



Automatiser le photomontage des projets de construction

Bruno Della Casa

7 juillet 2025

Domaine Ingénierie et Architecture

Master conjoint UNIGE-HES-SO en développement territorial

Orientation Ingénierie géomatique HES-SO

Directeur : Dr. Adrien GRESSIN

Expert : Ing. Géomètre breveté Sébastien HÄMMERLI

Mémoire n° : 1124



**UNIVERSITÉ
DE GENÈVE**

Hes·so
Haute Ecole Spécialisée
de Suisse occidentale

RÉSUMÉ

Ce mémoire propose une alternative innovante pour améliorer la visualisation des projets de construction lors de la mise à l'enquête publique. Traditionnellement, les gabarits de construction permettent de visualiser l'emprise du bâtiment sur le terrain, mais ils ne montrent pas les éléments architecturaux détaillés tels que les balcons, les fenêtres, les avant-toits ou les décrochements de façade. Bien que ces éléments soient présents sur des plans ou des maquettes 3D, leur compréhension reste difficile pour les riverains non familiarisés avec ces représentations techniques.

L'objectif de ce travail est de permettre une visualisation complète et compréhensible du projet sur le terrain via une photo prise par un voisin. Pour ce faire, le projet repose sur plusieurs étapes clés :

- le calcul du référencement et de l'orientation des photos,
- la définition des paramètres de calibration de l'appareil photographique,
- l'élaboration d'une carte de profondeur,
- l'évaluation de la visibilité du projet sur l'image,
- la création d'un rendu texturé du projet le plus réaliste possible.

Cette approche vise à rendre la visualisation d'un projet plus accessible et intuitive pour l'ensemble des parties prenantes : riverains, architectes, autorités communales ou cantonales. Elle peut ainsi faciliter la compréhension du projet, améliorer la transparence du processus de mise à l'enquête et soutenir les démarches de concertation.

Mot-clé : Photogrammétrie, Projet de construction, Gabarits, Programmation, Carte de profondeur, Photomontage

REMERCIEMENTS

Je tiens à exprimer ma profonde gratitude à toutes les personnes qui ont contribué par leur aide à la réalisation de ce travail de master :

- Monsieur Adrien Gressin, Dr. ing. ENSG, professeur à la HEIG-VD et directeur de ce travail, pour ses conseils avisés, sa disponibilité et son accompagnement tout au long du projet. Son expertise et ses retours pertinents ont grandement contribué à la qualité de ce travail.
- Monsieur Sébastien Hämmerli, expert de ce travail, pour l'intérêt porté à mon travail et les remarques effectuées lors de la défense intermédiaire.
- Monsieur Raphaël Guenat, Architecte HES/SIA et Directeur de l'atelier PAT architecte SA, pour la fourniture de la maquette 3D d'un futur projet de construction.
- Madame Véronique et Monsieur Marc-Henry Keuffer Dit Barrelet, propriétaire du projet de construction, pour leur autorisation d'utilisation et pour m'avoir permis d'accéder à leur parcelle.
- Madame Corinne Della Casa, pour la relecture du rapport.
- Le bureau d'étude Rossier SA, pour m'avoir permis d'utiliser leur système de mesure GNSS.

ABRÉVIATIONS

Orientation externe	: Position et orientation d'une photographie
Calibration interne	: Détermination des paramètres internes de correction d'un appareil photo
IA	: Intelligences artificielles
RA	: Réalité augmentée
depthmapIA	: Valeur issue d'une carte de profondeur obtenue avec de l'IA
dproj	: Distance projetée sur un plan, ici sur le plan image des photographies
uv	: Coordonnées d'un pixel sur une image
valuesdepthmap	: Jeux de donnée (uv, depthmapIA, dproj)
GCP	: Points de contrôle au sol (Ground Control Points)
GPU	: Processeur graphique permettant de faire des calculs mathématiques à grande vitesse (Graphics Processing Unit)
CPU	: Processeur central de traitement (Central Processing Unit)
RLATC	: Règlement d'application sur l'aménagement du territoire et les constructions du canton de Vaud
ReLATeC	: Règlement d'exécution de la loi sur l'aménagement du territoire et les constructions du canton de Fribourg
WKT	: Well-Know Text, format standard en mode texte utilisé pour représenter des objets géométriques vectoriels
EPSG	: European Petroleum Survey Group, liste de code associé à des systèmes de coordonnées

TABLE DES MATIÈRES

Résumé	I
Remerciements	II
Abréviations	III
Table des matières	IV
Table des figures	VII
Liste des tableaux	IX
1 Introduction	1
1.1 Contexte	1
1.2 Objectif	2
2 État de l'art	3
2.1 Géoréférencement et orientation d'une image	4
2.1.1 Positionnement et orientation en temps réel	5
2.1.2 Positionnement et orientation avec GCP	5
2.1.3 Positionnement et orientation par rapport à des images géoréférencées et orientées	8
2.1.4 Conclusion	16
2.2 Maquette 3D de projet	17
2.3 Position du projet dans l'image	17
2.4 Rendu visuel du projet	17
2.4.1 Illumination de Phong	17
2.4.2 Lancer de rayons	19
2.4.3 Conclusion	20
2.5 Détermination de la visibilité du projet	21
2.5.1 Combinaison d'image ou stéréo-image	22
2.5.2 Monoplotting d'un nuage de points ou d'un maillage sur une image	23
2.5.3 Utilisation de deep learning	23
2.5.4 Conclusion	26
2.6 Récapitulatif de l'état de l'art	27
2.6.1 Géoréférencement et orientation d'un image	27
2.6.2 Maquette de projet	27
2.6.3 Position du projet dans l'image	27
2.6.4 Rendu visuel du projet	27
2.6.5 Détermination de la visibilité via carte de profondeur	27
3 Méthode proposée	28
3.1 Calibration interne et orientation externe de la photographie	29
3.1.1 Modèle photogrammétrique	29
3.1.2 Recherche des points homologues	29
3.1.3 Détermination approximative des paramètres de calcul	32
3.1.4 Détermination précise de l'orientation externe et de la calibration interne	34
3.1.5 Validation de la méthode	39
3.2 Position du projet	41
3.3 Nuage de points et carte de profondeur	41

3.3.1	Transformation simple de la carte de profondeur	44
3.4	Rendu du projet dans l'image	44
3.5	Calcul de visibilité du projet dans l'image	45
4	Application	47
4.1	Page d'accueil	47
4.2	Les projets	47
4.2.1	Liste des projets	47
4.2.2	Calcul de photomontage	48
5	Mise en œuvre sur un projet existant	49
5.1	Acquisition du modèle de base photogrammétrique	50
5.1.1	Acquisition terrain	50
5.1.2	Traitement Agisoft	50
5.1.3	Préparation des données	51
5.2	Spécialité du projet	52
5.3	Résultat	53
5.3.1	Résultat automatique	53
5.3.2	Résultat semi-automatiquement	54
6	Conclusion	55
6.1	Utilisation	55
6.2	Perspectives d'amélioration	56
6.2.1	Optimisation des traitements de calcul	56
6.2.2	Amélioration du rendu visuel du projet	56
6.2.3	Évolution de l'interface utilisateur	56
6.2.4	Intégration d'informations réglementaires et contextuelles	57
6.2.5	Modification des éléments supprimés	57
Annexe		58
A	Manuel	58
A.1	Installation	58
A.2	Manuel utilisateur	58
A.2.1	Démarrer l'application web	58
A.2.2	Tutoriel	58
A.2.3	Création d'un nouveau projet de photomontage	58
A.3	Manuel développeur	60
A.3.1	Dossier du code	60
A.3.2	Architecture du code	60
A.3.3	Diagramme des traitements du code	65
B	Modèle photogrammétrique utilisé	66
B.1	Banc de calibration	66
B.2	Modèle photogrammétrique dans un champ	67
B.2.1	Choix du lieu	67
B.2.2	Matériel utilisé	68

B.2.3	Acquisition terrain	69
B.2.4	Traitement bureau	69
C	Fonction mathématique	70
C.1	Méthode des moindres carrés	70
C.1.1	Résolution pour le calcul de points homologues 3D	71
C.1.2	Résolution de la DTL	72
C.2	Algorithme RANSAC	73
C.3	Produit scalaire	74
C.4	Séparation des données par cluster	75
C.4.1	K-Means	75
C.4.2	DBSCAN	75

TABLE DES FIGURES

1	Exemple de champ de garbarits	1
2	Exemple de garabits en toiture	2
3	Photomontage Mies van der Rohe de 1922	3
4	Réalité augmentée de panorama	4
5	Schéma de la procédure standard de création d'un photomontage	4
6	Système GNSS KickTheMap pour smartphone	5
7	Les différents éléments de l'équation de colinéarité	6
8	Schéma d'un réseau de neurones	9
9	Points homologues GIM-RoMa sur la paire d'image complexe	11
10	Points homologues GIM-DKM sur l'ensemble des paires d'images	11
11	Points homologues avec LoFRT sur des images drones	12
12	Points homologues LightGlue sur l'ensemble des paires d'images	12
13	Images utilisées pour le test de détection	13
14	Représentation du filtre par voisinage	14
15	Représentation du filtre par chaînage	15
16	Représentation du filtre épipolaire	15
17	Exemple de projet Archicad exporter en Wavefront	17
18	Illustration du modèle de Phong	18
19	Illumination diffuse	18
20	Illumination spéculaire	19
21	Facteur de brillance dans l'intensité spéculaire	19
22	Illustration de lancer de rayons	20
23	Comparaison des rendus	20
24	Étapes de retouche manuelle d'une photo	21
25	Distances des cartes de profondeurs	21
26	Exemple de photo et de carte de profondeur	22
27	Stéréo-image	22
28	Carte de profondeur de la photogrammétrie	23
29	Maillage issu d'un nuage de points	23
30	Carte de profondeur avec différents modèles IA	24
31	Comparaison entre Depth Anything et Depth Pro	25
32	Diagramme de synthèse des phases de traitement	28
33	Filtre des points homologues avec « cv2.findFundamentalMat »	30
34	Évolution des poids de correspondance relatifs à l'image principale selon trois différences d'orientation	31
35	Liste des étapes de la détection des points homologues par rapport à l'image utilisateur	31
36	Schéma de la détection des images de référence	32
37	Schéma des déterminations approximatives des inconnues	32
38	Exemple de détection de points faux par reprojection	33
39	Analyse de la DLT par RANSAC avec des valeurs fausses et peu précise	34
40	Banc de calibration	34
41	Erreurs selon des combinaisons de deux paramètres de calibration	35
42	Erreurs selon différentes combinaisons de paramètres de calibration	36

