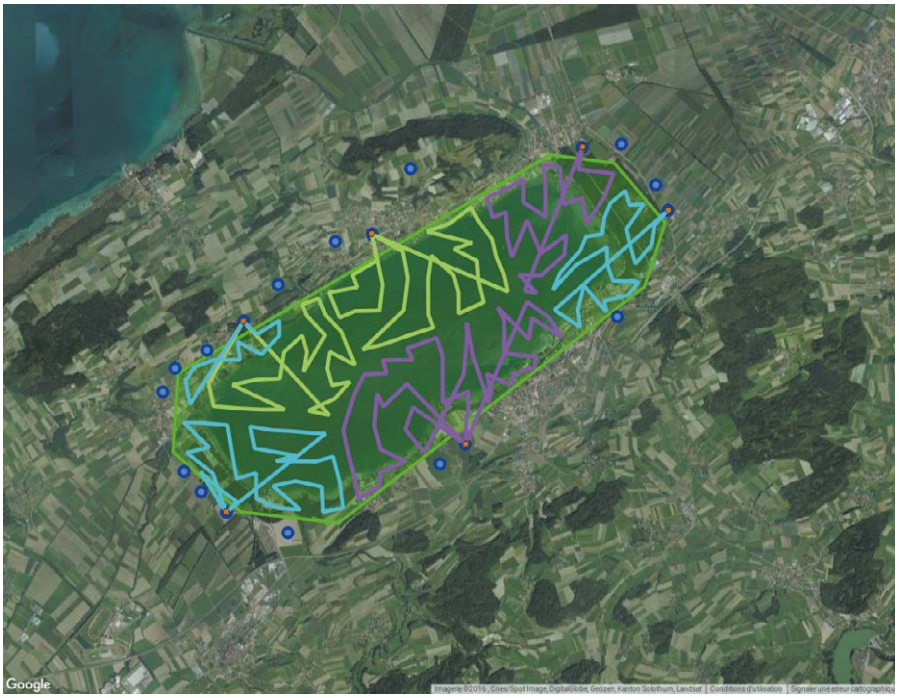

Figure 8. **Un autre exemple : la couverture du lac de Morat**

L'attribution des parcelles aux drones : La proposition qui a été faite repose sur l'usage d'une file de priorité pour les drones et d'un index spatial pour les parcelles à parcourir. Le problème s'approche de la résolution d'une tournée de voyageur de commerce. La solution proposée est parallélisable moyennant un certain nombre de précautions.

Il serait intéressant de poursuivre ce projet avec des algorithmes d'amélioration, qui prendraient la solution obtenue et tenteraient de l'améliorer encore. D'ailleurs, la plupart des méthodes pour traiter efficacement des problèmes de type NP-complet font appel à ces deux phases, construction puis amélioration. Un autre point où il serait possible de continuer, serait de comparer les résultats obtenus avec ceux d'une méthode alternative, par exemple [Lak13]. ●

Remerciement

Merci à l'équipe de senseFly, qui a accepté de collaborer sur ce projet.

Contacts

Bertrand MERMINOD, professeur à l'EPFL (École polytechnique fédérale de Lausanne)
Laurent JOSPIN, étudiant en master à l'EPFL
laurent.jospin@alumni.epfl.ch
Marthe ORTOLO, étudiante en master à l'EPFL
marthe.ortolo@alumni.epfl.ch
Emmanuel CLÉDAT, PhD à l'EPFL
emmanuel.cledat@epfl.ch

Bibliographie

[API15] Gustavo S. C. Avellar, Guilherme A. S. Pereira *, Luciano C. A. Pimenta, and Paulo Iscold. *Multi-UAV routing for area coverage and remote sensing with minimum time.* Sensors, November 2015.

[Bie15] Michel Bierlaire. *Optimization : Principles and Algorithms, volume 1.* EPFL press, 1015 Lausanne, 2015.

[BV07] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization.* Cambridge University Press, 2007.

[CD85] Bernard Chazelle and David P. Dobkin. *Optimal convex decompositions.* Computational Geometry, pages 63–133, 1985.

[CLRS09] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduciton to Algorithms, Third* Edition, volume 1. The MIT Press, 2009.

[DB16] Steven Diamond and Stephen Boyd. *CVXPY : A Python-embedded modeling language for convex optimization.* Journal of Machine Learning Research, 17(83) :1–5, 2016.

[DFG99] Qiang Du, Vance Faber, and Max Gunzburger. *Centroidal voronoi tessellations : Applications and algorithms.* SIAM Rev., 41(4) :637–676, December 1999.

[fdlc17] Office fédéral de l'aviation civile. *Ordonnance du detec sur les aéronefs de catégories spéciales, 2017.* https://www.admin.ch/opc/fr/classified-_compilation/19940351/index.html.

[LA05] Jyh-Ming Lien and Nancy M. Amato. *Approximate convex decomposition of polygons.* Computational Geometry, pages 100–123, 2005.

[Lak13] Lakeside Labs. *Flying High* - Multi-UAV Area Coverage, August 2013.

[Meh12] Mohsen Mehdizade. *A Decomposition Strategy for Optimal Coverage of an Area of Interest using Heterogeneous Team of UAVs.* PhD thesis, Concordia University, 2012.

[Sla97] Petr Slavik. *A tight analysis of the greedy algorithm for set cover.* Journal of Algorithms, 25 :237–254, 1997.

ABSTRACT

Unmanned Aircraft System (UAS) are an innovative solution for mapping. With one UAS flying 100 meters above ground level, working inside a control perimeter, one can generate orthophotos or 3D models with a precision of the order of 1cm. This low-cost alternative to spatial remote sensing can deliver high quality results, especially when looking for a high-res image. But a major problem remains : what if the area to map is too big to be covered by a single flight ? Some flight plans have to be prepared to cover efficiently the area with one or more UASs.
In this study, with the Swiss Company senseFly as a partner, two EPFL master students have compiled and analysed a series of existing algorithms that can be used to get a flight plan optimized to reduce the total time needed to cover the area. The complete algorithm is able to treat problems even with heterogeneous UASs swarms, loaded with different models of cameras. It can be parallelized during the main phases of solution search and is easy to implement.