Optimisation d'une chaîne de production photogrammétrique : détection automatisée de points de calage

Léa Corduri

CONSERVATOIRE NATIONAL DES ARTS ET MÉTIERS

ÉCOLE SUPÉRIEURE D'INGÉNIEURS GÉOMÈTRES ET TOPOGRAPHES

MÉMOIRE

présenté en vue d'obtenir

le DIPLÔME D'INGÉNIEUR CNAM

SPÉCIALITÉ : Géomètre et Topographe

par

CORDURI Léa

Optimisation d'une chaîne de production photogrammétrique : détection automatisée de points de calage

Soutenu le 02/09/2024

JURY

Madame SIMONETTO Elisabeth	Présidente du jury
Monsieur SEMLER Quentin	Maître de stage
Monsieur CHARLET Christophe	Enseignant référent

REMERCIEMENTS

Je tiens à remercier toutes les personnes ayant contribué à la réalisation et à la rédaction de ce mémoire.

Je voudrais tout d'abord remercier mon maître de stage, Mr SEMLER Quentin, pour m'avoir suivie tout au long de la préparation de ce travail, pour ses conseils avisés, sa disponibilité et son soutien constant.

Je tiens de même à remercier Mr HAENEL Raphaël pour son aide précieuse et son soutien indéfectible tout au long de cette aventure. Ses suggestions et son encouragement ont été d'une grande aide pour surmonter les défis rencontrés.

Je souhaite également exprimer ma gratitude à toute l'équipe de Syslor pour m'avoir offert l'opportunité de réaliser mon alternance au sein de leur entreprise. Cette expérience a été enrichissante et formatrice.

Enfin, j'adresse mes remerciements les plus sincères à ma famille et à mes amis pour leur soutien moral et leur compréhension durant cette période intense.

LISTE DES ABRÉVIATIONS

API : Interface de Programmation Applicative

BTP : Bâtiment et Travaux Publics

DT : Déclaration de Travaux

DICT : Déclaration d'Intention de Commencement de Travaux

GNSS : Système de Navigation Globale par Satellite

ICP : Iterative Closest Point

RANSAC : RANdom SAmple Consensus

YAML : Yet Another Markup Language

YOLO : You Only Look Once

GLOSSAIRE

COCO (Common Objects in Context) : Base de données importante utilisée pour entraîner des modèles neuronaux dans le contexte de l'apprentissage profond.

Centre de Certification des Technologies Avancées (CCTA) : Organisme certificateur spécialisé dans l'évaluation et la certification de technologies de pointe.

Clustering : Méthode de classification non supervisée dont le but est de regrouper des données similaires.

Erreur de reprojection : Erreur quadratique sur les différences de coordonnées des points connus en coordonnées terrain et issues de la construction du modèle 3D.

Firebase : Plateforme cloud de Google pour le développement d'applications web et mobiles, offrant des services comme l'authentification ou le stockage de données.

Framework : Structure logicielle préfabriquée qui fournit un ensemble de fonctions et d'outils permettant un usage plus efficace.

Géoréférencement direct : Méthodologie qui détermine précisément l'orientation externe des images en photogrammétrie, en utilisant des données de position et d'orientation fournies directement lors de la capture des images.

Main : Point d'entrée principal d'un programme.

Paramètres externes : Paramètres qui définissent la position et l'orientation de la caméra dans l'espace par rapport à l'objet ou à la scène à photographier. Ces paramètres permettent de transformer les coordonnées du système de la caméra en coordonnées du système global.

Paramètres internes : Englobent des variables telles que la focale de la lentille, les coefficients de distorsion et la position du centre optique, déterminant ainsi la projection géométrique des images capturées.

Photogrammétrie rapprochée : Technique de mesure et de modélisation en 3D d'objets à en utilisant des photographies acquises à courte distance. Elle est souvent employée pour des applications en architecture, archéologie et ingénierie civile.

Points de calage : Points connus en coordonnées terrain utilisés pour géoréférencer un modèle 3D et le mettre à l'échelle.

Points de contrôle : Points connus en coordonnées terrain, non utilisés lors du processus de reconstruction photogrammétrique.

Tracking GNSS : Connaissance de la localisation des images acquises pour le traitement photogrammétrique grâce à la combinaison GNSS/caméra.

Transfert Learning : Méthode en apprentissage automatique où un modèle préalablement entraîné sur une tâche est ajusté pour une nouvelle tâche similaire, permettant ainsi d'améliorer sa performance.

TABLE DES MATIERES

Introduction

La photogrammétrie est part intégrale de nombreuses réalisations allant de l'orthophotographie à la projection en réalité augmentée. Cette technique consiste en la reconstitution d'objets et de scènes à partir de plusieurs prises de vue photographique. Elle s'est développée exponentiellement depuis sa théorisation, au milieu du XIXe siècle, par le français Aimé Laussedat. Bien qu'aujourd'hui un simple smartphone permet au néophyte d'appréhender ce domaine via les nombreuses applications disponibles depuis les plateformes de distributions consacrées telles qu'Apple ou Google store.

Syslor, l'entreprise dans laquelle a pris part ce mémoire, tire parti de ces avancées technologiques en ouvrant à l'usager des applications permettant notamment le levé, l'implantation, et le géoréférencement des réseaux enterrés via un processus de photogrammétrie mobile couplée à un récepteur GNSS pour assurer une précision de géolocalisation avérée. Tournées vers un public de professionnels du BTP, ces applications se veulent simples et rapides d'utilisation et sont issues de la volonté d'améliorer la sécurité de l'opérateur à proximité du réseau en lui permettant de capturer une fouille depuis une distance raisonnable. Elles visent également à renforcer la sécurité globale en facilitant le marquage des réseaux conformément à la législation française et ainsi prévenir les dommages aux infrastructures souterraines. Elles permettent enfin de fournir les livrables nécessaires aux études du géomètre sans nécessité son déplacement ce qui lui a valu d'obtenir le label Solar Impulse décerné aux solutions écologiques et économiquement viables.

Cette simplicité apparente masque néanmoins un développement très technique, nécessitant à la fois les compétences de développeurs et les connaissances spécialisées d'ingénieurs géomètres. Les chaînes de production amenant à l'envoi des livrables demandés ont ainsi été pensées dans un souci d'optimisation des temps de traitements. De nombreux processus d'automatisation prennent place notamment à travers l'utilisation d'API logicielles.

Cependant, l'optimisation de ces chaînes de traitements reste un sujet au cœur des préoccupations de l'entreprise, qui vise à l'automatisation globale de ses procédés. Le sujet ici développé découle de la volonté d'optimiser le traitement photogrammétrique de l'application Récolement , encore aujourd'hui semi-automatique. En effet, l'utilisation généralisée, par les chefs de chantier, de marquages peinturés comme points de calage pose un problème. Bien

que se présentant comme une solution plus pérenne et économique que l'utilisation de cibles dédiées au géoréférencement, ces marquages nécessitent aujourd'hui une intervention humaine. Il sera ainsi question de trouver une réponse à la question suivante : comment peut-on se soustraire à la nécessité d'une intervention humaine lors de la reconnaissance de points de calage non paramétrés ?

Le développement visant à résoudre cette problématique sera défini en tenant compte de contraintes spécifiques, en particulier le respect de la procédure antérieure. L'objectif est de proposer une solution qui s'inscrit dans la continuité des méthodes actuelles, tout en évitant d'introduire des changements révolutionnaires susceptibles d'être perçus comme trop complexes par le client et de compromettre ainsi sa relation avec l'entreprise avec celui-ci. De plus, on considère ici l'alignement initial des caméras / photos comme étant un succès.

Afin de comprendre les objectifs réels de l'entreprise, la première partie de ce travail est consacrée à une mise en situation présentant l'état actuel de la chaîne de production, suivi d'une exploration de l'état de l'art sur le géoréférencement automatique des nuages 3D obtenus par traitement photogrammétrique. Ensuite, la démarche d'automatisation sera décomposée en deux parties : la détection des cibles à l'aide du Deep Learning, et le déploiement final au sein de l'entreprise via le logiciel Agisoft Metashape.

I. Contextualisation et exploration du sujet

I.1. État des lieux de la chaîne de production actuelle

I.1.1. La nécessité du géoréférencement des réseaux enterrés

Le géoréférencement des réseaux enterrés est une obligation en France depuis la mise en vigueur de la réforme du 15 février 2012 [1], dite réforme anti-endommagement DT - DICT. Celle-ci préconise que tout chantier réalisé à proximité de réseaux doit prévoir le marquage et la géolocalisation de ceux-ci en amont du chantier. Trois classes de précision sont ainsi mises en place selon la juste connaissance de l'emplacement des réseaux (Figure 1). Cette précision s'applique depuis la génératrice supérieure des réseaux en planimétrie, mais aussi en altimétrie. Cette génératrice correspond à la ligne fictive marquant le point le plus élevé d'un ouvrage, d'une gaine. La législation demande que l'ensemble des réseaux enterrés tendent à être connus sous une précision de classe A [1]. Depuis 2018, cela constitue une obligation pour les réseaux neufs et rénovés avec l'arrêté du 14 décembre 2017 venant modifier la réglementation fixée par l'arrêté du 15 février 2012.

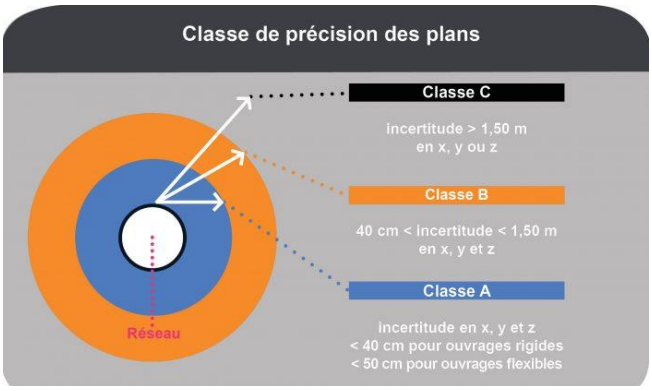


Figure 1: Classe de précision des réseaux enterrés. Source : https://sigerly.fr/detection-georeferencement-reseaux-electriques-souterrains/

Syslor dispose ainsi de solutions permettant notamment le jumelage numérique de réseaux sur la base de la photogrammétrie. Plus concrètement, l'entreprise dispose d'une solution d'acquisition vidéo, avec prise de points de calage, permettant le géoréférencement d'un nuage de points pouvant alors être la base d'un tracé de plan de récolement en classe A.

L'entreprise est donc un prestataire certifié CCTA pour la localisation des réseaux enterrés. Cette qualification est requise afin de produire des plans de récolement de classe A. Il s'agit alors de respecter certaines conditions techniques et, notamment, l'utilisation des points de calage est impérative au processus photogrammétrique pour atteindre cette précision selon le fascicule 2 du guide technique de cette réglementation [2]. En effet, il est stipulé en son sein, concernant l'utilisation de la photogrammétrie comme base aux mesures : « Le

géoréférencement du nuage de points corrélé à partir de ces images revient à déterminer les paramètres d'orientation externe du bloc d'images. Cette étape est essentielle pour des relevés de récolement classe A. Il est impératif de disposer de points de référence comme des piquets de part et d'autre de la tranchée avec mesurage GNSS et/ou tachéométrique de ces points visibles sur plusieurs images. ». Cette prescription est rappelée comme étant d'ordre obligatoire à l'utilisation de cette technique.

I.1.2. L'application récolement vidéo et sa chaîne de traitement

L'application de récolement vidéo est au cœur de notre problématique d'optimisation. Son objectif est de créer un nuage de points géoréférencé à partir d'une acquisition vidéo et d'un traitement de photogrammétrie rapprochée. Cette section détaille le processus intégral de son utilisation, de l'acquisition sur le terrain aux rendus finaux, en passant par le traitement.

I.1.2.1. Récolement vidéo : acquisition terrain

L'opérateur réalise une vidéo de l'objet qu'il souhaite reproduire en ayant au préalable pris attention de placer et relever les points de calage qui permettront le géoréférencement de l'ensemble. Ces points sont relevés dans la majorité des cas au moyen d'un récepteur GNSS directement par l'opérateur, le plus souvent chef de chantier, mais peuvent aussi faire l'objet d'un relevé a posteriori en cas de présence de masques perturbant la récupération des signaux GNSS. Plusieurs méthodes ici sont possibles : l'utilisation de cibles Metashape ou par matérialisation au sol comme observé sur la Figure 2.

Bien évidemment, l'acquisition par biais photogrammétrique amène à un certain nombre de prérequis à respecter afin que le produit final soit de qualité acceptable. Dans le cas type d'un relevé de tranchée, l'opérateur devra veiller à respecter certaines recommandations. Ainsi, pour maximiser les chances d'un recouvrement optimal, il est demandé à l'opérateur réalisant la vidéo de ne pas trop s'éloigner de la tranchée et de rester à maximum 5 à 7 mètres de l'objet avec une vitesse de marche lente, d'environ 3 km/h, et constante sans pauses ni arrêts sauf en cas de coupure vidéo afin de sécuriser le passage au-dessus de la tranchée. Les repères de calage doivent aussi être visualisables depuis les deux côtés de la tranchée dans l'idéal. Un aller-retour est donc recommandé comme nous le voyons

en bleu sur la Figure 2. Enfin, l'objet d'acquisition doit être fixe. Ainsi le chantier, lors de la prise vidéo, doit ainsi être à l'arrêt, les ouvriers hors de la tranchée.

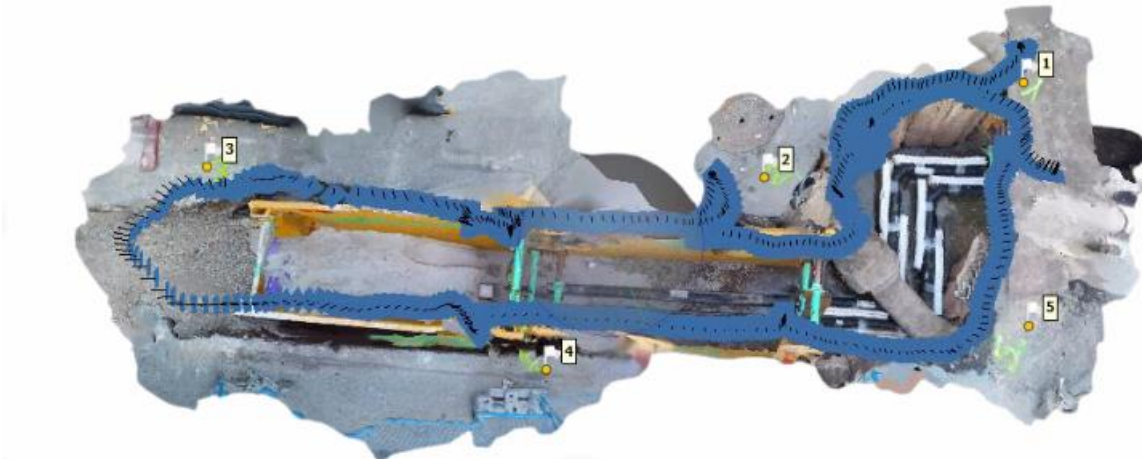


Figure 2 - Capture issue de Metashape de l'orthophoto du nuage 24XXXXX10632 avec placement des caméras en bleu et visualisation des repères de calage de 1 à 5.