43	Illustration des points faux détectés par moindre carré	38
44	Points homologues pour déterminer la précision du modèle	39
45	Points homologues réduits pour déterminer la précision du modèle	40
46	Illustration de la projection du projet sur l'image	41
47	Séparation du nuage de points selon la prise de vue	41
48	Comparaison des valeurs terrains avec les cartes de profondeur IA	42
49	Schéma de visibilité du nuage de points par rapport à la photo	42
50	Calcul de visibilité des points	43
51	Résultat de l'analyse de visibilité du nuage de points	43
52	Transformation approximative des valeurs de la carte de profondeur	44
53	Deuxième transformation approximative des valeurs de la carte de profondeur	44
54	Illustration de calcul pixel par pixel de rendu par phong	45
55	Détermination de la valeur terrain du pixel	46
56	Page d'accueil de l'application	47
57	Page de la liste de projet	48
58	Exemple de page de projet	48
59	Projet de construction	49
60	Photographie drone de la parcelle du projet	49
61	Photographie drone de la parcelle du projet	50
62	Photographie drone de la parcelle du projet	50
63	Réduction du nuage de points	51
64	Maquette du projet originale et réduite	52
65	Maquette du projet avec le nuage de points	52
66	Emprise du projet	53
67	Résultat automatique du photomontage	53
68	Résultat semi-automatique du photomontage	54
69	Structure par arbre des dossiers de projet	59
70	Exemple de fichier <i>position_orientation.txt</i> d'orientation externe	59
71	Architecture de <i>projet.json</i> comprenant la liste des projets dans <i>appviewer</i>	60
72	Diagramme des étapes de programmation	65
73	Banc de calibration	66
74	Calcul Agisoft	67
75	Photos drones du secteur test	67
76	GoPro Max 360	68
77	REDcatch	68
78	DJI mini 2	69
79	Schéma d'acquisition des photographies	69
80	Produit scalaire	74
81	Projection d'un point sur un plan	74
82	Exemple de cluster avec K-Means	75
83	Séparation d'une donnée en 5 groupes avec K-Means	75

LISTE DES TABLEAUX

1	Classement des modèles IA de détermination de points homologues	10
2	Récapitulatif des résultats visibles de la détection de points homologues par IA	13
3	Comparaison de la détection de points via l'IA LightGlue et via le logiciel Agisoft	14
4	Comparaison des modèles IA de carte de profondeurs	24
5	Comparaison des modèles de carte de profondeurs	26
6	Erreur des points de contrôle sur le banc de calibration	35
7	Comparaison des résultats entre les valeurs réelles et calculées	39
8	Analyse de la précision sur les points de contrôle	40
9	Comparaison des résultats entre les valeurs réelles et calculées avec des points homologues réduits	40
10	Coordonnées des points de contrôle du banc de calibration	67

1 INTRODUCTION

1.1 Contexte

Dans le canton de Vaud, les constructions nouvelles ou les agrandissements sont soumis à autorisations municipales selon l'art. 68 du règlement d'application de la loi sur l'aménagement du territoire et les constructions (RLATC). Dans ce cadre, certaines communes exigent systématiquement la pose de gabarits de construction, tandis que d'autres peuvent en demander la mise en place en fonction de la nature ou de l'impact du projet. À titre de comparaison, le canton de Fribourg impose quant à lui la pose de gabarits sur l'ensemble de son territoire, comme le stipule l'article 91 du règlement d'exécution de la loi sur l'aménagement du territoire et les constructions (ReLATEC).

Cette pratique permet de matérialiser le volume du futur bâtiment sur le terrain afin de faciliter l'évaluation du projet par les riverains et les autorités. La procédure pour la mise en place de gabarit est la suivante :

- Implantation sur site et relevé des altitudes au sol de la position du projet par un géomètre.
- Mise en place de gabarit dont la hauteur est calculée par rapport à l'altitude au sol et à l'altitude du projet.

Actuellement, il n'y a pas de norme sur la façon de poser les gabarits de construction. Suivant les entreprises mandatées, ceux-ci peuvent être implantés aux angles de la toiture ou par rapport à l'emprise au sol du bâtiment. Dans ce dernier cas, il est probable que la hauteur des gabarits ne soit pas recalculée en fonction du nouveau point de départ mais simplement reprise depuis l'altitude prévue pour la corniche, ce qui engendre une imprécision sur la représentation réelle du volume bâti.

De plus, ces gabarits présentent plusieurs limites : ils ne permettent pas de représenter précisément des éléments comme les cheminées ou les balcons qui dépassent de l'emprise générale du bâtiment. De plus, l'aspect esthétique du projet n'est pas mis en valeur et il devient difficile pour les voisins de repérer facilement l'emplacement des ouvertures. Lorsqu'il s'agit de projets de grande envergure, comme des quartiers entiers, il est encore plus complexe d'obtenir une vision réaliste de l'ensemble.



FIGURE 1 – Exemple de champ de garbarits ¹.

L'installation de ces gabarits n'est également pas aisée dans des situations particulières, telles que lors de travaux de surélévation ou lorsque l'accès à la parcelle est limité.

1. source : <https://dievolkswirtschaft.ch/fr/2019/10/trente-indicateurs-dun-coup/>



FIGURE 2 – Exemple de garabits en toiture².

Les projets de construction de tours ou d'antennes, en raison de leur hauteur, ne peuvent pas être représentés efficacement sur le terrain à l'aide des gabarits traditionnels.

Une solution envisageable pour pallier ces limitations consiste à créer une maquette numérique détaillée du projet, permettant ainsi de générer des vues en 3D. Cependant, bien que cette approche offre des avantages, elle n'est pas idéale. En effet, elle ne permet pas à chaque résident de visualiser précisément l'emprise du projet depuis son domicile. Le rendu peut parfois paraître peu réaliste et ressembler à une simple illustration publicitaire. De plus, le traitement des données nécessaires à cette modélisation peut se révéler très lourd du point de vue humain et logiciel et peut constituer un frein dans le processus de conception.

La création de photomontage de projet de construction est également utilisée de manière occasionnelle et souvent pour des projets de petite envergure. Cette approche présente l'avantage de permettre une visualisation réaliste de l'implantation du bâtiment dans son environnement existant. Toutefois, sa mise en œuvre reste complexe et peu adaptée à une reproduction à partir de nombreux points de vue.

Un photomontage réaliste est généralement réalisé manuellement à l'aide d'un logiciel de traitement d'image. Cette méthode nécessite :

- Une photographie géoréférencée et orientée
- Une modélisation 3D du bâtiment
- Calcul de la position du projet dans l'image
- Définition manuelle d'un masque de transparence dans un logiciel de retouche permettant de définir la visibilité du projet en fonction de sa position relative aux éléments de l'image (devant ou derrière).

Ce processus, long et répétitif, doit être appliqué individuellement à chaque image capturée sur le terrain. Des compétences en géomatique et en graphisme sont nécessaires afin de garantir la conformité du positionnement du projet dans l'image. L'exactitude du photomontage est primordiale, notamment pour prévenir d'éventuels litiges juridiques liés à une discordance entre la visualisation et la réalisation effective du projet.

1.2 Objectif

Ce travail a pour objectif de développer une application simple et intuitive, accessible à tout type d'utilisateur, permettant de visualiser automatiquement un projet de construction intégré dans une photographie prise directement depuis le terrain. L'ambition principale est de réduire au maximum la complexité d'utilisation de l'outil, de manière à ce que l'utilisateur n'ait qu'à importer une photographie, sans nécessiter de compétences techniques particulières. L'utilisateur, qu'il soit un riverain, un architecte ou une autorité communale, pourra ainsi obtenir une image réaliste du projet intégré dans son environnement réel.

L'application doit permettre l'utilisation de photographies prises depuis n'importe quel point de vue accessible autour du site du projet : qu'il s'agisse d'un balcon, d'une fenêtre ou d'une terrasse, l'outil doit s'adapter à la diversité des situations de prise de vue.

2. source : <https://gabaritsconstruction.ch/realisations/en-toiture/>

2 ÉTAT DE L'ART

« Dès le XIX^e siècle, les photographies retravaillées ont été utilisées dans des présentations pour illustrer l'impact d'un projet sur la vue urbaine et le paysage. » [Pousin *et al.*, 2023] Dans un premier temps, les photomontages étaient effectués avec des découpages et des collages et servaient principalement dans le domaine artistique ou pour de la propagande politique. L'architecte Ludwig Mies van der Rohe (1886-1969) est un des précurseurs de l'utilisation de photomontage que ce soit de manière artistique ou pour la conception de projet. Le projet de gratte-ciel proposé à Berlin en 1922 constitue un exemple emblématique (voir Figure 3).



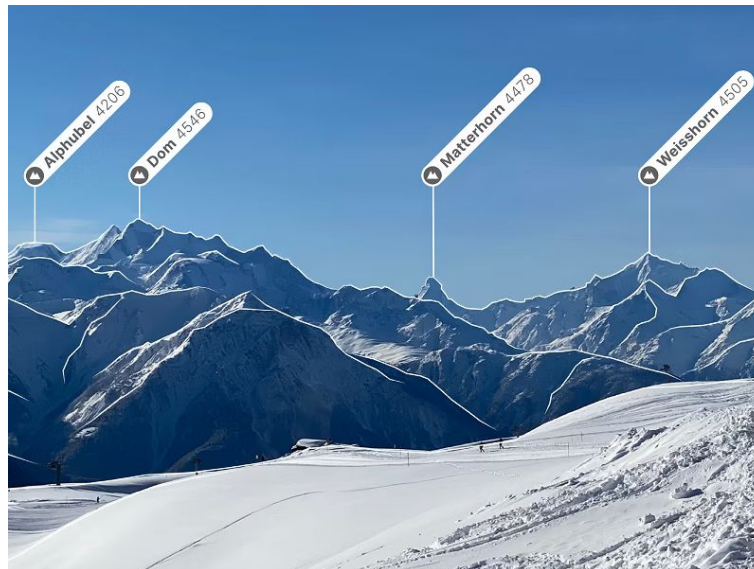
FIGURE 3 – Photomontage Mies van der Rohe de 1922³

A la fin du XX^e siècle, le photomontage connaît une transformation majeure avec l'avènement de l'informatique, de la numérisation des images et des logiciels de traitement graphique comme Adobe Photoshop (lancé en 1990). Ces outils permettent des montages plus précis, réalistes et facilement modifiables.

Depuis le début du XXI^e siècle, cette évolution se prolonge avec l'essor de technologies immersives telles que la réalité augmentée (RA). Bien que le concept de RA ait été initié dès 1966 par Ivan Sutherland, ce n'est qu'au cours des années 2000 que ces technologies commencent à se diffuser auprès du grand public grâce aux smartphones et aux casques de visualisation. La RA permet de superposer en temps réel des objets virtuels dans le monde réel. Actuellement, les projets de constructions utilisent principalement la RA pour faire des contrôles d'exécution ou pour visualiser l'agencement de pièce ou d'aménagement extérieur.

La réalité augmentée est également utilisée en combinant les coordonnées GNSS et le système d'orientation du smartphone. L'outil « panorama » dans l'application de @swisstopo est un exemple d'utilisation. Elle permet d'afficher en temps réel les noms des sommets de montagnes visibles à travers la caméra du téléphone (voir Figure 4).

3. source : https://www.researchgate.net/figure/a-Mies-van-der-Rohe-Friedrichstrasse-skyscraper-1921-Photomontage-Bauhaus-Archiv_fig5_324969880

FIGURE 4 – Réalité augmentée de panorama⁴

Actuellement, la démarche pour la création d'un photomontage pour un projet est la suivante :

- Obtention d'une photographie géoréférencée et orientée
- Une modélisation 3D du bâtiment
- Calcul de la position du projet dans l'image
- Définition manuelle d'un masque de transparence dans un logiciel de retouche permettant de définir la visibilité du projet en fonction de sa position relative aux éléments de l'image (devant ou derrière).

