École Polytechnique Fédérale de Lausanne

Projet de Master

Pic2Map: intégration de photographies dans QGIS

Auteur :
Gillian Milani

Superviseurs :
Timothée Produit
Prof. François Golay

Un rapport de projet soumis pour l'accomplissement du titre de Master en ingénieurie de l'environnement

dans le

Laboratoire de systèmes d'information géographique
Ingénieurie de l'environnement

Juillet 2014, version corrigée

"Il est bien difficile, en géographie comme en morale, de connaître le monde sans sortir de chez soi."

Voltaire

"Sous nous, comme pour nous faire honneur en accompagnant notre marche, la terre se déroule en un immense tapis sans bords, sans commencement ni fin, aux couleurs variées où la dominante est le vert, dans tous ses accents comme dans tous ses mariages. Les champs en damiers irréguliers ont l'air de ces « couvertes » en pièces multicolores mais harmoniques rapportées par l'aiguille patiente de la ménagère. Il semble qu'une inépuisable boîte à joujoux vient d'être répandue profuse par cette terre, la terre que Swift nous découvrit vers Lilliput, comme si toutes les fabriques de Carlsruhe avaient vidé là leur stock. Joujoux ces petites maisons aux toits rouges ou ardoisés, joujoux cette église, cette prison, cette citadelle, les trois habitacles où se résume toute notre civilisation présente. Joujou bien plus encore ce soupçon de chemin de fer qui nous envoie de tout en bas son aigre petit cri de sifflet comme pour forcer notre attention, et qui tout mignon file si lentement — pourtant avec ses quinze lieues à l'heure — sur son rail invisible, panaché de sa petite aigrette de fumée... Et qu'est cet autre flocon blanchâtre que j'aperçois là-bas flottant par l'espace : la fumée d'un cigare ? Non, un nuage."

Nadar (Gaspard Felix Tournachon)

ECOLE POLYTECHNIQUE FEDERALE DE LAUSANNE

Résumé

ENAC
Ingénieurie de l'environnement

Master en ingénieurie de l'environnement

Pic2Map: intégration de photographies dans QGIS

par Gillian Milani

La technique du monoplotting consiste à combiner un modèle numérique de terrain et une photographie afin d'en extraire des informations spatiales. Bien que de nombreux outils existent déjà, le monoplotting n'a, jusqu'à maintenant, jamais trouvé sa place au sein même d'un logiciel SIG. Pour répondre aux changements actuels de la cartographie, un plugin de monoplotting a été développé pour QGIS : Pic2Map.

Facile d'installation et de prise en main, Pic2Map s'adresse avant tout à un public initié à l'utilisation de SIG. Une soin particulier a été apporté pour concilier la simplicité d'utilisation, la flexibilité et les performances. Entre autre, Pic2Map permet l'orthorectification de photographies obliques, la visualisation de couches vectorielles dans la photographie, la digitalisation d'objets vectoriels géoréférencés directement dans la photographie, etc. Par rapport aux outils de monoplotting déjà disponibles, Pic2Map offre l'intégration ultime dans le monde SIG.

Concernant l'estimation de la pose de la caméra, les algorthimes DLT et Levenberg-Marquardt ont été implémentés. Six points de contrôles sont nécessaires à la pose, dont les paramètres peuvent être fixés ou variables. Aucune information a priori sur la pose n'est nécessaire. La précision des projections entre la photographie et le modèle de terrain a été mesurée. Avec un modèle à une résolution spatiale de 200m, des écarts typiques de 50m sont observés pour des points situés à quelques kilomètres de la caméra. Des erreurs de 20m en moyenne sont observées avec un DEM à 25m de résolution spatiale pour une distance comparable.

Remerciements

Ce travail n'aurait pas eu la présente qualité sans l'intervention de nombreuses personnes m'ayant apporté leur aide. J'aimerais d'abord remercier Monsieur Timothée Produit, pour m'avoir offert la possibilité de travailler avec lui et m'avoir incité à m'instruire à un nouveau domaine de la science. Merci à Monsieur Matthew Parkan pour les discussions et les idées qui ont voltigé en GC D2 414. Merci à chaque personne qui est venu voir l'avancée du travail, qui a donné son avis et quelques idées. Un mot va naturellement aux sites de tutorial et forum (opengl-tutorial, nehe.gamedev, zetcode, stackexchange, etc.) sur lesquels de nombreux anonymes au grand coeur partagent leurs expériences. Finalement, je remercie laboratoire LASIG pour m'avoir permis d'effectuer ce travail dans une atmosphère sereine et dynamique. Plus qu'un travail de master, le temps passé au LASIG sera un grand souvenir.

Table des matières

Table des figures

Liste des tableaux

Abbréviations

DEM	**D**igital **E**levation **M**odel (modèle d'élévation)
WSL	Eidg. Forschungsanstalt für **W**ald, **S**chnee und **L**andschaft
	Institut fédéral de recherches sur la forêt, la neige et le paysage
GCP	**G**round **C**ontrol **P**oint
	Point de contrôle
DLT	**D**irect **L**inear **T**ransform
	Transformation Linéaire Directe
LM	algorithme de **L**evenberg **M**arquardt
WMS	**W**eb **M**ap **S**ervice
API	**A**pplication **P**rogramming **I**nterface
PnP	**P**erspective-**n**-**P**oint problem
	Problème de perspective à n points
DPW	**D**igital **P**hotogrammetric **W**orkstation

A mes collègues d'études

Chapitre 1

Introduction

Avant d'explorer les recherches déjà réalisées dans le domaine du monoplotting, il convient de fixer certaines bases et d'expliciter les objectifs du travail. Ce chapitre contient un bref survol historique des techniques photogrammétriques employées jusqu'à aujourd'hui. Les concepts principaux du présent travail y sont présentés ainsi qu'un résumé des méthodes et résultats.

1.1 De la photogrammétrie à la photogrammétrie

C'est en 1855 que Gaspard Felix Tournachon prit la première photographie aérienne de l'Histoire. Selon [1], le dessein de ces premières photographies aériennes fut de produire des cartes topographiques à des fins militaires, comme pour beaucoup d'innovations malheureusement. La technique photographique se développa progressivement pour donner finalement naissance en 1862 à la photogrammétrie (cf. [2]), l'art de combiner plusieurs images aériennes afin d'en extraire des informations en trois dimensions. Les produits cartographiques se sont depuis raprochés de plus en plus du monde réel.

Ces dernières années donnèrent lieu à de nombreux chamboulements dans le domaine de la cartographie. Les globes virtuels et géo-portails, largement répandus envers le grand public, créèrent un intérêt encore jamais vu pour les données géo-spatiales. Actuellement, dans les milieux professionnels, les logiciels et le matériel se perfectionnent de jour en jour comme, par exemple, avec les recours de plus en plus fréquents à des caméras hyperspectrales ou l'utilisation de mesures Lidar.

Plus spécifique au domaine de la photogrammétrie, des logiciels comme Pix4D réalisent des ajustements de photographies automatisés. Des caméras grand public, de moindre qualités que les caméras photogrammétriques, sont utilisables avec de tels logiciels.

D'autres développement, venant de la vision par ordinateur, font leur entrée dans le monde de la cartographie ; par exemple Photosynth, un logiciel récent de Microsoft. Il permet de combiner des images urbaines afin d'en extraire l'information 3D et de reconstruire un monde virtuel. Ces dernières applications montrent à quel point les principes de la photogrammétrie ont été développés et répandus dans de nombreux médias. On peut s'attendre à une augmentation de la demande pour des données de qualités dans les années à venir, ou tout au moins à une augmentation de la qualité des données désirées par le grand public.

Des solutions de cartographie pour des zones restreintes et des petits projets commencent par ailleurs à émerger. Prenons l'exemple d'une société se reposant sur des petits drones semi-automatiques afin de prendre des photographies [1]. Ces photographies sont automatiquement traitées dans des logiciels qui fournissent des produits de haute qualité. La cartographie par drone repose sur les mêmes principes et équations que ceux utilisés il y a plus de 150 ans par les pionniers de la photogrammétrie. Toutefois, il manque peut-être encore un solide pont entre les solutions pour le grand public et les technologies de pointe. Le livre [3] donne un bon aperçu de l'avancée actuelle de la photogrammétrie.

1.2 Mesures en 3D à partir d'images

La photogrammétrie se repose sur les principes de la vision stéréoscopique. Celle-ci permet, en combinant deux images, de connaître l'information de profondeur d'un objet. En d'autres termes, la photogrammétrie propose d'extraire des informations en trois dimensions à partir d'objets en deux dimensions. Elle permet notamment de créer des ortho-images. Une ortho-image n'est pas une photographie ou un dessin en vue perspective, telle que la voit l'œil humain, mais une photographie rectifiée en une vue orthogonale. La création d'une ortho-image passe par l'élaboration, implicite ou explicite, d'un modèle du terrain en trois dimensions. Ce modèle de terrain, créé en général pour ortho-rectifier des images aériennes, peut être enregistré et peut donc servir également à d'autres applications.

Beaucoup d'appellations existent pour nommer un modèle numérique de terrain. Nous allons ici utiliser l'abréviation DEM pour « Digital Elevation Model ». Un DEM peut être créé avec plusieurs images aériennes, comme déjà cité, mais aussi avec d'autres outils, tels que le lidar ou le radar.

1. Easy2map SA, CH-1066 Epalinges

La photogrammétrie n'est pas la seule solution pour extraire de l'information 3D depuis des images. Il est également possible d'en extraire en combinant un DEM et une ortho-image. Le DEM est drapé avec les informations couleurs de l'ortho-image. Il est donc possible de reconnaître des objets sur le DEM et d'y faire des mesures. Une approche conceptuellement similaire mais techniquement très différente consiste à utiliser une photographie à la place de l'ortho-image. Dans ce cas, une correspondance avec le DEM doit être établie au préalable. Une transformation mathématique projective modélise la relation entre l'espace image et l'espace 3D du DEM. Cette dernière approche est appelée "monoplotting" (c.f. [4–6]). L'auteur n'a malheureusement pas eu accès à l'ouvrage [7], souvent considéré comme étant la publication de naissance du monoplotting.

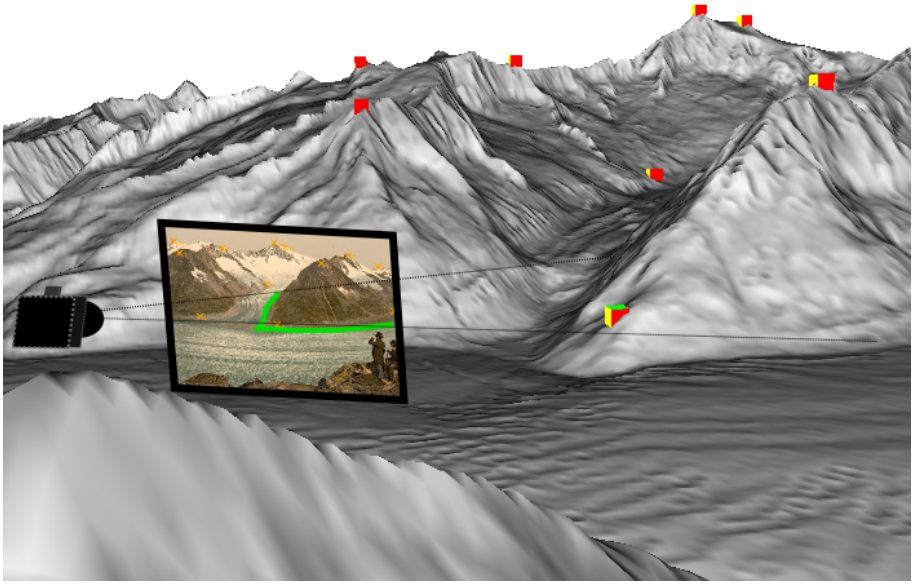


FIGURE 1.1: Principe de monoplotting

1.3 Monoplotting sur photographies

Utiliser un DEM et une photographie à la place d'images aériennes apporte certains avantages et inconvénients. Dans le cas du monoplotting, la position et l'orientation des caméras doivent être déterminées par rapport au DEM. Ce positionnement se fait en général grâce à des points de contrôle, points connus sur le DEM et sur l'image. Dans le cas de la photogrammétrie, il n'est pas toujours nécessaire de connaître la pose de la caméra. Certaines approches utilisent cette information, mais pas toutes. En aérotriangulation, des correspondances entre images, sans utiliser de DEM, fournissent les informations de positionnements relatifs. Toutefois, en photogrammétrie, comme en monoplotting, de tels

points de contrôle sont requis, soit pour la pose de la caméra, soit pour la correspondance entres images ou relativement au sol.

La présence d'autres différences plus importantes mérite tout de même d'être relevée. Par exemple, une estimation de la pose est souvent connue *a priori* par des mesures GPS et inertielles lors d'un travail par photogrammétrie aérienne. Cela facilite les calculs en permettant l'utilisation d'une pose initiale dans un calcul itératif. En monoplotting, la pose initiale peut être totalement inconnue. Il s'agit donc de mettre en place des outils et algorithmes capables de retrouver une solution initiale de pose ou d'aider l'utilisateur à trouver une pose estimée. Un autre point à relever est le type de données utilisées. La photogrammétrie travaille toujours avec plusieurs images, où un chevauchement important est nécessaire. Le monoplotting, quant à lui, ne requiert qu'un DEM en plus d'une photographie. Les modèles de terrain étant de plus en plus accessibles par un grand public, tel que le modèle ASTER [2], le monoplotting devient de plus en plus attrayant.

Ce qu'apporte le monoplotting par photographie par rapport à la photogrammétrie aéroportée peut se résumer ainsi :

— possibilité d'extraire de l'information à haute fréquence ; par exemple avec une webcam fixe pour étudier des étendues de neiges dans une vallée.
— possibilité d'extraire de l'information rapidement ; par exemple après une avalanche ou un glissement de terrain.
— utilisation de simples photographies terrestres ; ce qui libère la contrainte du vol.
— utilisation de photographies anciennes, même aéroportées ; par exemple si la photo est unique, sans recouvrement avec d'autres photographies.
— cartographie de zones non-propices à un survol aérien fréquent ; par exemple lors d'éruptions volcaniques ou de présence de brouillard ou de pollution continue.
— possibilités de traitement par des non-photogrammètres.

En outre, les différences sont toutes reliées de près ou de loin aux instruments employés : instruments aéroportés ou instruments au sol. Il est donc raisonnable, pour ne pas dire judicieux, de vouloir développer outil de monoplotting d'une part accessible par chacun et une simple d'utilisation et d'autre part simple de développement.

Les avantages précités sont obtenus notamment au détriment de la précision. Un outil de monoplotting n'a pas la prétention de pouvoir un jour remplacer les techniques photogrammétriques. C'est par contre un complément extrêmement prometteur é la photogrammétrie aéroportée.

2. Advanced Spaceborne Thermal Emission and Reflection Radiometer

1.4 Nécessité d'un nouvel outil

La question du besoin avéré d'un tel outil se pose. Le chapitre 2, "État de l'art" parcourt entre autres les solutions disponibles actuellement. Les techniques utilisées dans le monoplotting ont été largement explorées. Certains logiciels récents paraissent proposer les fonctionnalités de monoplotting. Malgré tout, aucun logiciel open source répandu ne propose d'outil intégré à un SIG, facilitant son utilisation et accessible à un large public.

Quand ce projet a démarré, quelques outils de monoplotting facile de prise en main semblaient être disponibles. Il en existe quelques-uns dans la famille des logiciels libres, comme ILWIS par exemple. Leur complexité rend cependant leur utilisation uniquement possible par des utilisateurs experts en la matière. D'autres, comme le WSL Monoplotting Tool, tendent à rendre l'expérience plus aisée pour l'utilisateur. Pourquoi donc refaire le même travail ?

Sur le plan technique, il était très intéressant et encourageant de pouvoir créer ses propres algorithmes. Nous avons donc essayé d'intégrer quelques innovations par rapport aux outils déjà disponibles sur le marché. Bien que certaines d'entre elles n'aient pas pu être réalisées, il y a bon espoir que d'autres laboratoires développent la cartographie par monoplotting et les algorithmes sous-jacents, de même que l'industrie du SIG intègre dans ses propres logiciels des outils de monoplotting. De plus, certaines implémentations apportent certainement quelques nouveautés aux solutions déjà existantes.

D'autre part, l'intrégration du monoplotting directement dans un logiciel SIG, permettant une synergie entre les modules, n'a pas encore été réalisée. L'argument le plus convaincant est certainement celui du plugin face à l'application indépendante. En effet, le logiciel QGIS propose déjà de nombreux outils et permet le développement de plugin interagissant avec les autres parties du logiciel. Par exemple, la gestion des différents systèmes de coordonnées est déjà prise en compte, de même que la lecture de fichier raster, la création de couches vectorielles, les symbologies graduées et catégorisées, la création de produits cartographiques finis, etc. Un plugin permet donc d'être simple dans son développement, tout en étant riche de synergie avec l'application mère, en proposant également une extension facilitée dans le futur, de même qu'une distribution facile.

Sur le plan conceptuel, la programmation open source a de nombreux avantages pour le domaine académique. Grâce à elle, il est possible de modifier un projet, de le compléter ou de s'inspirer des algorithmes utilisés. Dans le milieu académique, le développement open source est primordial. Le code est devenu un moyen traditionnel de développer de nouvelles approches scientifiques. Faire un logiciel académique sans open source revient à développer une nouvelle approche numérique sans dévoiler les équations. L'open source

est un acteur clé d'une science forte et efficace. De plus, les logiciels open source n'ont aujourd'hui pas le même public que les logiciels propriétaires.

Les quelques retours déjà reçus avant la fin du travail ont souvent été en lien avec des envies de développement et d'adaptation des techniques utilisées dans le plugin pour d'autres logiciels ou tâches spécifiques. Le pouvoir de développement de l'application open source est donc un point à ne pas sous-estimer.

1.5 Objectifs

L'essence de ce travail est de rendre publique et fonctionnelle une étude pensée et réalisée par Monsieur Timothée Produit ([8]) durant sa thèse de doctorat. Le workflow y a été défini. Le présent travail se focalise donc sur son adaptation et son implémentation. Les objectifs exposés sont concrets, ambitieux et raisonnables. Ils ont été fixés avec une part d'incertitude, puisque le développement logiciel était alors un milieu encore peu exploré par l'auteur. Les objectifs du travail de master sont listés ci-dessous.

- Créer un plugin de monoplotting pour le logiciel QGIS ;
- Rendre le plugin, pratique, rapide et ergonomique ;
- Obtenir des résultats précis, fiables et déterministes ;
- Obtenir un code structuré, commenté et lisible par un tiers initié aux langages ;
- Permettre de calculer la pose avec des points de contrôle ;
- Faciliter la prise de points de contrôle ;
- Fonctionnalités :
  - Permettre la digitalisation de nouvelles entités vectorielles sur la photographie ;
  - Permettre la visualisation de couches vectorielles ;
  - Permettre l'ortho-rectification de photographies ;
  - Offrir une certaine synergie avec d'autres objets informatiques (EXIF, KML, utilisation d'ortho-image, etc.) et logiciels (QGIS, Google Earth, autres SIG)

Le travail de master porte sur l'élaboration d'un outil informatique. Ce présent travail n'est donc qu'une mince fenêtre donnant un bref survol de l'accomplissement des objectifs et de l'effort fourni dans l'implémentation.

1.6 Résumé des résultats

Le plugin a été réalisé en Python, utilisant notamment le framework Qt. Les algorithmes Direct Linear Transform (DLT) et Levenberg-Marquadt ont été implémentés pour l'estimation de la pose. Ce duo permet notamment de calculer une pose sans aucune information a priori et de pouvoir fixer les paramètres de pose indépendamment les uns des autres.

Le plugin crée répond aux attentes initiales. Les processus les plus lents ont été optimisés afin d'offrir des temps de calculs acceptables. De nombreux petits efforts ont été consacrés à la facilité de digitalisation de points de contrôle, comme l'implémentation de la vue 3D visible sur la figure 1.2. L'estimation de la pose peut s'effectuer soit avec des points de contrôle, soit manuellement. Dans l'approche par points de contrôle, au moins six

points sont nécessaires. L'algorithme est capable de trouver une position initiale sans information a priori et de prendre en compte les paramètres connus par l'utilisateur.

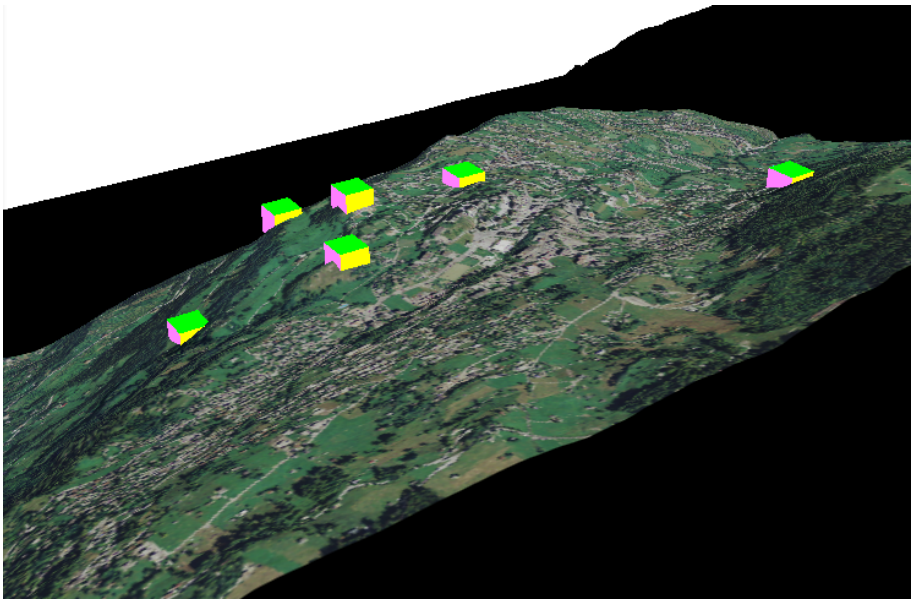


FIGURE 1.2: Une Ortho-photo peut être drapée sur le DEM. Les points de contrôle sont représentés en 3D. L'utilisateur peut se déplacer et digitaliser directement ses points de contrôle dans cette vue, représentés ici par des cubes.