I.1.2.2. Récolement vidéo : traitement photogrammétrique

À la suite de cette acquisition, la vidéo est envoyée via l'application et réceptionnée sur les serveurs de l'entreprise. Un script télécharge automatiquement la vidéo depuis la Firebase et crée un dossier de traitement. Le traitement est réalisé semi-automatiquement, chaque étape, présentées avec la Figure 3, pouvant être lancée indépendamment via une interface en cas de besoin.

```
Alignement(9),Densecloud(10),Models(11),Exports(12),Post(13),Potree(14)
```

Figure 3 - Étapes semi-automatiques du script de traitement Metashape issues du model_manager. Source : Syslor.

Une intervention humaine est ainsi nécessaire au moment du géoréférencement à la fin de l'étape 9 comme schématisé sur la Figure 4, particulièrement en cas d'utilisation de marquages peints en tant que points de calage.

Les traitements et les rendus sont renvoyés sous 48 heures. La phase de traitement photogrammétrique de chacun de ces chantiers est estimée entre 2 à 3 heures. Le pointage et l'optimisation durent, quant à eux, environ 40 minutes.

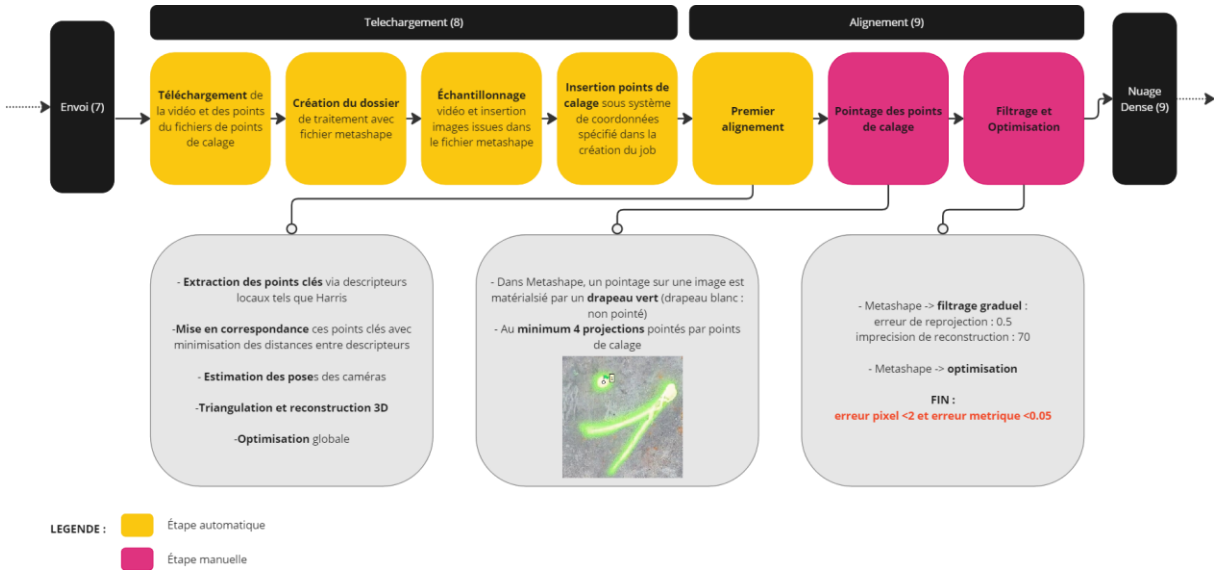


Figure 4 - Détails des étapes 8 et 9 du processus de traitement semi-automatique de l'application récolement vidéo via Metashape

I.1.3. Pistes d'amélioration : le pointage des points de calage.

L'application étant destinée aux chefs de chantier, ceux-ci privilégient la rapidité et l'efficacité. Ainsi, une grande majorité de cette clientèle tend à ne pas utiliser de cibles pré-codées, ce qui aurait permis à la chaîne de traitement d'être automatisée dans son entièreté. Ils matérialisent alors les points de calage avec de la bombe de peinture. Ces marquages sont adjoints d'un numéro permettant leur différenciation.

Figure 5 - Matérialisation à la bombe de peinture de points de calage. Source : Syslor, chantier n° 24XXXXX09471, 24XXXXX10379, 24XXXXX10318 (de gauche à droite)

Ils se présentent essentiellement sous deux formes : un point ou une croix. Des exemples sont visualisables sur la Figure 5. La peinture utilisée est quant à elle de couleur variable, parfois également au sein d'un même chantier.

Ce projet utilisera donc les données collectées sur les chantiers clients afin de développer un script automatisant le pointage de ces repères peints.

Le logiciel interne utilisé est Agisoft Metashape. Son API, nous le verrons plus tard, permet la liaison logiciel – script Python [2](1).

L'ensemble des intégrations Python doivent pouvoir fonctionner sur la version 3.8 utilisée par les développements de l'entreprise.

I.2. État de l'art

I.2.1. Le géoréférencement des chantiers photogrammétriques

Le géoréférencement (2) est le procédé permettant le positionnement d'un objet dans un système de coordonnées. Ainsi, lors de la reconstitution d'un nuage de points à partir de photographies, sans coordonnées propres, celui-ci sera dans un repère local. Ce n'est que lors de la détermination de points connus en coordonnées et pointés dans le nuage, dits points de calage, que le nuage sera référencé dans leur système.

Les points de calages permettent de déterminer les paramètres intrinsèques de la caméra, de mettre à l'échelle le modèle et peuvent être utilisés comme points homologues permettant d'affiner le placement des caméras.

Ils peuvent être matérialisés sur le terrain, par des cibles ou des marquages. On distingue alors les cibles non paramétrées, des cibles paramétrées.

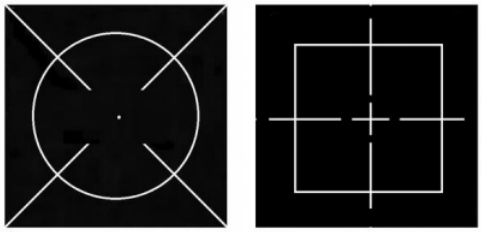
I.2.1.1. Les cibles et repères non paramétrées

Les cibles non paramétrées peuvent être des marquages mis en place sur le chantier par l'opérateur et pouvant être identifiées facilement, mais aussi de véritables cibles sans identification propre comme celles présentent sur la Figure 6.

Certains chantiers, nécessitant une reconstitution plus esthétique, notamment, préfèrent utiliser des éléments déterminants du paysage afin que les rendus ne soient pas entachés par les marquages. On parle ici par exemple de la réalisation de jumeaux numériques de monuments historiques.

Ces repères et cibles non paramétrés ont plusieurs avantages. Concernant les marquages ou les éléments du paysage, leur pérennité est assurée et leur mise en place sur le terrain rapide. Quant aux cibles non paramétrées, elles sont facilement discernables.

Néanmoins un désavantage certain est constaté. En effet, ces cibles et marquages nécessitent un alignement préalable des photos afin de rendre leur identification possible par mise en correspondance de leurs coordonnées reprojetées et connues (partie III.2).



On note que Agisoft Metashape permet la détection de cibles non paramétrées telles que la Figure 6. Leur couleur d'impression n'est alors pas déterminante, seul un bon contraste est nécessaire.

Figure 6 - Cibles non paramétrées utilisées par Metashape. Source : [3]

I.2.1.2. Les cibles pré-paramétrées et méthode assimilée

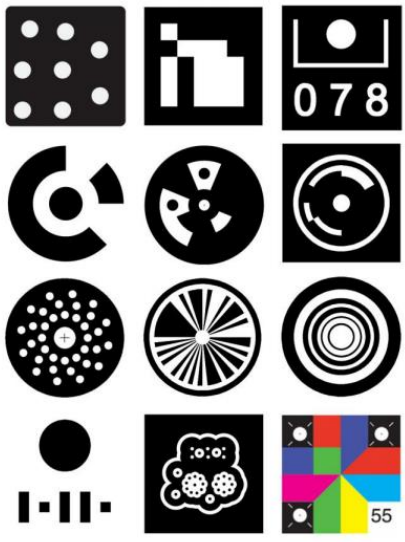


Fig. 1. Various types of coded targets used in industry

Figure 7- Exemples de différentes cibles codées pouvant être utilisée lors d'un géoréférencement. Source : [4]

Les cibles pré-paramétrées, quant à elles, permettent un pointage automatique presque aussi précis que manuellement. En effet, le pointage manuel des cibles peut amener des erreurs de reprojection de l'ordre de 0.5 pixel contre 1 à 3 pixels pour une cible paramétrée. On peut visualiser les différents types de cibles paramétrées utilisées dans le domaine de la photogrammétrie Figure 7. Celles-ci représentent essentiellement des code-barres circulaires, mais l'on peut aussi retrouver, en haut à gauche par exemple, des cibles de type QR code ou encore, en haut à

Figure 8 - Cible circulaire 12 bits pointé automatiquement dans Metashape. Source : Syslor.

droite, des cercles de pointé adossés de chiffres numériques. Toutes ces cibles disposent alors d'un centre de référence, ou centre de pointé, qui peut être défini comme le centre d'un cercle, d'un carré, ou encore d'une croix.

Agisoft Metashape permet l'utilisation de certaines de ces cibles telles que notamment les cibles circulaires 12 bits (Figure 8) [2]. Leur détection repose principalement sur des algorithmes de traitement d'images. Les images sont généralement binarisées en noir et blanc avant d'en extraire les contours, de déterminer les régions candidates et enfin les correspondances géométriques.

En marge de ces usages dits traditionnels, dits de géoréférencement indirect, d'autres techniques permettent le géoréférencement d'un nuage de point dès l'acquisition, un géoréférencement direct.

On parle ici particulièrement de la connaissance en amont de certaines données : les paramètres internes et externes des caméras. Ces informations peuvent permettre géoréférencement direct via notamment l'utilisation d'un GNSS. Plusieurs applicatifs, principalement sur drone, sont disponibles aujourd'hui. On parle ici par exemple des solutions RealityCapture (3) ou encore Pix4DMapper (4).

Syslor a développé une application similaire sur smartphone: EasyScan. En combinant acquisition vidéo et suivi GNSS, cette application permet de reconstruire un modèle 3D géoréférencé avec une précision globale inférieure à 15 cm. Cependant cette utilisation reste imparfaite dans le contexte de la localisation des réseaux enterrés qui oblige l'utilisation de points de contrôle comme nous l'avons défini dans la partie I.1.1. Ainsi, ce travail pourrait également bénéficier à cet outil.

Tableau 1 – Synthèse des avantages et inconvénients des méthodes de géoréférencement vues dans ce chapitre.

TYPE DE GÉORÉFÉRENCEMENT	AVANTAGES	INCONVÉNIENTS
ELEMENTS DU PAYSAGE	• Précis (0.5 pixels) • Pérennes • Conditions d'acquisition optimales non déterminante dans leur détection (œil humain)	• Pointage manuel
CIBLES NON PARAMÉTRÉES	• Caractéristiques spécifiques • Détection automatiques	• Détection post alignement : n'aide pas à celui-ci • Moins précis (1 à 3 pixels)
CIBLES PARAMETREES	• Détection automatique • Caractéristiques spécifiques • Détection et identification après import des images: aide à l'alignement	• Moins précis (1 à 3 pixels)
GÉOREFÉRENCEMENT DIRECT	• Géoréférencement immédiat • Conditions d'éclairage / de capture optimales compensées • Pas de détection nécessaire	• Moins précis (<15cm) • Calibration préalable recommandé • Ne suis pas les directives de la réglementation sur la localisation des réseaux

I.2.2. Le Deep Learning comme solution à la détection de points de calage

Le Deep Learning, ou apprentissage profond en français, est un sous-domaine du Machine Learning, lui-même issu du domaine de l'intelligence artificielle. L'objectif de l'apprentissage profond est alors d'entraîner un ordinateur à penser comme le pourrait un humain. En effet, il est aujourd'hui possible à un ordinateur de reconnaître des visages, d'imiter des voix, de compléter une phrase, d'anticiper le développement de codes, etc.

Tous ces développements ont été possibles grâce à la reprise, dans les années 2000, du connexionnisme ayant vu le jour dans les années 60 et ayant pour objectif de modéliser les comportements humains à travers l'interconnexion de réseaux d'unités simples (5). L'apprentissage profond reprend donc ce principe en se basant sur le modèle neuronal humain et en créant des modèles d'apprentissage complexes via des couches neuronales imbriquées, interconnectées.

Deux grandes méthodes d'apprentissage sont définies suivant les données d'entraînement introduites au sein de ces réseaux : l'apprentissage supervisé et l'apprentissage non supervisé. L'apprentissage supervisé demande obligatoirement une annotation manuelle des données afin de guider la machine. L'apprentissage non supervisé, quant à lui, ne nécessite pas d'intervention humaine outre le choix des données et est plutôt basé sur des prédictions statistiques. On retiendra néanmoins que la quantité de données d'entraînement nécessaire à l'apprentissage supervisé est moindre vis-à-vis de l'apprentissage non supervisé.

Étant ici question de géoréférencer des modèles 3D créés via photogrammétrie, l'idée d'utiliser l'apprentissage profond comme solution à la détection des points de calage a été vue comme une solution intéressante, notamment pour pallier l'utilisation de cibles pré-paramétrées comme vu durant la partie I.2.1. Cette possibilité a d'autant plus été envisagée par l'utilisation de Metashape Agisoft qui permet déjà la détection d'objets sur des orthoimages (6). Le logiciel et son API supportent donc ce type d'utilisation.

Une détection directement sur les images alignées, similairement aux cibles non-paramétrées, mais via apprentissage profond sera alors ici développée. En effet, bien qu'une tentative d'identification préalable à l'alignement des chiffres adjoints aux marquages soit mise en avant lors de ce travail, il convient de constater qu'à ce jour, cette idée n'a pas encore été pleinement explorée.

II. La détection de cibles peintes

Dans cette partie, un premier module de détection et de pointage du centre des cibles peintes a été développé. À cet effet, un modèle d'apprentissage profond a été créé et une méthode de pointage des centres des repères a été mise au point. L'ensemble de ces deux processus nous a finalement permis de tirer des conclusions quant à la précision du pointage.