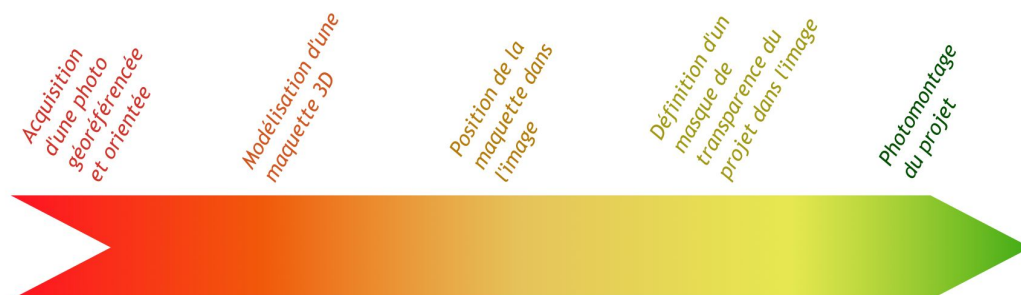


FIGURE 5 – Schéma de la procédure standard de création d'un photomontage

2.1 Géoréférencement et orientation d'une image

La position et l'orientation de la prise de vue est nécessaire à la création de photomontage de projet de construction. La qualité des valeurs de coordonnées et de direction doivent être prise en compte. En effet, il ne s'agit plus seulement d'afficher des éléments informatifs approximatifs, mais bien de positionner un objet architectural avec une précision comparable à celle d'un gabarit physique utilisé sur le terrain. La précision des gabarits est dépendante de l'altitude fournie par une mesure GNSS RTK et de la mesure manuelle de la hauteur dudit gabarit. En tenant compte des incertitudes liées au positionnement et à des facteurs environnementaux comme le vent, une estimation de précision de l'ordre de 20 cm semble raisonnable.

À l'inverse, les capteurs GNSS intégrés aux smartphones, dont la précision est de plusieurs mètres, ne permettent pas d'atteindre ce niveau d'exactitude. Par conséquent, une application de visualisation de projets intégrés dans des photographies ne peut pas s'appuyer exclusivement sur les capteurs mobiles comme le fait une application de type « panorama ».

4. source : <https://www.swisstopo.admin.ch/fr/realite-augmentee>

Il existe deux méthodes principales avec plusieurs variantes possibles permettant de positionner et orienter une photographie :

- En temps réel avec l'utilisation des capteurs internes (GNSS et IMU). Il s'agit de méthode utilisée par les smartphones lors de réalité augmentée.
- Issus d'un calcul photogrammétrique :
 - Positionnement et orientation en utilisant des points de référence (Ground Control Points : GCP).
 - Positionnement et orientation en utilisant des photographies déjà géoréférencées et orientées.

2.1.1 Positionnement et orientation en temps réel

Le positionnement et l'orientation d'une image en temps réel est dépendante des capteurs internes à l'appareil. Il doit au minimum être équipé d'un récepteur GNSS (Global Navigation Satellite System) et un capteur IMU (unité de mesure inertielle). Le capteur GNSS permet d'obtenir la position de la photo tandis que l'IMU, composée d'un gyroscope, d'un accéléromètre et d'un magnétomètre, fournit une estimation de l'orientation de l'appareil (inclinaison, azimuth, roulis). Cependant, cette méthode présente plusieurs limites. Elle ne permet pas de déterminer les paramètres de calibration interne de la caméra, comme la position du point principal (PPA), la focale ou les distorsions optiques. De plus, la qualité des capteurs intégrés dans les appareils grand public (comme les smartphones ou tablettes) reste souvent insuffisante pour garantir une position exacte, en raison de leur faible précision et des perturbations possibles du signal GNSS, notamment en présence d'obstacles tels que des bâtiments ou des arbres. Le système GNSS requiert une vue dégagée vers le ciel. Il devient donc inopérant lors de prises de vue en intérieur, par exemple depuis une fenêtre ou un balcon couvert.

Un exemple concret de cette approche est l'application *Pix4Dcatch* ou *KickTheMap*, qui permet en rajoutant un récepteur GNSS de meilleure qualité de capturer des images avec des précisions centimétriques de positionnement.



FIGURE 6 – Système GNSS KickTheMap pour smartphone⁵

2.1.2 Positionnement et orientation avec GCP

Cette deuxième méthode s'appuie sur les principes de la photogrammétrie qui repose sur la géométrie projective pour relier les coordonnées 3D d'un point dans l'espace à ses coordonnées 2D sur une image. Cette relation est formalisée par l'équation de colinéarité (1), qui est définie par l'alignement entre un point dans l'espace, le centre de projection de la caméra et le point pixel. Cette équation peut s'écrire de la manière suivante :

$$m = F - \frac{k^T F R (M - S)}{k^T R (M - S)} \quad (1)$$

5. source : <https://www.kickthemap.com/>

En la développant, trois équations sont obtenues :

$$x_m = x_c + f \cdot \frac{r_{11}(X_M - X_S) + r_{21}(Y_M - Y_S) + r_{31}(Z_M - Z_S)}{r_{13}(X_M - X_S) + r_{23}(Y_M - Y_S) + r_{33}(Z_M - Z_S)}$$

$$y_m = y_c + f \cdot \frac{r_{12}(X_M - X_S) + r_{22}(Y_M - Y_S) + r_{32}(Z_M - Z_S)}{r_{13}(X_M - X_S) + r_{23}(Y_M - Y_S) + r_{33}(Z_M - Z_S)}$$

$$z_m = 0$$

$\vec{\uparrow}^{cam} = (o, x, y, z)$: Système de coordonnées de la caméra

x, y : Centre du capteur de l'image

z : Axe de la visée de l'appareil photo

$\vec{\uparrow}^{glob} = (O, X, Y, Z)$: Système de coordonnées globale

$m = [x_m, y_m, 0]$: Coordonnée d'un point terrain sur le capteur, $z=0$ dans $\vec{\uparrow}^{cam}$

$M = [X_M, Y_M, Z_M]$: Coordonnée d'un point terrain dans $\vec{\uparrow}^{glob}$

$S = [X_S, Y_S, Z_S]$: Coordonnée du point de la caméra dans $\vec{\uparrow}^{glob}$

$$R_{glob}^{cam} = R_\omega \cdot R_\phi \cdot R_\kappa = \begin{bmatrix} r_{11} & r_{21} & r_{31} \\ r_{12} & r_{22} & r_{32} \\ r_{13} & r_{23} & r_{33} \end{bmatrix} : \text{Matrice de rotation du } \vec{\uparrow}^{glob} \text{ vers } \vec{\uparrow}^{cam}$$

$F = [x_c, y_c, -f]$: Point S dans $\vec{\uparrow}^{cam}$

(Distance focale f)

(x_c, y_c sont égales à 0 dans un modèle théorique)

$k^T = [0, 0, 1]$: vecteur de direction unitaire de l'axe des z en $\vec{\uparrow}^{cam}$

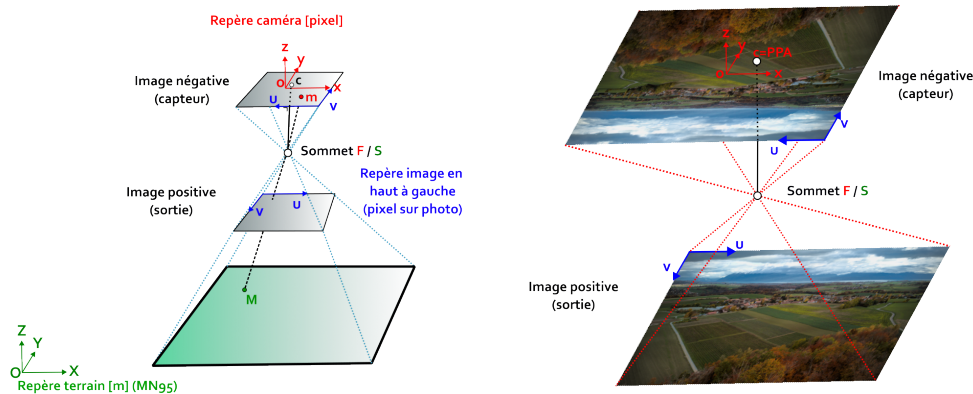


FIGURE 7 – Les différents éléments de l'équation de colinéarité

Pour que cette modélisation soit précise, il est nécessaire de connaître les paramètres internes de la caméra utilisée. Celle-ci vise à corriger les déformations introduites par l'optique de l'appareil. Deux types de distorsions principales doivent être prises en compte :

- La distorsion radiale est causée par la forme sphérique des lentilles, qui réfractent différemment la

lumière du centre vers le bord du capteur. Cette distorsion fait apparaître des courbes à la place de ligne droite. Elle est fortement visible avec des objectifs de petite focale (Exemple : 12 mm).

- La distorsion tangentielle se produit lorsque l'objectif et le plan de l'image ne sont pas parallèles ce qui engendre un déplacement linéaire des points de l'image

Enfin, un autre paramètre fondamental est le point principal d'autocollimation (PPA), il correspond au point d'intersection entre l'axe optique de la caméra et le plan du capteur. Ce point n'est généralement pas parfaitement centré sur l'image et doit être estimé avec précision pour garantir la qualité de la projection inverse.

La calibration interne comprend le PPA et les coefficients de correction des différentes distorsions de l'appareil photographique.

La calibration externe comprend les valeurs propres à la prise de vue soit la position et l'orientation.

Une image comporte six paramètres extrinsèques, trois de position $[X_S, Y_S, Z_S]$ et trois de rotation $[R_\omega, R_\phi, R_\kappa]$, ainsi qu'un certain nombre de paramètres de calibration interne dont la focale f , le centre principal PPA= $[x_c, y_c]$ et les coefficients de distorsion radiale K_{1-4} et tangentielle P_{1-2} . Cela représente donc au total au moins 11 inconnues à estimer. Chaque point de contrôle au sol (GCP) fournit deux observations (coordonnées image u_m et v_m), ce qui implique qu'au moins six GCP sont nécessaires pour contraindre le système. Toutefois, afin de garantir une solution stable et robuste, il est courant d'utiliser environ 15 GCP, idéalement répartis sur l'ensemble du champ de l'image pour couvrir toute la zone de distorsion et maximiser la précision.

Il existe de nombreux programmes permettant d'effectuer des calculs photogrammétriques. Nous pouvons notamment citer les programmes commerciaux Pix4D et Agisoft ainsi que le programme opensource en ligne de commande MicMac. L'avantage de ces trois logiciels est qu'ils permettent d'effectuer des géoréférencements sur des points GCP ou sur les coordonnées internes des photos. Agisoft a une API python qui permet ainsi de récupérer les valeurs d'un projet directement dans un script.

Agisoft a développé plusieurs modèles de calibration interne qui permettent de passer d'une caméra théorique (sans distorsion et sans PPA) vers une caméra réelle [Agisoft, 2025]. Le modèle principalement utilisé est « Frame ». Il correspond, en principe, au modèle de calibration pour des photos prises avec un smartphone :

$$\begin{aligned}
 x &= -x_m / f \\
 y &= y_m / f \\
 r &= \sqrt{x^2 + y^2} \\
 x' &= x(1 + K_1 r^2 + K_2 r^4 + K_3 r^6 + K_4 r^8) + (P_1(r^2 + 2x^2)) + 2P_2 xy \\
 y' &= y(1 + K_1 r^2 + K_2 r^4 + K_3 r^6 + K_4 r^8) + (P_2(r^2 + 2y^2)) + 2P_1 xy \\
 u &= w \cdot 0.5 + x_c + x' \cdot f + x' \cdot B_1 + y' \cdot B_2 \\
 v &= h \cdot 0.5 + y_c + y' \cdot f
 \end{aligned} \tag{2}$$

x_m, y_m : Coordonnées en pixel dans le système caméra parfait

f : Longueur focale en pixel

x_c, y_c : Point principal d'autocollimation(PPA)

K_{1-4} : Coefficient de distorsion radiale

P_{1-2} : Coefficient de distorsion tangentielle

B_{1-2} : Coefficient de distorsion affine

w, h : Largeur et hauteur d'image en pixel

La détermination de la position et de l'orientation d'une image implique la résolution d'un système d'équations non linéaires. Pour résoudre ce type de système, des valeurs initiales approximatives sont indispensables.