Les couches vectorielles de QGIS peuvent être digitalisées dans la photographie. La visualisation de couches vectorielle est implémentée pour les points, lignes et polygones, incluant le support pour des symbologies simples, catégorisées et graduées. Un exemple est exposé par la figure 1.3. L'ortho-rectification est implémentée et permet de définir les coordonnées limites de l'ortho-image ainsi que sa résolution telles que présentées à la figure 1.4. Une fois la pose déterminée pour une photographie, il est possible de la sauvegarder et de l'importer dans le logiciel Google Earth, géo-référencée et orientée.

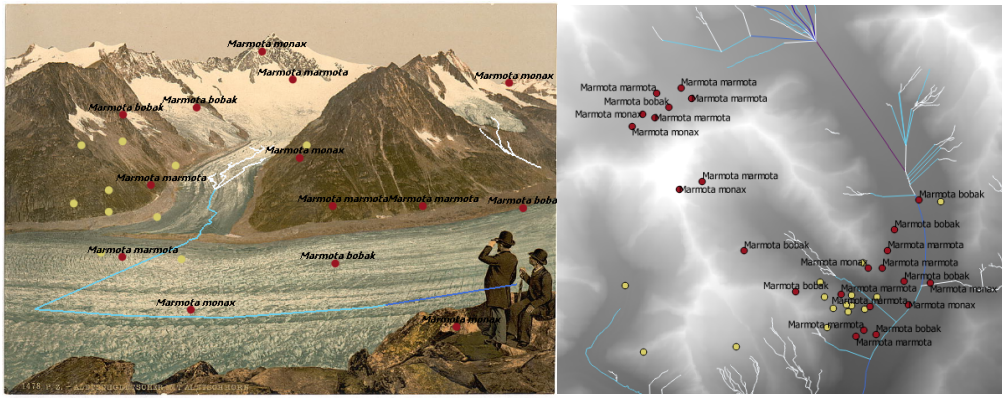


FIGURE 1.3: A gauche : visualisation de couches vectorielles sur la photographie avec des étiquettes et une symbologie graduée pour les rivières. A droite : représentation orthogonale standard de couches vectorielles dans QGIS. Les deux couches de points sont totalement fictives, existantes pour le seul besoin de la démonstration.

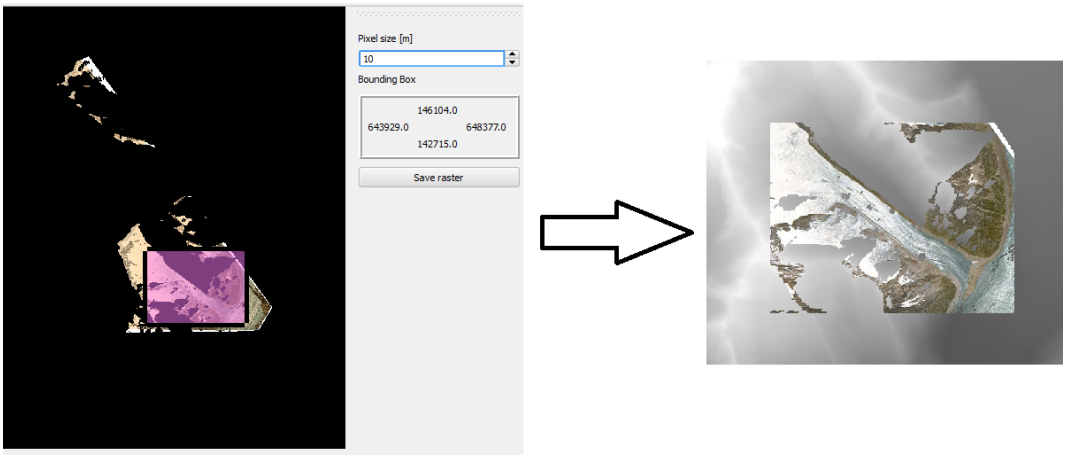


FIGURE 1.4: Exemple d'ortho-rectification. Il est possible d'y définir les coordonnées limites et la résolution du fichier raster.

1.7 Structure du travail

Le chapitre introductif clarifie les principes conceptuels sous-jacents à l'implémentation du plugin. La pertinence de l'outil est présentée, de même que les objectifs du travail. Le chapitre 2, "État de l'art", détaille les solutions logicielles déjà existantes et fait l'historique du développement du monoplotting. Les références des algorithmes implémentés sont citées également au chapitre 2. Le chapitre 3, "Théorie et techniques", explicite les mathématiques sous-jacentes à l'implémentation. Certains concepts sont présentés en profondeur. De nombreux détails concernant les équations sont à consulter dans les annexes. Le chapitre 4, "Réalisation", expose l'implémentation du plugin. Les outils informatiques sont rapidement survolés, suivis de l'adaptation des concepts mathématiques pour le cas spécifique du plugin. La précision de l'outil est particulièrement considérée dans le chapitre 4. Finalement, les limites du plugin et les possibles développements sont exposés en fin de document.

1.8 Clarifications

Le mot "pose" est utilisé ici pour exprimer l'orientation externe de la caméra, c'est-à-dire la position et l'orientation. L'estimation de la pose inclus également l'estimation de paramètres internes comme la focale ou le point central.

Le terme "reprojection" correspond au mot anglais "back-projection". Lorsqu'un point est transformé, d'une espace 3D sur un plan, on parle de projection. La reprojection exprime la transformation inverse, c'est-à-dire la transformation d'un point sur un plan jusque dans l'espace 3D. Ces deux transformations sont très différentes.

Chapitre 2

État de l'art

> « Beaucoup de conséquences en découlent, en particulier : [...]
> – des développements qui progressivement ont quitté le domaine académique (avec de nombreuses publications), pour devenir quasi-exclusivement du domaine industriel, sans aucune publication. Ceci s'est traduit dans les dernières décennies, lors du passage au numérique, par de véritables boîtes noires sans aucun moyen pour l'usager de savoir ce qu'il s'y faisait exactement. On a d'ailleurs assisté régulièrement à des travaux de recherche, dans des domaines connexes potentiellement usagers de cette technique, qui en réinventaient tout ou une partie, par faute de publications accessibles. » [1]

Les stations de photogrammétrie numérique (Digital Photogrammetric Workstation, DPW), existent depuis quelques décennies déjà et permettent notamment de travailler dans un mode mono-photogrammétrique, ou en d'autres termes, permettent le monoplotting (c.f. [9]). Ici, nous nous intéressons à un autre type de matériel. Comme proposé dans [10], c'est plus exactement l'interaction entre une photographie et un SIG qui est le centre d'intérêt. Les spécificités de l'approche proposée par rapport à la DPW sont l'équipement nécessaire (PC contre station de travail), l'intégration d'objets standards utilisés dans les SIG (rasters tiff, couches shapefile, etc.) et la capacité d'être utilisée par des personnes novices dans le domaine de la photogrammétrie.

2.1 Logiciel de monoplotting

La technique de monoplotting n'est pas récente. Certains logiciels ont déjà été développés il y a plus de 25 ans, comme celui présenté dans [5]. La publication [7] peut être considérée comme le début du monoplotting. Suivant les descriptions extraites de différentes

publications citées dans ce chapitre, ces premiers logiciels semblent s'apparenter en de nombreux points aux logiciels actuels de monoplotting en ce qui concerne les fonctionnalités proposées. Toutefois, la distribution et la prise en main devaient être significativement plus complexes, car de nombreux outils, notamment d'interfaces, n'existaient pas.

Aujourd'hui, il faut reconnaître l'existence d'outils déjà disponibles sur le marché, open source ou propriétaire. Seulement deux outils actuels seront cités ici, par soucis de pertinence. Il est, en quelque sorte, surprenant de n'en trouver qu'un nombre restreint. Les recherches en bibliothèque ou sur internet ne dirigent que vers peu de solutions.

Pour commencer avec les logiciels propriétaires, le WSL Monoplotting-Tool ([11]) propose les fonctionnalités standards de monoplotting, c'est-à-dire la digitalisation de couches vectorielles et la visualisation de couches vectorielles dans la photographie. De plus, ce logiciel propose un algorithme d'estimation de pose, module qui n'est pas disponible dans tout logiciel de monoplotting. Il faut également reconnaître à ce logiciel l'effort mis en œuvre pour l'accessibilité de cette technique à un public plus large qu'auparavant. L'auteur a assisté à une démonstration du logiciel mais ne l'a pas testé à proprement parler. Ce logiciel est certainement le plus abouti et le plus utilisable par un large public à l'heure actuelle. Cet outil de monoplotting a par ailleurs déjà servi à quelques études scientifiques, telles que [12] ou [13].

Un autre exemple de solution, cette fois-ci au niveau des logiciels open source, est le logiciel ILWIS de [14]. Il propose depuis quelques années un module de monoplotting. Les fonctions de monoplotting proposées sont les fonctions basiques de digitalisation. La prise en main difficile d'ILWIS ne facilite pas la considération de ce logiciel. Ce point est d'ailleurs très important, puisqu'il constitue un obstacle à une utilisation par un large public. Le logiciel ILWIS a cependant l'avantage de proposer une estimation de la pose avec un nombre points de contrôle, dépendamment de l'algorithme de pose choisi et des données entrées par l'utilisateur.

Un autre exemple de solution serait les globes virtuels, comme World Wind ou Google Earth. Plus particulièrement, ce dernier permet d'inclure des photographies de paysages, de les géo-référencer et de les orienter. Il permet également de dessiner des objets vectoriels sur le DEM. Cependant, les fonctionnalités spécifiques de dessin directement sur l'image ne sont pas (encore) disponibles. Une telle solution n'est pas inintéressante, mais propose des capacités et des applications différentes par rapport à du monoplotting sur photographie. Elles sont en outre complémentaires sur bien des aspects.

D'autres logiciels, plus anciens, ont été développés dans des instituts de recherches. Par exemple, [15] ont développé un ensemble de logiciel supportant des fonctionnalités de

monoplotting. Un premier logiciel est utilisé pour la détermination de la pose de la caméra. Un minimum de trois points connus est nécessaire. La focale est déterminée par un processus itératif et sous certaines hypothèses restrictives se basant sur les caméras disponibles sur le marché à cette époque. Le deuxième logiciel sert à rectifier l'image oblique et peut également projeter des couches vectorielles planimétriques dans la photographie. La méthode est décrite comme fonctionnelle jusqu'à un angle d'incidence de 60 °. Ce fait, couplé à l'exemple décrit dans la publication, indique que des photos obliques aériennes sont tout de même nécessaires. Du moment qu'un vol aérien est nécessaire, la photogrammétrie classique devient avantageuse par rapport au monoplotting. Ceci peut être un facteur expliquant la faible distribution apparemment de cet outil.

Ainsi, les techniques utilisées dans le monoplotting ont été largement explorées. Certains logiciels récents paraissent proposer des fonctionnalités intéressantes. Malgré tout, aucun logiciel open source répandu ne propose d'outil capable d'être utilisé simplement et développé au cas par cas, comme proposé dans [10].

2.2 Solutions indépendantes

De nombreuses solutions indépendantes semblent également exister. Des outils spécifiques ont été développés, chacun pour leurs propes études telles que [16] ou [17]. Ces solutions spécifique, à la carte, émergent peut-être par manque de solutions appropriées ou par manque de flexibilité ou de capacité de développement (open source) des solutions sur le marché. Comme mentionné à la section 1.4, des outils open source possédant une grande capacité de développement sont nécessaires au monde académique.

Un projet de master de l'ETH [18] s'était intéressé à l'élaboration d'un logiciel qui visait à introduire les étudiants en géomatique à la problématique du monoplotting. La particularité de ce logiciel est de n'offrir qu'un simple aperçu du monoplotting : l'introduction de jeux de données autres que les originaux est impossible. Ce logiciel n'apparaît pas ainsi comme un d'outil de monoplotting fonctionnel.

2.3 Photogrammétrie terrestre et laser scanner

La photogrammétrie terrestre (close range photogrammetry) n'est pas vraiment adaptée pour du monoplotting comme on l'entend en cartographie. La photogrammétrie terrestre a donné naissance à de nombreuses applications (c.f. [19], [20]). Cependant, les approches sont souvent différentes. Par exemple, on travail souvent avec un nuage de points brut en photogrammétrie terrestre, tandis que l'on travaille plutôt avec un DEM en cartographie.

Par ailleurs, l'intégration de la 3D n'est pas encore complètement intégrée aux SIG [1] et reste très spécifique à l'environnement urbain, comme présentée dans l'article [22].

La photogrammétrie terrestre est souvent complétée ou comparée au laser scanner pour la mesure de l'information 3D à petite échelle (cf. [23]). Ceci est certainement expliqué par la distance à laquelle un laser scanner est utilisable, qui est du même ordre de grandeur que la distance des objets en photogrammétrie terrestre. Dans ces deux applications, le nuage de points est préféré au modèle de terrain pour la représentation et le travail en 3D. Le modèle de terrain est en effet approprié pour des représentation en 2.5D. L'utilisation de modèle par polygones pour des objets de haute complexité spatiale n'est pas appropriée ; c'est pourquoi le nuage de point est préféré, par exemple pour la représentation d'arbres ou de falaises.

Certaines institutions sont allées très loin dans le développement de logiciels pour la gestion de photographies terrestres. Comme décrit dans [24], certains laboratoires ont émis des efforts très importants dans ce domaine. Ces efforts restent toutefois très spécialisés et spécifiques. Aucune application grand public n'a été apparemment construite sur cette base, peut-être pour la simple raison que le besoin ne se faisait pas encore ressentir.

Certaines recherches ont également porté sur la combinaison de Lidar et photographie aériennes utilisant des caméras à petit format, tel que dans [25]. L'utilisation de nuages de points pour du monoplotting sur de l'imagerie aéroportée rend le processus extrêmement similaire à la photogrammétrie terrestre.

2.4 Algorithmique

Au niveau des algorithmes utilisés dans le plugin, de nombreuses solutions ont proposées dans des domaines parfois divers. L'estimation de la pose est un problème exploré par beaucoup de chercheurs ([26], [27], [28]). Bien que depuis longtemps utilisée en photogrammétrie, elle a été revue par la vision, comme décrit dans [29].

Le livre [30] regroupe les solutions explorées par la photogrammétrie et la vision par ordinateur. De très nombreuses techniques sont présentées pour tous types de problèmes. L'estimation de la pose a donc atteint des limites difficilement extensibles. Par exemple, l'algorithme efficient-PnP ([31]) est capable d'estimer une pose précise avec quatre points seulement. Citons finalement [32] qui propose une estimation de la pose avec seulement quatre point, qui prend en compte l'estimation de la focale et des déformations radiales.

1. Le terme 2.5D est souvent utilisé pour différentier la représentation spatiale d'un SIG avec un espace en 3D (cf. [21]).

Cette approche paraît prommeteuse mais n'a pas été considéré ici pour des questions de complexité.

D'autres algorithmes ont été utilisés, comme Levenberg-Marquadt le décrit dans [33]. Les bases de l'approche par moindre carré peuvent être consultées dans [34]. L'adaptation de l'algorithme Levenberg-Marquardt avec les équations de colinéarités a déjà été proposé dans [35].

Chapitre 3

Théorie et techniques

Dans ce chapitre, les bases mathématiques sous-jacentes au plugin sont présentées. De nombreuses équations sont explicitées dans les annexes. La problématique de l'estimation de la pose est traitée dans une large mesure. Avant d'arriver à ce point, les bases de la géométrie d'une prise de vue photographique sont présentées. Le but de ce chapitre est de dévoiler l'entier du processus permettant le positionnement de l'image par rapport au DEM.

Ce but est atteint grâce aux points de contrôle (GCP). Chaque GCP possède une coordonnée 2D dans l'image et une coordonnée 3D dans le DEM. Estimer la pose de la caméra revient à trouver une configuration unique de l'image : chaque ligne allant du point focale à la coordonnée du GCP sur le DEM doit traverser l'image par la coordonnée du GCP sur l'image. Une représentation schématique est donnée par la figure 3.1.

3.1 Géométrie de la photographie

En terme mathématique, une photographie est assimilable à une projection d'un espace 3D sur un plan. Le modèle utilisé ici est appelé "modèle sténopé". Afin d'introduire cette notion solidement, les systèmes de références seront d'abord définis, suivi des transformations entre systèmes de références. Le modèle sténopé sera ensuite introduit, suivi des équations de colinéarités. Certaines notions vont être redéfinies et précisées dans cette section, ceci pour une clarification des conventions utilisées.

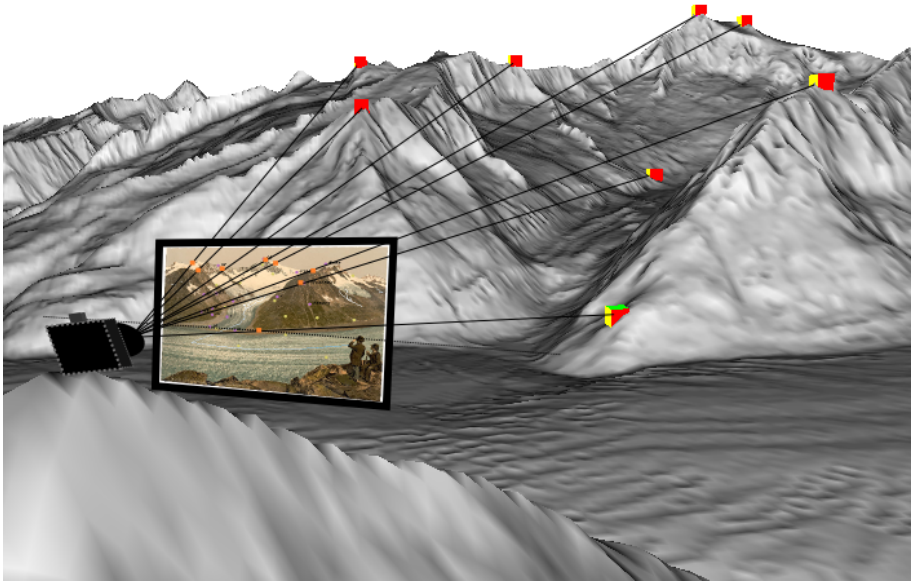


FIGURE 3.1: Principes du positionnement par GCP : les coordonnée de l'image et du terrain doivent être alignés sur le point focal. Les informations sont les coordonnées des GCP dans l'image, d'une part et les coordonnées dans le modèle de terrain d'autre part.

3.1.1 Systèmes de référence

Afin de se repérer dans l'espace, l'homme a tendance à mettre des mots sur ce qu'il voit : «Je suis devant l'entrée de la gare».

Les descriptions précises peuvent faire place au concept : «je vais au centre-ville». On ne voit pas le centre d'une ville ; ce n'est pas un objet physique que l'on peut toucher et le centre n'est en général pas clairement limité par une ligne précise.

Quand les descriptions et les concepts ne suffisent pas, pour des raisons de précisions, de généralités ou autre, un système numérique plus rigoureux peut être utilisé : «J'habite à la deuxième maison de la rue, qui est la première à gauche depuis la grand-rue.»

On peut étendre cette notion de spatialité numérique en quadrillant le territoire selon deux axes. Chaque point peut être repéré dans ce système unique de quadrillage que l'on dénommera « système de coordonnée ». Les systèmes de coordonnées, introduits ici par la logique, ont également une définition mathématique et répondent à des règles précises (cf. [36]).

Deux grandes familles de système de coordonnées cohabitent. La première comprend les « systèmes de coordonnées géographiques », où chaque point est défini par une longitude, une latitude et accessoirement une élévation. En terme mathématique, chaque point est

donc défini par deux angles (longitude, latitude) et une distance (élévation). Ce type de système est optimal pour décrire un positionnement sur une sphère. La Terre étant ronde, ou presque, on comprend tout de suite son utilité.

La seconde famille comprend les « systèmes de coordonnées projetées ». Chaque point y est défini selon deux axes horizontaux et un axe vertical. En termes mathématiques, chaque point est donc défini par trois distances (Nord, Est, altitude). Ce type de système-ci est optimal pour un positionnement sur un plan. Le terme « projeté » vient de la projection des points de la sphère (système géographique) sur un plan, un cône ou un cylindre. De nombreux types de projections existent, chacune avec ses avantages et inconvénients respectifs ; certains conservent les distances, d'autres les angles, d'autres les aires, etc.). Durant le travail, les systèmes de coordonnées projetées seront uniquement présents. Le terme « projeté » sera dès lors implicite.

Nous utiliserons ici principalement trois systèmes de coordonnées : Le système objet, le système caméra et le système image. Ces trois systèmes sont définis dans l'annexe A.1. La figure 3.2 représente ces trois systèmes distincts. Les deux systèmes de références spatiaux, c'est-à-dire objet et caméra, sont faciles à se représenter. Le système image requiert quelques précisions en plus, n'étant pas un système en trois dimensions, ni un système métrique.

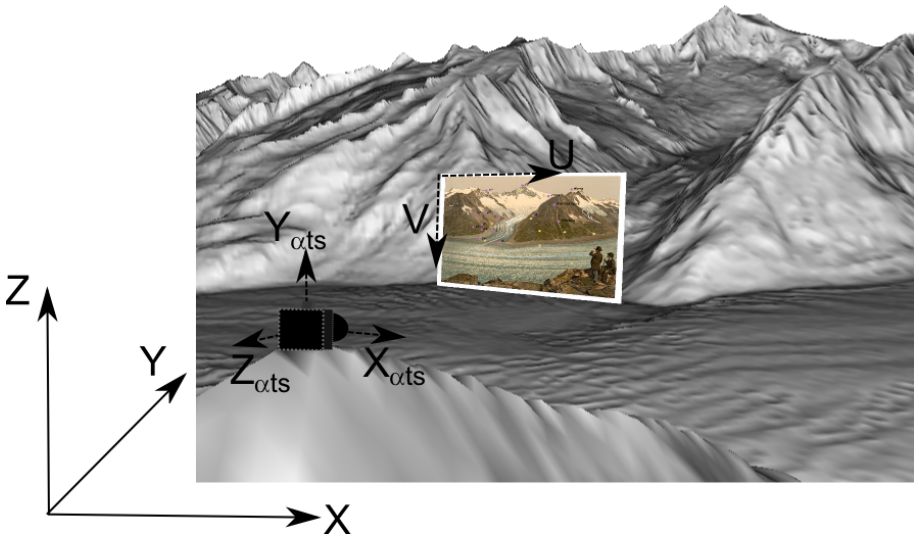


FIGURE 3.2: Représentation des systèmes de coordonnées objet (X, Y, Z), caméra $(X_{\alpha ts} Y_{\alpha ts} Z_{\alpha ts})$ et image (U, V).