II.1. La création d'un jeu de données

II.1.1. Annotation des données

II.1.1.1 : Technique choisie : la segmentation d'instance

Pour tirer parti des données mises à disposition et maximiser les chances de réussite, l'apprentissage profond supervisé a été utilisé. L'annotation des données d'entrées a ainsi été nécessaire.

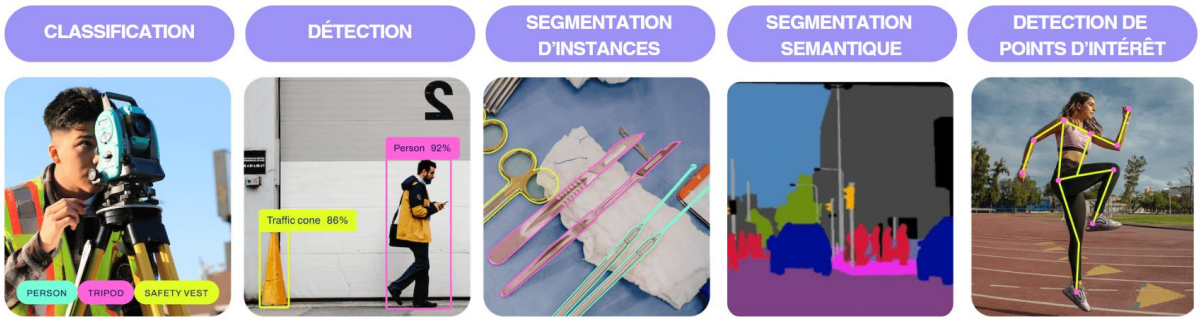


Figure 9 - Exemples de tâches d'apprentissage profond. Source des images : (6) et (9)

Plusieurs techniques de reconnaissances d'objets et donc d'annotation d'images sont disponibles (Figure 9) suivant les entrées et les sorties souhaitées : la détection d'objets, la classification, la segmentation sémantique et d'instance ou la détection de points d'intérêts. L'intention étant ici la reconnaissance et le pointage du centre des points de calages bombés sur images, la segmentation d'instance a été choisie. Cette technique permet non seulement la détection, mais également la segmentation des objets identifiés. Ainsi, les données de sortie comprennent la classique boîte englobante ainsi qu'un masque de segmentation qui nous permettra, comme nous le verrons plus tard, de déterminer le centre pointé.

Pour la segmentation d'instance, les images doivent être annotées avec des contours précis délimitant chaque instance d'objet visée. L'utilisation de polygones est essentielle pour

cette tâche, afin d'obtenir des contours détaillés permettant à l'ordinateur d'apprendre de manière plus fidèle.

II.1.2. Logiciel d'annotation CVAT

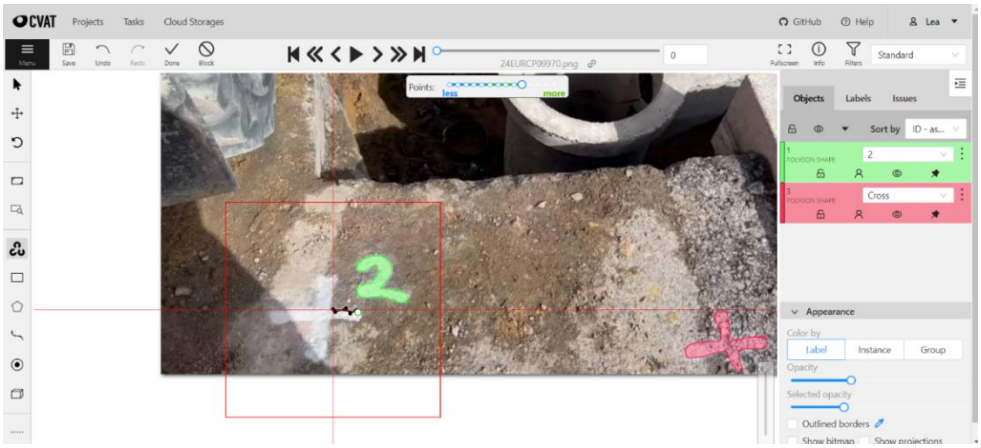


Figure 10 - Interface d'annotation CVAT. Utilisation de l'outil « Ciseau intelligent ». Source : (8)

Un logiciel d'annotation a été nécessaire. Syslor disposant d'une équipe Recherche et Développement s'étant déjà penchée sur le sujet de la détection par apprentissage profond, non des points de calage mais des réseaux enterrés, le logiciel antérieurement utilisé a été mis à disposition. Il s'agit d'un logiciel d'annotation d'images en ligne, disponible en installation locale : CVAT – Computer Vision Annotation Tool (8). Ses fonctionnalités et sa prise en main sont conviviales et intuitives. Il dispose aussi d'un outil de segmentation intelligent utilisant le module Python OpenCV, « ciseaux intelligents » (Figure 10). Ils permettent de détourer les objets au plus proche de leur contour détecté. Cet outil a largement optimisé le temps d'annotation manuel nécessaire tout en laissant la possibilité de modification des polygones en cours d'utilisation.

Le logiciel dispose en plus de nombreux formats d'exports dont les formats COCO, LabelMe, KITTI, Mask Segmentation et YOLO entre autres. Ce point est important car différents modèles d'entraînements supportent différents formats d'annotation en entrée. Plusieurs modèles d'entraînement peuvent ainsi être considérés sans avoir de difficultés liées au changement de formats d'annotation, en principe.

II.1.3. Annotations et classes finales

Initialement, 12 classes à annoter ont été définies afin d'identifier non seulement les deux principaux marquages, les croix et les points, mais aussi leurs numéros associés, de 0 à 9. Cette étape a été premièrement réalisée chantier par chantier. Cependant, pour améliorer la qualité des résultats, il a été nécessaire de diversifier les images et annotations, de même que d'homogénéiser les classes afin d'éviter au possible les biais. Le diagramme présent Figure 11 permet de visualiser cette évolution. On compte près de 6 000 instances annotées dans la version finale, *markers_and_digits_v2*.

Nous observons néanmoins, avec la Figure 11, une évidente disparité entre les classes. Elle est perceptible entre les chiffres et les marquages, mais aussi en leur sein même. Il a donc été nécessaire après un premier entraînement, comme nous le verrons dans la partie II.1.3, d'opter pour une reconnaissance a priori des marquages avant de s'intéresser à leur correspondance via leurs coordonnées reprojetées. De plus, la classe point (dot) a été exclue du fait de sa surreprésentation au sein des chantiers hors de son cas d'usage recherché. On parle ici notamment de la présence de surbrillances de soudage sur les réseaux.

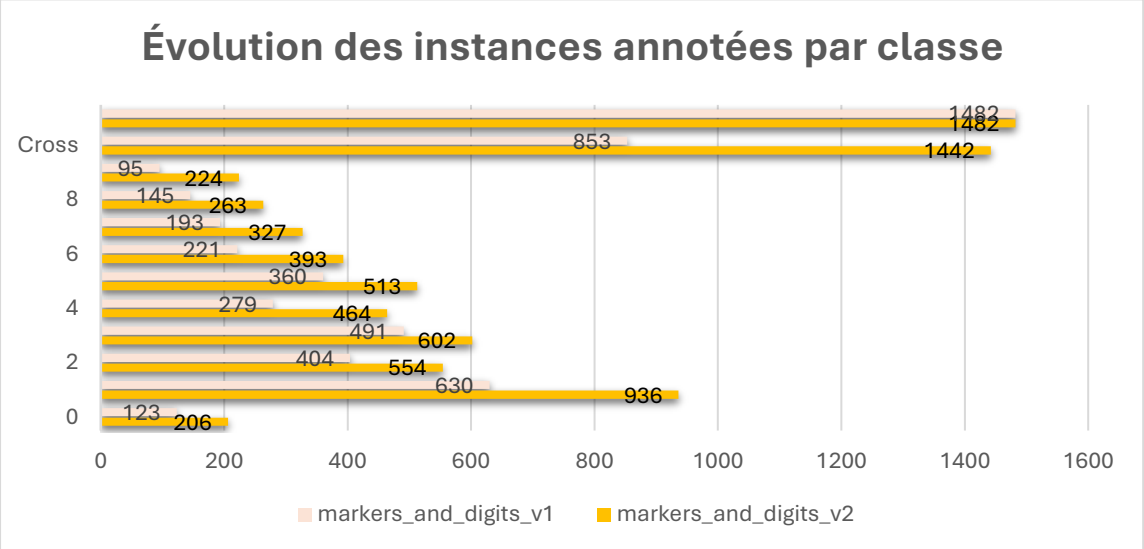


Figure 11 - Annotations initiales (markers_and_digits_v1) et finales (markers_and_digits_v2)

II.1.2. Modèle et entraînements

II.1.2.1.Le choix du modèle YOLOv8-seg-m

Un choix significatif a ici été fait, celui d'utiliser un modèle simple à prendre en main et efficace en termes de performance. Le modèle You Only Look Once (YOLO) dans sa

dernière version v8 a ainsi été retenu. Mask-RCNN aurait pu lui être préféré, étant la référence dans le domaine de la segmentation d'instance, cependant YOLO offre des atouts certains.

En effet, le framework Ultralytics (9) permet un usage simple de l'ensemble de ses modèles et permet des renvois de métriques et graphiques incidents aux entraînements et validations. Ce framework admet aussi l'ajout de paramètres d'augmentation en interne. Des modèles pré-entraînés sur une diversité de jeux de données sont aussi disponibles sur la plateforme. Enfin, l'utilisation d'un framework unique permet d'éviter les conflits de packages.

De plus, YOLOv8 présente des performances équivalentes à celles de Mask-RCNN tout en étant plus rapide. Plusieurs études démontrent cette équivalence [5][6] et, dans certains cas, mettent en évidence que les performances de YOLOv8 dépassent celles de Mask-RCNN [7].

Le modèle YOLO se distingue alors par une architecture particulière permettant une détection et segmentation en une seule étape, contrairement à Mask-RCNN qui utilise une approche en deux étapes. La première version de YOLO a été créée par Joseph Redmon, Santosh Divvala, Ross Girshick et Ali Farhadi en 2015 (10).

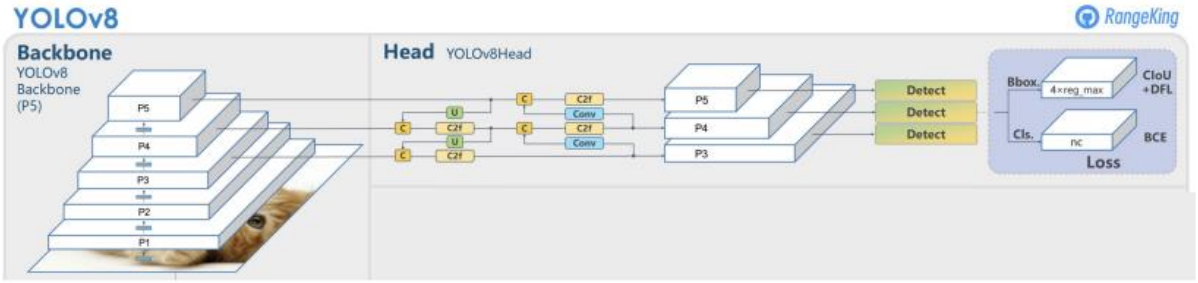


Figure 12 - Représentation graphique de l'architecture du modèle YOLOv8. Source : [7]

YOLOv8 [8], sa version state-of-the-art sortie en janvier 2023, intègre un réseau de pyramides caractéristiques permettant de réduire l'image d'entrée tout en augmentant le nombre de canaux. Cette architecture facilite la détection d'objets à différentes échelles et résolutions (Figure 12). Cela s'avère particulièrement pertinent dans notre cas, qui traite des images de résolution HD de 1080x1920 pixels, alors que le modèle YOLO prend en entrée une résolution de 640x640 pixels.

YOLOv8 a également été entraîné sur une plus grande variété de données par rapport à son prédécesseur, YOLOv5, qui reposait uniquement sur les données COCO. Cela permet au modèle de mieux se généraliser et d'être plus efficace sur de nouvelles données, en appliquant le principe du transfert d'apprentissage.

Model	size (pixels)	mAP^{box} 50-95	mAP^{mask} 50-95	Speed CPU ONNX (ms)	Speed A100 TensorRT (ms)	params (M)	FLOPs (B)
YOLOv8n-seg	640	36.7	30.5	96.1	1.21	3.4	12.6
YOLOv8s-seg	640	44.6	36.8	155.7	1.47	11.8	42.6
YOLOv8m-seg	640	49.9	40.8	317.0	2.18	27.3	110.2
YOLOv8l-seg	640	52.3	42.6	572.4	2.79	46.0	220.5
YOLOv8x-seg	640	53.4	43.4	712.1	4.02	71.8	344.1

Figure 13 - Les différents variables du modèle YOLOv8 segmentation et leurs métriques issues de leur entraînement sur données COCO 2017. Source : https://docs.ultralytics.com/tasks/segment/

Plusieurs variantes du modèle YOLOv8, pré-entraînées pour la segmentation, sont disponibles sur Ultralytics (Figure 13). Leurs dénominations dépendent du nombre de paramètres pris en charge, les valeurs ajustables apprises pendant l'entraînement. Ce sont essentiellement les poids et les biais en entrée et sortie de neurones. Ainsi, le modèle YOLOv8n-seg (nano) est plus léger mais moins complexe et profond que YOLOv8m-seg (medium). Plus un modèle contient de paramètres, plus il sera précis (mAP), mais cette précision se fait au détriment de sa rapidité d'inférence.

Le variant utilisé ici sera donc le modèle YOLOv8m-seg, offrant un bon compromis entre précision et rapidité.

II.1.2.2. Entraînements

Les données annotées sont divisées en trois ensembles distincts selon la répartition suivante : 70 % pour le jeu d'entraînement, 20 % pour le jeu de validation et 10 % pour le jeu de test, conformément à la pratique usuelle [9]. Les jeux d'entraînement et de validation sont utilisés lors de l'entraînement du modèle. En effet, le jeu d'entraînement va servir de base à la reconnaissance de caractéristiques ; l'algorithme analyse les caractéristiques de l'image et ajuste ainsi les paramètres du modèle. Le jeu de validation, quant à lui, intervient en fin de chaque cycle d'entraînement, appelé époque, afin d'évaluer le modèle à chaque moment de son apprentissage. Enfin, le jeu de test vient évaluer les performances du modèle en fin d'entraînement en restant totalement indépendant.

Au début de chaque entraînement, Ultralytics va nous rappeler les paramètres et hyperparamètres du modèle choisi. Ceux-ci existent par défaut, mais peuvent être modifiés par l'utilisateur, soit dans la ligne de code lançant l'entraînement, soit dans le fichier de configuration du modèle : data.yaml. Ce dernier renseigne également les chemins vers les dossiers contenant les jeux de données annotées. Dans l'annotation YOLO, ce chemin pointe vers un dossier contenant deux sous-dossiers : *images* et *annotations*. Chaque image du dossier *images* détient une annotation correspondante dans le dossier *annotations*, les deux fichiers partageant le même nom.