Une méthode couramment utilisée pour obtenir une première estimation de ces paramètres est la transformation linéaire directe (Direct Linear Transformation, DLT). Cette approche permet de linéariser le problème en exprimant les coordonnées image en fonction d'une combinaison linéaire des inconnues.

Le modèle DLT comporte 11 inconnues et fournit une estimation approximative des paramètres internes et externes de la caméra, sans tenir compte des distorsions optiques. Il constitue ainsi une étape utile pour initier des méthodes d'optimisation non linéaire plus précises

$$\begin{aligned} x_m &= \frac{L_1 X_M + L_2 Y_M + L_3 Z_M + L_4}{L_9 X_M + L_{10} Y_M + L_{11} Z_M + 1} \\ y_m &= \frac{L_5 X_M + L_6 Y_M + L_7 Z_M + L_8}{L_9 X_M + L_{10} Y_M + L_{11} Z_M + 1} \end{aligned} \quad (3)$$

La DLT contient 11 inconnues (L_{1-11}) tandis que l'équation de base (Equation 1) contient 9 inconnues ($S_M, S_M, S_M, R_\omega, R_\phi, R_K, x_c, y_c, f$). Il faut donc au minimum 6 points de références pour déterminer les paramètres de la DLT et 5 points de références pour l'équation de base. L'équation de base n'étant pas linéaire, il est nécessaire d'avoir des valeurs approchées des paramètres inconnus.

Les 11 paramètres de la DLT permettent de retrouver des valeurs approchées des inconnues de l'équation de colinéarité :

$$\begin{aligned} L &= \frac{-1}{\sqrt{L_9^2 + L_{10}^2 + L_{11}^2}} \quad (4) \\ x_c &\approx L^2(L_1 L_9 + L_2 L_{10} + L_3 L_{11}) \\ y_c &\approx L^2(L_5 L_9 + L_6 L_{10} + L_7 L_{11}) \\ c_x &\approx \sqrt{L^2(L_1^2 + L_2^2 + L_3^2) - x_c^2} \\ c_y &\approx \sqrt{L^2(L_5^2 + L_6^2 + L_7^2) - y_c^2} \\ p &\approx (c_x + c_y)/2 \\ r_{11} &\approx \frac{L(x_c L_9 - L_1)}{c_x}, r_{12} \approx \frac{L(y_c L_9 - L_5)}{c_y}, r_{13} \approx L \cdot L_9 \\ r_{21} &\approx \frac{L(x_c L_{10} - L_2)}{c_x}, r_{22} \approx \frac{L(y_c L_{10} - L_6)}{c_y}, r_{23} \approx L \cdot L_{10} \\ r_{31} &\approx \frac{L(x_c L_{11} - L_3)}{c_x}, r_{32} \approx \frac{L(y_c L_{11} - L_7)}{c_y}, r_{33} \approx L \cdot L_{11} \\ \begin{bmatrix} X_S \\ Y_S \\ Z_S \end{bmatrix} &\approx - \begin{bmatrix} L_1 & L_2 & L_3 \\ L_5 & L_6 & L_7 \\ L_9 & L_{10} & L_{11} \end{bmatrix}^{-1} \cdot \begin{bmatrix} L_4 \\ L_8 \\ 1 \end{bmatrix} \end{aligned}$$

2.1.3 Positionnement et orientation par rapport à des images géoréférencées et orientées

Le principe fondamental de cette méthode repose sur l'utilisation d'images de référence dont les paramètres de calibration interne et externe sont connus dans le but d'orienter une image cible. Cette approche s'apparente à la précédente car elle repose sur l'équation de colinéarité (1) comme fonction de base. La différence est qu'elle utilise la détection de points homologues entre des paires d'image à la place d'avoir des points de références. La méthodologie correspond à chercher des points similaires entre les images de référence et l'image à traiter (points homologues). Plusieurs possibilités de détection de points homologues existent :

- Saisie manuelle des points homologues dans les images
- Détection des points homologues par recherche de points d'intérêts et d'appariement entre les images à l'aide de descripteur. Un algorithme couramment utilisé pour ce type de traitement est SIFT (Scale-Invariant Feature Transform)
- Détection des points homologues par IA

La **saisie manuelle** consiste à sélectionner manuellement des points communs entre les images. Cette méthode, bien que précise, présente plusieurs inconvénients majeurs dans un cadre de traitement automatisé. Elle requiert un temps considérable, d'autant plus qu'un nombre suffisant de points (généralement plus de 10) doit être défini pour garantir une orientation stable et précise. De plus, la qualité de l'orientation dépend fortement de la pertinence des points sélectionnés, ce qui implique des compétences spécifiques en photogrammétrie.

Les familles d'algorithmes reposant sur une méthode d'extraction de points d'intérêt et d'appariement de ces points entre deux images sont les plus répandues actuellement dans les logiciels de photogrammétrie. « Le comportement de l'image au voisinage de chaque point d'intérêt détecté est décrit au moyen d'un descripteur qui lui est associé, la mise en correspondance des points homologues s'effectuant par la suite dans l'espace de ces descripteurs. » [Le Bris, 2010]. Un algorithme couramment utilisé pour ce type de traitement est **SIFT** (Scale Invariant Feature Transform). Il est disponible librement et présente l'avantage d'être invariant à l'échelle et à la rotation locale, ce qui le rend robuste pour certaines variations de point de vue. Cependant, il montre ses limites lorsque les images présentent des différences importantes d'éclairage, de saison ou d'angle de prise de vue. Par exemple, deux photographies d'un même lieu prises à des périodes différentes ou selon des directions très divergentes peuvent ne produire que peu, voire aucun point homologue détectable via SIFT.

Les approches basées sur l'**intelligence artificielle**, et notamment sur le deep learning, offrent une alternative plus flexible et performante pour la détection de points homologues. Ces méthodes sont capables de généraliser la notion de similarité visuelle, de manière similaire à un être humain, en prenant en compte le contexte global de l'image. Elles sont donc plus robustes face aux variations d'éclairage, de saison et de perspective.

Le deep learning repose sur l'utilisation de réseaux de neurones artificiels. On distingue trois types de neurones schématisés sur la figure 8. Les neurones d'entrée reçoivent les données initiales, les neurones des couches cachées traitent et transforment ces données et les neurones de sortie produisent les résultats du traitement. Ces neurones sont interconnectés et chacun des liens possède un poids spécifique.

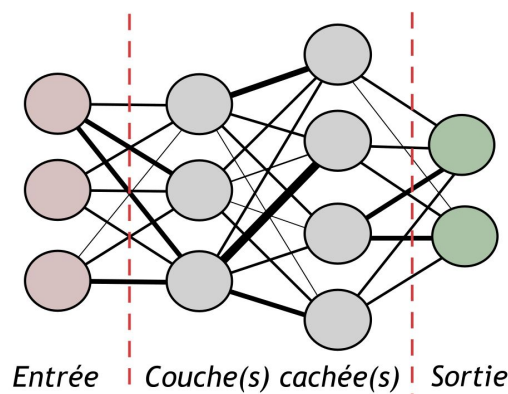


FIGURE 8 – Schéma d'un réseau de neurones

Les couches cachées peuvent être composées d'un nombre variable de rangées de neurones. Ces neurones sont composés de deux fonctions principales (pré-activation ou partie linéaire et activation). La pré-activation consiste à calculer la somme pondérée des valeurs issues de la couche précédente, à laquelle on ajoute un biais. Cette somme est ensuite transformée par la fonction d'activation. Après la propagation des données à travers toutes les couches cachées, le réseau génère une sortie correspondant à une prédiction pour chaque neurone de la couche de sortie.

Pour que le résultat soit pertinent, le modèle doit être entraîné à l'aide d'un jeu de données comportant des exemples et les résultats attendus (par exemple : une image de chien doit indiquer chien). En fournissant les données d'entrée ainsi que les résultats attendus, le modèle ajuste automatiquement les poids des connexions et les biais des neurones pour minimiser l'erreur entre sa prédiction et la vérité. Le bon fonctionnement du réseau repose donc sur la qualité et la justesse des données d'apprentissage.

Dans le cas d'une image, les données en entrée peuvent être les valeurs RGB de chaque pixel. Chaque neurone d'entrée ne peut recevoir qu'une seule valeur, ce qui signifie que pour une image de 100x100 pixels, le réseau reçoit 30'000 données (100 x 100 x 3 canaux de couleur). Cependant, cette approche a pour inconvénient de perdre la notion de position spatiale des pixels, car l'image 2D est transformée en un vecteur 1D.

Pour conserver cette structure spatiale, on utilise des réseaux de neurones convolutifs (**CNN** – Convolutional Neural Networks). Le principe général reste similaire à celui d'un réseau de neurones classique, mais les

données d'entrée sont traitées par des opérations de convolution, de sous-échantillonnage et de normalisation. Ces opérations permettent d'analyser des groupes de pixels tout en préservant les relations spatiales. Les CNN sont utilisés pour la classification, la détection, la segmentation ou encore l'appariement d'images. Le Deep Learning permet, par exemple, de détecter des points communs entre des images. Plus récemment, de nouvelles approches reposant sur les *Vision Transformers* (ViT) ont émergé et ont tendance à être privilégiées. Le Deep Learning est optimisé pour des calculs sur GPU car il est spécialement conçu pour le calcul parallèle massif, ce qui le rend bien plus adapté que le CPU pour ces tâches de traitement.

Il existe plusieurs modèles IA de détection de points homologues entre 2 images. Le site <https://paperswithcode.com/> fournit gratuitement des codes, des ensembles de données et l'évaluation de résultat. En utilisant la tâche « Image Matching »⁶ sur le jeu de donnée « ZEB », des ressemblances entre les différents modèles IA sont présents et sont indiquées dans la Table 1.

Comme pour les algorithmes de type SIFT, le modèle IA détecte des points d'intérêts pour chaque image et recherche ensuite les similitudes entre eux. La détection de points d'intérêt se fait image par image. Il faut ensuite lui indiquer les paires d'images à traiter. Il en ressort deux tableaux contenant les coordonnées images des points homologues pour chaque image et un tableau des scores de justesses des points homologues.

Classement	Code Git	Modèle IA	Année
1	xuelunshen/gim	GIM-RoMa	2025
2	xuelunshen/gim	GIM-DKM	2024
3	parskatt/roma	RoMa	2023
4	parskatt/dkm	DKM	2022
5	xuelunshen/gim	GIM-LoFTR	2024
6	xuelunshen/gim	GIM-LightGlue	2024
7	zju3dv/LoFTR	LoFTR	2021
8	ubc-vision/image...	RootSIFT	2012
9	cvg/lightglue	LightGlue	2023
10	magicleap/SuperGlue	SuperGlue	2019

TABLE 1 – Classement des modèles IA de détermination de points homologues

Le code « xuelunshen/gim » reprend des modèles IA existants (RoMa, DKM, LoFTR et LightGlue). Deux modèles les plus performants selon « paperswithcode » (GIM-RoMa et GIM-DKM) et deux modèles indépendants sans GIM (LoFTR et LightGlue) ont été testés. Afin d'évaluer et comparer les performances de ces modèles, trois paires d'images, acquises avec des appareils différents, ont été sélectionnées selon leur niveau de complexité :

- Une paire complexe, avec des points de vue très différents et peu d'éléments communs clairement identifiables.
- Une paire moyenne, présentant des points de vue distincts mais avec plusieurs éléments communs visibles.
- Une paire simple, avec des points de vue rapprochés et de nombreux éléments partagés.