Une photographie numérique possède également un système de coordonnées, bien qu'il ne soit pas en trois dimensions. L'espace mathématique d'une image défini un plan possédant seulement deux dimensions, alors que les objets représentés sur l'image existent

également dans le monde à trois dimensions spatiales. Il est pourtant possible de trouver une correspondance entre ces deux espaces.

Afin de bien comprendre cette correspondance, remarquons les différences entre un espace non-métrique en deux dimensions et un espace métrique en trois dimensions. Afin de ne pas trop s'égarer, nous utiliserons les termes « images » et « photographie » pour des fichiers stockés numériquement. Ceci précise le fait qu'une photographie est, ici, un objet en deux dimensions. Le système de coordonnées utilisé pour une image est en général défini selon sa résolution. Au temps des photographies argentiques, il aurait été difficile de comprendre ce terme. Mais aujourd'hui, la résolution d'une image est clairement définie et comprise par chacun. Le pixel de l'image sera défini comme la référence unitaire pour exprimer sa taille. Nous verrons plus tard que l'expression de distance dans un espace comprenant l'image comme plan sera utile pour exprimer la transformation entre le système objet et le système image. Ce troisième espace est appelé système caméra, déjà cité plus haut.

Avant de s'intéresser à cette correspondance à proprement parler, il convient de définir encore clairement les transformations entre les systèmes objet et caméra ; car le système caméra n'est qu'un intérmédaire, tel que représenté sur la figure 3.3, entre ces deux systèmes et ne présente pas donc d'autre intérêt.

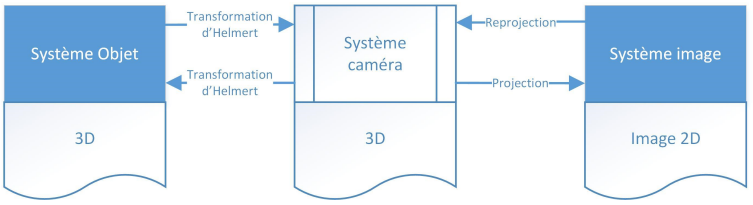


FIGURE 3.3: Représentation des systèmes de coordonnées et de leurs transformations.

3.1.2 Changement de système de référence et forme homogène

Afin de définir les transformations entre systèmes de références, une introduction aux matrices de rotation est présentée à l'annexe A.4. Cette introduction a notamment le mérite de fixer et d'exposer la convention d'angles utilisée. En effet, il est apparu souvent que des ambiguïtés, voire des erreurs, apparaissent quand les conventions sont mélangées.

Quand la position d'un point est connue dans un système de coordonnées, il est possible d'exprimer cette position dans un autre si l'on connaît la transformation entre ces deux systèmes. En considérant un cas restreint, la transformation d'un système de coordonnées projetées à un autre revient à une transformation entre deux systèmes cartésiens plus une contrainte (d'échelle). Ceci équivaut à trouver une translation et une rotation pour

déterminer la transformation. On notera désormais $\boldsymbol{x'}$ un vecteur exprimé dans le système caméra. La transformation peut être exprimée par :

$$\boldsymbol{x'} = R \cdot (\boldsymbol{x} - \boldsymbol{X_L}) = \begin{pmatrix} m_{1,1} & m_{1,2} & m_{1,3} \\ m_{2,1} & m_{2,2} & m_{2,3} \\ m_{3,1} & m_{3,2} & m_{3,3} \end{pmatrix} \begin{bmatrix} x - X_L \\ y - Y_L \\ z - Z_L \end{bmatrix} \tag{3.1.1}$$

Où $\boldsymbol{X_L}$ est un vecteur translation et R est une matrice de rotation en trois dimensions. Une astuce mathématique souvent utilisée consiste à exprimer un problème en y ajoutant une ou plusieurs dimensions afin de l'écrire sous forme linéaire. En géométrie dans l'espace, on parle alors de coordonnées homogènes. Le nom fait référence aux polynômes homogènes avec lesquels il est possible de réécrire la transformation.

Dans le cas d'une translation et d'une rotation du système de référence, il est possible d'écrire la transformation d'eun point X vers un point X' à l'aide d'une matrice 4x4 $\boldsymbol{Q_{RT}}$ donnée par :

$$\begin{bmatrix} X' \\ Y' \\ Z' \\ 1 \end{bmatrix} = \begin{pmatrix} m_{1,1} & m_{1,2} & m_{1,3} & 0 \\ m_{2,1} & m_{2,2} & m_{2,3} & 0 \\ m_{3,1} & m_{3,2} & m_{3,3} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & -X_L \\ 0 & 1 & 0 & -Y_L \\ 0 & 0 & 1 & -Z_L \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \boldsymbol{Q_{RT}} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{3.1.2}$$

3.1.3 Projection sur un plan - modèle Sténopé

Maintenant que nous avons défini la transformation du système objet au système caméra, nous pouvons nous focaliser sur la transformaton entre le système caméra et le système image. La projection d'un point selon un modèle de perspective parfait dans un espace plan est défini en annexe A.5. Nous nous intéressons ici plus particulièrement au modèle sténopé ayant comme paramètres internes la focale et le point central, représentés sur la figure 3.4.

Dans le cas d'une projection réelle (qui s'oppose à la projection mathématique, parfaite, définie plus haut), le point focal de l'image n'est pas aligné avec le centre de l'image. La projection sur le plan image du point focal est appelée point central et écrit ici $(u_0, v_0)^t$. En utilisant la grandeur de focale f, on réécrit l'équation d'une projection idéale :

Soit le point central (u_0, v_0) exprimée en unitée du système image. La transformation perspective du point $\boldsymbol{X'}$ sur le plan $z = -f$ est donnée par :

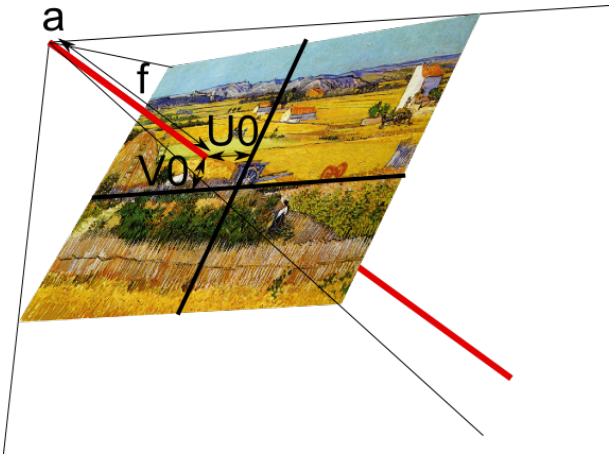


FIGURE 3.4: Représentation schématique du modèle sténopé : la projection est définie selon la distance focale f qui lie le point focal a au point central défini par $U_0 et V_0$.

$$\begin{bmatrix} zx \\ zy \\ z \end{bmatrix} = P\boldsymbol{X'} = \begin{pmatrix} 1 & 0 & u_0 \\ 0 & 1 & v_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1/f \end{pmatrix} \begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} \tag{3.1.3}$$

Pour chaque point $\boldsymbol{X'}$ dans le système caméra, une projection (x,y) unique sur le plan existe. La matrice de projection P défini la transformation. Ici, P est représenté comme le produit de deux matrice : la matrice du point central et la matrice focale. On remarque que la transformation $(X', Y', Z') \rightarrow (x, y)$ n'est pas inversible. La transformation n'est en fait pas injective. La fonction n'est donc pas bijective et ne possède donc pas d'inverse. Il n'est donc pas possible de trouver un point unique $\boldsymbol{X}$ correspondant à un point (x, y) de l'image. En d'autre terme, il est facile de projeter un point, mais plus difficile de trouver un point correspondant à une projection.

3.1.4 Equation de colinéarité

En multipliant chaque côté de l'équation 3.1.3 par la matrice inverse du point central, en incluant le changement de système de référence selon l'équation 3.1.1 et divisant les deux premières lignes par la troisième, on trouve les équations dites de colinéarité :

$$x = u_0 - f\frac{m_{1,1} \cdot (X - X_L) + m_{1,2} \cdot (Y - Y_L) + m_{1,3} \cdot (Z - Z_L)}{m_{3,1} \cdot (X - X_L) + m_{3,2} \cdot (Y - Y_L) + m_{3,3} \cdot (Z - Z_L)}$$

$$y = v_0 - f\frac{m_{2,1} \cdot (X - X_L) + m_{2,2} \cdot (Y - Y_L) + m_{2,3} \cdot (Z - Z_L)}{m_{3,1} \cdot (X - X_L) + m_{3,2} \cdot (Y - Y_L) + m_{3,3} \cdot (Z - Z_L)} \tag{3.1.4}$$

Le nom "colinéarité" provient du fait que les deux vecteurs partant du point focale, l'un allant jusqu'à un point quelconque et l'autre sur le plan image, sont toujours colinéaires. Une des approches standards utilisée en photogrammétrie pour le géoréférencement, aussi appelée aérotriangulation par faisceaux, se base sur la résolution par moindres carrés des deux équations précédentes. De nombreuses variantes existent (modèle indépendants, par blocs, par faisceaux, etc.) dont la plupart reposent sur les équations de colinéarité couplées à des méthodes numériques de moindres carrés.

Les méthodes itératives de moindres carrés nécessitent une solution initiale approchée. En photogrammétrie, une solution approchée peut toujours être estimée. Au début de la photogrammétrie, les images étaient toujours prises environs verticalement, ce qui donnait une valeur estimée pour le tangage et le roulis. Aujourd'hui, la valeur initiale approchée est donnée par la navigation par satellites et par des mesures inertielles.

Dans le cas d'une photographie oblique, il n'est pas forcément facile ni possible d'estimer les paramètres de pose. Une solution analytique serait donc préférable à une solution numérique ; ainsi, aucune initialisation estimée ne serait nécessaire.

Dans le but de ne pas requérir à une solution initiale, il est possible de reformuler le problème précédant par changement de variables et de transformer les équations de colinéarité en un système linéaire homogène. Ce système peut dès lors être résolu par la méthode des moindres carrés, qui est non-itérative pour le cas linéaire. Le fait de reformuler le problème en un système linéaire homogène puis de le résoudre par moindres carrés est appelé "Direct Linear Transform" (transformation linéaire directe). Il est donc possible de reformuler les équations de colinéarité 3.1.4 à l'aide d'une seule matrice :

$$\begin{bmatrix} zu \\ zv \\ z \end{bmatrix} = L \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{pmatrix} L_1 & L_2 & L_3 & L_4 \\ L_5 & L_6 & L_7 & L_8 \\ L_9 & L_{10} & L_{11} & 1 \end{pmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{3.1.5}$$

Les coordonnées images de la projection sont données dans l'annexe A.6, de même que les relations entre les paramètres de la DLT et les paramètres du modèle sténopé. Notons déjà que, connaissant les paramètres de projection, il est facile de trouver les paramètres de la DLT. Le contraire n'est pas trivial.

3.2 Estimation de la pose

La pose de la caméra comprend ici neuf paramètres [1] : trois pour la position, trois pour l'orientation, deux pour le point central et un pour la focale. L'estimation de la pose se fait en deux étapes. D'abord, une première estimation est calculée par l'algorithme Direct Linear Transform (DLT). Ensuite, une approche par moindres carrés est utilisée afin de fixer certains paramètres. L'algorithme de Levenberg-Marquardt a été implémenté pour la partie itérative, car les équations de colinéarités sont décrites comme hautement non-linéaires ; la forte instabilité de la solution itérative doit être prise en compte.

Pourquoi ne pas choisir un algorithme de pose plus performant, comme le PnP ? Ce dernier permet l'estimation de la pose avec seulement trois points (quatre pour enlever l'ambiguïté de positionnement propre au P3P).

L'algorithme DLT, contrairement à PnP, permet d'estimer certains paramètres de l'orientation interne de la caméra. Plus précisément, DLT permet l'estimation de la focale et du point central. L'implémentation d'une minimisation par moindres carrés est également nécessaire pour fixer certains paramètres pouvant être connus. Ce duo d'algorithmes n'a donc pas été choisi pour sa précision de pose, mais pour ses caractéristiques d'estimation de paramètres rendant très flexible son utilisation.

En résumé, ce duo apporte deux fonctionnalités essentielles au coût de deux GCPs de plus :

— l'obtention d'une estimation de chaque paramètre sans aucune valeur initiale, y compris la focale ;
— l'ancrage de certains paramètres, donnant la possibilité de fixer les paramètres connus.

3.2.1 Transformation linéaire direct

L'algorithme DLT est ici implémenté selon une utilisation spécifique. Souvent, on ne recherche que la matrice de projection homographique, c'est-à-dire le résultat direct de la DLT. Dans ce cas, l'algorithme DLT est très puissant. Tout ce complique lorsqu'il s'agit d'estimation de pose : une deuxième étape est nécessaire à l'extraction des paramètres depuis la matrice d'homographie. Tout l'art de l'implémentation de la DLT ce trouve dans cette seconde étape.

1. attention à ne pas confondre les paramètres de pose, ici au nombre de neuf, et les paramètres de la DLT, toujours au nombre de onze. Alors que les paramètres de pose ont une signification physique, les paramètres de la DLT sont des paramètres purement mathématiques.

(3.2.1)

$$\begin{pmatrix} X_1 & Y_1 & Z_1 & 1 & 0 & 0 & 0 & 0 & -u_1X_1 & -u_1Y_1 & -u_1Z_1 \\ 0 & 0 & 0 & 0 & X_1 & Y_1 & Z_1 & 1 & -v_1X_1 & -v_1Y_1 & -v_1Z_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ X_n & Y_n & Z_n & 1 & 0 & 0 & 0 & 0 & -u_nX_n & -u_nY_1 & -u_nZ_n \\ 0 & 0 & 0 & 0 & X_n & Y_n & Z_n & 1 & -v_nX_n & -v_nY_1 & -v_nZ_n \end{pmatrix} \begin{bmatrix} L_1 \\ L_2 \\ L_3 \\ L_4 \\ L_5 \\ L_6 \\ L_7 \\ L_8 \\ L_9 \\ L_{10} \\ L_{11} \end{bmatrix} = \begin{bmatrix} u_1 \\ v_1 \\ \vdots \\ u_n \\ v_n \end{bmatrix}$$

Où (X_i, Y_i, Z_i) représente les coordonnées d'un point de contrôle dans le système objet et (u_i, vi) les coordonnées du même point de contrôle dans le système image.

Le vecteur de position (X0, Y0, Z0) et la focale peuvent être facilement extraits des paramètres de la DLT. Cependant, il est plus difficile d'extraire les paramètres de rotation. L'extraction des rotations est faite en deux étapes : D'abord une approximation de la matrice de rotation est extraite de la projection homographique, ensuite, on recherche la matrice de rotation qui approxime le mieux l'estimation initiale (estimation qui n'est pas une matrice de rotation, car elle ne respecte pas les propriétés spécifiques). La passionnante recherche de la matrice de rotation est détaillée dans l'annexe A.8.4.

3.2.2 Algorithme Levenberg-Marquardt

La résolution de problème par moindres carrés a été largement explorée. Cette approche est utilisée afin de déterminer les paramètres d'une fonction (modèle) sur la base d'observations et de mesures. Le but étant de minimiser le carré de l'erreur entre les mesures et le modèle. La résolution par moindres carrés a été choisie car elle permet une flexibilité totale parmis les paramètres. Ainsi, si la position, ou la focale, ou les deux sont connues a priori, il est possible de les fixer. Quand un paramètre est fixé, le modèle le considère comme une simple constante et cherche à optimiser seulement les paramètres libres.

Originellement développée pour l'astronomie et la géodésie, l'approche par moindres carrés est employée partout et déclinée en de nombreuses variantes. Les méthodes d'optimisation par itération sont, en général, basées sur une solution initiale et sur un algorithme

de mise à jour de la solution. Afin de trouver les meilleurs paramètres pour le modèle fonctionnel étudié, on pose une fonction dite "objective".

Une fonction objective est une fonction quelconque représentant l'erreur entre le modèle et les mesures. On comprend en général la fonction objective comme fonction de coût, les paramètres optimaux se trouvant au minimum de la fonction objective.

Dans le cas général d'optimisation par itération, on étudie le comportement de la fonction objective dans le voisinage de la solution initiale pour ensuite mettre à jour les paramètres de la fonction une fonction f au point x_0, on déduit le comportement de la fonction objective pour trouver son minimum. Soit :

$$\begin{aligned} \delta x &= g(y - f(l, X_n)) \\ x_{t+1} &= x_t + \delta x \end{aligned} \tag{3.2.2}$$

où g() définit une fonction de l'erreur de prédiction, ou en d'autres termes, le comportement de la fonction objective au point xi, l représente le vecteur d'observation, x représente le vecteur de paramètres à optimiser et f représente le modèle fonctionnel. La fonction f lie les observations l_i (abscisses) aux mesures y_i (ordonnées).

Pour des problèmes non-linéaires, la résolution par moindres carrés possède de nombreuses variantes, chacune avec leurs avantages et inconvénients. Une des variantes les plus répandues est celle de Levenberg-Marquardt. Avant d'entrer dans le détail de cette variante, il importe d'introduire un autre algorithme : la descente de gradient.

Pour la descente de gradient, comme pour toutes les approches par moindres carrés, la fonction objective correspond à la somme des carrés des erreurs de prédiction :

$$Objective(x) = \sum_{i=1}^{n} \left[\frac{y_i - f(l_i, x)}{w_i} \right]^2 \tag{3.2.3}$$

où y_i représente la valeur observée de f à l'indice i et w_i un poids donné à l'observation i [2]. Le pas de mise à jour de la descente de gradient est égal à :

$$\delta x = \alpha J^T W (y - f(l, x)) \tag{3.2.4}$$

2. La matrice de poids est souvent remplacé par une matrice identité, et n'est plus écrite dans les équations. Une matrice de poids sert notamment à prendre en compte des différences de précision de mesures ; le modèle va donc s'ajuster plus précisément sur les mesures avec un poids élevé.

où J est la matrice jacobéenne $[\frac{\delta f}{\delta x}]$ et α la taille du pas dans la direction opposée au gradient.

La descente de gradient est conceptuellement très simple ; les paramètres sont modifiés dans la direction opposée au gradient de la fonction objective. La descente de gradient est reconnue comme étant un algorithme hautement convergent pour des fonctions objectives simples. Le désavantage est le temps de convergence très lent de cette approche, qui a besoin d'un très grand nombre d'itérations. Cette approche est notamment utilisée lorsque le nombre de paramètres à optimisé devient très grand, ce qui n'est pas le cas ici.

La méthode de base de résolution de problème par moindres carrés est appelée méthode de Gauss-Newton. Le nom de cette méthode est très variable et souvent remplacée par « résolution par moindres carrés ». Cette méthode fait l'hypothèse que la fonction objective est approximativement quadratique proche de la solution optimale. Le pas de mise à jour de Gauss-Newton est égal à :

$$\delta x = (J^T W J)^{-1} J^T W (y - f(l, x)) \tag{3.2.5}$$

Cette solution vient de la relation $\frac{\delta Objective}{\delta \delta x} = 0$; on estime directement les paramètres qui minimise la fonction objective.

Levenberg-Marquardt crée un équilibre entre la descente de gradient et Gauss-newton. Si la fonction objective diverge, plus de poids est donné à la descente de gradient : la convergence est plus lente mais assurée. Si une convergence est observée, plus de poids est donné à Gauss-Newton, ce qui accélère la convergence. Le pas de mise à jour de Levenberg-Marquardt est donné par :

$$\delta x = (J^T W J + \lambda \, diag(J^T W J))^{-1} \, J^T W \, (y - f(l, x)) \tag{3.2.6}$$

Où λ est un facteur de pondération entre les deux méthodes. Levenberg-Marquardt ajoute également certaines opérations pour le calcul de α. L'algorithme est ainsi composé :

$\lambda \leftarrow \lambda_0$
$x \leftarrow x_0$
$S_{old} \leftarrow Objective(x)$
while dx $>$ threshold AND iteration $<$ maxIteration **do**
    $\delta x \leftarrow g(\lambda, y - f(x))$
    $x_{new} \leftarrow x + \delta x$

$S_{new} \leftarrow Objective(x_{new})$
if $S_{new} < S_{old}$ **then**
$\lambda \leftarrow \lambda/10$
$x \leftarrow x_{new}$
$S_{old} \leftarrow S_{new}$
else
$\lambda \leftarrow \lambda \cdot 10$
end if
end while

Dans notre cas, la fonction étant décrite comme hautement non-linéaire, une pose initiale éloignée de la solution optimale peut amener à un résultat insensé. Si des points de contrôles sont également faux, ou très mal mesurés, un résultat insensé peut également survenir. Différentes variantes de LM existent, surtout concernant la mise à jour du poids λ et la matrice $diag(J^T W J)$ utilisée dans la mise à jour.

3.2.3 Principes du "Perspective-n-Points" (PnP)

Une approche différente de la DLT permet d'estimer une position initiale. Cette deuxième approche a été remis au goût du jour par le domaine de la vision (ou computer vision). Ceci a eu pour conséquence d'apporter une nouvelle façon de voir dans divers domaines, notamment en photogrammétrie. Ainsi, ce ne sont plus les équations de colinéarités qui sont utilisées, mais les équations de coplanarité. Ce processus est également appelé résection spatiale.