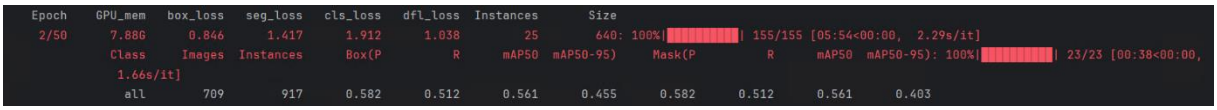


Figure 14 - Log d'entraînement d'une époque avec ultralytics et le modèle YOLOv8-seg-m

Une fois l'entraînement lancé, nous pouvons consulter le journal des logs, comme vu sur la Figure 14. Ce journal nous permet d'extraire les métriques clés, essentielles pour suivre l'évolution de la performance du modèle. Ces métriques, combinées à celles de la validation, seront cruciales pour évaluer si le modèle converge correctement ou s'il y a un risque de surapprentissage.

II.1.2.3. Les Métriques

Les métriques à minimiser sont les suivantes :

- **Box_loss / seg_loss** : mesure la différence de localisation des boites/masques prédits par rapport aux boites / masques vrais, issus de l'annotation.
- **Cls_loss (class loss)**: défini la perte de classification, évalue la précision de prédiction de la classe par le modèle.

Pour les métriques suivantes, qui dépendent des masques de segmentation et des boîtes englobantes, l'objectif est inverse : il convient de se rapprocher d'un maximum de 1. Des informations détaillées sur ces métriques sont disponibles dans l'Annexe n°1.

- **P (precision)**: la précision est définie comme le pourcentage de prédictions correctes parmi toutes les prédictions faites.
- **R (recall)** : le rappel est défini comme le pourcentage de prédictions correctes parmi toutes les instances réelles positives.

- **mAP50 et mAP50-90 (mean Average Precision) :** métrique qui évalue la performance des modèles de détection d'objets et de reconnaissance de formes en calculant la moyenne des précisions pour toutes les classes d'objets à différents seuils de rappel (recall). La précision moyenne (AP) est définie comme l'aire sous la courbe de précision-rappel.
- **IoU (Intersection Over Union)** : L'IoU (Intersection Over Union) quantifie le recouvrement entre la boîte englobante prédite ou la région segmentée et la boîte englobante réelle ou la région annotée dans un jeu de données.

II.1.3. Résultats de l'entraînement

L'entraînement des modèles a été réalisé sur une période de 50 époques. Ce paramètre a été obtenu après un entraînement préliminaire sur 100 époques ayant montré une stagnation des métriques vers 45/50 époques. La taille du batch est resté à 16, paramètre par défaut de YOLOv8. Enfin, il faut ici préciser que le module Ultralytics utilise déjà des paramètres d'augmentation par défaut (12) qui seront ajustés à la suite de cette partie.

II.1.3.1. Élimination des chiffres

Ont été comparés ici les deux entraînements du modèle sur les jeux de données suivants : *markers_and_digits_v2*, rassemblant les 12 classes initiales et *cross_only*, basé sur la classe unique des croix. Le graphique, Figure 15, permet alors de faire ressortir le modèle entraîné uniquement sur une classe comme étant largement supérieur. Et en effet, comme vu dans la partie II.1.3, le biais de classe important constaté dans les annotations du jeu de données *markers_and_digits_v2* limite grandement ses performances. Les erreurs de classification sont également très courantes comme nous pouvons le voir sur la Figure 16. Celles-ci découlent des ressemblances morphologiques entre chiffres, aggravées par les variations de leur orientation au sein des images. On observe donc souvent, par exemple, une détection erronée d'un 4 au lieu d'un 7 et vice versa.

En raison de l'insuffisance de données annotées et des difficultés liées à la rotation ainsi qu'à la variabilité graphologique, la détection des chiffres a été abandonnée. L'accent est désormais mis exclusivement sur la détection des croix. Les résultats globaux de l'entraînement de ce premier jeu de données, qui a été finalement laissé de côté, sont présentés en Annexe n° 2, sous forme de graphiques générés par le framework Ultralytics.

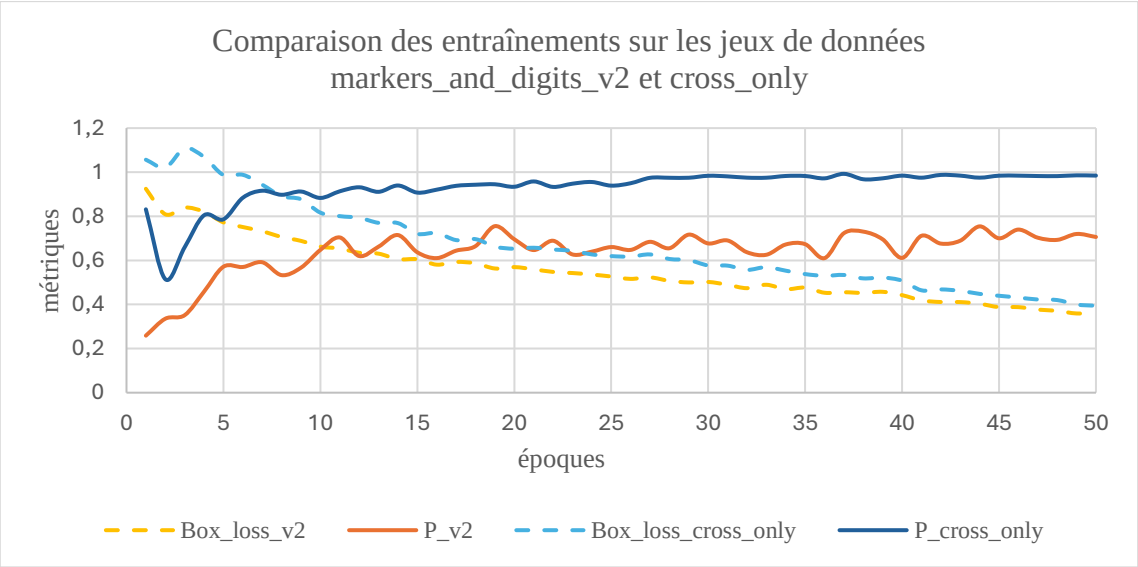


Figure 15 - Graphique de comparaison des métriques entre les jeux de données markers_and_digits_v2 et cross_only.

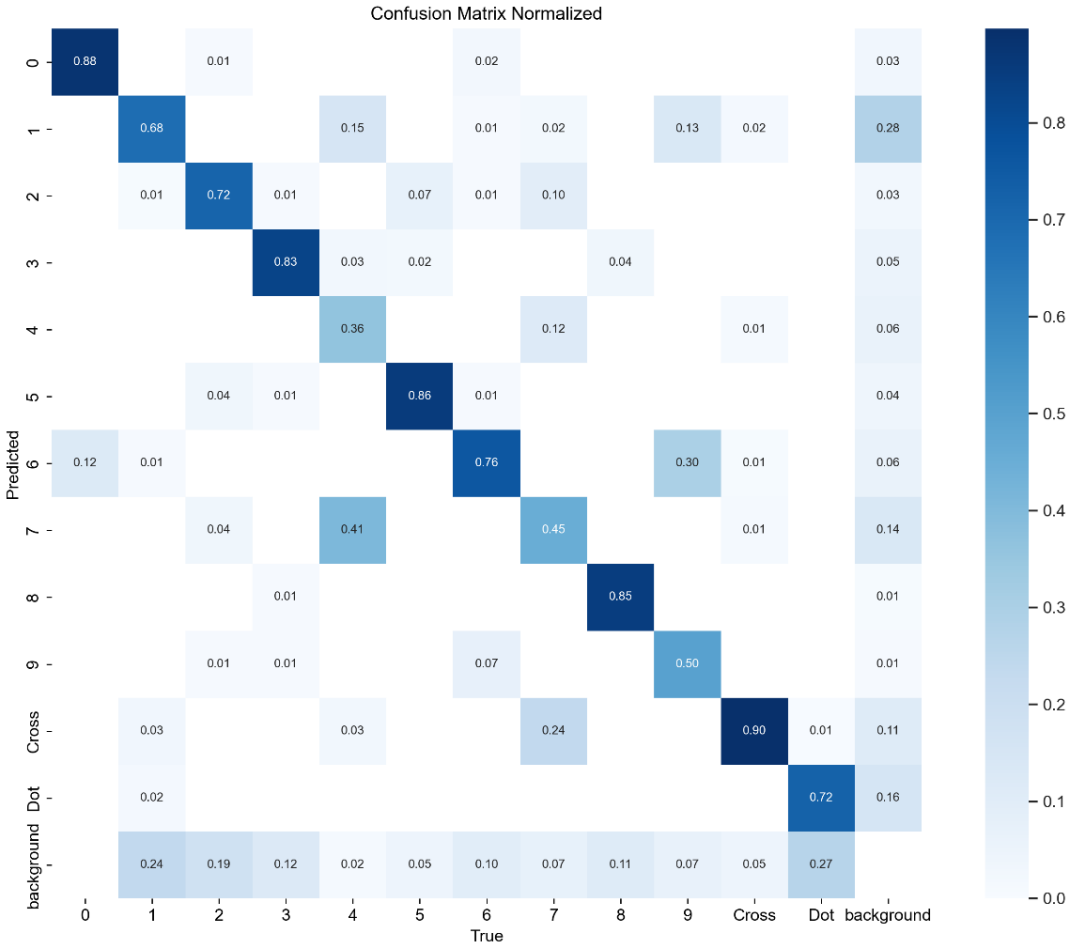


Figure 16 - Matrice de confusion normalisée du modèle markers_and_digits_v2.

II.1.3.1. Augmentation du modèle cross_only

Le modèle cross_only a été entraîné à nouveau, afin d'améliorer sa performance, avec cinq variations de paramètres d'augmentation. Trois éléments ont essentiellement été ici considérés : les variations hsv, la rotation des images (degrees) ainsi que la perspective (shear). Les détails de ces augmentations peuvent être retrouvés dans l'Annexe 3 de ce rapport.

La validation des modèles a été réalisée sur un jeu de données test, contenant 109 images de croix représentatives. Le VARIANT 2 apparaît donc comme le plus performant, avec le meilleur équilibre entre précision et rappel (Tableau 2). Il sera donc utilisé pour la suite de ce travail. Il a permis, à 80 % de confiance, de détecter correctement 85 croix contre 82 pour le modèle original, sans aucun faux positif.

Tableau 2 - Métriques de validation des variants augmentés du jeu de données cross_only. En bleu et orange les métriques et résultats concernant respectivement les métriques des boites englobantes et des masques de segmentation.

	P	R	MAP50	MAP50-95	P	R	MAP50	MAP50-95
ORIGINAL	0,969	0,854	0,904	0,814	0,948	0,836	0,886	0,616
VARIANT 1	0,925	0,782	0,865	0,739	0,914	0,773	0,843	0,554
VARIANT 2	0,96	**0,869**	**0,917**	**0,83**	0,94	**0,851**	**0,89**	0,615
VARIANT 3	0,96	0,862	0,909	0,814	0,933	0,845	0,893	0,622
VARIANT 4	0,951	0,879	0,919	0,826	0,841	0,87	0,899	0,628
VARIANT 5	0,916	0,891	0,918	0,814	0,897	0,873	0,888	0,62

II.2. Méthodes d'estimation du centre de pointé

La détection des croix peintes amène à se poser la question de l'estimation de leur centre de pointé. Ainsi, pour chaque détection, le masque de segmentation a été extrait. Celui-ci correspond aux points de contour des croix dans le référentiel image. Afin de déterminer le centre, les masques de détection ont été transformés en images binaires sur lesquelles les méthodes de détermination du centre seront testées.

Le centre théorique dont il sera question lors des comparatifs suivants correspond au centre des croix sur images originelles, pointé par un opérateur humain. Le biais humain ne sera pas pris en compte dans notre analyse.

II.2.1. Le centre de masse

Le premier test vise à déterminer le centre de masse, également appelé centroïde, des croix détectées. Pour cela, le module Moments d'OpenCV (13) a été utilisé. Les moments géométriques sont issus des intégrales pondérées de la fonction d'intensité de l'image. Pour une image donnée, les moments (p,q)sont définis comme suit :

$$M_{pq} = \sum_x \sum_y x^p y^q I(x,y)$$

Où I(x,y) est l'intensité de l'image à la position (x,y) avec :

- M_{00} : moment d'ordre zéro, correspond à l'aire de la forme ;
- M_{10} et M_{01} : moments d'ordre un, utilisés pour calculer le centre de masse.

On peut ainsi calculer le centre de masse (cX, xY) à partir des moments du contour, dans notre cas :

$$cX = \frac{M_{10}}{M_{00}}, \quad cY = \frac{M_{01}}{M_{00}}$$

Bien évidemment ce centroïde dépend fortement de la forme de la croix. Une croix avec des branches parfaitement perpendiculaires aura un centre de pointé proche, voire recoupé, au centre théorique. Cependant, la majorité des croix apposées par les opérateurs terrain sont loin d'être parfaites et présentent souvent des branches plus ou moins courbes et de longueurs variables ainsi qu'une perspective plus ou moins forte liée à l'acquisition.

II.2.2. Le centre typologique

Autre possibilité de recherche du centre de croix explorée : le centre typologique. L'idée est ici de trouver les 4 extrémités des branches de la croix afin de retrouver le point d'intersection des deux droites reconstruites, notre centre considéré (Figure 17).

Pour déterminer ces extrémités le module itertools.combination (14) a été utilisé afin de boucler sur 4 points du contour et de déterminer ceux définissant la plus grande aire possible. Leur centre de masse a, par la suite, été déterminé similairement à la méthode décrite dans la partie II.2.1.

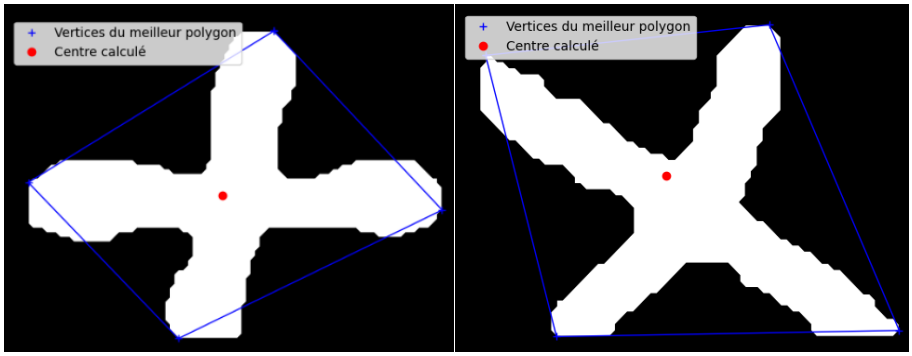


Figure 17 - Exemples de centres typologiques.

