MicMac V1/V2 état d'avancement

Ajustement de faisceaux

Marc Pierrot Deseilligny

Univ. Gustave Eiffel – IGN/ENSG, LaSTIG lab.- France

novembre 2022

MicMac V1 et V2

Quelques jalons(1)

2003 code d'appariement d'images pour les MNS/MNT : multi-échelle, multi-images ; pour l'orientation utilise `OriLib`, modèle `Grille`;

2005 ajout de méthode d'orientation et d'ajustement pour la calibration géométrique des caméras numériques du LoEMI;

2007 intégration des point SIFT comme points homologues , pipeline d'orientation automatique à partir d'un "paquet" d'images:

2008 début d'une collaboration structurante avec le MAP-CNRS dans le domaine du patrimoine;

2009 intégration des modèle fisheye (+- "tous" les modèles)

2010 diffusion assez large de MicMac dans différentes communautés scientifiques (patrimoine, environnement)

Quelques jalons(2)

2011 pipeline de calcul de modèles en "vrai 3D" en 100% automatique , (i.e, pas du 2.5 d):

2012 début d'une collaboration structurante en science de la terre avec l'IPGP;

2013 portage sous Windows;

2015 prise en compte de l'ajustement de faisceau sur des capteurs satellitte;

2018 . . . décision de lancer une deuxième version V_2.

2020 début de recherches utilisant le deep learning dans V_1;

Bilan-1 Points positifs

1. une chaine photogrammétrique libre open source complète (satellite, aérien, terrestre)
2. une chaine photogrammétrique centrée sur la métrologie, offrant un contrôle fin des étapes de calcul (!= boite noire);
3. un outil qui a été largement utilisé à l'IGN (calcul de MNS France par les services de production, études à IGN-Espace, apprentissage de la photogrammétrie à l'ENSG, prestation en photogra terrestre par le SGM, utilisation au Matis/Lastig);
4. a été largement utilisé par des scientifiques, ingénieur . . . dans le patrimoine et l'environnement (2010-2016) ;
5. a été financé par plusieurs acteurs privés ou public (4 thèses industrielles, 1 ANR, 1 FUI, projet CNES-TOSCA).

Bilan -2 , points negatifs

Un outil qui s'est dévloppé depuis 15 ans, sans réelle stratégie globale, et pour l'essentiel avec un seul programmeur pour le noyau. Conséquences :

1. une chaine sans interface et compliquée à utiliser : noms de commandes sans rapport avec ce qu'elles font, messages d'erreur peu clairs;
2. un code peu ou mal documenté; pas de test unitaire, fonctionnel ...
3. des choix de conception complexes, notamment des optimisations (CPU, mémoire) moins justifiées avec le matériel actuel; le gros de la librairie est orientée traitement d'image, alors que l'usage majoritaire est photogrammétrique;
4. utilise peu de librairies externes, beaucoup de "home made" sur des fonctionnalité courante (lecture tiff maison, certaines interface "attaquent" directemnt XLib);
5. des contribution externes réalisées par des contractuels, pas facile à maintenir;
6. une chaine "viellissante" à l'ère du deep-learning, big data ...

Décisions

MicMac V1 ne pourra pas évoluer sur le long terme :

1. soit on "abandonne" à moyen terme, en se limitant à une maintenance currative;
2. soit on fait une nouvelle version.

Décisions

MicMac V1 ne pourra pas évoluer sur le long terme :

1. soit on "abandonne" à moyen terme, en se limitant à une maintenance currative;
2. soit on fait une nouvelle version.

Il y a un interêt a avoir une chaine photogrammetrique open source complète développée à l'IGN En faisant une deuxième itération grace au recul, on espère reprendre les bonnes idées et éviter les ecueil de V1.

1. on n'abandonne pas et on fait une nouvelle version!

Ulisateurs prioritaires

Utilisateurs prioritaires :

1. étudiants et enseignants en photogrammétrie, notamment ENSG;
2. chercheurs en STIC utilisant la photogrammétrie, notamment au LaSTIG, CEREMA, UGE ...
3. expert/ingénieur en photogrammétrie terrestre, aérienne et spatiale, notamment à l'IGN et dans le cadre de collaborations formalisées(CERN).