Tous les modèles ont été testés en priorité sur la paire complexe. Les images sont réduites pour avoir une largeur de maximum 1'000 pixel. En cas de résultats incohérents, des tests supplémentaires ont été réalisés sur les paires moins complexes.

GIM-RoMa

Ce modèle permet de détecter un grand nombre de points homologues sur la paire complexe. Toutefois, il nécessite une carte graphique NVIDIA pour fonctionner correctement. Sur un ordinateur équipé uniquement d'un processeur Intel Core i13 3GHz et de 8 Go de RAM, le calcul échoue et provoque un blocage du système.

6. <https://paperswithcode.com/sota/image-matching-on-zeb>

La Figure 9 montre les points homologues détectés sur la paire d'images complexe.



FIGURE 9 – Points homologues GIM-RoMa sur la paire d'image complexe

GIM-DKM

GIM-DKM présente beaucoup d'erreur avec l'appariement de points sur la paire complexe (voir première paire d'image sur la Figure 10). Il est plus efficace avec les deux autres paires avec une grande quantité de points.

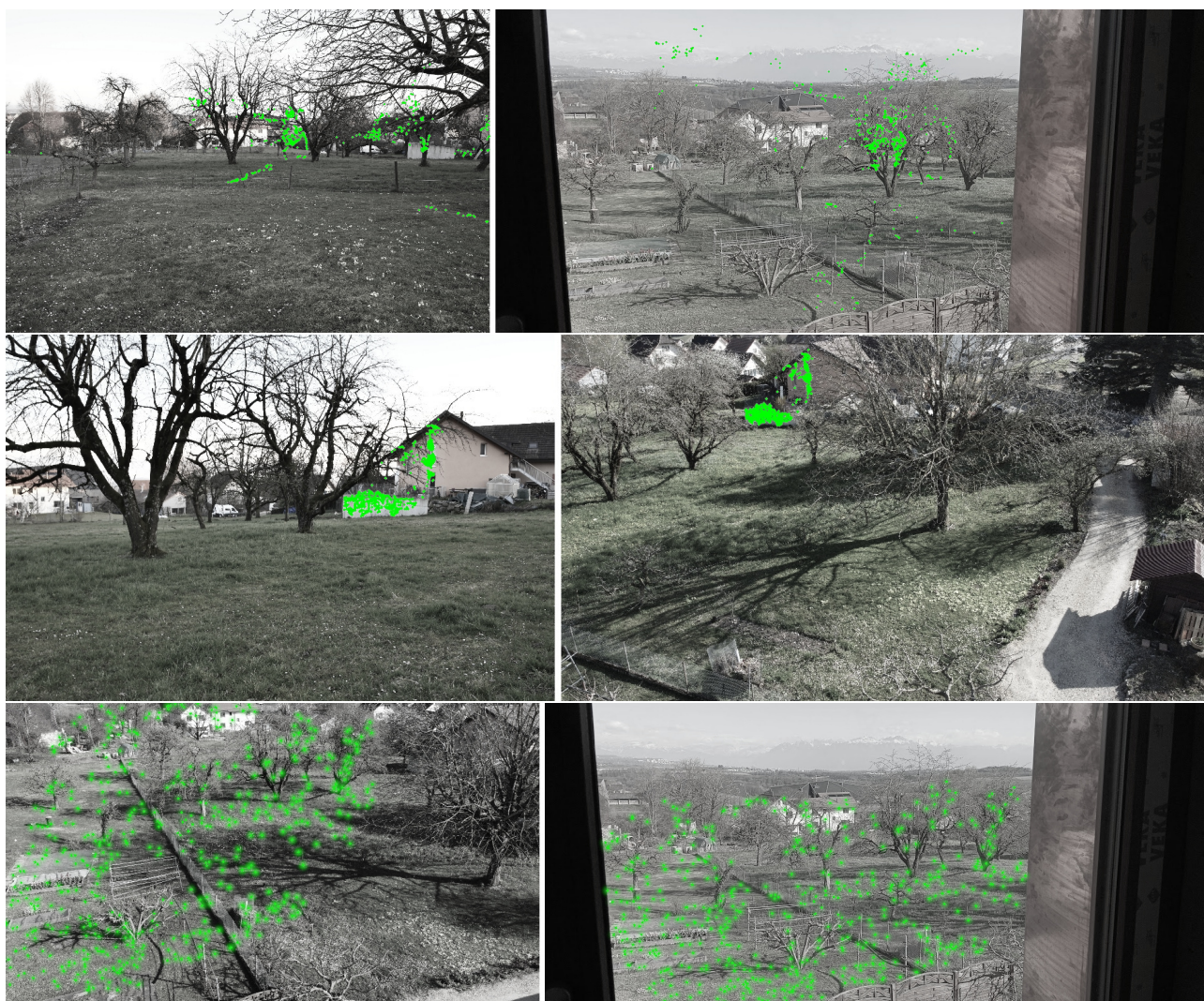


FIGURE 10 – Points homologues GIM-DKM sur l'ensemble des paires d'images

LoFTR

LoFTR nécessite des images de la même taille. La Figure 11 montre que la détermination de points homologues entre deux images acquises avec le même appareil et proche l'une de l'autre ne fonctionne pas. Il y a des points

repartis sur l'ensemble des images, ce qui ne devrait pas être le cas. De plus, en regardant visuellement les points, il est difficile de trouver des bonnes correspondances. Il semble que ce modèle ne soit pas fiable pour des images extérieures. L'analyse n'a pas été effectuée plus profondément sur d'autres images.

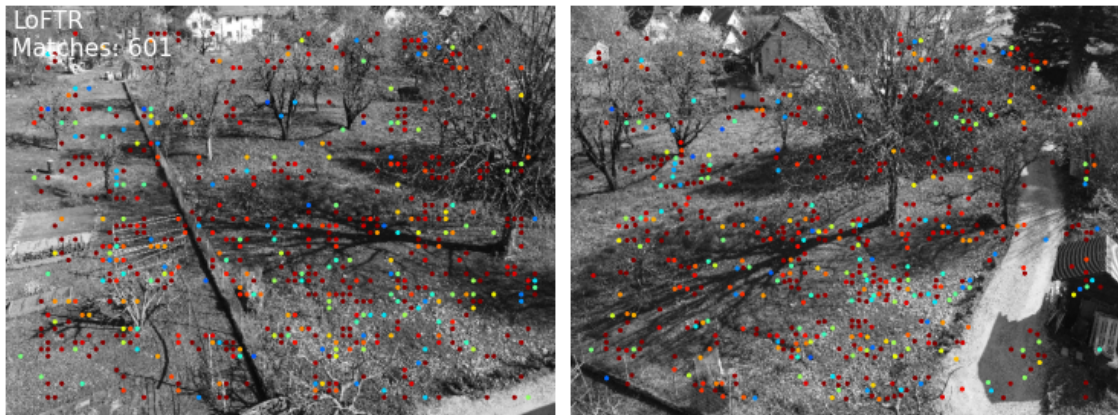


FIGURE 11 – Points homologues avec LoFRT sur des images drones

LightGlue

LightGlue détecte des points homologues cohérents à partir de la paire d'image moyenne. Les points homologues ont une densité plus faible que pour les précédents modèles comme présenté sur la Figure 12.

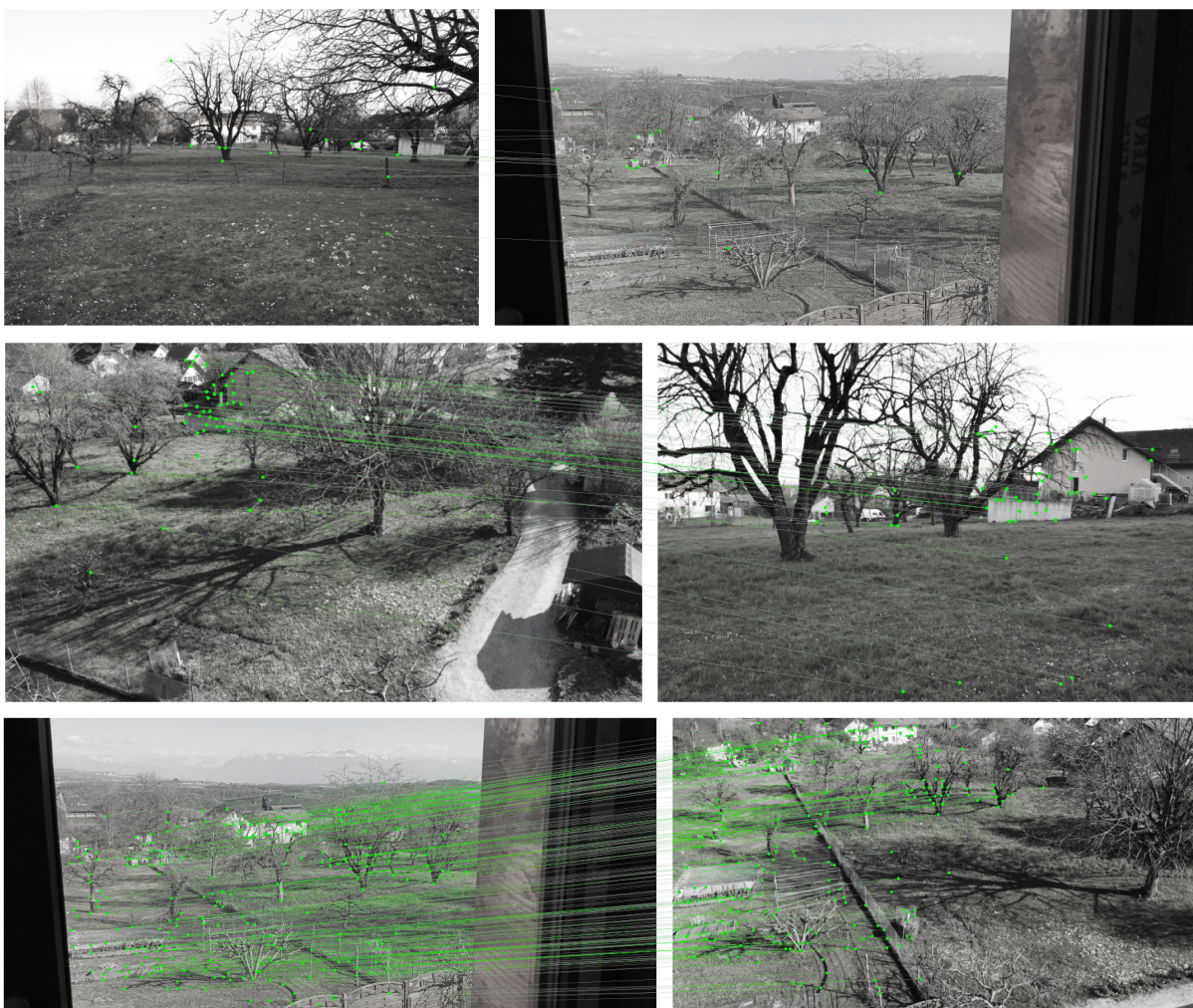


FIGURE 12 – Points homologues LightGlue sur l'ensemble des paires d'images

La Table 2 décrit de manière succincte les différents modèles IA testés. Le modèle le plus juste est GIM-RoMa et le modèle le plus rapide en tenant en compte d'un calcul sur CPU est LightGlue.