II.2.3. Le centre de convolution

L'utilisation d'un noyau de convolution spécifique a été prise en compte afin de déterminer le pixel le plus entouré. Un noyau personnalisé a été défini avec une pondération croissante des pixels environnants vers le centre, de manière symétrique. Voici sa représentation sous forme de matrice :

$$\begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 2 & 1 & 0 \\ 1 & 2 & 0 & 2 & 1 \\ 0 & 1 & 2 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

En faisant passer ce noyau sur l'ensemble du masque suivant un nombre d'itérations importantes, 100 dans notre cas, nous pouvons extraire le pixel ayant le score le plus élevé, qui sera en théorie le pixel central. Cependant, nous pouvons voir que cette solution n'est pas toujours cohérente. Il en résulte plutôt le centre de l'espace le plus important du masque, comme nous pouvons l'observer sur la Figure 18.

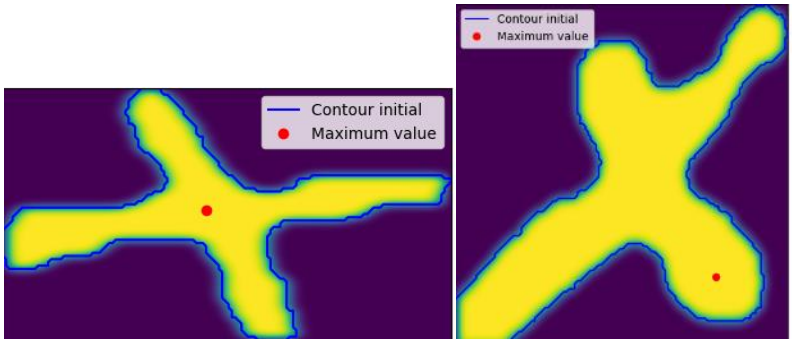


Figure 18 - Exemples de centres de convolution extraits, avec à droite un pointage réussi et à gauche une aberration.

II.2.4. Le centre détecté

Un autre entraînement, cette fois-ci de détection de point clé, a été réalisé avec le modèle YOLOv8. Près d'un millier d'images binaire issues des masques de détection ont été annotées d'un point en leur centre. Le modèle pré-entraîné considéré est ici YOLOv8m-pose. L'entraînement s'est déroulé sur 50 époques avec les paramètres Ultralytics par défaut. Nous pouvons observer Figure 19 un exemple de résultat issu de ce modèle.

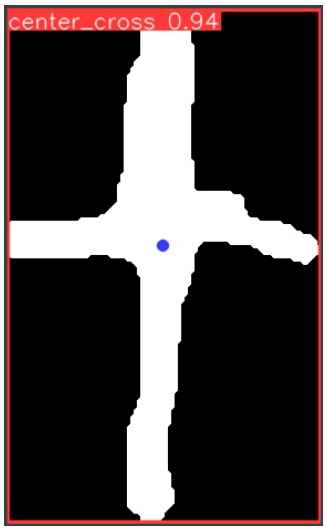


Figure 19 - Exemple de centre détecté.

II.2.5. Le centre via le module skeleton

Enfin, le module skeletonize de scikit-image (15), vu dans l'article [10] pour tracer des linéaires de câbles, est une méthode permettant d'extraire le squelette d'un masque binaire. IL permet de supprimer itérativement les bords du masque afin de conserver son ossature centrale représentée par des polylignes d'une épaisseur d'un pixel. Le squelette des croix détectées a été extrait grâce à ce module, ses pixels caractéristiques classifiés en fonction de leur voisinage (Figure 20) :

- Extrémités : voisinage de 1 pixels
- Continuité : voisinage de 2 pixels
- Branchement : voisinage de 3 pixels
- Intersection : voisinage supérieur à 3 pixels

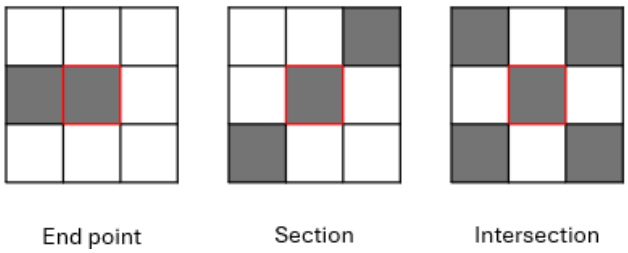


Figure 20 - Extrémité (End point), Continuité (Section), Intersection mis en surbrillance suivant leur voisinage. Source : [10]

Les branchements et intersections sont filtrés en fonction de leur distance aux extrémités pour éliminer les faux embranchements pouvant résulter de l'utilisation du module skeletonize. Par la suite, un clustering est appliqué pour identifier les points d'intersection

pertinents. En cas d'intersections multiples, le centre de masse des points est retenu comme point central, tel qu'illustré sur l'image de droite de la Figure 21.

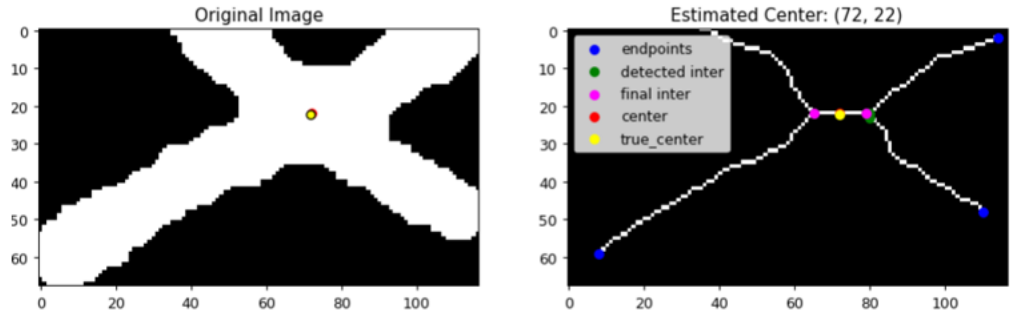


Figure 21 – Exemple type de pointage d'un centre extrait par la méthode skeleton.

II.3. Résultats du pointage automatique et comparaisons

Pour évaluer la méthode la plus performante, un jeu de test comprenant 109 croix, variées en forme, couleur et placement, a été constitué. Les coordonnées des centres théoriques ont été déterminées par un pointage manuel sur les images d'origine. Ces images ont ensuite subi une détection, permettant d'en extraire leurs masques finalement transformés en images binaires.

Les graphiques présentés ici (Figure 22 et Figure 23) ont été réalisés en comparant les coordonnées image théoriques, et les coordonnées image retrouvées automatiquement suivant les différents algorithmes présentés partie II.2.

La Figure 22 illustre le graphique de dispersion des écarts-types en X et Y des coordonnées issues des différentes méthodes dans le référentiel image. La méthode Skeleton se distingue par sa précision supérieure, avec un écart-type moyen de 4,61 pixels, et son ellipse d'imprécision est la moins étendue. Toutefois, il est important de noter que cette méthode présente des dérives importantes, pouvant atteindre jusqu'à plus de 13 pixels, comme le montre la Figure 23.

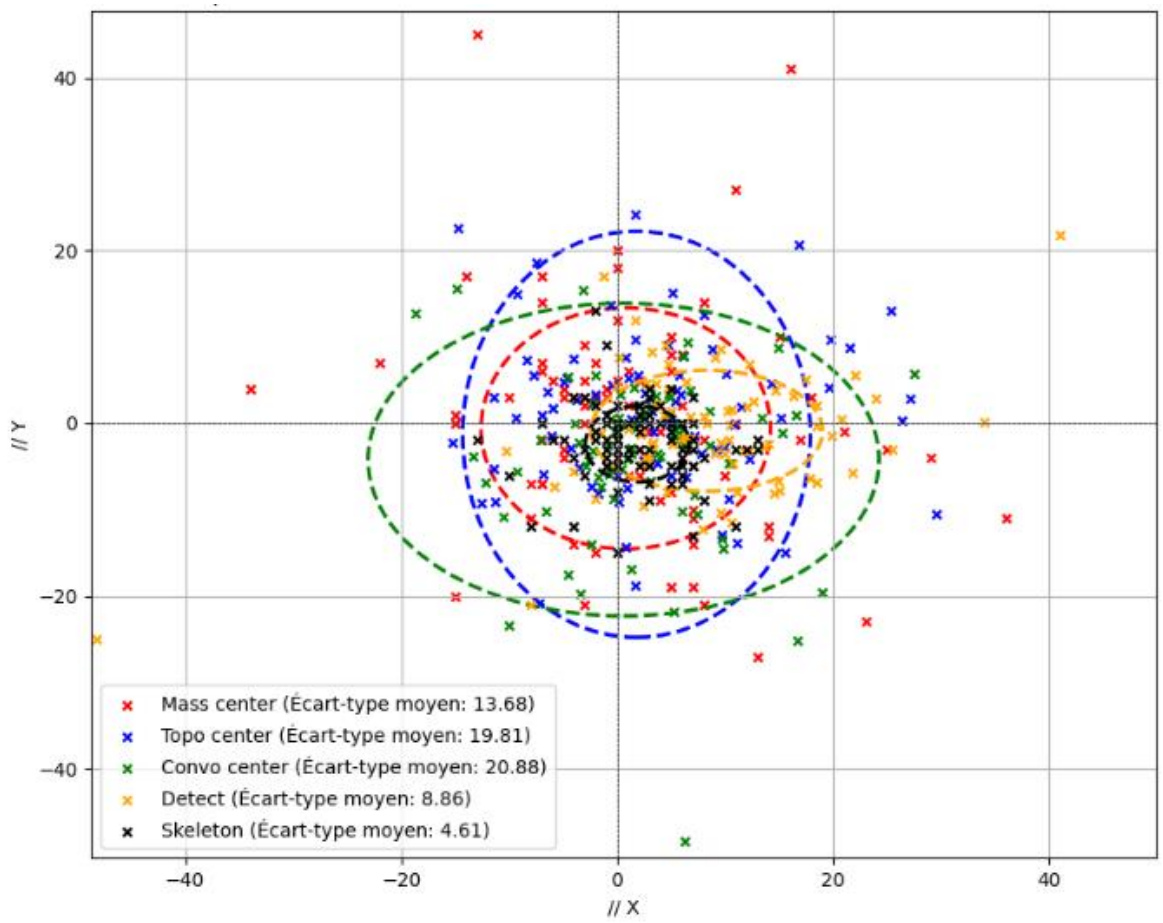


Figure 22 - Graphique de dispersion et ellipses d'imprécision entre pointé théorique et pointé automatique suivant les méthodes considérées.

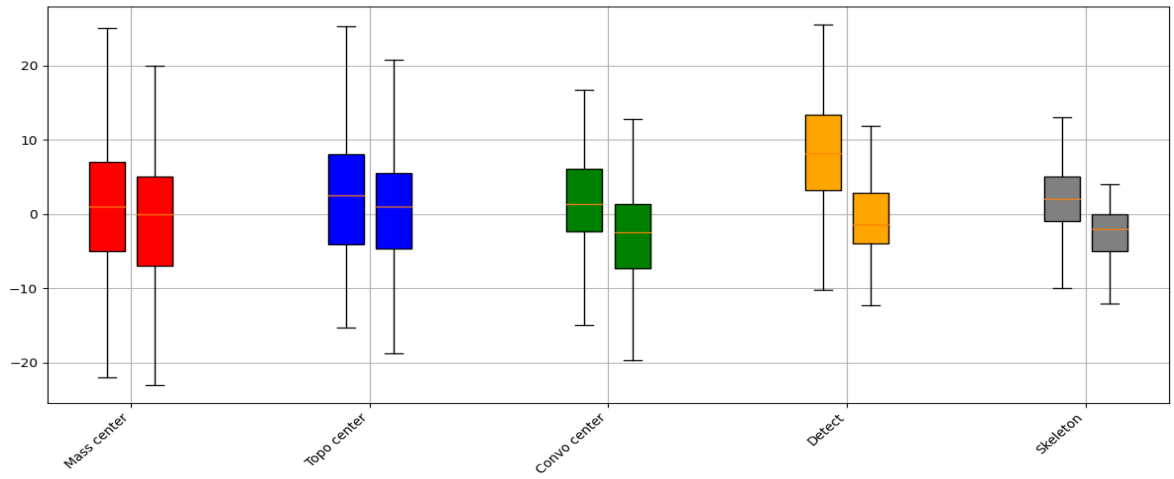


Figure 23 - Diagramme en boîtes des écarts en position X (sur la gauche) et Y (sur la droite) des différentes méthodes dans le référentiel image.

On constate, après une analyse détaillée, que les erreurs dépassant 20 pixels proviennent principalement de deux sources : d'une part, la détection de croix à trois branches, et d'autre part, des problèmes de reconnaissance d'intersections multiples entraînant l’assimilation d’une intersection à un centre. La première difficulté pourrait être atténuée en exigeant un marquage plus rigoureux de la part des opérateurs sur le terrain. Il s'agirait d'améliorer la qualité des croix peintes au sol, avec des branches bien définies, tout en évitant les prises vidéo sous un angle supérieur à 30° par rapport au sol. La seconde difficulté est plus complexe et nécessiterait des recherches approfondies, ainsi qu'une révision de l'algorithme de pointage des centres des masques.

III. Intégration et déploiement via le logiciel de traitement Metashape

Cette section finale abordera l'intégration de l'algorithme de détection dans le script global PyAutoMarkers, ainsi que l'adaptation de cette fonctionnalité au traitement photogrammétrique via Metashape. Nous détaillerons ensuite les différents processus d'alignement testés pour faire correspondre les ensembles de détection avec les points de calage relevés sur le terrain et ainsi géoréférencer le chantier, avant de présenter les résultats globaux obtenus.

III.1. L'API Metashape et la schématisation du processus d'intégration

III.1.1. L'utilisation de l'API Metashape

L'API de Metashape a été utilisée en développement externe. Le script est actuellement exécuté via une fonction *main* sous PyCharm plutôt que directement depuis le logiciel. Il a plus tardivement été intégré dans le processus de traitement de l'application Récolement explicité partie I.1.2.2. Cependant, il serait possible de l'appeler au sein même de Metashape en créant un menu personnalisé. Un exemple de cette application est visible à la Figure 24.

```
#Ajoute un custom menu dans metashape -> .py a mettre dans : C:YourPATH\Agisoft\Metashape Pro\scripts
import Metashape

def add_your_definition():
    #
    doc = Metashape.app.document
    chunk = doc.chunk
    #
    # Ajouter les éléments de la definition
    doc.save()

Metashape.app.addMenuItem("Custom/Menu_Item",add_your_definition)
```

Figure 24 - Exemple de création d'un menu personnalisé dans Metashape

Ce type d'ajout, de même que l'utilisation de certaines fonctions via l'API de Metashape, n'est possible qu'avec une licence Metashape Pro valide et activée.