$\rightarrow$ Les utilisations plus "grands publics" pour avoir des modèles visuellement plaisants ne sont pas une priorité, mais pourront se développer dans le cadre de collaborations (ex développement pour les archives dans le cadre du stage d'Alexane NGhien).

Les "interfaces"

1. **programmeur**, au niveau du code lui-même pour les programmeurs, avec objectif d'avoir une qualité de code et de documentation le permettant;
2. **expert**, au niveau ligne de commandes : noms rationnels, complétions automatique, spécifications des commandes (entrée, sortie, objectif);
3. **étudiant**, vCommand systématisée
4. **expert,étudiant**, binding python permettant un accès relativement simple, à "grain plus fin" que les commandes;
5. **"grand public"**, dans le cadre de collaboration extérieur, interface finalisée, contacts pris avec Map-CNRS et meshroom.

Organisation : table rase . . .

1. évolution progressive incompatible avec les ambitions de correction des problèmes
2. on part "from scratch"

Organisation : table rase . . .

1. évolution progressive incompatible avec les ambitions de correction des problèmes
2. on part "from scratch"

****

. . . ou presque, il y a provisoirement des liens avec V1 :

1. on link avec V1, qui est utilisé comme librairie externe pour quelques fonctionnalités avant que l'on choisisse la "bonne" lib (essentiellement lecture/ecriture des images, gestion des systèmes de projection);
2. on peut importer des données V1 au format V2 pour prototyper des chaines complètes ; par exemple import d'orientation initiale, calcul de points homoloques ;
3. on peut faire du copié-collé de code sur des petits morceaux autonomes (marginal).

Organisation : équipe . . .

Actuellement un peu plus de 2 ETP . A gros traits :

1. Mehdi Daakir, dans le cadre du CERN, test Window, développemet de tests fonctionnels;
2. Celestin Huet, dans le cadre du CERN, développements spécifiques, ajout de librairies externes;
3. Yann Meneroux, détection de cibles pour PRISMA
4. Christophe Meynard, informatique (portage windows, git, vcommand et complétion), dérivée formelle (avec MPD);
5. Jean Michaël Muller, binding python, ajout de la topographie;
6. Marc Pierrot Deseilligny, développement du noyau;
7. Ewelina Rupnik, estimation de pose initiale rapide et précise.

Plus un ingénieur de recherche (Christian Staron) travaillant dans `MMVII` dans le cadre d'une collaboration IGN/CNES-Tosca/IPGP.

Etat d'avancement, général

1. noyau (70% ??) : classe pour les commandes, la gestion de chantier, la serialization, les dérivées partielles, l'optimisation non linéaire,
2. portage windows, ok en interne, premier beta testeur CERN en janvier;
3. vCommand et complétion, premier proto, en beta;
4. binding python : premiers exercices avec les PPMD en 2023;
5. tests unitaires : 80%
6. tests fonctionnel : 40%
7. documentation algorithmique (60% ??) de l'existant
8. documentation programmeur à haut niveau (50% ??) de l'existant
9. documentation programmeur dans le code (70%) .

Etat d'avancement, thématique

1. détection de cible codées, pour l'IGN et le CERN : 90%;
2. classe capteur avec perspective central quasi complet, incluant plusieur modèle de fisheyes et de nombreux modèle de distorsion;
3. estimation de poses initiales quelques algorithmes : relèvement dans l'espace (calibré ou non), matrice essentielle;
4. ajustement de faisceaux cas conique : assimilation de : points d'appuis, points de liaison, contraintes de bloc rigide, inclinomètres (70%) ; possibilité de contrainte dans les équations (i.e. optimisation sous contrainte stricte);
5. quelques développements prototype pour intégrer de la topo;
6. ajustement de faisceau cas spatial ($30 - 50\%??$) : gestion des RPC, gestion des changement de coordonnées, calcul de modèle ajusté purement $2d$; reste à faire : gestion de modèle initiaux physique (grilles ? modèles type "airbus") , gestion de corrections "physiques" $3d$ (nécessite discussion et collaboration avec le service de l'information spatiale);
7. développement de surface (cas motivé par une application concrète) : quelque modèle d'égalisation de la radiométrie (purement paramétrique).