Modèle IA	Temps de calcul	Justesse
GIM-RoMa	10 secondes sur GPU <i>NVIDIA L40S de 48Go de RAM</i>	Grande quantité de points avec des bonnes correspondances et peu d'erreur flagrante visible directement sur une paire de photo complexe
GIM-DKM	10 secondes sur GPU <i>NVIDIA L40S de 48Go de RAM</i>	Grande quantité de points pour chaque paire d'image. Un problème de justesse des correspondances est visible avec la paire d'image complexe.
LoFTR	75 secondes sur CPU <i>Intel Core i3 3GHz et de 8 Go de RAM</i>	Testé uniquement sur une seule paire d'image. Beaucoup de points mais avec des erreurs flagrantes sur l'ensemble des points homologues.
LightGlue	10 secondes sur CPU <i>Intel Core i3 3GHz et de 8 Go de RAM</i>	De manière général, il y a moins de détection de points que sur les modèles GIM-RoMa et GIM-DKM. Un problème de justesse des correspondances est visible avec la paire d'image complexe.

TABLE 2 – Récapitulatif des résultats visibles de la détection de points homologues par IA

Afin d'analyser l'apport de l'IA, une détermination des points homologues a été effectuée avec les deux méthodes (IA avec LightGlue / algorithme de type SIFT effectué dans le logiciel Agisoft) via deux paires d'images représentées sur la Figure 13. Ces images ont des points de vue très différents d'un lieu.



FIGURE 13 – Images utilisées pour le test de détection

La Table 3 donne, pour les deux paires d'images, le nombre de points homologues détectés ainsi que leurs validités. Il est possible de constater que la détection par IA permet de trouver plus de points homologues.

Paires d'images	IA LightGlue	Algorithme de type SIFT de Agisoft
1	Valide : 134 points Invalide : 29 points Total : 163 points	Valide : 5 points Invalide : 5 points Total : 10 points
2	Valide : 19 points Invalide : 28 points Total : 47 points	Valide : 0 points Invalide : 0 points Total : 0 points

TABLE 3 – Comparaison de la détection de points via l'IA LightGlue et via le logiciel Agisoft

Dans le cadre d'une création automatique des points homologues, que ce soit avec l'IA ou avec SIFT, des erreurs sont souvent existantes. Il existe plusieurs méthodes permettant de les détecter et de les supprimer avant d'effectuer un calcul photogrammétrique [Le Bris, 2010] :

- Filtre aller-retour (uniquement pour SIFT)
- Filtre signe (uniquement pour SIFT)
- Filtre de voisinage
- Filtre par reprojection
- Filtre épipolaire
- Filtre par « chaînage »

Le filtre de voisinage consiste à analyser les proximités des points homologues voisins entre deux images. Si le voisinage des points de l'image 1 n'est pas, de manière générale, le même que pour l'image 2, le point est éliminé comme le montre la Figure 14. L'algorithme est le suivant :

Pour tout point homologue P entre 2 images :

$S_1 = N$ points les plus proches dans l'image 1

$S_2 = N$ points les plus proches dans l'image 2

$nb_{communs} = \text{nbre de points communs entre } S_1 \text{ et } S_2$

Si $\frac{nb_{communs}}{N} < \text{seuil}$ alors :

Le point P est éliminé.

sinon

Le point P est conservé.

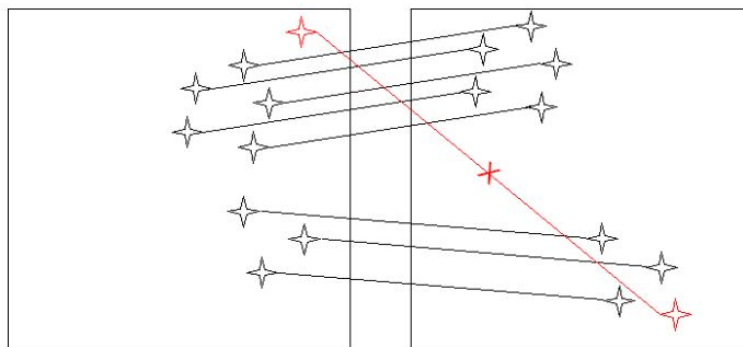


FIGURE 14 – Représentation du filtre par voisinage [Le Bris, 2010, illustration reprise de la page 30]

Le filtrage par reprojection repose sur la connaissance précise de la position et de l'orientation des images. En appliquant les équations de colinéarité, il est possible de déterminer la coordonnée 3D d'un point homologue à partir de deux ou plusieurs vues. Ce point 3D est ensuite reprojété dans les images d'origine. On compare alors la position du pixel reprojété avec celle du point homologue détecté. Si l'écart dépasse un certain seuil fixé, le

point est considéré comme incorrect et est éliminé. Cette méthode permet d'évaluer la cohérence géométrique des points homologues à travers un nombre potentiellement illimité d'images.

La méthode de chaînage, schématisée sur la Figure 15, consiste à suivre les correspondances de points homologues à travers une séquence d'images successives. L'objectif est de former une boucle, c'est-à-dire que le point identifié dans la première image doit pouvoir être retrouvé, par correspondances successives, dans l'image initiale. Si une boucle cohérente ne peut pas être établie, le point est alors rejeté. Cette méthode est utile pour évaluer la robustesse des correspondances sur plusieurs vues, mais elle pourrait éliminer des points valides en milieu de boucle.

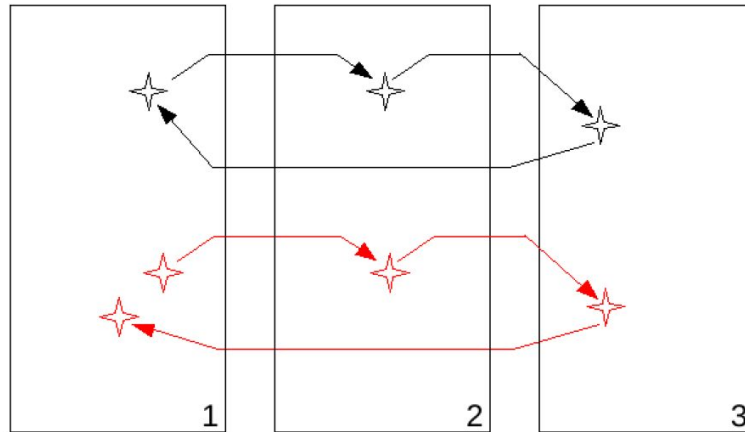


FIGURE 15 – Représentation du filtre par chaînage [Le Bris, 2010, illustration reprise de la page 33]

Le filtrage épipolaire exploite la géométrie fondamentale entre deux images. La droite épipolaire (droite rouge sur la Figure 16) est définie par les paramètres de calibration en intersectant le plan défini par la position des deux images et d'un point homologue (S_1 , S_2 et m_1) avec le plan défini par la deuxième image. Une fois l'épipolaire calculée à partir des paramètres de calibration, tout point identifié dans une première image doit correspondre à un point situé à proximité de la ligne épipolaire correspondante dans la seconde image. L'écart entre la position réelle du point détecté et cette ligne est mesuré. Un seuil est appliqué pour décider de la validité de la correspondance. Ce filtre permet de supprimer les correspondances incohérentes sur le plan géométrique.

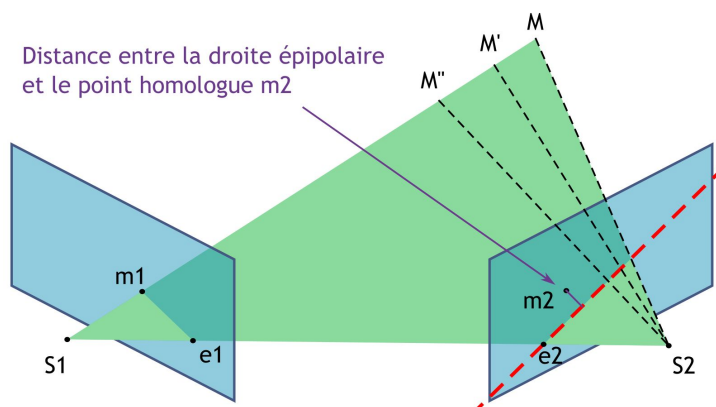


FIGURE 16 – Représentation du filtre épipolaire

La détection de points homologues entre plusieurs images constitue une étape clé pour réaliser une aérotriangulation. Cette dernière vise à déterminer l'ensemble des inconnues, soit les paramètres d'orientation externe (position et orientation) et interne (calibration de la caméra) d'un ensemble d'images dans un traitement global. Afin de garantir la cohérence de ce calcul, il est essentiel d'identifier les images présentant un recouvrement suffisant. Dans le cas de l'orientation d'une image isolée vers un ensemble d'images de référence, il convient de trouver lesquelles recouvrent notre image cible. Une sélection précise des paires d'images à comparer permet

de réduire les temps de calcul tout en assurant la qualité de la reconstruction vu les erreurs fréquentes des points homologues déterminés par un procédé automatique.

Plusieurs approches permettent de déterminer les couples d'images susceptibles de se recouvrir :

- **Méthode exhaustive par sous-échantillonnage** : les images sont d'abord réduites (par exemple, avec un facteur de 8) afin de diminuer le coût de calcul. Des points homologues sont ensuite recherchés pour tous les couples possibles (i images donnent $\frac{i(i-1)}{2}$ paires). Les images ayant un nombre de correspondances supérieur à un certain seuil n sont retenues comme ayant un recouvrement. Une fois ces paires identifiées, la détection des points homologues peut être affinée sur des images à plus grande résolution.
- **Méthode séquentielle** : lorsque les images ont été acquises selon un ordre logique (ex. le long d'un vol ou d'un trajet), on suppose que chaque image recouvre principalement les m images suivantes. Le nombre de paires à traiter est alors réduit à $i \cdot m - \frac{m \cdot (m+1)}{2}$. Si la prise de vue est en boucle, ce nombre devient simplement $i \cdot m$.
- **Méthode basée sur des informations de position/orientation** : si des données approximatives sur la position et/ou l'orientation des images sont disponibles (par exemple via GPS/IMU), on peut déterminer les paires en fonction de leur proximité spatiale ou direction de prise de vue similaire, ce qui permet une sélection plus juste et efficace des images à comparer.

2.1.4 Conclusion

Trois grandes familles de méthodes de géoréférencement ont été analysées :

- GNSS/IMU (smartphone) : accessible mais trop imprécis pour une représentation fiable du projet avec des appareils disponibles pour le grand public.
- GCP (points de contrôle au sol) : très précis, mais nécessite une préparation sur le terrain, du matériel et des compétences techniques.
- Appariement avec images déjà orientées via points homologues :
 - Cette méthode offre un bon compromis en permettant un géoréférencement automatique, sans matériel spécifique.
 - L'appariement peut être manuel, automatique avec un algorithme de type SIFT ou basé sur l'intelligence artificielle (IA).
 - L'IA, en particulier via les réseaux convolutifs, surpasse les autres techniques en robustesse, notamment en conditions variées (lumière, angle, saison).