III.1.2. Définition globale des modules

Deux modules ont été codés pour lancer le processus de détection, de pointage et de géoréférencement des repères peinturés. Le cheminement intégral du code développé, PyAutoMarkers, est détaillé en Annexe n°4.

III.2. Alignement entre le modèle local et le modèle terrain.

Dans cette section, nous détaillerons l'étape cruciale d'alignement des clusters détectés sur les points de calages connus en coordonnées, extraits d'un fichier .csv existant. Nous commencerons par décrire le processus de clustering utilisé pour identifier les ensembles pertinents. Puis, nous explorerons différents algorithmes d'alignement. Enfin, nous mettrons en avant l'efficacité de la fonction de mise en correspondance sans labels de Metashape.

III.2.1. Clustering agglomératif

Un clustering des détections, basé sur leurs coordonnées reprojetées, a été nécessaire pour définir un ensemble de détection à faire correspondre avec le fichier de points de calage levés, et ainsi géoréférencer le chantier.

Scikit.learn, propose plusieurs solutions de clustering (16). Des essais ont été réalisés sur différentes méthodes en tenant compte du fait que le nombre de clusters n'est à priori pas connu, les détections pouvant échouer.

Le choix retenu, offrant la plus grande cohérence, est le clustering agglomératif (17). Cette méthode appartient à la famille des clusterings hiérarchiques, qui regroupent les entrées en les fusionnant ou en les séparant de manière itérative et ordonnée. Le clustering agglomératif commence par initialiser chaque point comme un cluster individuel. Les similarités, mesurées selon un paramètre métrique, permettent de fusionner ces clusters jusqu'à ce que le seuil prédéterminé soit atteint. Les paramètres ont été définis avec la métrique par défaut de distance euclidienne à un seuil de 2 mètres avec une connexion simple

III.2.2. Revue des algorithmes d'alignements testés

III.2.2.1. L'algorithme GO-ICP

Cet algorithme est disponible sur GitHub (18). Cependant, sa compilation en une bibliothèque utilisable avec Python est quelque peu obsolète (19). En effet, Syslor utilise actuellement des scripts sous Python 3.8, une version qui n'est pas compatible avec ce package, dont la compilation s'arrête à Python 3.7. Pour tester cet algorithme, le code original en C++ a donc été exécuté en tant que sous-processus.

L'idée de Go-ICP [11] est de modéliser l'espace à l'aide de cubes définissant la rotation et translation entre deux ensembles de points. Son objectif est ainsi de trouver la matrice de transformation qui minimise les distances euclidiennes entre les points des deux nuages. Pour ce faire, les cubes sont subdivisés suivant un algorithme de séparation et d'évaluation, dit Branch and Bound, permettant de déterminer la translation et rotation optimale entre les ensembles. Cette position est ensuite affinée à l'aide d'un algorithme ICP. Ce processus mathématique itératif constitue une solution efficace pour éviter les minima locaux, un problème courant lorsque l'ICP est utilisée sans un alignement initial adéquat.

Cependant, les résultats ne sont pas satisfaisants dans notre cas. En effet, parmi les 14 chantiers où la détection a été correcte, seulement 50 % d'entre eux sont correctement alignés.

Et en effet, avec un nombre de points par nuage dans compris entre 5 et 20, cet algorithme n'est pas assez robuste. S'ajoute à cela la non-gestion du facteur d'échelle, estimé ici par comparaison des aires entre les nuages normalisés.

III.2.2.2. L'algorithme RANSAC

Une autre méthode, explorée dans le but de faire coïncider les détections du référentiel local aux points de calage géoréférencés, est RANSAC. Celle-ci est beaucoup utilisée en vision par ordinateur ou traitement d'images afin d'estimer les valeurs aberrantes d'un ensemble de données. Cet algorithme cherche à trouver la plus proche estimation d'un modèle (droite, plan, transformation géométrique, etc.) à partir d'un consensus de points choisi au sein des entrées. Plusieurs itérations peuvent être réalisées jusqu'à trouver le meilleur modèle et donc l'ensemble de points correspondant à ses valeurs acceptées.

RANSACRegressor de scikit-learn (20) permet d'utiliser cet algorithme afin d'ajuster un ensemble de données tests à un ensemble cible avec l'outil RANSACRegressor.fit(). Cette définition prend en entrée deux ensembles de données de tailles égales et permet d'en extraire les erreurs selon chaque correspondance prédite. Ces erreurs sont exprimées suivant l'estimateur choisi, par défaut le modèle linéaire donnant l'erreur quadratique moyenne entre la position prédite et théorique.

Dans notre cas les ensembles sont de taille variable suivant le nombre de clusters issus des détections. Il a donc été nécessaire d'itérer sur la plus grande taille de combinaisons possible entre nuages. Par exemple, si le nuage local de clusters détectés ne comporte que 5 points alors que le nuage géoréférencé, issu des points GNSS, comporte 7 points ; nous itérerons sur des combinaisons de 5 points sélectionnés parmi les points GNSS. Cela nous fera donc considérer 21 correspondances possibles sur lesquelles RansacRegressor.fit() sera appliqué. Cette définition combinatoire est la suivante :

$$C(n,k) = \frac{k!}{(n-k)!\,n!}$$

où n = nombre total, k = ensemble choisi

De plus, ces combinaisons seront multipliées des permutations (factoriel du nombre de points du sous-ensemble) possibles parmi les points sélectionnés afin de trouver la meilleure correspondance entre points cluster et points GNSS :

Tableau 3 - Tableau de synthèse des alignements selon la méthode RANSAC pour différents chantiers.

N° chantier	Nb de pts de calage	Nb de pts détectés	Combinaisons	Permutations	Itérations totales	Temps	Erreur (m)
23CHJRP07979	7	7	1	5040	5040	00:00:12	0,045
23MOTEP08569	5	5	1	120	120	00:00:01	0,01
24CHJRP09678	6	5	6	120	720	00:00:02	0,058
24EURCP09486	8	7	8	5040	40320	00:01:54	0,014
24CHJRP09681	5	4	5	24	120	00:00:00	0
24CHJRP09809	10	5	252	120	30240	00:01:24	0,006
24EURCP09471	9	7	36	5040	181440	00:08:30	0,022
24SPACP08809	5	4	5	24	120	00:00:00	0

Cette méthode a donc une rapidité exponentiellement décroissante en fonction du nombre de correspondances à considérer (Tableau 3). Il faut, de plus, au minimum 5 clusters détectés afin de pouvoir estimer les erreurs quadratiques, comme mis en avant en rouge dans le Tableau 3. En effet, une transformation 3D avec facteur d'échelle va nous demander 12

paramètres minimum en entrée afin d'être résolue. Aucune erreur ne peut donc être extraite d'une transformation déterminée avec uniquement 4 clusters détectés, car considérée « parfaite ». Enfin, le placement longitudinal et en quinconce des points de calage le long des réseaux en fait résulter des erreurs de correspondance dû aux proches symétries possibles. Cela pose d'autant plus problème que ces mauvaises correspondances varient suivant l'initialisation des combinaisons et permutations. On note alors un phénomène de dépendance à l'initialisation.

Cette solution ne répond pas non plus aux exigences de notre cas d'utilisation.

III.2.3. Fonction de mise en correspondance Metashape.

Les algorithmes testés précédemment sont loin de satisfaire à la volonté d'un alignement correct entre clusters détectés et points de calages connus en coordonnées. Cependant en approfondissant la recherche des définitions Metashape, la mise en correspondance de ces données est possible lors de l'import des coordonnées des points de calage [2].

En effet, au sein du logiciel, il est possible d'utiliser les coordonnées levées et de les faire correspondre aux marqueurs grâce à l'option « ignore labels » dans la fonction *ImportReference*. Nous avons choisi ici un seuil de correspondance de 0.5 mètre et défini l'*item* utilisé comme étant les marqueurs. Cette définition est aussi disponible via l'API comme le montre la définition utilisée

```
def match_gcps_to_cluster(self):
    self.chunk.importReference(self.gcps_path, delimiter =";",format = Metashape.ReferenceFormatCSV, columns = "nxyz[XYZ]",
                               crs = self.chunk.crs,create_markers=True,
                               ignore_labels=True,threshold=0.5, items=Metashape.ReferenceItems.ReferenceItemsMarkers)
    self.doc.save()
```

Figure 25 - Définition de mise en correspondance clusters/points de calage connus en coordonnées.

Cette solution semble satisfaire à notre application. Il en résulte en effet un taux de correspondance correcte de 100% à partir de 4 clusters non-alignés issus de la détection. De même, son temps de calcul reste bien en dessous des quelques minutes que nous avons pu observer avec les autres algorithmes avec un traitement sous quelques secondes.

Le script mis en place utilise donc cette solution pour la mise en correspondance des clusters et points de calages connus, ce malgré l'absence d'informations disponibles sur la théorie sous-jacente à son développement.

III.3. Synthèse des résultats et commentaires

III.3.1. Précision du géoréférencement global : objectif classe A.

Une étude sur les écarts avec les coordonnées estimées par reprojection des repères automatiques sur un chantier référencé manuellement a été réalisée afin d'obtenir les écarts de géoréférencement globaux avec un pointage manuel (théorique) et automatique (estimé). Ces écarts cherchent à respecter les conditions requises afin d'obtenir la classe de précision A présentée lors de la partie I.1.1. Nous nous référons donc au guide technique d'application de la réglementation en la matière [2] en sa partie 4.3.1.

Afin de respecter cette classe A, nous nous plaçons dans le cas d'application de Syslor : les fouilles ouvertes. Nous nous référons donc au gabarit n°1, extrait ci-dessous dans la Figure 26.

Dimensions	Précision	Écart moyen inférieur à	1er seuil	2e seuil (incertitude maximale de localisation)
Planimétrie	10 cm	11,25 cm	entre 0 et ±27 cm au moins 99% des écarts	entre 0 et ±40 cm 100% des écarts
Altimétrie	10* cm	15* cm	entre 0 et ±35* cm au moins 99% des écarts	entre 0 et ±40 cm 100% des écarts

Figure 26 - Tableau extrait du fascicule 2 de la réglementation afférente à la classification des réseaux. Page 70/234

Nous pouvons observer (Annexe n°5) que l'ensemble des écarts calculés en planimétrie et altimétrie nous amène à une précision de pointage automatique par rapport au pointage manuel de 2.3 et 3.2c cm respectivement. Ainsi en utilisant la formule de cumul des erreurs page 72 du fascicule réglementaire [2] nous avons :

$$E_T = \sqrt{e_1^2 + e_2^2 + e_3^2 \cdots e_n^2}$$

Où nous pouvons considérer comme erreurs à prendre en compte :

- Précision du pointage automatique par rapport au manuel : 2.3cm / 3.2 cm
- Précision du levé GNSS : 3cm / 5cm
- Précision de rattachement au système de référence : 5 cm

- Précision de reconstruction du nuage de point 3D : 5 cm

Ainsi, dans notre cas d'étude nous avons :

$$E_{plani} = \sqrt{2.3^2 + 3^2 + 5^2 + 5^2} = 8.02\ cm$$

$$E_{alti} = \sqrt{2.3^2 + 5^2 + 5^2 + 5^2} = 9.23\ cm$$

Ces deux écarts de géoréférencements moyens se situent bien en deçà de l'écart moyen demandé afin de faire partie de la classe A, sans considérer l'erreur de pointage théorique. De même, en excluant la valeur aberrante qui sera vue dans la partie III.3.2, l'ensemble des écarts altimétriques et planimétriques ne dépassent pas le premier seul d'incertitude présenté Figure 26. Nous nous situons bien, avec cette méthode, en classe de précision A.

III.3.2. Analyse de la donnée aberrante

Cependant nous pouvons détecter une valeur aberrante au sein des 7 chantiers détaillés en Annexe n°5. Celle-ci se trouve chantier 23X08569 au niveau du cluster_0 avec un écart altimétrique dépassant les deux seuils d'incertitude maximales autorisés vu avec la Figure 26. On voit ici une limite posée par la photogrammétrie. En effet, la croix de calage ici détectée l'a été sur des images très floutées comme vu sur la Figure 27. Ainsi l'acquisition peut ici être mise en cause avec notamment une vitesse de déplacement trop importante à cet endroit. En plus d'amener une erreur de détection ce flou incite également à une grande incertitude de reconstruction à cet endroit du essentiellement à un manque de points homologue pour l'alignement (Figure 28). Ce problème de reconstruction est d'autant plus exacerbé du fait que l'on se trouve ici en bord d'acquisition (Figure 29).

Figure 28 - Zoom sur une des images du chantier 23X08569 présentant une détection aberrante causé par le flou présent

Figure 27 - Chantier 23X08569, capture du nuage de point épart issu de Metashape avec positionnement des pointés manuels (numéros) et des clusters identifiés selon l'algorithme automatique mis en place. En bleu le placement des caméras.

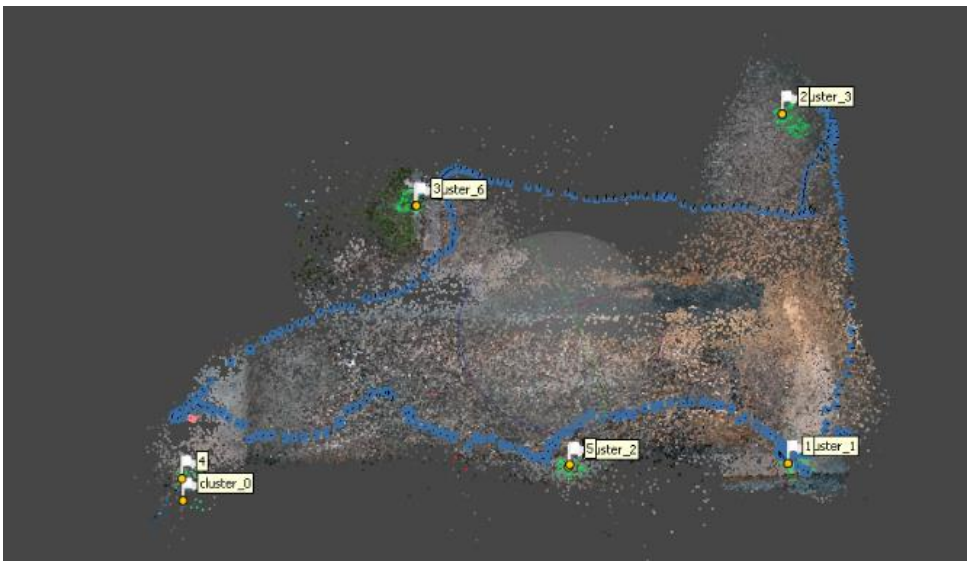


Figure 29 - Nuage de point dispersé au niveau de l'alignement des images floues du chantier 23X08569

III.3.3. Remarques et pistes d'amélioration

La méthode de pointage automatique entre donc dans les tolérances. Cependant elle entache tout de même le géoréférencement de quelques centimètres. Cette dégradation est due à plusieurs facteurs :

- Le manque de rigueur lors de la matérialisation au sol par l'opérateur
- L'imprécision du masque extrait de la détection
- L'imprécision de l'algorithme de recherche du centre de la croix

Chacun de ces facteurs peuvent être étudiés et améliorés. En effet, en collectant les données pour l'annotation, nous avons constaté une détérioration visible de la qualité du marquage au sol au cours des deux années prises en compte. Une plus grande rigueur lors de cette étape peut être demandée.