Contrairement à DLT, la connaissance de la focale est requise pour PnP, ce qui en fait son principal inconvénient pour le plugin. La focale ne peut pas être un paramètre de l'algorithme classique. La variante PnP qui permet de déterminer la pose avec seulement trois points s'appelle le P3P. Dans ce cas, une ambiguïté de quatre solutions existe. Il est possible de réduire cette ambiguïté sans utiliser plus de points. Cependant, on utilise en général quatre points comme minimum pour calculer la pose, sans ambiguïté cette fois-ci. L'algorithme portera donc le nom de P4P dans ce cas.

En plus de permettre le calcul de la pose avec moins de points, PnP garantit une meilleure précision selon [37]. PnP n'a pas l'instabilité observée de la DLT dans une configuration planimétrique des GCPs. Cependant, PnP possède un cylindre dangereux, sur lequel peut reposer le centre de projection. Dans ce cas, la solution est instable.

Chapitre 4

Réalisation

Pic2Map a été rendu public le 13 juin 2014, dans la revue de presse de geotribu.net. Dans ce chapitre, nous verrons comment les concepts du chapitre 1, développés au chapitre 2, ont été établis sous forme de code informatique.

Un survol de l'implémentation est présenté par la figure 4.1. On peut y découvrir le diagramme de processus à l'intérieur de plugin. Deux sources de données de bases sont nécessaires : Une photographie et un DEM. Il est possible d'utiliser une ortho-photo pour faciliter l'estimation de la pose ou plus exactement, faciliter la digitalisation des GCPs.

Après avoir chargé les données, l'estimation de la pose peut être réalisée soit avec des GCPs, soit dans une vue 3D. Dans les deux cas, il est possible de charger ou d'enregistrer la pose en format KML [1]. Les deux approches font intervenir la vue 3D : l'approche par GCPs utilise la vue 3D pour vérifier la pose et pour faciliter la digitalisation de GCPs.

Afin de déterminer la pose, il n'y a pas forcément besoin de passer par les GCPs. Une approche complètement manuelle a été développée sur la base de la vue 3D, appelée "virtual 3D". L'approche virtual 3D utilise la vue 3D pour positionner la photographie par rapport au DEM et estimer les paramètres de pose visuellement, sans GCP ni algorithme. La photographie est fixée à l'arrière plan et l'on peut jouer sur la transparence du DEM afin de trouver une correspondance.

Dans le cas de l'approche par GCP, il est possible de charger ou d'enregistrer les points de contrôle. Ceux-ci peuvent être édités sur l'image, dans le canevas QGIS ou dans la vue 3D. Une fois les GCPs digitalisés, l'estimation de la pose peut avoir lieu. Une indication de la précision de la pose est donnée par la projection des GCPs XYZ dans l'image, et la reprojection des GCPs UV sur le DEM.

1. Le format KML permet de stocker de l'information spatiale pour divers types d'objets. Pour l'intégration de photos dans Google Earth, KML possède un standard d'écriture de pose de caméra, ce qui n'est pas le cas pour tous les standards.

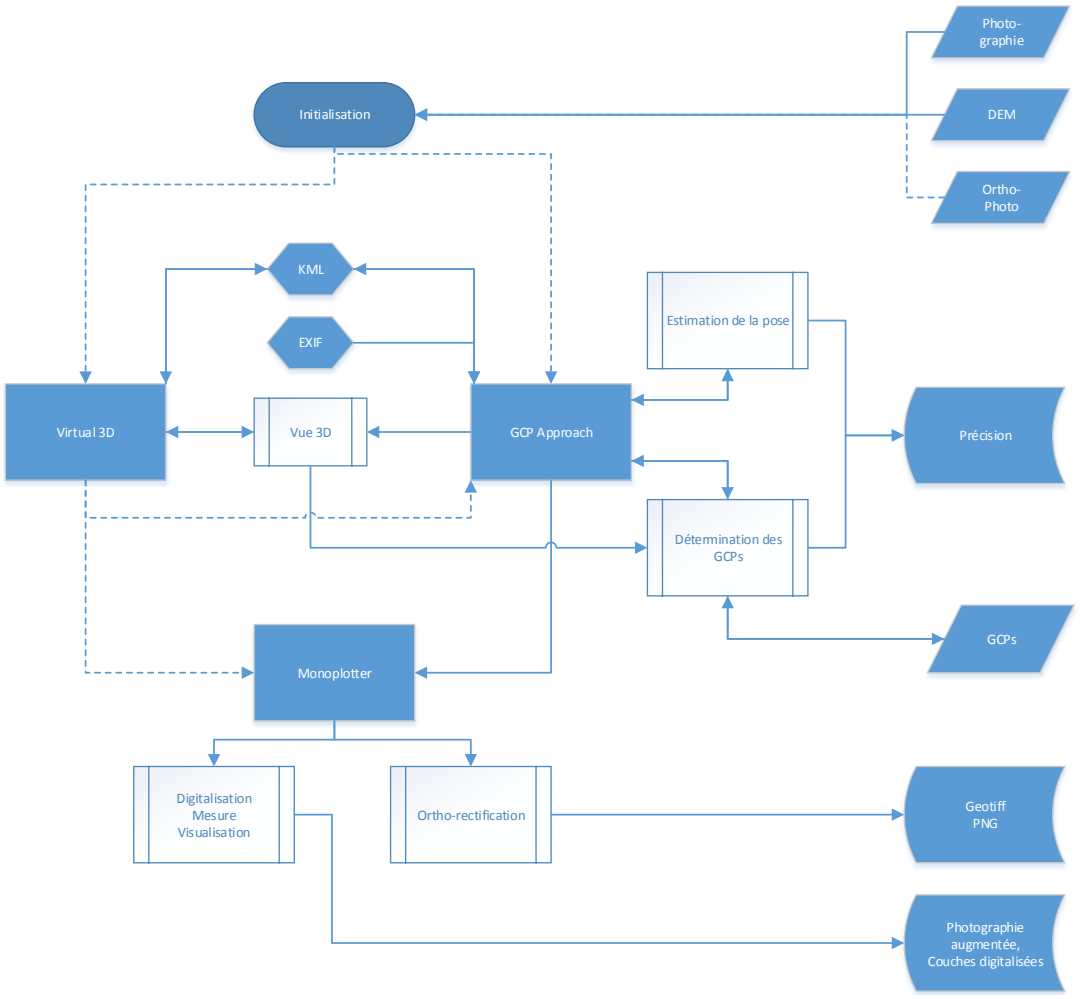


FIGURE 4.1: Diagramme de processus du plugin. On y distingue trois paliers : L'initialisation, l'estimation de la pose et le monoplotting. Les flèches représentent des interactions et échanges de données. Les traitillés représentent des choix possibles.

Il est également possible de déterminer la pose manuellement par l'approche virtual 3D, puis d'utiliser l'approche par GCP en utilisant la pose initiale pour faciliter la digitalisation des GCPs.

A l'ouverture du monoplotter, il n'est plus possible de changer de pose. Différentes fonctionnalités deviennent disponibles : La mesure sur l'image, la visualisation de couches vectorielles et la digitalisation de nouvelles entités. Un outil d'ortho-rectification peut être ouvert. On peut y enregistrer la photographie ortho-rectifiée, en raster au format png, qui sera une simple image, ou au format GeoTiff, qui sera géo-référencé.

4.1 Outils

Différents outils sont utilisés pour le plugin, mais chacun a son utilité et ses particularités. Toutes les modules utilisés sont inclus dans l'application QGIS et accessibles depuis du code Python. La figure 4.2 représente schématiquement cette organisation. QGIS est codé en C++ et utilise abondamment le framework QT. Le module PyQt est utilisé afin de créer l'interface et des signaux entre les différents objets du plugin. Le module GDAL est utilisé pour gérer les couches de données et l'information spatiale. La 3D est gérée par les modules QtOpenGL et PyOpenGL.

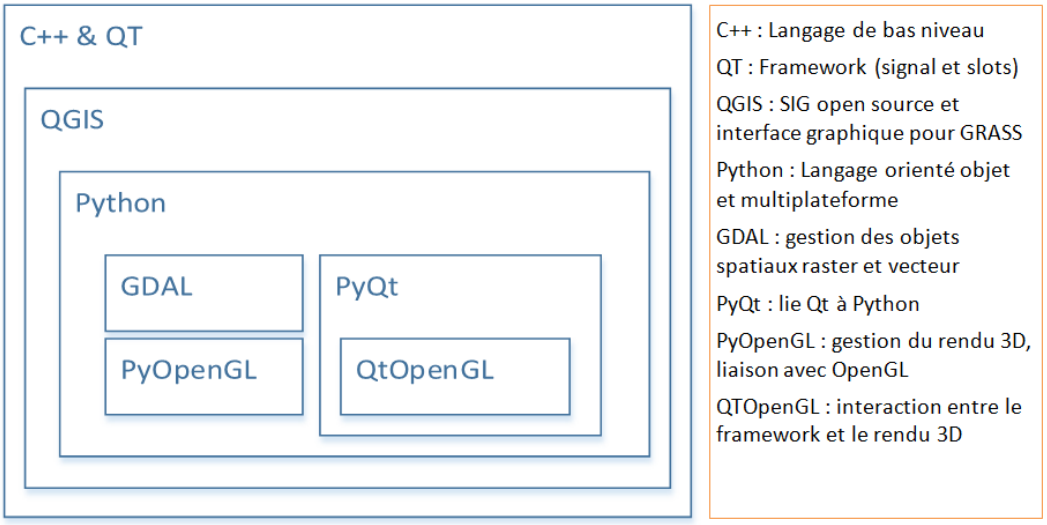


FIGURE 4.2: Représentation schématique de l'organisation des modules

4.1.1 python

En mai 2014, Python se classa au 8ème rang mondial [2] des langages de programmation. Il peut être décrit comme le langage populaire pour la programmation objet. On lui reconnaît notamment la qualité d'offrir un outil optimisant la productivité des programmeurs. Un bon équilibre est réalisé entre la perte de temps de calculs et le confort de programmation de haut niveau. Cela est d'autant plus vrai que les machines deviennent performantes. Une très bonne introduction à Python est à découvrir : [38].

En outre, Python est un langage multiplateforme et possède de nombreuses extensions. Ces arguments font de Python un langage de choix pour le développement logiciels. De plus, Python possède de nombreuses extensions scientifiques, ce qui en fait un candidat sur mesure pour le développement logiciel dans le milieu académique.

2. TIOBE Index pour mai 2014

Malgré tous ces avantages, ce sont deux autres raisons qui ont conduit à utiliser Python comme langage du plugin. La première est l'utilisation de Python dans une première version d'un monoplotter, réalisée par Timothée Produit. Malheureusement, une grande partie de ce code n'a pas été réutilisée, notamment la bibliothèque de pose OpenCV. La deuxième raison est simplement que les plugins pour QGIS doivent être écrits en Python. Il est techniquement possible de les écrire en C, mais leurs partages et leurs installations sont infiniment plus compliquées.

Le code python existant n'a pas été réutilisé, car la bibliothèque OpenCV ne permet pas d'estimer une pose avec une focale inconnue. Certaines astuces permettent tout de même de l'utiliser. Par exemple, on peut itérer la pose avec une focale variable. On choisira finalement la meilleure itération. La deuxième difficulté par rapport à OpenCV était l'intégration dans QGIS. Cet argument a été assez convaincant pour opter pour une algorithme de pose plus artisanal, qui permet de prendre la focale comme un paramètre de pose. Une autre raison pour n'avoir pas utilisé le code originel fut la stratégie de calcul du z-buffer. L'intégration de OpenGL au plugin a permis de s'implifier l'écriture du code et d'utiliser des calculs sur le GPU.

4.1.2 Qt

Parmi les types de bibliothèques informatiques de programmation, il existe une variante appelée « framework ». Un framework se distingue des librairies standards par ses fonctions très générales. Un tel outil sert à structurer la programmation. Cette structure est imposée par le simple fait d'utiliser ce framework.

Qt est un framework répandu, utilisé dans de célèbres logiciels. Qt permet en outre de travailler avec des signaux et des slots à l'intérieur du programme. Ce dernier n'est donc plus un script déroulant, un algorithme linéaire, une recette de cuisine, mais devient un objet vivant, qui réagit aux actions de l'utilisateur ; tel est le principe de la programmation par signal. Plus de détails peuvent être obtenus dans [39].

Qt est notamment utilisé pour créer des interfaces utilisateurs, puisque des signaux peuvent être envoyés suivant les actions de l'utilisateur sur l'interface.

Qt est également une API d'interface graphique offrant divers composants tant pour des interfaces que pour d'autres tâches telles que le chargement de photographie dans le script, la gestion de données, l'intégration d'OpenGL, etc. De plus, Qt est multiplateforme et facilite donc largement la tâche de programmation.

Plus précisément, c'est le module PyQt qui a été utilisé pour le plugin. En effet, le plugin, codé en Python, requiert une bilbiothèque pythonnisée. Il n'est donc pas possible

d'importer directement Qt. Cependant, PyQt ne reprend pas les modules Qt pour les coder en Python. PyQt fait le lien entre le langage Python et les modules Qt. On parle de « liaison de langage ».

4.1.3 OpenGL

Pour comprendre la force d'OpenGL, il faut comprendre la finance. En bref, une API propriétaire crée du revenu, mais peine à se répandre. Une API ouverte basée sur des standards uniformisés crée des opportunités de marchés bien plus grands pour les partenaires. Ce ne sont donc pas les membres de Khronos Group, contrôlant et développant OpenGL, qui s'enrichissent, mais les mécènes de Khronos Group (Apple, Nokia, Nvidia, Intel, etc.).

Au niveau technique, OpenGL apporte deux avantages majeurs. Le premier est le codage déjà réalisé de nombreuses fonctions de 3D ; par exemple, récupérer une coordonnée en cliquant sur un DEM. Le deuxième est l'optimisation de tous les processus 3D. Par exemple, la précision de la coordonnée après un clic sur le DEM dépend de la distance à la caméra. Ce comportement est intuitif : Plus de précision est donnée aux objets proches, car on les voit mieux, avec plus de détails. Une aperçu du fonctionnement d'OpenGL est à explorer dans [40].

4.1.3.1 Z buffer

OpenGL n'utilise pas un algorithme de « ray-tracing » pour déterminer les coordonnées 3D d'un pixel d'une image. Un objet, qui est construit à chaque changement de vue, stocke la distance de l'objet à la caméra. En d'autres termes, la coordonnée Z du système caméra est enregistrée dans un objet annexe à l'image. Cet objet est appelé « depth buffer » ou « Z-buffer », en référence à la dimension de profondeur dans l'image, souvent exprimée sur l'axe Z. La figure 4.3 représente le Z-buffer ; plus l'éloignement est important, plus le terrain est clair. En connaissant la distance à la caméra et la position de l'objet dans l'image, il est possible d'appliquer une projection en retour sur le terrain.

L'avantage d'utiliser un Z buffer par rapport au ray-tracing est la capacité de calcul sur le GPU. Le GPU permet d'effectuer des calculs simples massivement parallèle. C'est exactement pour cette caractéristique du GPU que certaines bibliothèques, comme OpenGL, ont été créées.

Il peut être utile de préciser certaines règles de création du Z- buffer afin de mieux comprendre certains comportements qui peuvent parfois paraître illogiques.

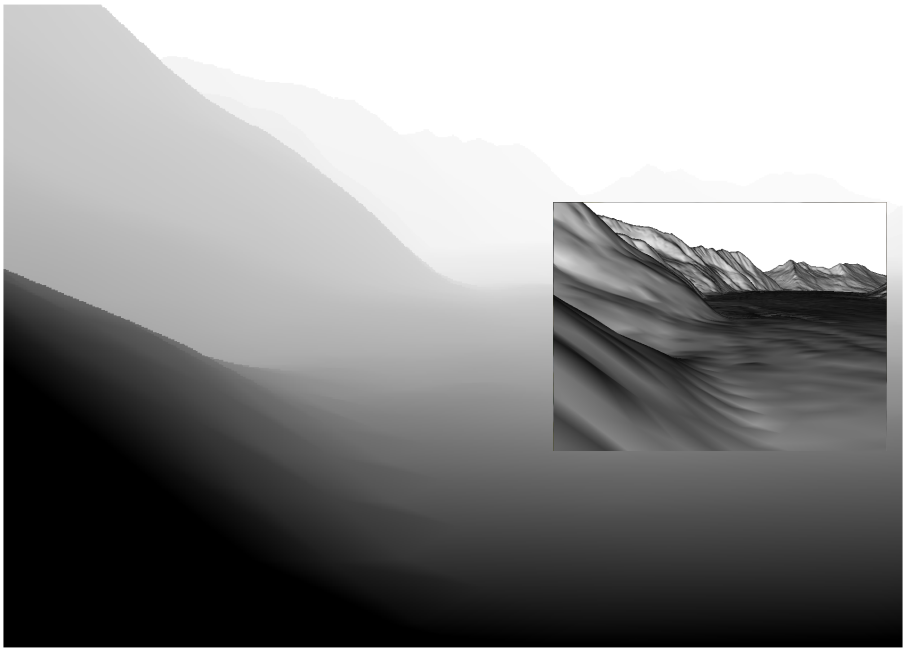


FIGURE 4.3: Représentation du Z-buffer : plus un objet est éloigné, plus il est clair. Un zoom dans le DEM ombré permet de mieux visualier le terrain.

Le Z-buffer contient des valeurs entre 0 et 1. La position de la caméra à une valeur z de 0, alors que le point visible dans la vue 3D à une valeur z de 1. On pourrait s'attendre à ce qu'un point à mi-chemin entre la caméra et le point le plus éloigné aie une valeur z de 0.5. Mais ceci ne ne produit pas. La valeur à mi-chemin serait plutôt de 0.99. On peut voir sur la figure 4.3 que les points très éloignés change peu de couleurs d'une montagne à l'autre ; alors que très proche, les changements de couleurs se font sur quelques mètres.

Ce comportement est expliqué par le fait que plus un objet est proche, plus il sera discernable. Donc, il sera plus facile d'être précis lors d'une interaction avec lui. Par contre, lorsqu'un objet est éloigné, l'erreur peut devenir grande. Il n'y a donc pas de raison d'être précis avec la valeur Z si les valeurs u et v ne sont de toutes façons pas précises. Par exemple, sur la figure 4.3, une différence de 1 pixel, sur les montagnes les plus éloignées, résultent en différence de 20 mètres sur le plan vertical.

La relation non-linéaire entre le z-buffer et la distance est finalement logique. Ce comportement a pour conséquence de corréler fortement la précision du z-buffer avec deux valeurs : le clip proche et le clip lointain. Ces deux valeurs définissent l'intervalle d'existence des objets dans OpenGL. Un objet plus proche que le clip proche sera invisible, et un objet plus loin que le clip lointain sera également invisible. Il faut donc trouver un équilibre entre ces deux valeurs et la précision du Z-buffer. C'est pourquoi, dans certains cas, le plugin peut se comporter étrangement si l'on travaille sur un objet très proche ou sur un autre très éloigné.

Bien que le ray-tracing soit plus précis que l'utilisation d'un z-buffer, il paraît difficilement implémentable pour une application qui doit effectuer des calculs rapidement. Le chapitre 8 du livre [41] appronfondit les différences entre les deux approches.

4.1.4 GDAL

"Geospatial Data Abstraction Library" (GDAL) est une bibliothèque essentielle pour le monde des SIG open source. A la base, c'est un outil de standardisation pour la gestion d'information spatiale ; comme son nom l'indique GDAL établit un modèle de donnée plus qu'un format [3]. Aujourd'hui, GDAL possède des outils d'interopérabilité, capable de traduire de nombreux formats, et de transformation de données. La plupart des logiciels disponibles, notamment QGIS, repose sur GDAL pour gérer les différentes formes numériques des couches raster et vectorielles. GDAL ne se limite pas aux logiciels open source mais il est également utilisé par nombre de logiciels propriétaires.

Dans le plugin Pic2Map, GDAL est utilisé pour des opérations de traitements de couches. Il est entre autre utilisé pour afficher les couches de polygones dans le monoplotter, ce qui requiert de nombreuses opérations intermédiaires, ainsi que pour enregistrer l'ortho-image en format GeoTiff.

4.1.5 QGIS

Dans les SIG open sources, le logiciel GRASS est en tête des fonctionnalités et de la distribution, comme décrit dans [42]. Le logiciel QGIS est une interface entre autre développée pour accéder aisément à GRASS. QGIS paraît donc approprié pour un développement open source sur un SIG. QGIS est de plus en plus utilisé par des entreprises. Certaines entreprises de développement informatique développent même des solutions spécifiques sur QGIS [4].

QGIS propose une commande Python intégrée à l'interface pour effectuer des calculs et lancer des commandes. Des plugins peuvent être codés en Python et lancés depuis QGIS. Les plugins python ne requirent pas de compilation et sont multiplateformes. La distribution des plugins se fait grâce à des dépôts (fichiers .xml) qui permettent de récupérer des méta-données sur les plugins et de les installer en quelques clics.

3. En informatique, un objet abstrait est une référence pour divers développements où une même structure interne est partagée. Imaginons que GDAL défini ce qu'est un "arbre" ; chacun peut ensuite créer un pommier, un cerisier, etc. L'important est que l'on va manger les fruits ou couper le tronc avec les mêmes outils.

4. Camptocamp SA, 1015 Lausanne

Notons finalement qu'aucun SIG répandu (Erdas, Arcgis, Mapinfo, etc.) ne paraît offrir la capacité de monoplotting pour des applications cartographiques.

4.2 Documentation

Une documentation du plugin a été écrite pour les utilisateurs et donne une vue d'ensemble de la réalisation. Pour des raisons de clarté et de cohérence, cette documentation n'a pas été inclue ici même. Elle donne également une vue d'ensemble du travail accompli et de l'effort fourni. Elle peut être consultée à l'adresse http ://lasig.epfl.ch/ dans l'onglet "Software".