L'utilisation de points homologues détectés automatiquement par IA apparaît comme la méthode la plus fiable et automatisable pour orienter une photo sans intervention manuelle. Une modèle photogrammétrique du lieu du projet est nécessaire à son fonctionnement.

2.2 Maquette 3D de projet

Pour réaliser un photomontage de projet de construction, il est nécessaire de disposer d'un modèle numérique représentant le projet. Historiquement, les photomontages étaient produits manuellement, en dessinant ou en collant des représentations du projet sur des photographies imprimées. Bien que ces méthodes permettaient une visualisation approximative de l'impact du projet, elles ne permettaient pas une évaluation précise de son emprise réelle dans l'espace, en raison des limitations techniques de l'époque.

Aujourd'hui, les outils numériques offrent de nouvelles possibilités. Les projets d'architecture sont généralement modélisés à l'aide de logiciels de conception BIM (Building Information Modeling) tels qu'Archicad ou Revit. Ces outils permettent de créer des maquettes numériques précises en trois dimensions intégrant des données de matériaux et d'implantation. Le géoréférencement de ces projets est souvent réalisé sur la base des données cadastrales officielles, ce qui assure leur alignement avec le contexte territorial existant.

Archicad, par exemple, permet l'exportation des projets au format Wavefront OBJ, un format ouvert (ASCII ou binaire) largement utilisé dans les workflows de visualisation 3D. Ce format contient des informations détaillées sur la géométrie du modèle, les matériaux et la gestion de la lumière (lumière diffuse, spéculaire et ambiante, voir Point 2.4), tout en conservant les textures définies dans le logiciel de conception. Cette capacité à exporter des modèles réalistes et géoréférencés constitue une base essentielle pour leur intégration crédible dans des photographies existantes.

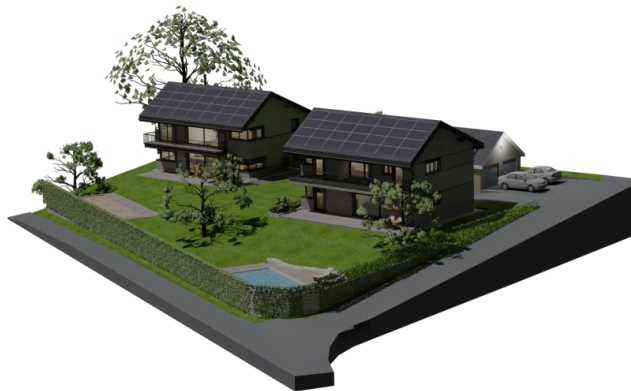


FIGURE 17 – Exemple de projet Archicad exporter en Wavefront⁷

2.3 Position du projet dans l'image

Une fois l'image calibrée, l'étape suivante consiste à localiser précisément le projet de construction dans la photographie. Cette localisation est indispensable pour identifier les pixels correspondant à l'implantation du projet. Pour ce faire, il est nécessaire de disposer des paramètres de calibration interne et externe. En combinant ces informations avec une maquette 3D géoréférencée du projet, il est possible de projeter chaque sommet du modèle 3D dans le plan image en utilisant l'équation de colinéarité (1).

2.4 Rendu visuel du projet

Ayant déterminé la position du projet dans une image, il reste à établir l'esthétique du projet dans l'image. Il existe plusieurs méthodes dont l'utilisation de l'illumination de phong ou le lancer de rayon.

2.4.1 Illumination de Phong

Ce modèle de représentation dépend de 3 types d'illumination :

- Diffuse : Permet d'affecter un ombrage en fonction de la direction de la lumière

7. source : Projet fourni par l'atelier pat Architectes SA

- Spéculaire : Permet d'affecter des reflets en fonction de la source de lumière et de la position d'observation de la caméra
- Ambiante : Affecte la même lumière sur l'entier de l'objet (la couleur est uniforme)

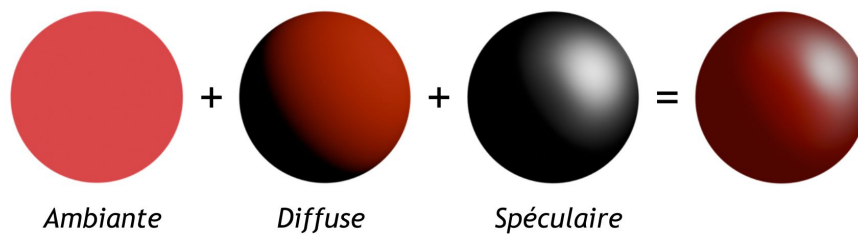


FIGURE 18 – Illustration du modèle de Phong

Ces illuminations, indiquée sur la Figure 18, sont représentées par des intensités.

$$I = I_{amb} + I_{diff} + I_{spec}$$

L'**intensité ambiante** est la plus simple à définir. Il s'agit de la multiplication de 2 paramètres :

$$I_{amb} = I_a \cdot k_a$$

I_a : Intensité de lumière ambiante pour l'ensemble des objets (souvent compris entre 0.2 et 0.3)

k_a : Coefficient de réflexion de la lumière ambiante en fonction du type de matière (compris entre 0 et 1)

L'**intensité diffuse**, décrite dans la Figure 19, dépend de l'intensité de la source lumineuse, de la propriété de la surface et du produit scalaire entre la normale de la surface ($\vec{N}$) et la direction de la source lumineuse ($\vec{L}$). Plus l'angle θ qui représente l'angle entre $\vec{N}$ et $\vec{L}$ est proche de 0 plus l'intensité est grande.

$$I_{diff} = I_l \cdot k_d \cdot \cos(\theta) = I_l \cdot k_d \cdot (\vec{N} \cdot \vec{L})$$

I_l : Intensité de la source lumineuse

k_d : Coefficient de réflexion diffuse en fonction du type de matière (compris entre 0 et 1)

$\vec{N}$ et $\vec{L}$ doivent être unitaires

θ doit être compris entre -90° et 90° sinon $I_{diff} = 0$

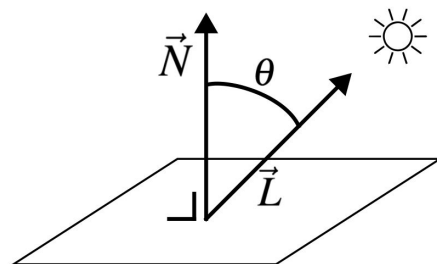


FIGURE 19 – Illumination diffuse

L'**intensité spéculaire**, décrite dans la Figure 20, est définie par l'intensité de la source lumineuse, la propriété de la surface et le produit scalaire entre la direction de l'observateur ($\vec{V}$) et la direction réfléchie de la source lumineuse ($\vec{R}$). Plus l'angle α qui représente l'angle entre $\vec{V}$ et $\vec{R}$ est proche de 0 plus l'intensité est grande.

$$I_{spec} = I_l \cdot k_s \cdot \cos(\alpha)^n = I_l \cdot k_s \cdot (\vec{R} \cdot \vec{V})^n$$

I_l : Intensité de la source lumineuse

k_s : Coefficient de réflexion spéculaire en fonction du type de matière (compris entre 0 et 1)

n : Facteur de brillance du reflet

$\vec{R}$ et $\vec{V}$ doivent être unitaires

α doit être compris entre -90° et 90° sinon $I_{spec} = 0$

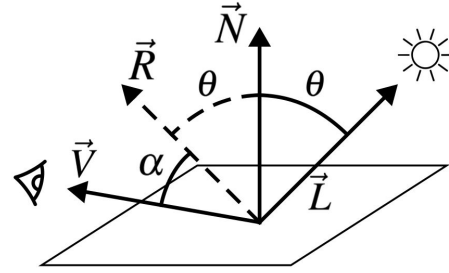


FIGURE 20 – Illumination spéculaire

Le facteur de brillance du reflet, illustrée sur la Figure 21, permet de définir la plage d'influence de l'illumination spéculaire.

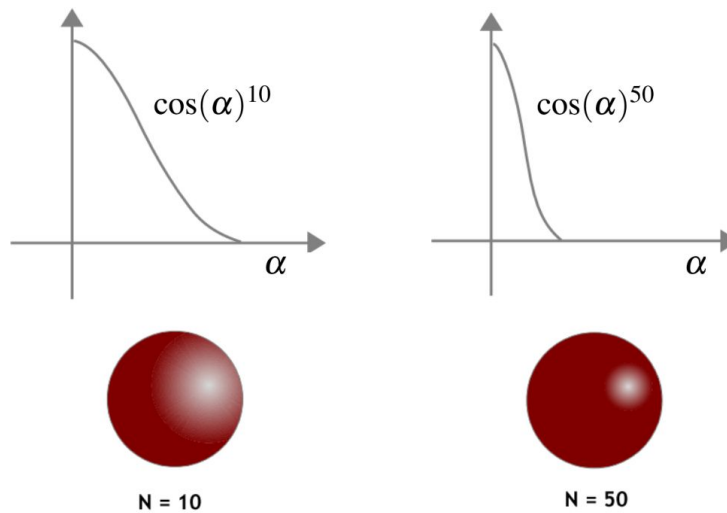


FIGURE 21 – Facteur de brillance dans l'intensité spéculaire

Ces différentes intensités permettent de définir la couleur vue par la caméra pour chaque face du maillage. La fonction complète est la suivante :

$$\begin{aligned} RGB_{rendu} &= RGB_{obj} \cdot (I_{amb} + I_{diff} + I_{spec}) \\ &= RGB_{obj} \cdot (I_a \cdot k_a + I_l \cdot k_d \cdot (\vec{N} \cdot \vec{L}) + I_l \cdot k_s \cdot (\vec{R} \cdot \vec{V})^n) \end{aligned}$$

2.4.2 Lancer de rayons

Le rendu par lancer de rayons enrichit le modèle d'illumination de Phong en intégrant la composante de transmission, permettant ainsi de simuler les réflexions et les réfractions de la lumière de manière plus réaliste. Ce processus repose sur le principe que la lumière interagit avec les surfaces en se réfléchissant ou en se transmettant à travers elles, en fonction de leurs propriétés matérielles.

Par exemple, lorsqu'un rayon lumineux frappe une surface hautement réfléchissante, comme un miroir, il est réfléchi selon un angle égal à celui d'incidence, éclairant potentiellement d'autres objets dans la scène. À l'inverse, une surface mate ou sombre, telle qu'un mur noir, absorbe la majeure partie de la lumière incidente, générant peu ou pas de rayons secondaires. La Figure 22 illustre 2 rayons lumineux qui sont réfléchis avec des intensités différentes en fonction des propriétés de la surface rencontrée. L'ombre au sol est alors plus claire à proximité des surfaces réfléchissantes.

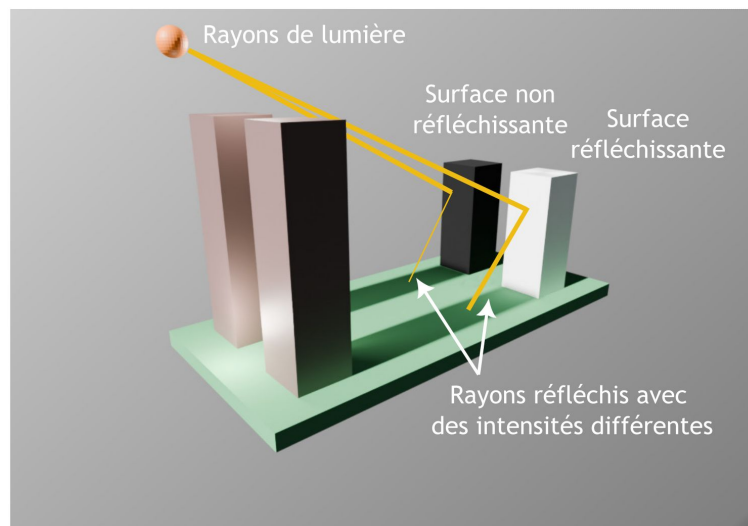


FIGURE 22 – Illustration de lancer de rayons

Dans le cadre du lancer de rayons, un rayon peut subir plusieurs réflexions successives sur différentes surfaces de la scène. À chaque interaction, l'intensité du rayon diminue en fonction des propriétés réfléchissantes des matériaux rencontrés, jusqu'à ce qu'elle devienne négligeable ou que le rayon sorte du champ de la scène.