Pipeline photogrammétrique

Le pipeline photogrammétrique est classiquement divisé en 3 étapes photogrammétriques et le reste :

1. calcul de mesure images éparses : points homologues , points d'appuis (détection de cible)
2. calcul de la géométrie des caméras : estimation initiale (cas conique terrestre) et ajustement ;
3. calcul de la scène $3D$: appariement dense contraint par la géométrie;
4. égalisation radiométrique ;
5. géneration du produit final : de nuage de points, de modèle numérique de surface, d'ortho photo, de mesh texturé . . .

Le point le plus avancé est le calcul de la géométrie, les autres points sont inexistant ou à l'état d'ébauche.

"Stratégie"

La ligne directrice est :

- avoir prioritairement une solution pour le calcul d'orientation (coeur de métier IGN) , éventuellement la topométrie ;
- intégrer d'abord des solutions libre externes pour le calcul d'appariement éparse et dense;
- compléter avec des solution interne pour : (1) être maitre de son destin (2) intégrer des résultat de recherches (par ex thèse de Mohamed Ali Chebbi);
- externe pour l'appariement éparse : SIFT + solutions basées sur l'apprentissage + méthodes "maison" (meilleure prise en compte des a priori) ?
- externe pour l'appariement dense, notamment des solution basée sur l'apprentissage fonctionnant en épipolaire (MMVII assurant toute la machinerie complémentaire en amont et en aval), puis méthodes "maison" (géométrie terrain, multi-image).

Organisation du code existant

Outils utilisés

- langage de développement principal `C++`, version minimale `C++17`
- compilateur supporté actuellement `g++, clang, MSVC++`
- portage testé régulièrement sur Linux et Windows, ca marche pour l'instant sous MacOS;
- automatisation de la compilation par cmake (3.15 au moins);
- gestion de version par `git` ;
- binding python (prototype) avec `pybind-11`;
- documentation élémentaire avec `doxygen`;
- documentation plus "haut niveau" avec un document `Latex`.

Bibliothèques Utilisées

On essaye d'éviter les deux extrêmes : MicMac-V1 *tout à la main*, outil comme Orpheo-too box : *reposer sur les épaules de géants* . Les deux posent des problèmes de maintenance à long terme . . . Bibliothèques utilisées :

- Eigen : algèbre matricielle;
- pybind11 : binding python;
- GDAL : lecture/écriture des images (en cours);
- PROJX : système de coordonnées (à venir/V1 pour l'instant);
- Xif... : méta-données des images (à venir/V1 pour l'instant);
- quelques code open source composé de fichier directement inclus dans MMVII (triangulation de delaunay Volodymyr Bilonenko-MIT, gestion du format ply Nick Sharp-MIT) ;

Bibliothèques non utilisées

Peut-être utilisé plus tard :

- CGAL pour la triangulation 2D-3D la tetrahédrization (mais problème de licence);

A priori Non utilisé :

- bibliothèque de traitement d'image
- boost (la librairie standard contient à peu près tout ce dont on a besoin);

Précautions logicielles(1)

Tout en restant du code de recherche, fait avec les moyen du bord, un certain nombre de précautions sont prises pour limiter les bugs :

- une variable d'environnement `The_MMVII_DebugLevel` fixe le niveau de vérification;
- à son plus haut niveau : tous les accès aux images sont vérifiés (débordement d'indice, de valeur) , les division par 0 , les racine carrés négatives . . .
- à son plus bas niveau seules les vérifications liées aux erreurs utilisateurs sont effectuées;

Pour la gestion mémoire les classes peuvent dériver de `cMemCheck` , dans ce cas (si on est en mode DEBUG):

- une vérification empirirque de non débordement en écriture sera faire à la destruction de l'objet;
- une vérification de la libération sera faite à la fin du process.