Pour les lecteurs intéressés à lire l'implémentation, le code est open source ; la lecture est possible une fois le plugin téléchargé. De plus, l'annexe C expose quelques exemples de développement en quelques tranches de code Python.

4.3 Positionnement dans OpenGL

L'estimation de la pose consiste à déterminer en neuf paramètres selon le modèle sténopé :

— trois définissant la position : X0, Y0 et Z0 ;
— trois définissant l'orientation : Tilt, Azimut et Swing ;
— un définissant la focal ;
— deux définissant le point central.

Il n'est cependant pas possible de les utiliser tels quels pour le positionnement dans OpenGL. Ce dernier utilise deux fonctions principales pour le positionnement de la vue. La première est la fonction gluLookAt. On y définit la position du point de vue, la direction de la vue et la verticalité de la vue. La deuxième fonction est gluPerspective. On y définit l'angle de vue et d'autres paramètres d'affichage indépendants de la pose.

Pour définir la pose dans OpenGL, il y a donc besoin de :

— la position de la caméra : X1, Y1 et Z1 ;
— un point situé au centre de l'image dans la direction de vue ;
— un angle définissant le roulis de l'image ;
— l'angle de vue verticale.

La position de la caméra est égale à la position estimée ; $(X1, Y1, Z1) = (X0, Y0, Z0)$. Afin de calculer un point situé dans la direction de vue, il suffit d'utiliser la rotation inverse du système DEM vers le système caméra. Un vecteur situé dans la direction de vue dans le référentiel caméra est transformé dans le système objet. Le détail des calculs peut être consulté à l'annexe A.9.

Il est également possible d'intégrer la notion de point central dans OpenGL. Aucune fonction n'est cependant disponible pour intégrer cette notion aisément. Afin d'intégrer le point central, il est nécessaire de créer à la main la matrice de projection perspective. Définir une telle matrice n'est bien sûr pas un problème, car sa construction est parfaitement maîtrisée, comme nous l'avons vu plus haut. La seule difficulté est l'adaptation des expressions au fonctionnement d'OpenGL. Par exemple, la point central, habituellement exprimé en millimètres ou en pixels, doit être entré dans OpenGL selon une valeur proportionnelle la largeur de l'image.

Les détails de l'implémentation du point central dans OpenGL peuvent être trouvé dans l'annexe A.10.

4.4 Ortho-rectification

Le processus d'ortho-rectification de la photographie reprend un processus standard pour des photographies aériennes tel que décrit dans [43] ; L'ortho-rectification est similaire entre des photographies terrestres ou aériennes pour la simple raison qu'il est très logique. L'ortho-rectification développée dans le plugin varie légèrement du processus habituel, notamment par l'utilisation d'OpenGL et de l'implémentation de la texturisation normalement utilisée pour des jeux vidéo. Les différentes étapes du processus habituel sont énumérées ci-dessous :

1. Détermination de l'empreinte limite de l'ortho-image ;
2. Définition de la résolution de l'ortho-image et de l'échelle de l'ortho-image. Ces deux paramètres définissent la largeur de grille dans le système objet δx ainsi que la largeur de la grille dans le système image ($\delta\widetilde{x} = \textit{échelle} \cdot \delta x$) ;
3. Pour chaque point de la grille, la valeur de Z est extraite du DEM ;
4. En utilisant la projection perspective, une coordonnée x', y' est calculée dans le système image pour un point de la grille X, Y, Z ;
5. La couleur du pixel à la coordonnée x', y' est extraite de l'image ;
6. La couleur extraite est attribuée à la position x, y de l'ortho-image.

Les différentes étapes de l'ortho-rectification dans le plugin sont énumérées ci-dessous :

1. En utilisant la projection perspective, une coordonnée x', y' est calculée dans le système image pour chaque vertex du DEM. Le point x', y' est reprojeté sur le DEM. Si la distance est plus grande qu'un seuil, le point est jugé invisible (par exemple, derrière une montagne) ;
2. Si le point est visible, la coordonnée projetée x', y' est attribuée au vertex ;
3. Une fois que tous les vertex visibles ont une coordonnée x', y', la photographie est interpolée sur le TIN créé par le DEMs ;
4. Définition de l'empreinte de l'ortho-photo (bounding box) et de la résolution ;
5. Le DEM colorié selon l'interpolation précédente est enregistré selon l'empreinte et la résolution définies ;

Certaines différences entre ces deux approches doivent être prises en considération. Tout d'abord, dans l'approche du plugin, chaque vertex du DEM est utilisé sans traitement ou interpolation. L'information spatiale est donc complètement utilisée.

Au niveau de l'extraction des couleurs, chaque pixel de l'image est projeté sur le DEM. Les pixels de l'image sont interpolés entre trois vertex du DEM comme sur la figure

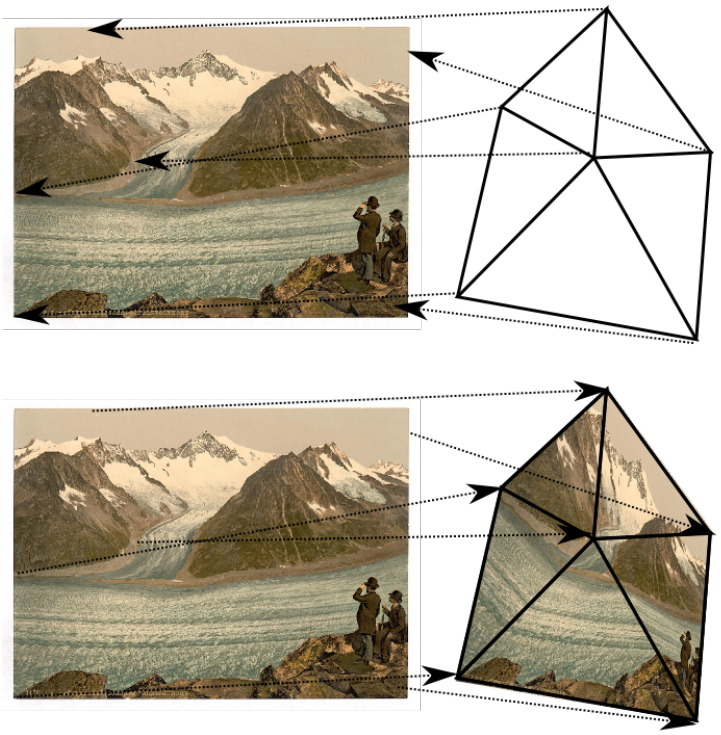


FIGURE 4.4: Représentation de la texturisation dans OpenGL

4.4. Une telle attribution des couleurs implique que chaque pixel de l'image est utilisé au mieux. L'attribution des couleurs est dépendante de la géométrie et de la pose de la caméra. Par exemple, beaucoup de pixels sont présents dans les triangles proches de la caméra, tandis que peu sont présents sur le DEM loin de la caméra.

Cette technique d'attribution de couleur est également optimale pour l'ortho-rectification. On peut se demander d'où vient cette puissance de calcul pour l'ortho-rectification. OpenGL a été développé pour les jeux vidéo. C'est un outil 3D extrêmement puissant. Tous les calculs sont optimisés et pensés pour la 3D uniquement. Ici, nous travaillons avec une caméra fixe, un terrain fixe et une image fixe. Il est donc intuitif de penser qu'OpenGL est surdimensionné pour traiter un tel cas. La puissance d'OpenGL nous permet donc d'être très efficaces dans ce procédé.

4.5 Précision

La précision de la pose est d'abord explorée par une simulation numérique basée sur l'algorithme implémenté. Ensuite, quelques résultats concrets seront présentés, réalisés avec le plugin et représentatif d'une utilisation typique. Les données des sections 4.5.1, 4.5.2 et 4.5.3 proviennent des mêmes cas d'études. Six positionnements différents sont présentés. Les photographies ont été prises dans la région des Diablerets, en Suisse.

Le DEM exploité est le MNT25/200 de Swisstopo. Ce DEM a une taille de maille de deux cents mètres. Les erreurs de positionnement sont donc à prendre en considération relativement à la maille du DEM. Dans un deuxième temps, un DEM à 25m de résolution spatiale est utilisée.

La qualité de pose et de projection dépendent de nombreuses variables telles que :

— la précision des GCPs (effort investi par l'utilisateur),
— le nombre de GCPs, c'est-à-dire la redondance,
— les distorsions de la photographie,
— la connaissance de certains paramètres (p.ex. positionnement par satellite),
— la résolution du DEM,
— la qualité du DEM,
— la distribution des GCPs dans l'image et dans le terrain,
— certains paramètres de l'algorithme (p.ex. nombre d'itérations),
— les erreurs de modèle (p.ex. négligé la corrélation entre la position et la focale),

4.5.1 Précision de la pose, simulation numérique

Le tableau 4.1 présente les erreurs de pose selon une simulation numérique. Le bruit utilisé dans les simulations est un bruit blanc. Les résultats présentés sont moyennés sur mille itérations. Trois paramètres sont considérés : l'erreur de positionnement 3D, l'erreur de direction (RMSE des angles), et l'erreur de focale.

Paramètre	Algorithme	Cube	Asymétrie 0.25	Asymétrie 0.1
Position	DLT	0.09	0.16	0.35
	LM	0.09	0.11	0.17
	LM+Pos	0	0	0
Orientation	DLT	0.038	0.034	0.031
	LM	0.036	0.035	0.033
	LM+Pos	0.022	0.019	0.018
Focale	DLT	0.026	0.060	0.130
	LM	0.026	0.060	0.133
	LM+Pos	0.015	0.012	0.011

TABLE 4.1: Simulation de l'estimation de la pose à 10 % de bruit. Dans la variante LM, le point central est fixée à (0,0), la valeur vraie. Dans la variante LM+Pos, la position est fixée à la valeur vraie, en plus du point central. La précision est presque toujours ordonnée selon DLT, LM puis LM+Pos. La variante LM contient une initialisation avec DLT. La précision de la pose diminue quand l'asymétrie augmente. La précision de l'orientation augmente quand la précision augmente. La précision de la focale diminue quand la précision augmente.

L'erreur de positionnement est donnée par rapport à une distance unitaire entre les GCPS. Par exemple, si la distance entre les GCPs est typiquement de 1 km, on peut

s'attendre à une erreur de pose de 90m avec un bruit de 10% dans les mesures. L'erreur de direction est donnée en radians. L'erreur de la focale est donnée relativement à une focale unitaire. Par exemple, si l'erreur focale et de 0.026 et la focale de 600 pixels, l'erreur serait de 15.6 pixel.

Huit GCPs ont été créés. Ils ont été placés selon trois configurations différentes. Soit en cube, configuration théoriquement optimale pour la DLT, en parallélépipède rectangle, avec une profondeur égale à 1/4 de l'arête du cube (asymétrie 0.25) ou avec une profondeur égale à 1/10 de l'arête du cube (asymétrie 0.1).

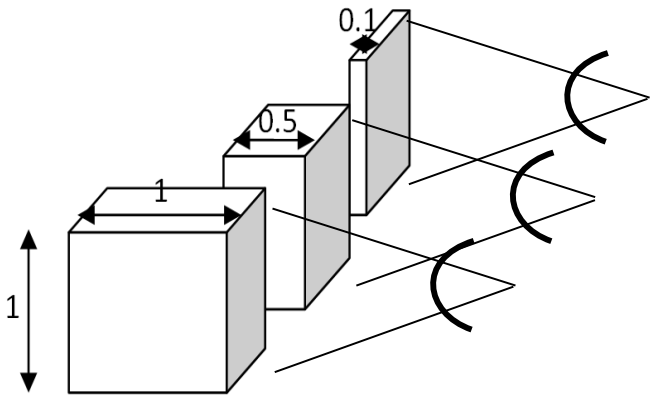


FIGURE 4.5: La simulation numérique a été répétée pour trois configuration différentes de GCPs. La valeur d'asymétrie représente le ratio entre la profondeur et le côté du prisme. Ici, les valeurs 1, 0.5 et 0.1 sont représentées. Les valeurs 1, 0.25 et 0.1 ont été utilisées.

On observe dans le tableau une augmentation des erreurs de positionnement quand la configuration des GCPs devient planimétrique, ce qui est attendu. En revanche, l'orientation est légèrement améliorée pour les variantes plus asymétriques. On peut expliquer ceci par l'asymétrie par la définition de l'asymétrie utilisée ici. Une asymétrie plus importante affecte seulement la pose des GCPs dans la profondeur. Ceci modifie la position et la focale, mais pas forcément l'estimation de l'orientation. Celle-ci augmente car les points se raprochent de l'image avec une asymétrie plus élevée. Des points plus proches aideront à obtenir une orientation plus précise, comme le bruit est ajouté sur l'image et non sur la coordonnée 3D.

Au niveau de la focale, l'asymétrie dégrade fortement la qualité de l'estimation. Cependant, lorsque la position est connue et fixée, la focale a tendance à être plus précise avec une configuration planimétrique. Ceci ne contredit pas le fait que la pose est moins précise quand les GCPs sont sur un plan. En effet, les problèmes de planimétries sont asymptotiquement liés à la configuration.

De plus, le tableau B.1 présente les erreurs des paramètres pour un niveau de bruit de 20 %. Les simulations montrent en outre un comportement quasi-linéaire des erreurs en fonction du niveau de bruit. On peut également observer une réponse très faible de la pose par Levenberg-Marquadt pour une asymétrie jusqu'à 0.3. Des résultats complémentaires sont présentés à l'annexe B.1.

Le tableau 4.2 présente une autre variante de configuration précédente. Les points sont cette fois-ci répartis sur les sommets d'un prisme plutôt que d'un cube. Par prisme, on entend une répartition telle que celle présentée à la figure 4.6. Les erreurs observées avec cette variante sont plus grandes que dans une configuration en cube. En général, elles sont environ le double des erreurs précédentes. Les comportements observés sont également les mêmes. Un point intéressant se trouve être l'estimation de la pose en connaissant la position de la caméra. Dans ce cas uniquement, les erreurs entre la configuration en cube ou en prisme sont les mêmes.

Paramètre	Algorithme	Asymétrie 1.0	Asymétrie 0.25	Asymétrie 0.1
Position	DLT	0.15	0.23	0.61
	LM	0.13	0.15	0.20
	LM+Pos	0	0	0
Orientation	DLT	0.056	0.066	0.119
	LM	0.046	0.051	0.058
	LM+Pos	0.021	0.020	0.020
Focale	DLT	0.043	0.071	0.170
	LM	0.039	0.051	0.110
	LM+Pos	0.014	0.013	0.013

TABLE 4.2: Erreur de positionnement selon une simulation numérique où les GCPs sont distribués sur les sommets d'un prisme. L'asymétrie, dans ce cas, fait référence au plan incliné.

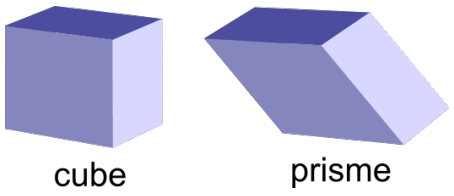


FIGURE 4.6: Les GCPs ont d'abord été répartis sur les sommets d'un cube. Une deuxième variante a consisté à les répartir sur les sommets d'un prisme ; ceci pour mieux simuler une répartition dans un paysage montagneux. Les erreurs sont plus grandes que dans le cas d'un cube.

4.5.2 Précision de la pose, cas réels

Le tableau 4.3 présente les erreurs de positionnement planimétrique pour six positionnements différents (cf. introduction de la section 4.5). Les positions détaillées sont disponibles à l'annexe B.2. Les erreurs sont à considérer dans le cadre d'un DEM à 200 mètres

de résolution spatiale. En général, on remarque une erreur de l'ordre de grandeur de la taille de maille, avec des erreurs pouvant augmenter jusqu'à quatre fois dans notre jeu de données. Cependant, l'erreur de pose n'est pas seulement dépendante de la résolution spatiale du DEM ; il serait incorrect d'inférer sur une relation linéairement proportionnelle entre la résolution spatiale et la précision de la pose. Cet exemple-ci a surtout le mérite de montrer les limites du plugin et les possibilités d'utilisation lorsque des données de moindres qualités sont exploitées.

Identifiant	# GCPs	Erreur [m]
1	9	179,6
2	8	384,5
3	8	315,8
4	14	219,8
5	9	142,6
6	12	778,0
moyenne	-	336,7

TABLE 4.3: Erreurs de positionnement planimétrique avec un calcul de pose totalement libre sur le DEM à 200m. Les erreurs sont variables, mais restent du même ordre de grandeur. Voir tableau 4.6 pour les mêmes observations avec un DEM à 25m.

4.5.3 Précision des projections

Afin de quantifier l'erreur de projection entre la photographie et la coordonnée 3D dans le système objet, les projections et reprojections des GCPs ont été considérées. Les six cas présentés dans le tableau 4.4 sont les mêmes que ceux du chapitres précédant.

ID	Projection sur la photo [pixel]				Reprojection dans le terrain [m]			
	Moyenne	Min	Max	RMSE	Moyenne	Min	Max	RMSE
1	6,51	3,10	13,48	3,42	19,86	3,83	76,06	21,29
2	5,28	0,60	13,21	4,47	52,57	3,24	179,29	60,02
3	8,59	3,09	17,72	5,06	63,60	9,03	153,62	51,26
4	20,07	0,76	48,27	12,95	44,31	1,44	237,39	65,76
5	10,85	3,26	19,26	4,31	50,98	8,88	117,60	34,91
6	24,57	2,58	60,53	20,36	63,00	13,48	184,68	48,62
Moyenne	12,64	2,23	28,74	8,42	49,05	6,65	158,11	46,98

TABLE 4.4: Erreurs de projection et reprojection avec le DEM à 200m. La diagonale de l'image est de 3'200 pixels. A gauche du tableau se trouvent les erreurs de projections sur la photographie ; à droite se trouvent les distances en mètres entre les GCPs du système objet et les reprojections des GCPs de l'image. L'information la plus pertinente est la moyenne de l'erreur de projection égale à 50 mètres.

Deux erreurs différentes sont étudiées. Dans la partie gauche du tableau, on trouve les erreurs de projection des GCPs du système objet sur l'image. L'unité est le pixel. Dans la partie de droite, on trouve les erreurs de reprojections des GCPs de l'image sur le terrain 3D. L'unité est le mètre.

La pose a été estimée sans fixer de position connue. Entre 8 et 14 GCPs ont été utilisés pour chaque cas. Les reprojections en-dessus de l'horizon n'ont pas été pris en compte dans le calcul des erreurs ; il n'y a jamais plus de deux par cas d'étude.

La précision de la projection de l'image est dépendante de la résolution de l'image. Les photographies utilisées ici ont toutes une résolution 2'560 x 1'920 pixels, c'est-à-dire une diagonale de 3'200 pixels. Une erreur de 12.64 pixels est donc égale à 0.4% de la diagonale de l'image. On peut donc considérer cette erreur comme faible. Une appréciation visuelle de cette erreur peut se faire sur l'image 4.7. Cette image correspond au cas 5. La moyenne des projections est représentative de la moyenne des autres cas d'études. La figure 4.8 montre les reprojections correspondantes sur le DEM. Les GCPs image et leurs reprojections sont en orange alors que les GCPs objet et leur reprojection sont en bleu.

La précision des reprojections se situe dans les environs de 50 mètres. Ce qui correspond ici à 1/4 de la résolution spatiale du DEM. Ce résultat est encourageant. De plus, on peut remarquer que la précision des reprojections est meilleure que la précision de la pose avec une différence d'un ordre de grandeur.

En ce qui concerne plus précisément le cas de la figure 4.7, deux points ont été ajoutés au premier plan. Ces GCPS ont été ajoutés dans le but de déterminer une meilleure estimation de la pose. Une erreur de seulement 6 mètres a été obtenue grâce à ces deux points. Ils ont été ajoutés autours de la maison visible sur la gauche de l'image.

FIGURE 4.7: Projection des GCPs du système objet dans l'image

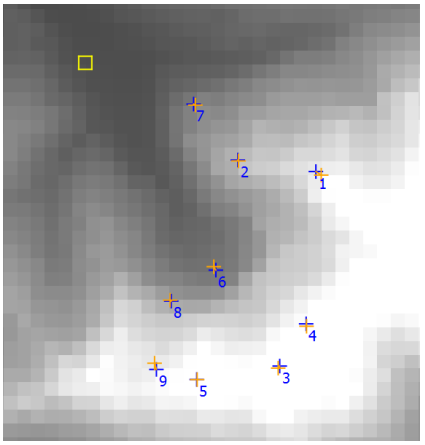


FIGURE 4.8: Projection des GCPs du système image sur le DEM

Comme la position à une précision de quelques mètres est connue, l'expérience de pose a été répétée, cette fois-ci en fixant les coordonnées Nord et Est. Le tableau 4.5 présente ces résultats. Les GCPs ont quelques fois été ajustés, le DEM étant à une résolution très grande et parfois approximatif. On peut noter que les erreurs sont du même ordre de grandeur que précédemment. Ceci peut s'expliquer par le fait que la quasi-totalité se

trouve loin de la caméra, à plusieurs kilomètres. Un meilleur positionnement de quelques dizaines de mètres n'a donc pas beaucoup d'influence sur la qualité des projections. Par exemple, un changement de focale de 1 % équivaut à un différence de pose de l'ordre de quelques dizaines de mètres.

Plus surprenant en revanche, la qualité des projections a tendance à être un peu plus élevée. Cette observation n'est pour l'instant pas explicable. Peut-être que, sans fixer de position, certaines erreurs, comme la calibration de la caméra, peuvent être compensées par un plus grand nombre de paramètres. Comme l'échantillon de tests est plutôt restreint avec six cas, ces variations sont peut-être purement stochastiques.