Le modèle de segmentation peut de même être amélioré en augmentant le nombre d'annotation mais aussi en testant d'autres architectures. Enfin, l'algorithme de recherche du centre de pointé via le module skeletonize pourrait faire l'objet de filtrage notamment en ne considérant pas les détections à 3 branches ou avec des ramifications importantes.

CONLUSION

Pour conclure, ce travail a permis d'optimiser la chaîne de traitement photogrammétrique de l'application Récolement de Syslor, en mettant en place le script python PyAutomarker dont le flowchart est présent en Annexe n°4. Celui-ci répond aux souhaits de l'entreprise en permettant l'automatisation du pointage de marques peintes utilisées comme points de calage. Plusieurs étapes ont ainsi amené à son développement.

Premièrement, un modèle d'apprentissage profond a été entraîné en utilisant l'architecture YOLOv8 et son framework Ultralytics. Une étape d'annotation des images a alors été indispensable, portant sur l'ensemble des données disponibles : les croix et points peints associés à leur numérotation. Cependant, les biais de classes présents entre chiffres, dus à des problèmes de gestion de la rotation et de graphologie, ont conduit à la détection et segmentation des marques en croix uniquement. Le modèle entraîné obtenu a été amélioré pour atteindre une performance maximale.

Ensuite, plusieurs algorithmes de recherche des centres des masques issus des détections ont été développés et testés, avant de retenir l'utilisation du module skeletonize, qui a permis un écart moyen de pointage de 4,61 pixels.

Un clustering a alors été mis en place afin de regrouper les détections comme projections des différentes croix. Une fois cette étape passée, une mise en correspondance de ces croix aux points de calage relevés sur le terrain a été nécessaire pour géoréférencer le chantier. Pour cela, deux algorithmes, GO-ICP et RANSAC, ont été testés avant de se tourner vers une solution opaque mais fonctionnelle de Metashape.

Finalement, la cohérence de l'automatisation a été évaluée au regard du respect de la réglementation de localisation des réseaux. La solution maintient la classe A si le processus d'acquisition est respecté et cohérent.

Il est aussi à noter que ce travail a été repris au sein de l'application EasyScan. La détection et le pointage des croix ont, cette fois, permis le respect de la réglementation en étant utilisé comme autocontrôle sur le terrain.

TABLE DES ILLUSTRATIONS

LISTE DES TABLEAUX

BIBLIOGRAPHIE

[1] **Arrêté du 15 février 2012**, Chapitre IV, Titre V, Livre V du code de l'environnement. (2012). Relatif à l'exécution de travaux à proximité de certains ouvrages souterrains, aériens ou subaquatiques de transport ou de distribution.

[2] **INERIS**. (2018). Guide d'application de la règlementation. Fascicule 2 version 3 relative aux travaux à proximité des réseaux (Version 3). Paris : INERIS. Retrieved from https://www.reseaux-et-canalisations.ineris.fr/gu-presentation/userfile?path=/fichiers/Guides_techniques/Fascicule2-Guidetechniquedestravaux-v3-2018-09.pdf

[3] **Agisoft LLC**. (2019). Manuel de l'utilisateur. Agisoft Metashape, Professional Edition, Version 1.5. Paris : Agisoft LLC. Retrieved from https://www.agisoft.com/pdf/manuals_other/metashape_pro_fr_1_5.pdf

[4] **Tushev, S., Sukhovilov, B., & Sartasov, E**. (2018). Robust Coded Target Recognition in Adverse Light Conditions. In Proceedings of the 2018 International Conference on Industrial Engineering, Applications and Manufacturing (ICIEAM) (pp. 15-18). Moscow, Russia.

[5] **Camacho, J., & Morocho-Cayamcela, M. E**. (2023). Mask R-CNN and YOLOv8 Comparison to Perform Tomato Maturity Recognition Task. In Communications in Computer and Information Science (Vol. 1885, pp. 382-396). Springer. https://doi.org/10.1007/978-3-031-45438-7_26

[6] **Ameli, Z., Nesheli, S., & Landis, E**. (2023). Deep Learning-Based Steel Bridge Corrosion Segmentation and Condition Rating Using Mask RCNN and YOLOv8. Infrastructures, 9(3). https://doi.org/10.3390/infrastructures9010003

[7] **Lyu, Z., Lu, A., & Ma, Y**. (2024). Improved YOLOv8-Seg Based on Multiscale Feature Fusion and Deformable Convolution for Weed Precision Segmentation. Applied Sciences, 14(5002). https://doi.org/10.3390/app14125002

[8] **Reis, D., Kupec, J., Hong, J., & Daoudi, A**. (2024). Real-Time Flying Object Detection with YOLOv8. arXiv:2305.09972 [cs.CV]. Retrieved from https://arxiv.org/abs/2305.09972

[9] **Charniak, E**. (2021). Introduction au Deep Learning. Malakoff: Dunod. ISBN 978-2-10-082578-3.

[10] **Caporali, A., Galassi, K., Zanella, R., & Palli, G**. (2022). FASTDLO: Fast Deformable Linear Objects Instance Segmentation. IEEE Robotics and Automation Letters, 7(4), 9075-9082. https://doi.org/10.1109/LRA.2022.3189791

[11] **Yang, J., Li, H., Campbell, D., & Jia, Y**. (2016). Go-ICP: A Globally Optimal Solution to 3D ICP Point-Set Registration. IEEE Transactions on Pattern Analysis and Machine Intelligence, 38(11), 2241-2254. Retrieved from https://arxiv.org/abs/1605.03344

SITOGRAPHIE

(1). Agisoft Metashape Python API 2.0. In: Agisoft. Agisoft Metashape, [en ligne]. Disponible sur : https://www.agisoft.com/pdf/metashape_python_api_2_0_0.pdf. (consulté le 12 octobre 2023).

(2). Canevas photogrammétrique. In: Association Française de Topographie (AFT). AFT, [en ligne]. Disponible sur : https://www.aftopo.org/lexique/canevas-photogrammetrique/. (consulté le 15 novembre 2023).

(3). Surveying. In: Capturing Reality. Capturing Reality, [en ligne]. Disponible sur : https://www.capturingreality.com/Surveying. (consulté le 5 avril 2023).

(4). Surveying and Mapping. In: Pix4D. Pix4D, [en ligne]. Disponible sur : https://www.pix4d.com/industry/surveying-mapping/. (consulté le 5 avril 2023).

(5). La révolution de l'apprentissage profond. In: Interstices. Interstices, [en ligne]. Disponible sur : https://interstices.info/la-revolution-de-lapprentissage-profond/. (consulté le 17 octobre 2023).

(6). Qu'est-ce que la segmentation d'images ? In: La Revue IA. [en ligne]. Disponible sur : https://larevueia.fr/quest-ce-que-la-segmentation-dimages/ (consulté le 17 octobre 2023).

(7). Automatic Detection of Objects on Orthomosaic. In: Agisoft. Agisoft, [en ligne]. Disponible sur : https://agisoft.freshdesk.com/support/solutions/articles/31000162552-automatic-detection-of-objects-on-orthomosaic. (consulté le 12 octobre 2023).

(8). CVAT. In: CVAT. CVAT, [en ligne]. Disponible sur : https://www.cvat.ai. (consulté le 15 novembre 2023).

(9). Ultralytics. In: Ultralytics. Ultralytics, [en ligne]. Disponible sur : https://www.ultralytics.com/fr. (consulté le 15 novembre 2023).

(10). Comparative Analysis of YOLOv8 and Mask-RCNN for Meniscus Pathology. In: Simran Jain. LinkedIn, [en ligne]. Disponible sur : https://www.linkedin.com/pulse/comparative-analysis-yolov8-mask-rcnn-meniscus-pathology-simran-jain-xovie#:~:text=Results%3A,despite%20the%20lower%20MAP%20value. (consulté le 24 février 2024).

(11). YOLOv8 Guide. In: Viso Suite AI. Viso Suite AI, [en ligne]. Disponible sur : https://viso.ai/deep-learning/yolov8-guide/. (consulté le 15 novembre 2023).

(12). Augmentation Settings YOLOv8. In: Ultralytics. Ultralytics, [en ligne]. Disponible sur : https://docs.ultralytics.com/usage/cfg/#augmentation-settings. (consulté le 28 janvier 2024).

(13). Image Processing. In: OpenCV. OpenCV, [en ligne]. Disponible sur : https://docs.opencv.org/4.x/d3/dc0/group__imgproc__shape.html#ga556a180f43cab22649c23ada36a8a139. (consulté le 10 novembre 2023).

(14). Itertools — Functions creating iterators for efficient looping. In: Python Software Foundation. Python Documentation, [en ligne]. Disponible sur : https://docs.python.org/3/library/itertools.html. (consulté le 29 janvier 2024).

(15). Skeletonize. In: scikit-image. scikit-image, [en ligne]. Disponible sur : https://scikit-image.org/docs/stable/auto_examples/edges/plot_skeleton.html. (consulté le 29 janvier

2024).

(16). Clustering. In: scikit-learn. scikit-learn, [en ligne]. Disponible sur : https://scikit-learn.org/stable/modules/clustering.html. (consulté le 2 février 2024).

(17). Hierarchical Clustering. In: scikit-learn. scikit-learn, [en ligne]. Disponible sur : https://scikit-learn.org/stable/modules/clustering.html#hierarchical-clustering. (consulté le le 2 février 2024).

(18). Go-ICP. In: Yang Jiaolong. GitHub, [en ligne]. Disponible sur : https://github.com/yangjiaolong/Go-ICP. (consulté le 7 avril 2024).

(19). Go-ICP Cython. In: aalavandhaann. GitHub, [en ligne]. Disponible sur : https://github.com/aalavandhaann/go-icp_cython. (consulté le 7 avril 2024).

(20). RANSAC Regressor. In: scikit-learn. scikit-learn, [en ligne]. Disponible sur : https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.RANSACRegressor.html. (consulté le 25 avril 2024).

(21). YOLO Data Augmentation Explained: Turbocharge Your Object Detection Model. In: Rumn. Medium, [en ligne]. Disponible sur : https://rumn.medium.com/yolo-data-augmentation-explained-turbocharge-your-object-detection-model-94c33278303a. (consulté le 19 mars 2024).

LISTE DES ANNEXES

Annexe n°1 : Précisions sur les métriques d'entraînement

La **précision** (P) et le **rappel** (R) sont des métriques définies par les équations suivantes :

$$P = \frac{VP}{VP + FP} \quad / \quad R = \frac{VP}{VP + FN}$$

- VP = vrais positifs
- FP = faux positifs
- FN = faux négatifs

L'image ci-contre, permet d'illustrer ces principes. Les faux négatifs sont alors des manques de détection, les faux positifs de mauvaises détections et les vrais positifs des détections réussies.

Nous pouvons aussi illustrer l'**IoU** (Intersection Over Union) avec l'équation suivante :

$$IoU = \frac{|O_p \cap O_r|}{|O_p U O_r|}$$

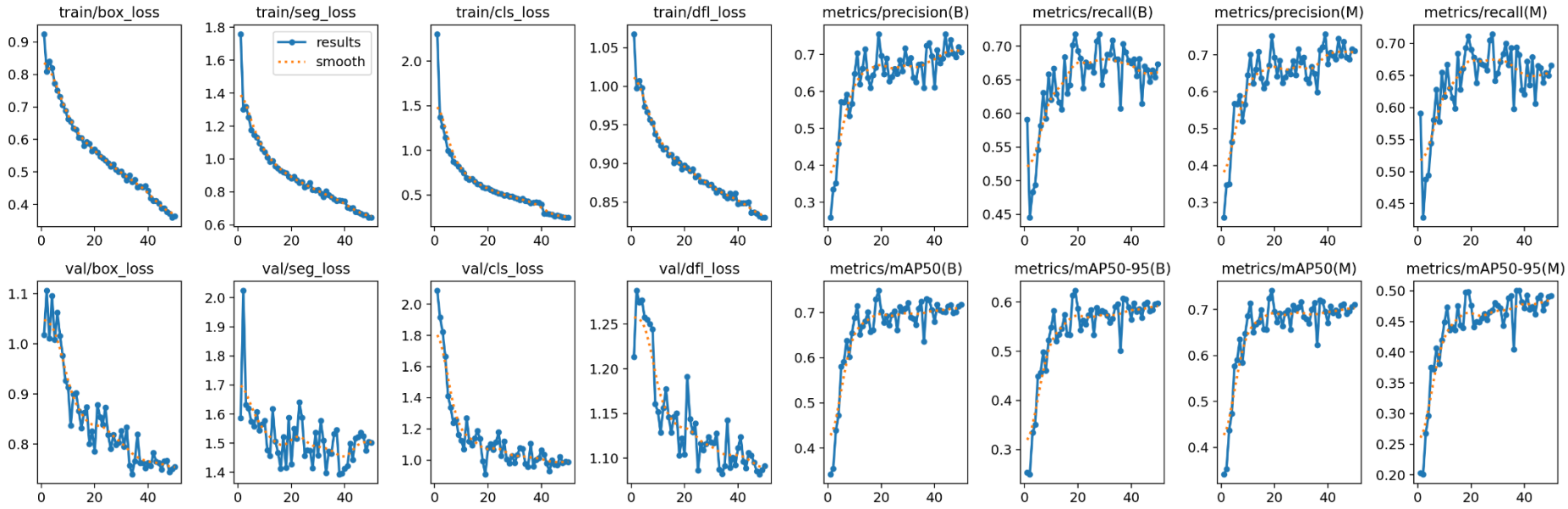
Où Op et Or sont respectivement les objets prédits et réels, annotés. La figure à droite nous donne un exemple avec en vert et blanc Or et en rouge Op.

Enfin le **mAP** (mean Average Precision) se définit mathématiquement tel qu'il suit :

$$AP = \int_0^1 P(R)\, dR \quad / \quad mAP = \frac{1}{C}\sum_{C=1}^{C} AP_C$$

On reprend ici le recall (R) et la precision (P). AP représente la précision moyenne sur une seule classe, mAP s'étendant sur l'ensemble des classes (C = nombre de classes).