Précautions logicielles(2)

Un programme , appelé abusivement `BENCH`, effectue des test unitaires assez "complets", par exemple :

- `MMVII Bench 1` lance les test sur toute les fonctionnalité;
- `MMVII Bench 200 PatBench=Geom` lance 200 fois les test sur les fonctionnalité géometrique, avec des configurations aléatoires différentes et . . . plante à l'itération 36

Plusieurs ensembles de commande s'executant consécutivement sont présents dans le dossier `MMVII-UseCaseDataSet`, en plus d'être des exemples didactiques, ce sont des précurseurs de test fonctionnels.

Quelques conventions/habitudes/styles

- on essaye de réduire la taille des header et la taille du code compilé : pas de *grosse* méthode *inline*, instantiation explicite des *grosse* méthodes template dans les cpp;
- pour les classes applications, le code de dérivation automatique, on passe par des allocateur qui évitent d'avoir à exporter tous les header (voir présentation suivantes);
- les header qui contiennet *beaucoup* de code *inline* s'appellent génélement `*Tpl*` ;
- les données membre commencent généralement par `m` et les variable par `a`, les noms de classes par `c`, les `typedef` par `t`, les classe enum par `e`,
- header sont essentiellement tous à plat sous `include`;
- quelques headers spécifiques à un ensemble de source sont au même niveau que les sources (exemple `src/Sensors/cExternalSensor.h`);
- tous les sources sont au même niveau `src/AA/BB.ccp` sauf le `src/main.cpp`.

Les formats MMVII (1) : dans la *jungle* des données photogrammétriques

Pas de standard permettant d'échanger simplement d'un logiciel à l'autre.
Rien qu'en interne IGN, au moins 3 familles de formats . . .
Au début de `MMVII`, réflexion sur l'utilisation du format `kapture`, format ouvert développé par `Naver-labs`, ayant des passerelle vers un certain nombre de solutions photogrammétrique. Mais un format d'échange, n'est pas forcément un bon format de travail. Format de travail doit pouvoir représenter finement toutes les étapes d'un calcul tel que modélisées par le logiciel, peu de chance qu'il puisse être représenté par un format standard.
Format interne spécifique `MMVII` : les grande lignes seront décrites dans la suite; permet pour chaque type de données de stocker autant d'état de calcul que nécessaire.
La capacité d'importer/exporter des données est considérée comme importante mais est dissociée du format interne.

Interface avec les autres formats

Les interfaces suivante existent ou sont prévues, essentiellement en import pour l'instant :

- import de donnée micmac-V1, V2 étant pour l'instant très incomplète;
- import de nombreuses type de donéees structurées sous forme *BSV* (blank separated value) ou CSV avec de commandes comme `ImportGCP`, `ImportTiePMul` , `ImportOri`;
- import d'une calibration sans distorsion avec la commande `EditCalcMTDI`;
- import d'orientation RPC `ImportInitExtSens` , autres format à venir ?
- import des données du SIA/SMD (cas particulier du BSV)
- import spécifique du format `ORGI` du `SIV` .

A moyen terme, import/export vers les solution courrantes, sans doute en se basant sur une solution type `kapture` (qui supporte COLMAP, bundler, nvm, OpenMVG, OpenSfM).

Documentation

Le point faible de tous les logiciels . . . Les ressources suivantes sont accessible :

- documentation au plus près du code avec `Doxygen` : lancer `doxygen Doxyfile`
- documentation *classique* sous forme d'un pdf à compiler à partir du Latex sous `MMVII/Doc`; ambitionne d'être une doc programmeur + utilisateur + algorithmique (cours de photogra), vaste programme . . . sans doute un quart de ce qui devrait être écrit l'a été;
- des cas d'usage sous `MMVII/MMVII-UseCaseDataSet/`;
- retour des sessions de formation : document rédigé par un ou deux stagiaires, vidéo.