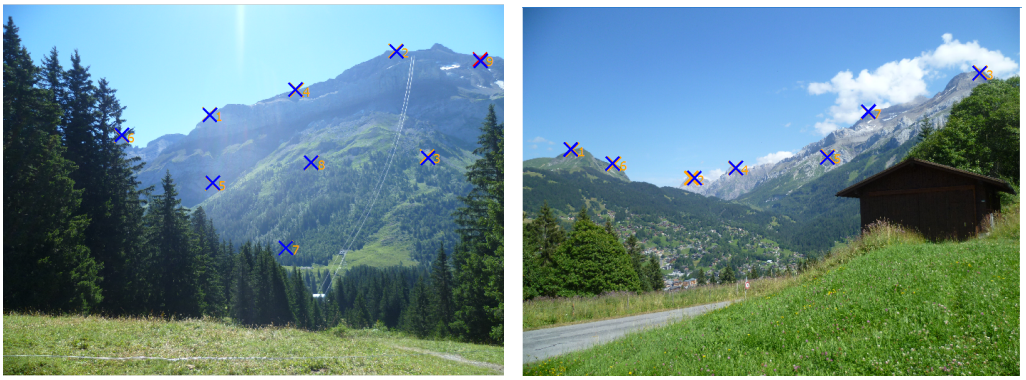
ID	Projection sur la photo [pixel]				Reprojection dans le terrain [m]			
	Moyenne	Min	Max	RMSE	Moyenne	Min	Max	RMSE
1	5.73	2.22	9.33	2.38	15.06	2.72	42.75	11.75
2	6.54	2.76	16.03	4.17	16.52	11.08	20.24	3.42
3	28.12	4.47	103.75	30.17	87.24	15.68	234.35	73.55
4	20.99	5.63	70.01	17.51	44.69	14.20	221.81	57.07
5	8.70	3.65	15.56	4.15	29.87	9.73	93.86	26.64
6	33.29	10.48	55.38	17.36	127.67	10.84	335.95	102.48
Moyenne	17.23	4.87	45.01	12.62	53.51	10.71	158.16	45.82

TABLE 4.5: Erreurs de projection et reprojection avec un DEM à 200m avec une position connue et fixée (mesures GPS). Les erreurs sont comparables à celles du tableau précédent. Les projections sont légèrement moins précises.

Le cas 6 est responsable d'augmenter la moyenne de l'erreur des reprojections considérablement. Sans prendre en compte ce cas, la moyenne est de 38 mètres, donc légèrement inférieur à une pose sans position connue. Le cas 6 est également celui qui a montré la plus grande différence de positionnement dans la section précédente. Des problèmes de définitions du DEM sont suspectés pour cette image. Cette affirmation n'est justifiée que par une analyse visuelle de la représentation 3D du DEM.

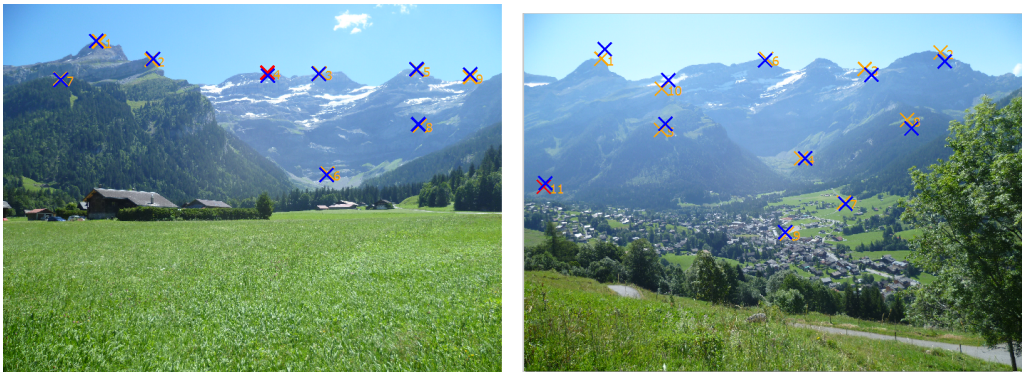
Notons tout de même que l'échantillon du cas 6 n'est pas représentatif d'une analyse complète, mais permet de se rendre compte des performances et des limites de l'outil. La figure 4.9 expose les six cas différents d'estimation de la pose. On peut y apprécier des cas divers et variés de configurations de GCPs, où les problèmes les plus grands ne viennent pas toujours des configurations les plus exotiques. Au fil des expériences, les GCPs ont quelque peu changé, c'est pourquoi sur les images, qui correspondent au tableau 4.5, le nombre de GCPs peut varier par rapport au tableau 4.3.

Une pose a également été estimée avec un DEM à 25 m de résolution spatiale. Le tableau 4.6 présente les résultats de cette expérience. La précision du positionnement planimétrique augmente considérablement. La moyenne des erreurs de reprojection diminue également, de même que la moyenne des projections. Le cas 6 a été pris à part, puisque



(a) cas 1 (b) cas 2

(c) cas 3 (d) cas 4



(e) cas 5 (f) cas 6

FIGURE 4.9: De gauche à droite et de haut en bas : les 6 cas d'études des projections. On peut voir, sur le cas 3, l'erreur maximum de 103 pixels, qui est en fait une erreur grossière, digitalisée sans ortho-photo ; ceci afin de montrer les possibilités si aucun ortho-photo n'est disponible. La configuration des GCPs du cas 2 n'est pas en cube du tout ; pourtant les projections sont assez précises. Le cas 6 a été le plus dur à traiter. Des problèmes de DEM sont suspectés.

l'algorithme n'arrive pas correctement à résoudre le problème de pose. Notons que le cas 6 possède les GCPs les plus éloignées de la caméra. Le problème est peut être lié à un roulis lègerement plus important que dans les autres cas.

Un test a également été effectué avec des images corrigées. Ceci permet de déterminer

ID	Projection [pixel]				Reprojection [m]				Position[m]
	Moy.	Min	Max	RMSE	Moy.	Min	Max	RMSE	
1	7.62	2.09	13.41	3.78	9.00	3.09	15.82	5.04	53.85
2	6.11	1.55	11.68	3.41	22.57	5.05	58.72	18.76	43.96
3	15.32	7.55	29.27	7.10	23.04	14.89	34.55	8.37	28.63
4	9.52	3.90	18.45	4.40	29.87	5.75	98.02	29.19	37.65
5	5.69	2.69	10.72	2.36	12.53	6.86	17.58	3.71	3.16
Moy.	8.852	3.556	16.706	4.21	19.402	7.128	44.938	13.014	33.45
6	43.98	4.73	74.88	19.84	212.11	14.97	488.87	162.97	88.60

TABLE 4.6: Erreurs de projection et reprojections avec un DEM à 25m. Les erreurs sont plus petites qu'avec le DEM à 200m. La moyenne de reprojection à 20m reste en-dessous de la résolution spatiale du DEM. La colonne de gauche représente l'erreur planimétrique de positionnement. Le cas 6 est considéré séparément ; l'algorithme est incapable de déterminer correctement la pose.

la signification des distortions de l'image. C'est donc un moyen de validé l'hypothèse de négligence de petits effets des distortions dans le processus d'estimation de pose. Les paramètres de correction ont été calculés avec openCV par Timothée. Pour ne citer qu'un résultat, concernant le cas 5, la prise en compte des corrections a permis de réduire l'erreur de projection, de 10 pixels en moyenne avec un DEM à 200m, à 5 pixels en moyenne avec un DEM à 25. En prenant en compte les distortions, l'erreur des projection est descendu jusqu'à environs 1 pixel. Cependant, en prennant en compte la résolution de l'image, les corrections des distortions n'ont rien apporté, puisque l'image corrigée a une diagonale de 625 pixel, contre 3'200 avant, donc un ratio de 5 environ.

Un dernier test a été fait sur le DEM à 200m. L'approche utilisée ici aurait dû être utilisée dans les tests précédents. Mais nous allons voir que ce n'est pas un problème. Afin de tester la précision des projections en terme de validation, l'erreur est mesurée sur des GCPs qui ne participent pas à la pose. Sur l'image 4.10, de tels GCPs ont leur reprojection en rose, alors que ceux utilisés dans l'algorithme de pose ont leur reprojection en bleu. Une synthèse des résulats est présentée dans le tableau 4.7. On peut remarquer que les erreurs des GCPs non utilisés dans le calculs ont le même ordre de grandeur que les GCPs utiles à la pose. On peut donc déduire que, si les GCPs sont bien répartis dans l'image et qu'un nombre suffisant sert à la pose, la précision est uniforme sur l'image.

Ii	Moy. Proj. [pixel]	Moy. Reproj. [m]	Pt 1	Pt 2	Pt. 3
1	6.28	15.45	7.34	4.52	21.16
2	15.15	37.93	27.26	9.1	17.95
3	8.19	36.54	45.95	18.66	21.78

TABLE 4.7: Erreurs de projection et reprojections avec un DEM à 200m. Ici, certains GCPs ont été ajoutés, mais n'ont pas été pris en compte dans l'estimation de la pose. Trois cas ont été étudiés, avec à chaque fois, trois GCP non utilisés dans la pose. Les erreurs de projections et reprojections des GCPs standards sont données dans les deux colonnes de gauches.

FIGURE 4.10: Tous les GCPs sont oranges. Ceux qui participent à la pose ont leur reprojection en bleu ; ceux qui ne partipent pas ont leur reprojection en rose. En résumé, les points ne participants pas à l'estimation de la pose en une précision similaire que les autres. Les précédants résultats peuvent donc être également lus ainsi, avec la retenue nécessaire.

4.6 Portabilité

Bien que Python et Qt soient tout à fait portables, la tâche de portée une application de machines en machines n'est jamais aisée. De nombreux problèmes de portabilité ont surgi durant la conception du plugin. D'autres apparaissent et de nombreux sont encore attendus. Le plugin a montré de bons comportements sur Windows. Cependant, les liaisons Python d'OpenGL semblent manquer sur beaucoup de systèmes Ubuntu et Mac OS. Les derniers tests sont cependant prometteurs et paraissent garantir une bonne portabilité. Les problèmes les plus récurents concernent OpenGL, surtout sur des environnements linux.

4.7 Limitations et problèmes

L'algorithme de pose utilisé montre certains défauts. Concernant, les paramètres internes de la caméra, seule la focale peut être déterminée ou fixée avec l'algorithme de pose. Le point central est hypothétiquement placé au centre de l'image ; les distorsions radiales

sont considérées comme nulles, de même que les distorsions tangentielles. Un développement succinct de l'inclusion des distorsions radiales et tangentielles peut être consulté à l'annexe A.11.

La précision de reprojections dépend de nombreux paramètres. Avec les tests réalisés, la digitalisation sur le monoplotter paraît possible, mais certainement pas assez précise pour nombre d'applications. Encore une fois, la précision dépendra de nombreux facteurs, il est toujours possible d'investir suffisamment d'efforts pour l'augmenter. Les paramètres internes n'étant pas encore pris en compte dans le plugin, il est nécessaire de corriger l'image au préalable pour prendre les déformations en considération.

Dans certains cas, la pose n'est bien réalisée, comme nous l'avons vu lors des tests, en référence au 6ème cas. Ces problèmes de pose semblent être liés à la présence d'un faible roulis. Quand la caméra est très penchée ($>30^\circ$), le roulis est très bien pris en compte. Cependant, des petit roulis ($< 12^\circ$) semblent poser problème.

Précisons encore que les erreurs dues à l'atmosphère sont négligeables. Elles sont de l'ordre de 1 mm à 1 cm par cent mètres de distance. La négligence totale de la rotondité de la terre inclue une erreur de 1.5 mètres à 10 kilomètres. Cette erreur reste inférieure à celles obtenues. Cependant, si l'outil devait se perfectionner à tel point que l'erreur de reprojection devienne de l'ordre du mètre, la rotondité de la terre devrait être prise en compte.

Chapitre 5

Conclusion

Dans la fin de ce travail, nous allons parcourir la réalisation des objectifs et plonger dans les suites possibles du travail.

5.1 Résumé

L'objectif de ce travail était d'implémenter un outil de monoplotting sous la forme d'un plugin Python pour QGIS. Diverses fonctionnalités étaient prévues. Certains points, comme la facilité d'utilisation ou l'aide à la digitalisation de GCPs, étaient mis en avant. Les objectifs du travail tels que posés au début du projet ont été atteints. Le plugin a été publiquement dévoilé dans la revue de presse du 13 juin de Geotribu [1].

Le plugin implémente l'intégralité des fonctionnalités proposées. Quelques fonctionnalités supplémentaires complètent le plugin : l'intégration de la symbologie QGIS, une interface pour gérer l'ortho-rectification, la visualisation et digitalisation des polygones ou encore la visualisation des étiquettes.

Au niveau de l'expérience utilisateur, sur lequel un effort devait être fourni, la possibilité de naviguer dans le DEM en 3D aide considérablement l'utilisateur. Ainsi, il est possible de digitaliser des points sur le DEM directement dans la vue 3D, ce qui aide dans le cas de sommet par exemple, souvent difficile à reconnaître sur une ortho-photo. Une approche sans GCPs est proposée, se reposant exclusivement sur cette capacité de voyage dans le DEM.

L'algorithme de positionnement a montré une précision de l'ordre de quelques dizaines de mètres pour des points situés à quelques kilomètres, en travaillant sur un DEM à

1. http ://geotribu.net/

deux cents mètres. Ceci représente les limites du plugin. L'outil paraît donc approprié pour des tâches telles que la visualisation de couches vectorielles dans une photographie. La précision peut être suffisante pour que la digitalisation rivalise avec une digitalisation standard, mais l'effort fournit pour déterminer la pose et la qualité du DEM et de l'image est déterminant.

Finalement, créer un objet concret, un produit fini, est très passionnant pour un projet de fin d'études. De plus, la création d'outil de monoplotting permet d'approfondir des problèmes rarement considérés ; par exemple, comment est stocké un fichier raster sur le disque, comment l'altitude est récupérée après un clic à l'écran, comment le géoréférencement est créé pour un géotiff, etc. De plus, le monoplotting se trouve à l'intersection entre la photogrammétrie et le domaine de la vision (ou Computer Vision). L'estimation de la pose de caméra est un sujet certes déjà traité à de nombreuses reprises, et ce depuis des décennies, mais s'y plonger au moment de clore ses études est un vrai plaisir dans la mesure où cela fait appel à plusieurs compétences acquises tout au long de mon cursus universitaire : la création d'outils cartographiques en master tissent un lien entre les équations d'algèbres linéaires de 1ère année en passant par les applications de concepts écologiques étudiées en 3ème année.

5.2 Perspectives

Trois axes principaux sont à considérer pour continuer le présent travail : la précision, l'utilisabilité et la mise à jour.

La précision de l'outil dépend de la précision de la pose, mais également de la prise en compte de déformation de l'image ou de l'utilisation du z-buffer.

Pour commencer par la précision de la pose, un algorithme pourrait être développé pour fixer la position de la caméra au sol, légèrement au-dessus de DEM. Actuellement, la caméra peut se retrouver sous le DEM si aucune contrainte n'est donnée. Un moyen de pallier ce problème existe déjà, mais il n'est pas automatique. Un tel algorithme pourrait aussi assurer la visibilité des GCPs, ce qui n'est pas encore le cas, bien qu'un message d'erreur apparaisse lorsqu'un GCPs est apparemment derrière une montagne.

Les déformations de l'image telle que proposées dans la section 4.7 sont à intégrer au plugin. De même que la prise en compte du point central serait certainement un bénéfice immédiat dans la qualité des projections.

L'optimisation de certains paramètres utilisés dans OpenGL permettrait de meilleures projections. Les points éloignés de la caméra souffrent beaucoup de la logique d'OpenGL.

Comme expliqué à la section 4.1.3.1, les données stockées dans le z-buffer ne sont pas linéairement dépendantes à l'éloignement. Les points les plus proches bénéficient d'une meilleure position, car l'hypothèse faite est que l'utilisateur est plus précis lorsqu'il clique sur un point proche ; ce qui est vrai, mais à quel point ?

L'utilisation de l'algorithme PnP serait préférable quand la focale est connue. En plus de permettre l'estimation de la pose avec seulement quatre points, l'algorithme PnP est connu pour être de meilleure précision que l'algorithme DLT. Comme discuté à la section 3.2.3, PnP est stable et convergent pour une configuration planimétrique des GCPs. Pour travailler avec DEM plat, comme par exemple dans les plaines d'Afrique ou sur un lac, l'implémentation d'une résolution complémentaire par PnP serait plus que bienvenue. Un accomplissement serait d'implémenté la méthode proposée par [32]. L'utilisation de PnP nous amène au deuxième point : l'utilisabilité.

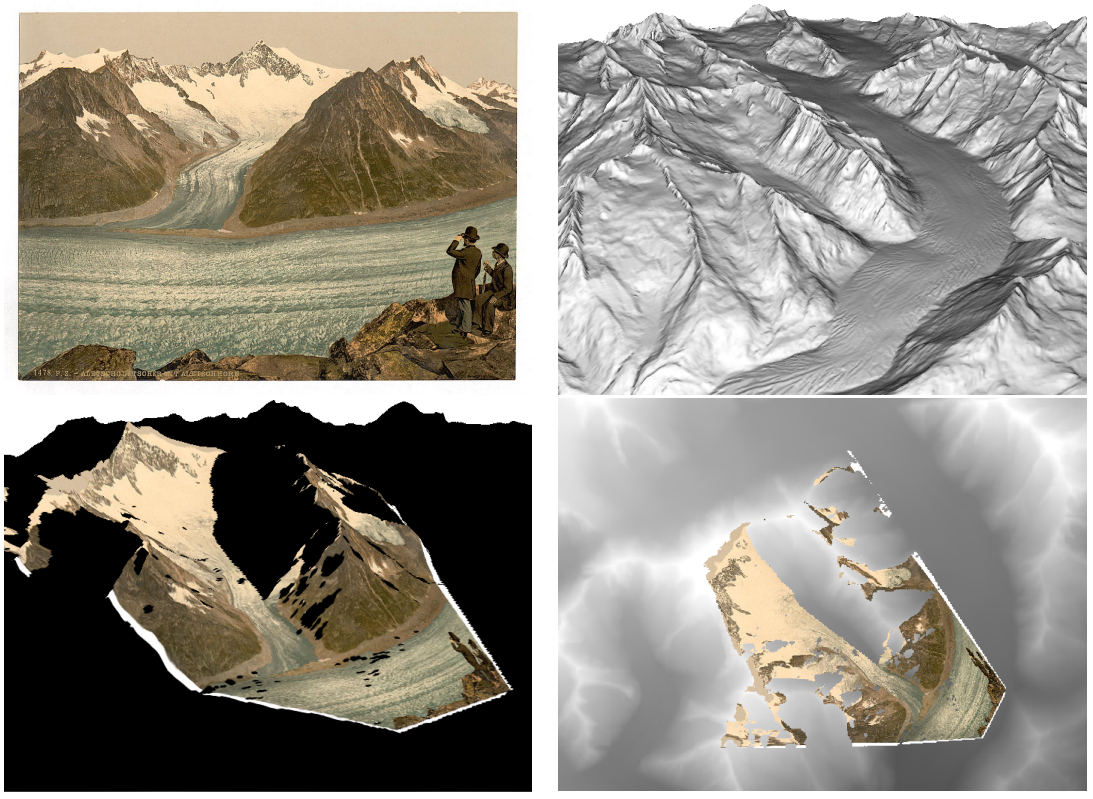
L'augmentation du confort de l'utilisateur peut passer par différent points ; une aide pour la digitalisation des GCPs, une nombre de GCPs requis plus faible, l'intégration de couches WMS [2]

Une prédétermination de la pose de la caméra pourrait être développée avec deux points de contrôle. Il doit être possible de faire fi des coordonnées Z et V des GCPs. afin d'avoir une détermination de la position et de l'orientation planimétrique. Les paramètres à estimer seraient donc X0, Y0, la focale et l'azimut.

Il serait également souhaitable d'inclure un WMS ou un objet équivalent pour récupérer des informations 3D quand aucun DEM n'est disponible facilement. L'utilisateur débutant en SIG n'aurait donc pas le souci de se procurer un DEM.

En ce qui concerne la mise à jour, le plugin doit être continuellement développé et adapté suivant les versions des modules utilisés et de l'intégration dans QGIS. Certaines fonctions peuvent disparaître. Il est donc nécessaire de tester le plugin sur chaque version béta de QGIS et de corriger le code en conséquence. Il serait également bienvenu de voir le plugin traduit, de même que la documentation en d'autres langues.

2. Web Map Service : Des données géo-référencées, comme un DEM, peuvent être importés d'un serveur directement dans le logiciel SIG.



Annexe A

Equations et définitions

A.1 Systèmes de référence

On appelle "système objet" le système de référence des objets présents dans la réalité. Le système objet est fixe par rapport à la Terre, contrairement aux deux autres systèmes. Ce système sera toujours considéré ici comme métrique. Sans perte de généralité, le système objet est ici assimilable au système de référence du DEM. La définition du système objet peut changer selon l'objet utilisé ou les choix de l'utilisateur.

On définit le système de coordonnées "caméra" ainsi : l'origine de système caméra se trouve sur au point focal. L'axe Z est dirigé perpendiculairement au plan de projection de l'image est dirigé à l'opposé de l'image. En regardant l'image, l'axe X est dirigé vers la droite et l'axe Y vers le haut. Contrairement au système objet, le système caméra n'est pas fixe par rapport à la Terre.

Le système image à généralement l'origine dans un coin de l'image. Il peut aussi être défini au centre. Le système image a seulement deux dimensions, l'une dite horizontale et l'autre verticale. La transformation entre le système image et le système objet nécessite une étape intermédiaire dans le système caméra.