Annexe n°2 : Résultats de l'entraînement sur le jeu de données *markers_and_digits_v2*



On observe sur ces résultats :

- Les métriques Precision/Recall/mAP augmentent au fil des époques, que ce soit pour la détection ou segmentation. On retrouve néanmoins une légère diminution autours de la 30^{e} époque.
- Les lr, taux d'apprentissage, augmentent avant de se stabiliser
- Les pertes/loss diminuent de manière générale

L'ensemble de ces indications tendent à montrer que le modèle apprend bien tant pour la détection que la segmentation. Cependant les fluctuations importantes observées le long des courbes d'apprentissage suggèrent une mauvaise gestion de la complexité du modèle. Cela peut venir de plusieurs facteurs présents dans notre jeu de données : le biais de classe important, le manque de données (on prescrit 1500 instances annotées par classes), ou un problème de généralisation. Cette dernière possibilité est évincée car les graphiques de validation avec l'observation des métriques Precision/Recall/mAP montrent une augmentation de ceux-ci. Enfin la validation permet de valider l'hypothèse de biais de classe. En effet, les métriques sont très bonnes pour la classe cross et dot mais très mauvaises sur les autres classes.

Annexe n°3 : Les différents variants d'augmentation du jeu cross_only.

Les 5 variants utilisées sont présentés dans le tableau ci-dessous :

	VARIANT 1	VARIANT 2	VARIANT 3	VARIANT 4	VARIANT 5
HSV_H	0,7	**0,2**	**0,2**	**0,2**	**0,2**
HSV_S	0,7	**0,8**	**0,8**	**0,8**	**0,8**
HSV_V	0,7	**0,6**	**0,6**	**0,6**	**0,6**
DEGREES	-60	**-10**	-20	-15	-5
SHEAR	-60	**-10**	-20	-15	-5

Les paramètres d'augmentation présenté dans les premières colonnes ont été implémentés directement lors de l'entraînement. Cela permet de rendre le modèle plus robuste et l'aide à généraliser sur de nouvelles entrées. Ces augmentations interviennent de manière aléatoire tout au long de l'entraînement. Ultralytics admet d'ailleurs déjà des paramètres d'augmentation par défaut (11).

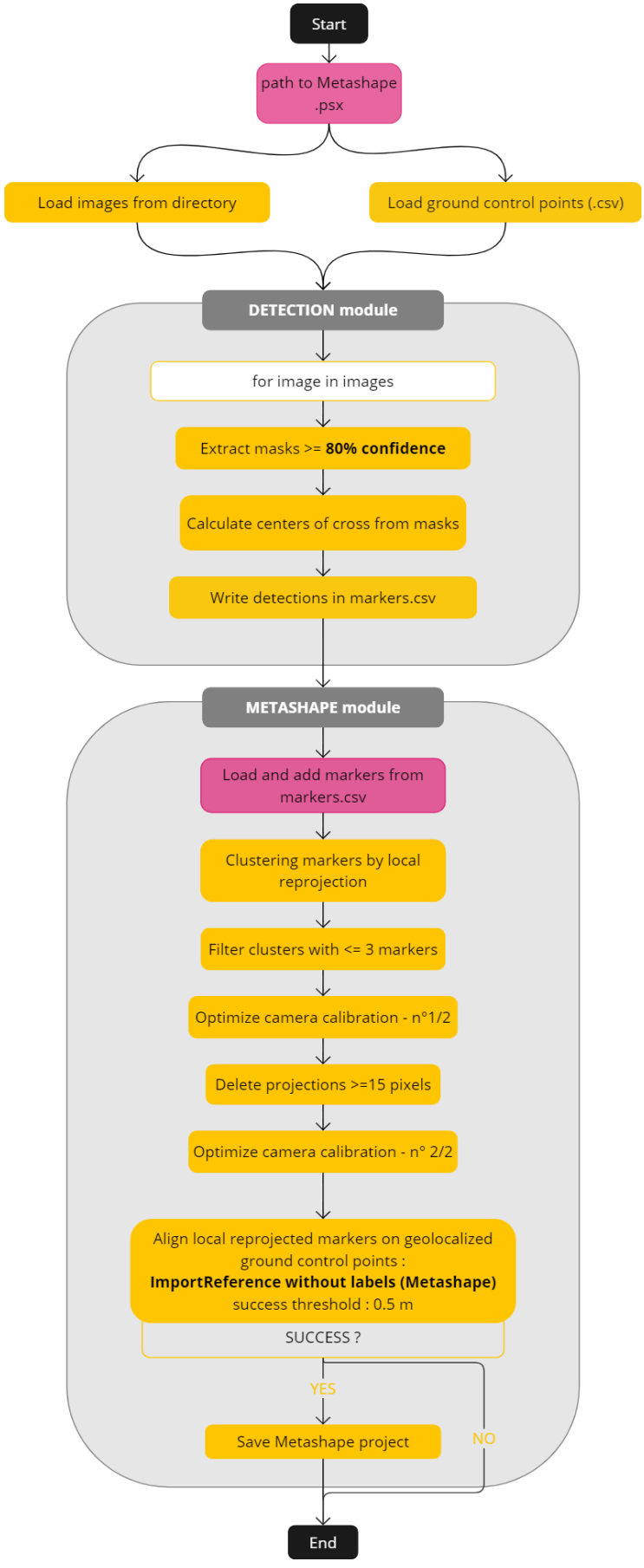
Les paramètres utilisés ici sont les suivants (20) :

- **HSV_H/_S/_V** : fait varier les niveaux HSV – Teinte, Saturation, Value - sur l' image d'entrée. Varie de 0 à 1. Le paramètre par défaut étant de 0.015, 0.7, 0.4 respectivement par défaut.
- **DEGREES** : effectue une rotation sur l'image d'entrée. Varie de +/- 180 degrés. Est nul par défaut.
- **SHEAR** : Admet des déformations géométriques le long des axes x et y. Permet de simuler différentes prises de vues possibles par l'opérateur. Est nul par défaut.

En jouant sur ces 5 paramètres l'idée a été de rendre le modèle plus robuste à l'utilisation de différentes couleurs de peintures mais aussi à la visualisation des croix sous différents angles.

Ci- contre, nous pouvons visualiser plusieurs de ces augmentations au sein d'un batch/lot d'entraînement. En effet, en haut à gauche nous observons un assombrissement de l'image par exemple tandis qu'en bas à droite une variation de saturation et de Teinte a été effectuée.

Annexe n°4 : Flowchart du script PyAutoMarkers



Annexe n°5 : Écarts entre coordonnées théoriques et coordonnées reprojetées par chantier correctement géoréférencé

Plusieurs chantiers ont été géoréférencés en s'appuyant sur un pointage des points de calage manuel (position connue). Les points de calages déterminés de manière automatique grâce à la méthode skeleton ont été reprojeté sur ceux-ci. Nous en retirons la moyenne des erreurs de position 3d par chantier.

eM et **eP** sont calculé par chantier et correspondent à l'erreur de reprojection en mètre et en pixel.

		Position connue				Position estimée							Ecarts moyens	
Nom du chantier	Gcp label	x	y	z	Cluster label	x	y	z	Delta X	Delta Y	Ecart alti	Ecart plani	alti	plani
24X09717	1	535221,947	191838,988	10,654	5	535221,944	191838,986	10,644	0,004	0,002	0,009	0,004	0,024	0,022
eM	2	535222,003	191839,840	10,628	4	535222,004	191839,842	10,625	-0,001	-0,001	0,002	0,002		
0,01	3	535221,908	191840,571	10,600	1	535221,908	191840,594	10,565	-0,001	-0,024	0,035	0,024		
Ep	4	535220,649	191840,750	10,615	2	535220,600	191840,820	10,532	0,049	-0,071	0,082	0,086		
1,045	5	535220,443	191839,918	10,634	6	535220,448	191839,914	10,639	-0,005	0,004	0,005	0,006		
	6	535220,393	191839,092	10,653	0	535220,380	191839,089	10,640	0,013	0,002	0,012	0,013		
24X09681	1	1643129,459	9078672,109	41,127	1	1643129,456	9078672,113	41,133	0,003	-0,004	0,006	0,005	0,008	0,015
eM	2	1643130,229	9078676,384	40,695	4	1643130,211	9078676,381	40,699	0,018	0,003	0,004	0,018		
0,02	3	1643128,016	9078681,467	40,446	2	1643128,017	9078681,466	40,455	-0,001	0,001	0,009	0,001		
eP	4	1643131,882	9078684,982	40,421										
0,683	5	1643128,582	9078684,379	40,359	0	1643128,594	9078684,344	40,372	-0,012	0,035	0,012	0,037		
23X08569	1	1422058,111	4192393,612	49,818	1	1422058,111	4192393,607	49,830	0,001	0,004	0,012	0,004	0,013	0,010
eM	2	1422054,414	4192395,416	49,981	3	1422054,410	4192395,419	49,970	0,004	-0,003	0,011	0,005		
0,044	3	1422053,809	4192390,988	50,050	6	1422053,807	4192390,981	50,029	0,003	0,006	0,021	0,007		
eP	4	1422055,532	4192387,524	49,617	0	1422055,681	4192387,396	49,175	-0,150	0,128	0,442	0,197		
1,083	5	1422057,116	4192391,386	49,696	2	1422057,131	4192391,369	49,690	-0,015	0,017	0,006	0,023		
24X09680	25	1924636,173	3184191,568	329,057	3	1924636,171	3184191,577	329,115	0,002	-0,009	0,058	0,009	0,024	0,017
eM	26	1924635,919	3184189,124	329,036	4	1924635,933	3184189,128	329,036	-0,014	-0,004	0,000	0,015		
0,03	27	1924633,266	3184188,403	329,003	2	1924633,255	3184188,372	328,980	0,011	0,031	0,023	0,033		
erreur pixel	28	1924633,013	3184185,763	329,002	0	1924633,005	3184185,778	329,014	0,008	-0,015	0,012	0,017		

1,637	29	1924629,807	3184184,517	328,928	1	1924629,812	3184184,526	328,953	-0,005	-0,009	0,025	0,010		
24X09471	1	535872,091	191843,395	10,507	7	535872,098	191843,397	10,530	-0,008	-0,003	0,024	0,008	0,030	0,015
eM	2	535872,282	191844,106	10,442	6	535872,294	191844,108	10,499	-0,012	-0,002	0,057	0,012		
0,018	3	535872,960	191845,024	10,534	4	535872,965	191845,023	10,538	-0,005	0,001	0,004	0,005		
eP	4	535873,246	191846,372	10,408	5	535873,257	191846,374	10,417	-0,011	-0,002	0,010	0,011		
0,871	5	535874,226	191847,375	10,421										
	6	535874,542	191846,506	10,445	0	535874,545	191846,501	10,439	-0,003	0,006	0,006	0,006		
	7	535874,387	191845,623	10,493										
	8	535873,553	191844,386	10,982	3	535873,560	191844,385	10,985	-0,007	0,001	0,003	0,007		
	9	535872,896	191842,973	10,677	1	535872,942	191842,945	10,572	-0,046	0,028	0,105	0,053		
24X09486	1	535800,145	191915,307	10,187	5	535800,142	191915,315	10,196	0,003	-0,007	0,009	0,008	0,038	0,013
eM	~~1~~	~~535800,145~~	~~191915,307~~	~~10,187~~	~~11~~	~~535800,158~~	~~191915,280~~	~~9,988~~	~~-0,013~~	~~0,027~~	0,200	0,030		
0,014	2	535799,151	191916,295	10,208	4	535799,146	191916,293	10,211	0,005	0,003	0,002	0,006		
eP	3	535797,603	191916,806	10,214	2	535797,584	191916,796	10,217	0,019	0,010	0,003	0,022		
0,818	4	535796,318	191917,196	10,216	9	535796,314	191917,195	10,233	0,005	0,002	0,017	0,005		
	5	535795,148	191917,404	10,251										
	6	535795,708	191916,201	10,240	1	535795,708	191916,205	10,246	-0,001	-0,004	0,006	0,004		
	7	535797,048	191915,744	10,219	3	535797,063	191915,748	10,232	-0,016	-0,004	0,013	0,016		
	8	535798,642	191915,186	10,190	0	535798,652	191915,193	10,242	-0,010	-0,007	0,052	0,012		
24X09678	1	1643144,397	9078680,377	40,726	0	1643144,425	9078680,336	40,681	-0,028	0,041	0,045	0,049	0,024	0,043
eM	2	1643142,313	9078682,259	40,565	3	1643142,278	9078682,218	40,582	0,035	0,041	0,017	0,054		
0,039	3	1643137,514	9078680,406	40,670										
eP	4	1643132,016	9078681,332	40,361	2	1643132,028	9078681,324	40,381	-0,012	0,008	0,020	0,014		
0,293	5	1643130,077	9078679,207	40,558	1	1643130,090	9078679,258	40,570	-0,013	-0,051	0,012	0,053		
	6	1643128,005	9078681,464	40,431										
											0,032	**0,023**		

Optimisation d’une chaîne de production photogrammétrique : détection automatisée de points de calage

Mémoire d'Ingénieur C.N.A.M., Paris 2024

RÉSUMÉ

L'application Récolement de l'entreprise Syslor est dédiée à la reconstitution, via photogrammétrie, de nuages de points. Ce produit facilite l'élaboration de plans de récolement relatifs aux réseaux enterrés. Actuellement, le géoréférencement des nuages de points est soit automatique, à l'aide de cibles pré-paramétrées, soit, dans la majorité des cas, manuel, reposant sur le pointage de marquages peints. Pour parvenir à une automatisation complète du processus et éliminer la nécessité d'une intervention humaine, un algorithme de pointage des marquages peints a été développé. Un modèle de segmentation d'instances YOLOv8 a été entraîné, et des méthodes ont été explorées afin de pointer avec exactitude le centre des masques extraits et, finalement, faire correspondre les clusters issus des détections avec les points relevés, tout en maintenant une classe de précision A.

Mots-clés : Photogrammétrie / Géoréférencement / Automatisation / Deep Learning / Segmentation d’instance

---

SUMMARY

Syslor’s Recolement application is dedicated to the reconstruction of point clouds via photogrammetry. This product facilitates the creation of as-built plans for underground networks. Currently, the georeferencing of point clouds is either automatic, using pre-configured targets, or, in the majority of cases, manual, relying on the marking of painted targets. To achieve complete process automation and eliminate the need for human intervention, a painted marking detection algorithm has been developed. An instance segmentation model, YOLOv8, has been trained, and methods have been explored to accurately pinpoint the center of the extracted masks and, ultimately, match the clusters resulting from the detections with the surveyed points, while maintaining an A-grade precision class.

Keywords: Photogrammetry / Georeferencing / Automation / Deep Learning / Instance Segmentation