A.2 Matrice de Rotation définie par tilt, azimut et swing

La matrice de rotation défini selon les angles tilt, azimut et swing est donnée par :

$$
\begin{aligned}
R_{1,1} &= -cos(azimuth) * cos(swing) - sin(azimuth) * cos(tilt) * sin(swing) \\
R_{1,2} &= sin(azimuth) * cos(swing) - cos(azimuth) * cos(tilt) * sin(swing) \\
R_{1,3} &= -sin(tilt) * sin(swing) \\
R_{2,1} &= cos(azimuth) * sin(swing) - sin(azimuth) * cos(tilt) * cos(swing) \\
R_{2,2} &= -sin(azimuth) * sin(swing) - cos(azimuth) * cos(tilt) * cos(swing) \\
R_{2,3} &= -sin(tilt) * cos(swing) \\
R_{3,1} &= -sin(azimuth) * sin(tilt) \\
R_{3,2} &= -cos(azimuth) * sin(tilt) \\
R_{3,3} &= cos(tilt)
\end{aligned} \tag{A.2.1}
$$

A.3 Matrices de rotation selon la convention roll, pitch, yaw

Les trois matrices correspondant à des rotations successives selon les X, Y et Z sont données par :

$$
R(\kappa) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos(\kappa) & sin(\kappa) \\ 0 & -sin(\kappa) & cos(\kappa) \end{pmatrix} \tag{A.3.1}
$$

$$
R(\phi) = \begin{pmatrix} cos(\phi) & 0 & -sin(\phi) \\ 0 & 1 & 0 \\ sin(\phi) & 0 & cos(\phi) \end{pmatrix} \tag{A.3.2}
$$

$$
R(\omega) = \begin{pmatrix} cos(\omega) & sin(\omega) & 0 \\ -sin(\omega) & cos(\omega) & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{A.3.3}
$$

A.4 Rotation et matrices de rotation

Plusieurs variantes existent afin de représenter une rotation en trois dimensions. Certaines ont neuf paramètres, représentant les éléments d'une matrice 3x3, d'autres variantes

utilisent des objets appelés quaternion possédant quatre paramètres. Une des variantes des plus répandues utilise trois paramètres : les angles roulis, tangage et lacet (roll, pitch, yaw en anglais). La rotation est alors le produit de trois matrices, chacune liée à un des angles :

$$R = R(\omega) \cdot R(\phi) \cdot R(\kappa) = \begin{pmatrix} m_{1,1} & m_{1,2} & m_{1,3} \\ m_{2,1} & m_{2,2} & m_{2,3} \\ m_{3,1} & m_{3,2} & m_{3,3} \end{pmatrix} \tag{A.4.1}$$

Dont les trois matrices sont formées selon l'annexe A.3.

A la place d'utiliser des rotations successives autour des axes (ω, ϕ, κ), il est possible d'exprimer ces mêmes rotations selon une autre convention. On nomme les angles de cette deuxième convention tilt, heading et swing. Le mot français "azimut" sera utilisé à la place de "heading". Bien que plus difficile à décrire mathématiquement, le trio tilt, azimut, swing est quelques fois préféré au trio roll, pitch, yaw pour sa logique plus intuitive.

On nommera $X_\alpha Y_\alpha Z_\alpha$ le système de coordonnées XYZ pivoté par un azimut α, $X_{\alpha t} Y_{\alpha t} Z_{\alpha t}$ le système de coordonnées XYZ pivoté par un azimut α et un tilt t, $X_{\alpha ts} Y_{\alpha ts} Z_{\alpha ts}$ le système de coordonnées XYZ pivoté par un azimut α, un tilt t et un swing s.

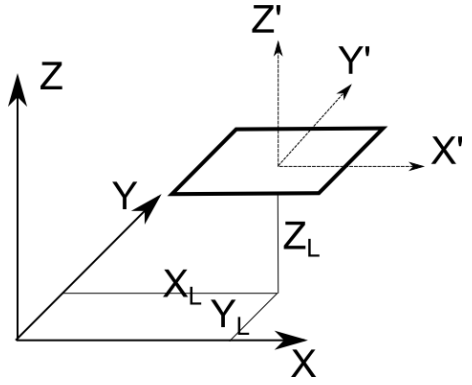


FIGURE A.1: Système de référence (X, Y, Z) translaté par (X_L, Y_L, Z_L)

Ce nouveau trio de rotation est appliqué selon l'ordre successif $Z', X_\alpha, Z_{\alpha t}$. Une première rotation autour de l'axe Z est appliquée dans le sens anti-trigonométrique. L'angle de rotation est appelé "azimuth" t noté α. Les coordonnées d'un point quelconque dans le système $(X_\alpha Y_\alpha Z_\alpha)$ peuvent être exprimées selon :

$$
\begin{aligned}
x^{\alpha} &= xcos(\alpha) - ysin(\alpha) \\
y^{\alpha} &= xsin(\alpha) + ycos(l\alpha) \\
z^{\alpha} &= z
\end{aligned}
\tag{A.4.2}
$$

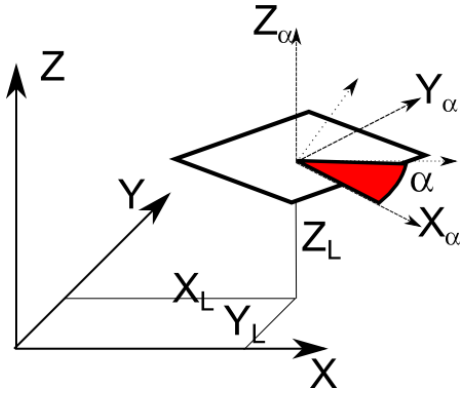


FIGURE A.2: Système de référence (X', Y', Z') pivoté par un azimut α

Les coordonnées d'un point quelconque dans le système $(X_{\alpha t}Y_{\alpha t}Z_{\alpha t})$ peuvent être exprimées selon :

$$
\begin{aligned}
x^{\alpha t} &= x^{\alpha} \\
y^{\alpha t} &= y^{\alpha}cos(t) + z^{\alpha}sin(t) \\
z^{\alpha t} &= -y^{\alpha}sin(t) + z^{\alpha}cos(t)
\end{aligned}
\tag{A.4.3}
$$

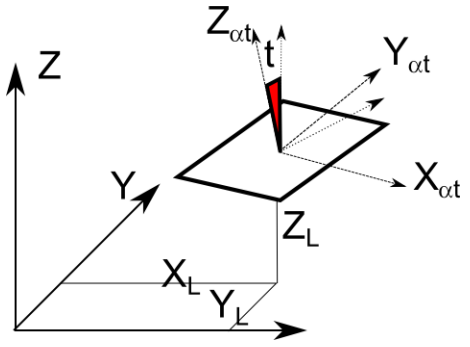


FIGURE A.3: Système de référence $(X_{\alpha}Y_{\alpha}Z_{\alpha})$ pivoté par un tilt t

Les coordonnées d'un point quelconque dans le système $(X_{\alpha ts}Y_{\alpha ts}Z_{\alpha ts})$ peuvent être exprimées selon :

$$
\begin{aligned}
x^{\alpha ts} &= -x^{\alpha t} cos(s) - y^{\alpha t} sin(s) \\
y^{\alpha ts} &= y^{\alpha t} cos(s) - z^{\alpha t} sin(s) \\
z^{\alpha ts} &= z^{\alpha t}
\end{aligned} \tag{A.4.4}
$$

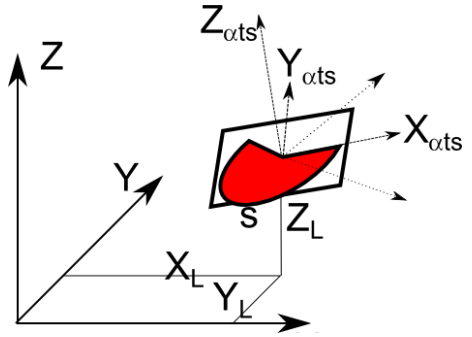


FIGURE A.4: Système de référence $(X_{\alpha t} Y_{\alpha t} Z_{\alpha t})$ pivoté par swing s

Une transformation d'un vecteur (x', y', z') par une rotation appliquée selon la formule A.4.5, avec l'annexe A.2 qui fait le lien avec les angles tilt, azimut and swing.

$$
\begin{aligned}
x_{\alpha ts} &= m_{1,1} \cdot x_L + m_{1,2} \cdot y_L + m_{1,3} \cdot z_L \\
y_{\alpha ts} &= m_{2,1} \cdot x_L + m_{2,2} \cdot y_L + m_{2,3} \cdot z_L \\
z_{\alpha ts} &= m_{3,1} \cdot x_L + m_{3,2} \cdot y_L + m_{3,3} \cdot z_L
\end{aligned} \tag{A.4.5}
$$

A.5 Projection perspective

Une transformation d'un système de coordonnées en trois dimensions sur un plan en deux dimension n'est pas similaire à un changement de système de référence. Les points projetés sur le plan ne possèdent plus les mêmes caractéristiques qu'avant. Par exemple, plusieurs points distinctement éloignés en trois dimensions peuvent se retrouver sur un même point une fois projetés.

La transformation perspective du point $\boldsymbol{X'}$ sur le plan $z = \frac{1}{a}$ est donnée par :

$$
\begin{bmatrix} zx \\ zy \\ z \end{bmatrix} = P\boldsymbol{X'} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & a \end{pmatrix} \begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} \tag{A.5.1}
$$

où le vecteur $(x, y)^t$ représente la projection dans l'espace image et $(X', Y', Z')^t$ représente la transformation d'un point d'un système quelconque vers le système de coordonnées caméra.

A.6 Projection par une forme homogène - paramètres de la DLT

(A.6.1)
$$u = \frac{L_1 x + L_2 y + L_3 z + L_4}{L_9 x +_{10} y + L_{11} z + 1}$$
$$v = \frac{L_5 x + L_6 y + L_7 z + L_8}{L_9 x +_{10} y + L_{11} z + 1}$$

On peut remarquer que L est exprimé par un produit matriciel dont les trois premières colonnes sont formées des paramètres internes et de rotations alors que la quatrième colonne exprime simplement la translation de l'origine :

(A.6.2)
$$\begin{pmatrix} L_1 & L_2 & L_3 & L_4 \\ L_5 & L_6 & L_7 & L_8 \\ L_9 & L_{10} & L_{11} & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & u0 \\ 0 & 1 & v0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1/f \end{pmatrix} \begin{pmatrix} m_{1,1} & m_{1,2} & m_{1,3} & -X_L \\ m_{2,1} & m_{2,2} & m_{2,3} & -Y_L \\ m_{3,1} & m_{3,2} & m_{3,3} & -Z_L \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

A.7 Coefficients de la DLT

Les équations A.7.1 expriment les coefficients de la DLT en fonction des paramètres utilisés dans les équations de colinéarité.

$$
\begin{aligned}
D &= -(x_0 m_{3,1} + y_0 m_{3,2} + z_0 m_{3,3}) \\
L_1 &= \frac{u_0 m_{3,1} - d_u m_{1,1}}{D} \\
L_2 &= \frac{u_0 m_{3,2} - d_u m_{1,2}}{D} \\
L_3 &= \frac{u_0 m_{3,3} - d_u m_{1,3}}{D} \\
L_4 &= \frac{(d_u m_{1,1} - u_0 m_{3,2}) x_0 + (d_u m_{1,2} - u_0 m_{3,2}) y_0 + (d_u m_{1,3} - u_0 m_{3,3}) z_0}{D} \\
L_5 &= \frac{v_0 m_{3,1} - d_v m_{1,1}}{D} \\
L_6 &= \frac{v_0 m_{3,2} - d_v m_{1,2}}{D} \\
L_7 &= \frac{v_0 m_{3,3} - d_v m_{1,3}}{D} \\
L_8 &= \frac{(d_v m_{2,1} - u_0 m_{3,2}) x_0 + (d_v m_{2,2} - u_0 m_{3,2}) y_0 + (d_v m_{2,3} - u_0 m_{3,3}) z_0}{D} \\
L_9 &= \frac{m_{3,1}}{D} \\
L_{10} &= \frac{m_{3,2}}{D} \\
L_{11} &= \frac{m_{3,3}}{D}
\end{aligned}
\tag{A.7.1}
$$

A.8 Extraction des paramètres de pose depuis la DLT

Une fois la matrice d'homographie calculée, l'extraction des paramètres de Pose se trouve être un problème ardus. Différentes approches existent, qui ne donnent pas toute les mêmes résultats, bien que similaire.

A.8.1 Position

On commence par extraire la position de la caméra :

$$
\begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix} = \begin{pmatrix} L_1 & L_2 & L_3 \\ L_5 & L_6 & L_7 \\ L_9 & L_{10} & L_{11} \end{pmatrix}^{-1} \begin{bmatrix} -L_4 \\ -L_8 \\ -1 \end{bmatrix}
\tag{A.8.1}
$$

Ensuite, on calcule la valeur de D, qui sera utile pour calculer les autres paramètres.

$$
D^2 = \frac{1}{L_9^2 + L_{10}^2 + L_{11}^2}
\tag{A.8.2}
$$

A.8.2 Point central

Les paramètres u0 et v0 sont facilement extrait selon les formules de transformation de variables appliquées auparavant.

$$\begin{aligned} u_0 &= D^2(L_1L_9 + L_2L_{10} + L_3L_{11}) \\ v_0 &= D^2(L_5L_9 + L_6L_{10} + L_7L_{11}) \end{aligned} \tag{A.8.3}$$

A.8.3 Focale

Les valeurs d_u et d_v expriment la longueur focale dans les directions verticale et horizontale de l'image. La focale est calculée par la moyenne de ces deux grandeurs.

$$\begin{aligned} d_u &= D^2[(u_0L_9 - L_1)^2 + (u_0L_{10} - L_2)^2 + (u_0L_{11} - L_3)^2] \\ d_v &= D^2[(v_0L_9 - L_5)^2 + (v_0L_{10} - L_6)^2 + (v_0L_{11} - L_7)^2] \end{aligned} \tag{A.8.4}$$

A.8.4 Matrice de rotation

Une première approximation de la matrice de rotation est donnée par :

$$R = D \begin{pmatrix} u_0L_9 - L_1 & u_0L_{10} - L_2 & u_0L_{11} - L_3 \\ v_0L_9 - L_5 & v_0L_{10} - L_6 & v_0L_{11} - L_7 \\ L_9 & L_{10} & L_{11} \end{pmatrix} \tag{A.8.5}$$

Si le déterminant de la matrice de rotation est négatif, la rotation est dite "left-handed". Dans ce cas, on prend l'opposé de D pour calculer la matrice de rotation. Un des problèmes principaux de la DLT est l'incertitude quant à l'orthogonalité de la matrice de rotation. Le problème a déjà été pris en compte, notamment en incluant des contraintes dans le système d'équations, notamment dans [44]. Dans le cas présent, il a été choisi de prendre cette première matrice comme une approximation et de trouver, ensuite, la matrice orthogonale la plus proche.

Traditionnellement, on calcule les paramètres internes de la caméra par une factorisation de cholesky des trois premières colonnes de la matrice d'homographie. On en ressort également la matrice de rotation. La matrice d'homographie est composée de $L = (L3|c_4)$,

$L3$ étant les trois premières colonnes de L. Il s'agit d'extraire R de $L3$ selon la relation $L3 = P \cdot R$, P étant la matrice de projection perspective. On peut donc écrire :

$$L3L3^t = (PR)(PR)^t = PRR^tP^t = PP^t \tag{A.8.6}$$

car l'inverse d'une matrice de rotation est égale à sa transposée ($RR^t = RR^{-1} = I$). Selon la définition de P, c'est une matrice triangulaire supérieure ; elle peut donc être trouvée par factorisation de Choleski. PP^t est appelé "Image de la Conique Absolue". Elle est souvent utilisée comme matrice de calibration interne. Après l'extraction de P par factorisation de Choleksi, une estimation de R est donnée par :

$$\widetilde{R} = P^{-1}L3 \tag{A.8.7}$$

Avec l'équation A.8.5, nous possédons déjà une approximation de la matrice de rotation, les dernières étapes ne sont donc pas nécessaires. On peut également noter qu'il est possible d'extraire une approximation par une décomposition RQ. Les équations développées plus haut sont en fait équivalents à une décomposition RQ.

Pour trouver la meilleure matrice R qui approxime $\widetilde{R}$, on applique une décomposition en valeur singulière à la matrice R.

$$\widetilde{R} = USV^t \tag{A.8.8}$$

Tel que S est une matrice diagonale possédant les valeurs propres de la décomposition sur ses éléments diagonaux, et tels que U et V sont orthogonaux par colonne. R est donné par :

$$R = UV^t \tag{A.8.9}$$

A.9 Positionnement dans OpenGL

La position de la caméra est directement obtenue dans les paramètres de pose. Afin de déterminer la direction de vue, un point au centre de la vue doit être donné. On le calcul selon la définition du système caméra :

$$v = \begin{bmatrix} X0 \\ Y0 \\ Z0 \end{bmatrix} + R^{-1} \cdot \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix} \tag{A.9.1}$$

avec R étant la matrice de rotation formée par les angles tilt, azimut et swing selon l'annexe A.7.

Le roulis κ (l'angle avec l'horizontale) de l'image est donné par :

$$\kappa = arcsin(R_{1,3}) = -sin(tilt) \cdot sin(swing) \tag{A.9.2}$$

Les paramètres à entrer dans gluLookAt sont donc :

$$gluLookAt \begin{pmatrix} X0 & Y0 & Z0 \\ v_1 & v_2 & v_3 \\ 0 & cos(\kappa) & sin(\kappa) \end{pmatrix} \tag{A.9.3}$$

avec v_i un élément du vecteur v de l'équation A.9.1. Il reste à entrer l'angle de vue verticale dans la fonction gluPerspective. L'angle de vue est calculé à partir de la taille de l'image en pixel et de la focal calculée, en pixel elle aussi. L'angle de vue vertical en degrés appelé FOV (pour Field Of View) est donné par :

$$FOV = 2 \cdot arctan(\frac{hauteur\ de\ l'image\ [pixel]}{focal\ [pixel]}) \cdot 180/pi \tag{A.9.4}$$

A.10 Intégration du point central dans OpenGL

Le fonctionnement interne d'OpenGL doit être maîtrisé afin d'inclure la notion de point central. Dans le cas où le point central est au milieu de l'image, la matrice de projection s'écrit ainsi :

$$projectionMatrix = \begin{pmatrix} \frac{n}{r} & 0 & 0 & 0 \\ 0 & \frac{n}{t} & 0 & 0 \\ 0 & 0 & -\frac{f+n}{f-n} & -1 \\ 0 & 0 & -\frac{2fn}{f-n} & 0 \end{pmatrix} \tag{A.10.1}$$

où r est la limite droite, l la limite gauche, t la limite haute, b la limite basse, n la limite proche et f la limite éloignée. La figure A.5 représente ces paramètres. Les unités de n et f sont les unités du DEM (ici en mètres). Les unités de r, l, t, et b sont un peu plus corsées. Ces variables sont données par rapport à la demi-hauteur de l'image. C'est pourquoi les valeurs habituelles sont $(t, b, r, l)^T = (a, -a, largeur/hauteur \cdot a, -largeur/hauteur \cdot a)$, avec $a = n \cdot tan(FOV/2 \cdot pi/180)$.

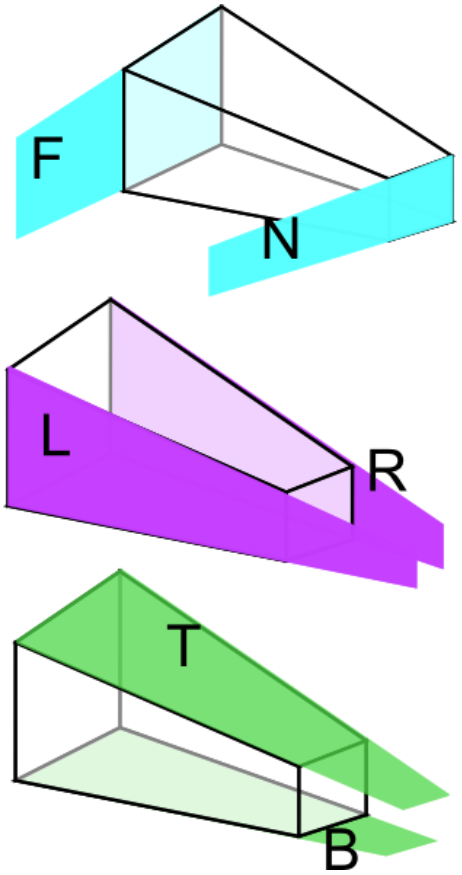


FIGURE A.5: Représentation des plans limites de projection

Si le point central n'est pas au centre de l'image, les limites gauches, droites, hautes et basses ne sont plus symétriques et centrés. On peut recalculer les valeurs de t, b, r et l en prenant en compte u0 et v0 selon :

$$\begin{aligned} a &= n \cdot 2 \cdot tan(FOV/2 \cdot pi/180) \\ t &= a \cdot (1/2 + v0) \\ b &= a \cdot (-1/2 + v0) \\ r &= a \cdot (1/2 + u0) \\ l &= a \cdot (-1/2 + u0) \end{aligned} \tag{A.10.2}$$

Dans ce cas, la matrice de projection est écrite ainsi :

$$projectionMatrix = \begin{pmatrix} \frac{2n}{r-l} & 0 & 0 & 0 \\ 0 & \frac{2n}{t-b} & 0 & 0 \\ \frac{r+l}{r-l} & \frac{t+b}{t-b} & -\frac{f+n}{f-n} & -1 \\ 0 & 0 & -\frac{2fn}{f-n} & 0 \end{pmatrix} \tag{A.10.3}$$

A.11 Distortions d'une image

Les distorsions radiales s'expriment par :

$$\begin{aligned} r &= \sqrt{\widetilde{u}^2 + \widetilde{v}^2} \\ u &= \widetilde{u} \cdot (1 + \kappa_1 r^2 + \kappa_2 r^4) \\ v &= \widetilde{v} \cdot (1 + \kappa_1 r^2 + \kappa_2 r^4) \end{aligned} \tag{A.11.1}$$

Les distorsions tangentielles s'expriment par

$$\begin{aligned} u &= \widetilde{u} + 2\kappa_3 \widetilde{u}\widetilde{v} + \kappa_4(r^2 + 2\widetilde{u}^2) \\ v &= \widetilde{v} + 2\kappa_4 \widetilde{u}\widetilde{v} + \kappa_5(r^2 + 2\widetilde{u}^2)) \end{aligned} \tag{A.11.2}$$

Ces distorsions viennent du modèle appelé Brown-Conrad, par exemple repris par [45]. Le problème de l'implémentation des distorsions optiques est leur intégration dans OpenGL. En effet, la correspondance 2D-3D étant largement basée sur les ressources d'OpenGL, il n'y a pas vraiment de solution possible à l'intégration des distorsions si OpenGL ne

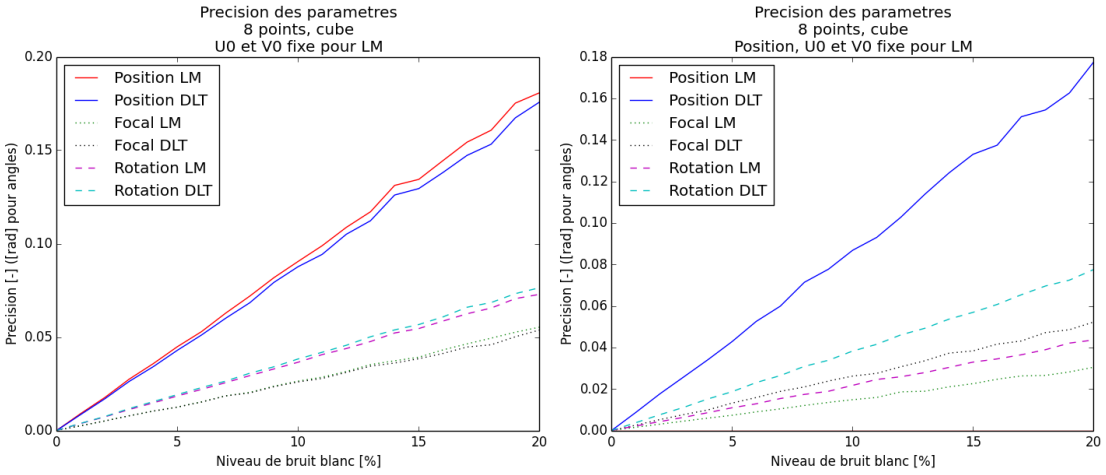
le permet pas. Une des approches possibles serait d'exprimer les distorsions dans la matrice de projection, comme il est fait pour le point central par exemple. Les choses se compliquent lorsqu'il s'agit d'inclure des déformations non-linéaires.

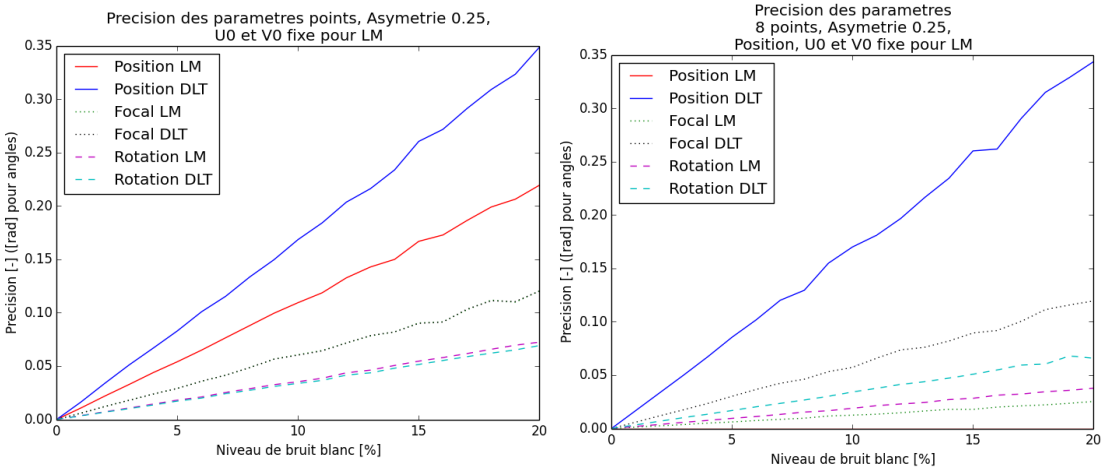
Annexe B

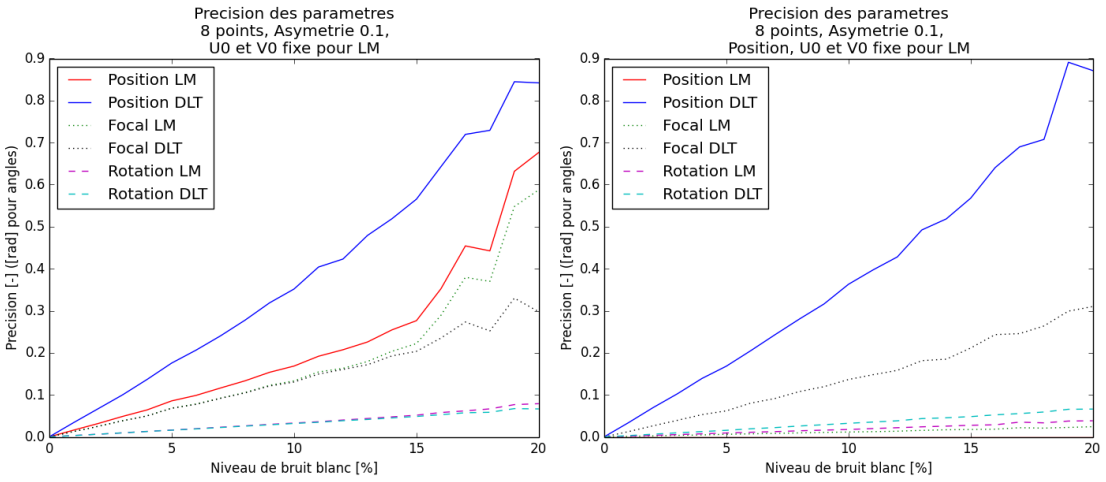
Précision de pose et de projection

B.1 Précision de projection - simulatons numérique

Les images suivantes présentent l'évolution des paramètres par rapport à une augmentation de bruit dans les mesures. L'image B.1 présente l'évolution des paramètres par rapport à l'asymétrie de la répartition des points dans l'espace voir 4.5.1.







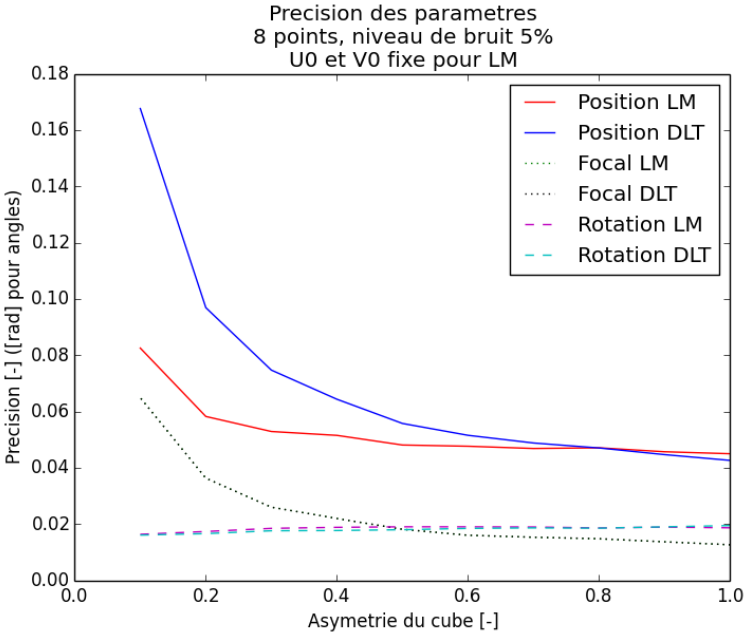


FIGURE B.1: Évolution de la précision selon l'asymétrie de la répartion spatiale des points de contrôle

B.2 Précision de projection à 20% de bruit

Ce tableau est comparable au tableau 4.1, mais pour un niveau de bruit de 20%. En soit, il n'apporte pas beaucoup plus d'information. Les figures de la section B.1 sont plus explicites pour analyser les différences de pose à des niveaux de bruit différents.

Paramètre	Algorithme	Cube	Asymétrie 0.25	Asymétrie 0.1
Position	DLT	0.17	0.35	0.84
	LM	0.18	0.22	0.67
	LM+Pos	0	0	0
Orientation	DLT	0.076	0.069	0.066
	LM	0.072	0.072	0.079
	LM+Pos	0.043	0.0377	0.038
Focale	DLT	0.053	0.120	0.297
	LM	0.055	0.120	0.588
	LM+Pos	0.030	0.025	0.024

TABLE B.1: Simulation de l'estimation de la pose à 20 % de bruit. les mêmes conclusions qu'avec un bruit de 10% peuvent être faite, avec ici une amplitude plus grande.

Le tableau B.2 détaille les erreurs de positionnement du tableau 4.3. Les positions sont données dans le système CH1903.

ID	GPS		Estimation		Erreur [m]
	Nord	Est	Nord	Est	
1	134135	581562	134011	581692	179,6
2	132893	577342	133219	577546	384,5
3	132091	578427	132405	578461	315,8
4	134302	580992	134391	580791	219,8
5	132688	578221	132724	578083	142,6
6	134280	577696	134750	577076	778,0
moyenne des erreurs					336,7

TABLE B.2: Erreurs de positionnement planimétrique

Annexe C

Exemples de développement

Cette annexe présente quelques exemples de pseudo-code représentant le travail effectué. On peut notamment y remarqué l'effort fourni pour certains processus paraissant simples comme la lecture des labels, alors que d'autres, d'apparence plus compliqué, peuvent être facilement mis en place comme la digitalisation de nouvelles entitées.

C.1 Paramètres dans les équations de colinéarités

```
#PARAM is a boolean vector with element 1 if the parameter is variable.
#p is the parameter vector reconstructed from the variable paramters (x_unkown)
# and the fixed parameters (x_fix)
p = zeros(shape=(len(PARAM)))
m = 0
n = 0
for k in range(len(PARAM)):
  if PARAM[k]:
    p[k] = x_unkown[m]
    m = m+1
  else:
    p[k] = x_fix[n]
    n = n+1
# R is calculated from p[3],p[4],p[5]
R = [...]
#Colinearity equation
ures = -p[6]\cdot (R[0,0]\cdot (x1-p[0])+R[0,1]\cdot (y1-p[1])+R[0,2]\cdot (z1-p[2]))/
    (R[2,0]\cdot (x1-p[0])+R[2,1]\cdot (y1-p[1])+R[2,2]\cdot (z1-p[2]))+p[7];
vres = -p[6]\cdot (R[1,0]\cdot (x1-p[0])+R[1,1]\cdot (y1-p[1])+R[1,2]\cdot (z1-p[2]))/
    (R[2,0]\cdot (x1-p[0])+R[2,1]\cdot (y1-p[1])+R[2,2]\cdot (z1-p[2]))+p[8];
```

C.2 Récupération des labels

```
def getSymbology
  [...]
  #Iterate over layer
  for name, layer in layers.iteritems():
    #Save symbology only of vector layer and visible
    if layer.type() ==0 and self.iface.legendInterface().isLayerVisible(layer):
      #create Label object and load the current labels
      palyr = QgsPalLayerSettings()
      palyr.readFromLayer(layer)
      #check if labels are activated
      if not palyr.enabled:
        labelParam.append(-1)
      else:
        If labels are activated, iterate over fields
        fields =  layer.pendingFields()
        id = -1
        for field in fields:
          #Get the ID of the field used for labels
          id+=1
          if str(field.name()) == palyr.fieldName:
            labelParam.append(id)
  [...]

def refreshLayers
  [...]
  #Iterate over layers
  for name, layer in layers.iteritems():
    #get Field ID used for labelling
    labelID = labelParam[count]
        iterator = layer.getFeatures(request)
        if labelID != -1:
        for feature in iterator:
      #Get label value for each feature.
          labels.append(feature.attributes()[labelID])
  [...]
```

C.3 Clic projeté dans le canvas

```
# Create a connection between click on OpenGL Window and the clickOnMonoplotter function
OpenGLWindow.clickSignal.connect(clickOnMonoplotter)

# This is called when an mouse event happens in the OpenGLWindow
mousePressEvent(event):
  [...]
  #read the z-buffer at the mouse location
  z = glReadPixels(x, y, 1, 1, GL_DEPTH_COMPONENT, GL_FLOAT)
  #Get the 3D coordinates
  result = gluUnProject( x, y, z, modelview, projection, viewport)
  if z != 1.0:
    # If the click is done out of the DEM (above horizon), z is equal to 1.0
    self.clickSignal.emit([-result[0],result[2]])

#This is called when the clickSignal is emited
clickOnMonoplotter(customEvent):
  #x and y get north and east coordinate.
  x,y = customEvent[0], customEvent[1]

  [...]
  #do various check depending on the layer selected and the tool in used

          #transform the North East to screen coordinates in the QGIS canvas
          u,v = self.WorldToPixelOfCanvasCoordinates(x,y)
          #Create a virtual click at the projected position in the canvas
          QMouseEvent(QEvent.MouseButtonRelease, [u,v], button, button, Qt.NoModifier)
          # Emit the click signal on the QGIS canvas at the projected location.
          self.currentTool.canvasReleaseEvent(event)
```

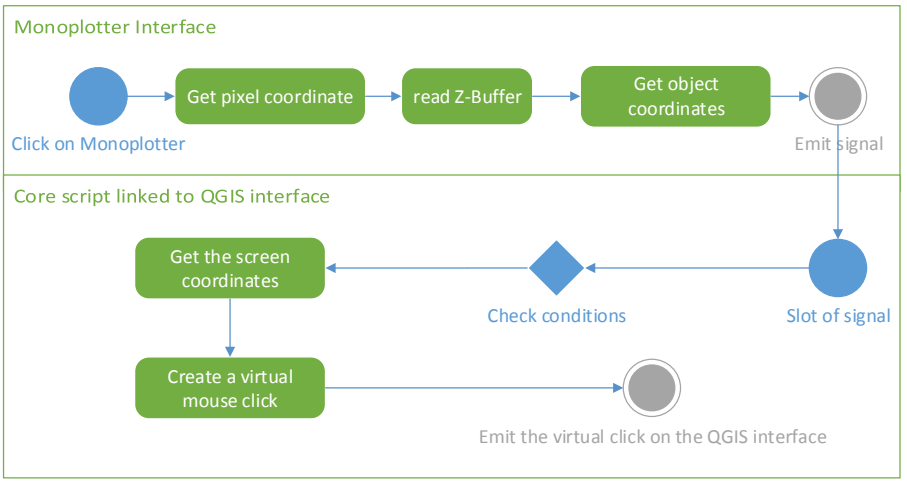


FIGURE C.1: Représentation schématique de l'évènement "Clic" lors de la digitalisation de couche dans le monoplotter

Bibliographie

[1] Kalantari, Kasser, et al. Photogrammétrie et vision par ordinateur. *Revue XYZ*, pages 49–54, 2008.

[2] Lemmens. *Geo-information : Technologies, Applications and the Environment.* Geotechnologies and the Environment. Springer, 2011.

[3] Wolf, Dewitt, and Wilkinson. *Elements of Photogrammetry with Applications in GIS.* Mac Graw Hill Education, 4th edition, 2014.

[4] Radwan and Makarovic. Digital mono-plotting system-improvements and tests. *ITC-Journal (Netherlands)*, 1980.

[5] Stokes. A photogrammetric monoplotter for digital map revision using the image processing system gop-300. *Int. Archives Photogramm. Remote Sensing*, 27 :B2, 1988.

[6] Jauregui, Vílchez, and Chacón. A procedure for map updating using digital monoplotting. *Computers and geosciences*, 28(4) :513–523, 2002.

[7] Makarovic. Digital mono-plotters. *ITC Journal*, 4 :583–599, 1973.

[8] Produit and Tuia. An open tool to register landscape oblique images and generate their synthetic models. In *Open Source Geospatial Research and Education Symposium (OGRS)*, 2012.

[9] Helpke. State-of-the-art of digital photogrammetric workstations for topographic applications. *Photogrammetric Engineering and Remote Sensing*, 61(1) :49–56, 1995.

[10] Loodts. From large-scale dtm extraction to feature extraction. In *Automatic Extraction of Man-Made Objects from Aerial and Space Images (II)*, pages 53–62. Springer, 1997.

[11] Bozzini, Conedera, and Krebs. A new monoplotting tool to extract georeferenced vector data and orthorectified raster data from oblique non-metric photographs. *International Journal of Heritage in the Digital Era*, 1(3) :499–518, 2012.

[12] Steiner. Reconstruction of glacier states from geo-referenced, historical postcards. Master's thesis, ETHZ, 2011.

[13] Corpataux. Cinématique des glaciers rocheux déstabilisés : le cas de tsaté-moiry. Master's thesis, UNIL, 2014.

[14] ITC. *ILWIS 3.0 Academic, User's Guide*. International Institute for Aerospace Survey and Earth Sciences (ITC), 2001.

[15] Doytsher and Hall. Fortran programs for coordinate resection using an oblique photograph and high-resolution dtm. *Computers & Geosciences*, 21(7) :895–905, 1995.

[16] Dumont, Sirguey, Arnaud, and Six. Monitoring spatial and temporal variations of surface albedo on saint sorlin glacier (french alps) using terrestrial photography. *The Cryosphere*, pages 759–771, 2011.

[17] Corripio. Snow surface albedo estimation using terrestrial photography. *International journal of remote sensing*, 25(24) :5705–5729, 2004.

[18] Fluehler, Niederost, and Akca. Development of an educational software system for the digital monoplotting, 2005. Eidgenössische Technische Hochschule Zürich, Institute of Geodesy and Photogrammetry.

[19] Luhmann, Robson, and Kyle. *Close Range Photogrammetry : Principles, Techniques and Applications*. Wiley, 2006.

[20] Belin, Cambier, Nabors, and Ratliff. Par corneal topography system (par cts) : the clinical application of close-range photogrammetry. *Optometry & Vision Science*, 72(11) :828–837, 1995.

[21] Koch and Heipke. Semantically correct 2.5 d gis data—the integration of a dtm and topographic vector data. *ISPRS journal of photogrammetry and remote sensing*, 61 (1) :23–32, 2006.

[22] Landes, Grussenmeyer, and Boulaassal. Les principes fondamentaux de la lasergrammétrie terrestre : acquisition, traitement des données et applications (partie 2/2). *XYZ*, (129) :25–38, 2011.

[23] Ressla, Haringa, Briesea, and Rottensteinerb. A concept for adaptive mono-plotting using images and laserscanner data. In *Proc. Symposium of ISPRS Commission III-Photogrammetric Computer Vision-PCV*, volume 6, pages 1682–1750, 2006.

[24] Jáuregui, White, Woodward, and Leitch. Noncontact photogrammetric measurement of vertical bridge deflection. *Journal of Bridge Engineering*, 8(4) :212–222, 2003.

[25] Mitishita, Machado, Habib, and Gonçalves. 3d monocular restitution applied to small format digital airphoto and laser scanner data. In *Proceedings of Commission III, ISPRS Congress, Istanbul*. Citeseer, 2004.

[26] Abdel-Azizand Karara. Direct linear transformation from comparator coordinates into object space coordinates in close-range photogrammetry. *In ASP Symp. on Close Range Photogrammetry*, 1971.

[27] Shih and Faig. Physical interpretation of the extended dlt-model. In *Proc. ASPRS Fall Convention, American Society for Photogrammetry and Remote Sensing, Reno, Nevada*, pages 385–394, 1987.

[28] Liu, Huang, and Faugeras. Determination of camera location from 2-d to 3-d line and point correspondences. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 12(1) :28–37, 1990.

[29] Förstner. New orientation procedures. *International Archives of Photogrammetry and Remote Sensing*, 33(part 3) :297–304, 2000.

[30] Huang. *Calibration and orientation of cameras in computer vision*. Springer, 2001.

[31] Lepetit, Moreno-Noguer, and Fua. Epnp : An accurate o (n) solution to the pnp problem. *International journal of computer vision*, 81(2) :155–166, 2009.

[32] Bujnak, Kukelova, and Pajdla. New efficient solution to the absolute pose problem for camera with unknown focal length and radial distortion. In *Computer Vision–ACCV 2010*, pages 11–24. Springer, 2011.

[33] Henri. The levenberg-marquardt method for nonlinear least squares curve fitting problems, 2013. Cours notes Department of Civil and Environmental Engineering Duke University.

[34] Merminod. *The Use of Kalman Filters in GPS Navigation*. PhD thesis, University of New South Wales, Kensington, 1989.

[35] Jianchao and Chern. Comparison of newton-gauss with levenberg-marquardt algorithm for space resection. In *Paper presented at the 22nd Asian Conference on Remote Sensing*, volume 5, page 9, 2001.

[36] Snyder. *Map projections–a working manual*. Geological Survey Bulletin Series. U.S. G.P.O., 1987.

[37] McGlone, Mikhail, Bethel, Mullen, American Society for Photogrammetry, and Remote Sensing. *Manual of photogrammetry*. American Society for Photogrammetry and Remote Sensing, 2004.

[38] Downey. *Think Python*. O'Reilly Media, 2012.

[39] Thelin. *Foundations of Qt Development*. Expert's Voice in Open Source. Apress, 2007.

[40] Martz. *OpenGL Distilled*. OpenGL. Pearson Education, 2006.

[41] Boreskov and Shikin. *Computer Graphics : From Pixels to Programmable Graphics Hardware*. Chapman & Hall/CRC Computer Graphics, Geometric Modeling, and Animation Series. Taylor & Francis, 2013.

[42] Neteler, Raghavan, et al. Advances in free software geographic information systems. *Journal of Informatics*, 3(2), 2006.

[43] Schowengerdt. *Remote Sensing : Models and Methods for Image Processing*. Elsevier Science, 2006.

[44] Hatze. High-precision three-dimensional photogrammetric calibration and object space reconstruction using a modified dlt-approach. *Journal of biomechanics*, 21 (7) :533–538, 1988.

[45] Fryer and Brown. Lens distortion for close-range photogrammetry. *Photogrammetric engineering and remote sensing*, 52(1) :51–58, 1986.