Le lancer de rayons permet également de simuler avec précision les ombres portées en traçant des rayons secondaires, appelés rayons d'ombre, depuis le point d'intersection d'un rayon principal vers chaque source lumineuse de la scène. Si un objet obstrue ce rayon, le point est considéré comme étant dans l'ombre par rapport à cette source lumineuse.

Cette méthode permet de générer des ombres nettes et réalistes, en tenant compte de la géométrie exacte des objets et de leur position relative aux sources de lumière. De plus, elle facilite la simulation d'ombres douces en modélisant des sources lumineuses de taille finie, ce qui crée des zones de pénombre et d'ombre complète, améliorant ainsi le réalisme visuel de la scène.

Il existe plusieurs logiciels permettant de faire du lancer de rayon, notamment :

- Blender : Logiciel open-source développé par la Fondation Blender. Il intègre les moteurs de rendu Cycles (rendu réaliste par lancer de rayons) et Eevee (rendu en temps réel).
- Cinema4D : Logiciel commercial développé par l'entreprise Maxon

2.4.3 Conclusion

Le modèle d'illumination de Phong repose sur une approche locale, ne prenant en considération que la géométrie, les propriétés de surface et les sources lumineuses directement associées à un point donné. De ce fait, il ne modélise pas les phénomènes d'occlusion ou de transmission lumineuse entre objets, tels que les ombres portées ou le couplage énergétique entre surfaces adjacentes. Ces différences sont visibles dans la Figure 23, l'image de droite comprend les ombres du bâtiment au sol. L'intégration de ces phénomènes dans des moteurs de rendu utilisant le lancer de rayon (ray tracing) demande des traitements de calcul plus conséquents. Le modèle par Phong est également plus simple à coder dans un script.

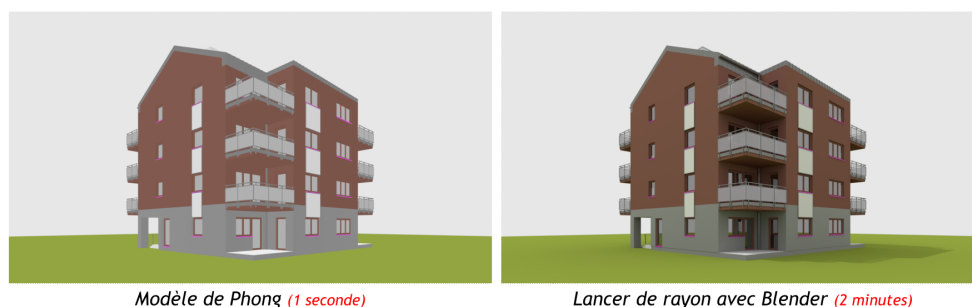


FIGURE 23 – Comparaison des rendus (Phong / Lancer de rayon)

2.5 Détermination de la visibilité du projet

La dernière étape du processus de photomontage consiste à déterminer si le projet de construction est partiellement ou totalement masqué par des éléments présents dans la photographie d'origine. Dans une procédure manuelle illustrée par la Figure 24, cette analyse repose sur l'observation intuitive de l'opérateur, qui identifie visuellement les zones où le projet devrait apparaître derrière des objets existants.

Pour intégrer le projet de manière réaliste, il est nécessaire de créer un masque de transparence qui définit les parties de l'image où le projet sera visible ou occulté. Ce masque est généralement élaboré à l'aide d'un logiciel de retouche d'image tel qu'Adobe Photoshop ou GIMP, en utilisant des outils de sélection et de calques pour isoler les zones concernées.

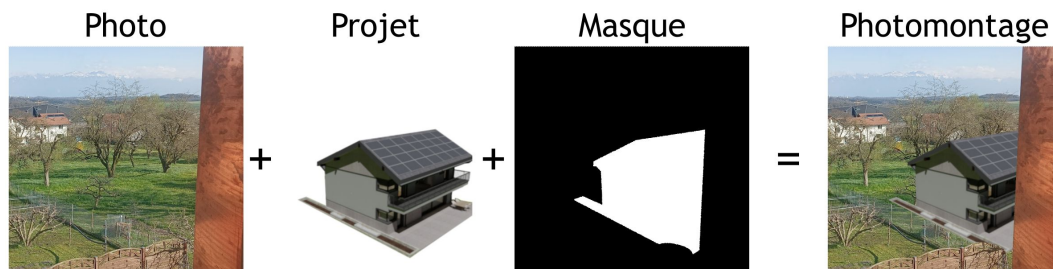


FIGURE 24 – Étapes de retouche manuelle d'une photo

Il est possible de générer automatiquement un masque de transparence en comparant deux types de distances :

- Les distances au terrain existant : distance entre la position de la prise de vue et les éléments présents dans l'image.
- Les distances au projet : distance entre la position de la prise de vue et les éléments qui composent le projet de construction.

En comparant ces distances pour chaque pixel de l'image, on peut déterminer un masque de visibilité du projet sur l'image. Si la distance au terrain est inférieure à la distance au projet, cela signifie qu'un élément du terrain se trouve devant le projet, et ce dernier doit être masqué à cet endroit. Ce masque de visibilité peut alors être transformé en masque de transparence.

Des images comprenant uniquement des valeurs de distance sont appelées cartes de profondeur ou depthmap (voir Figure 26). Ces distances peuvent soit être prises par rapport à la focale de l'image (en vert sur la Figure 25) ou projetée perpendiculairement à l'image (en rouge sur la Figure 25).

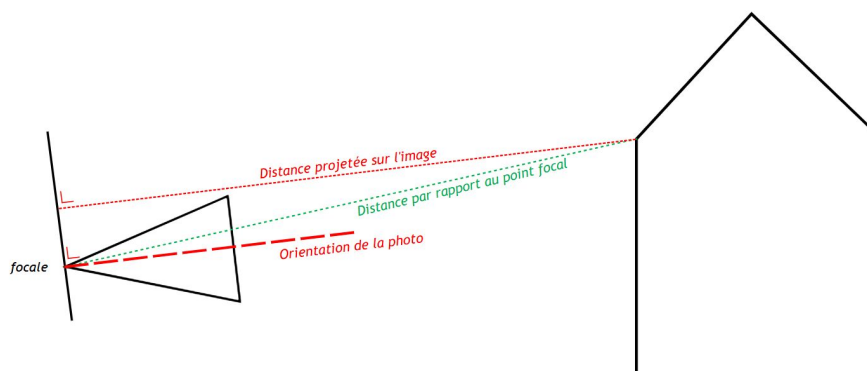


FIGURE 25 – Distances des cartes de profondeurs

Dans ce projet, il a été choisi d'utiliser la distance projetée. Sa détermination est explicitée par l'équation 5 :

R : Matrice de rotation de la caméra

S : Coordonnées de la caméra

M : Coordonnées du points à déterminer la distance de projection

$$\vec{N} = R^T \cdot \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix} : \text{Vecteur unitaire de l'orientation de la caméra} \quad (5)$$

$\vec{SM} = M - S$: Vecteur de la caméra au point M

$$D_{ProjImage} = \left| \frac{\vec{SM} \cdot \vec{N}}{\|\vec{N}\|} \right| = \left| \vec{SM} \cdot \vec{N} \right|$$

Ayant expliqué ce qu'est une carte de profondeur, il faut maintenant définir comment les calculer. Il existe trois méthodes principales :

- Combinaison d'images.
- En projetant un nuage de points sur l'image.
- Avec de l'intelligence artificielle.



FIGURE 26 – Exemple de photo et de carte de profondeur

2.5.1 Combinaison d'image ou stéréo-image

La combinaison d'image est une sous branche de la photogrammétrie. En combinant deux images à une certaine distance, la profondeur des pixels d'une image peut être déterminée en faisant des intersections de direction entre des objets communs (voir Figure 27).

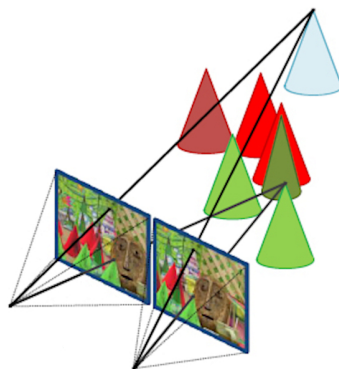


FIGURE 27 – Stéréo-image⁸

8. source : <https://fr.ids-imaging.com/technical-articles-details/items/whitepaper-depth-information-3d-images.html>

La qualité des cartes de profondeur dépend directement de la densité et de la précision du nuage de points calculé par stéréo. Il est possible que certaines zones de la carte de profondeur ne contiennent aucune valeur. L'incapacité de générer un nuage de points dense pour l'ensemble de l'image en est la cause. En effet, des surfaces trop homogènes ou la visibilité limitée d'un objet dans une seule image peuvent empêcher le calcul de correspondances denses des pixels. La Figure 28 montre le résultat d'une carte de profondeur avec deux prises de vue. L'arbre cache une partie du bâtiment ce qui fait qu'il manque des valeurs sur la carte de profondeur.



FIGURE 28 – Carte de profondeur de la photogrammétrie

2.5.2 Monoplotting d'un nuage de points ou d'un maillage sur une image

Une méthode pour générer une carte de profondeur à partir d'un nuage de points consiste à le transformer en maillage, puis à projeter ce dernier sur l'image à l'aide de l'équation de colinéarité.

Comme pour les méthodes basées sur la stéréo, la qualité de la carte obtenue dépend fortement de la densité et de l'homogénéité du nuage de points sur l'ensemble de l'image. Si le nuage est peu dense ou irrégulier, des discontinuités ou des trous peuvent apparaître dans le maillage (voir Figure 29)

Par ailleurs, il est essentiel que le maillage soit représentatif de la scène photographiée. Dans certains cas, des éléments présents sur l'image, comme un arbre, peuvent ne pas être visible dans le nuage de points et sont donc absents du maillage. La carte de profondeur générée ne tiendrait alors pas compte de ces objets, ce qui peut poser des problèmes d'interprétation.



FIGURE 29 – Maillage issu d'un nuage de points

2.5.3 Utilisation de deep learning

Comme pour la recherche de points homologues (voir Table 1), il existe une multitude de solutions Deep Learning permettant de créer des cartes de profondeurs à partir d'images. Des jeux de données test de code sont disponibles gratuitement sur le site <https://paperswithcode.com/> :

- "Monocular Depth Estimation" avec le jeu de données ETH3D. Le plus juste est Distill Any Depth. Il y a également MiDas et Depth Anything dans la liste des modèles.
- "Monocular Depth Estimation" avec le jeu de données NYU-Depth V2. HybridDepth est en 1ère place, Distill Any Depth en 2ème et Depth Anything en 4ème place.

Le jeu de données NYU-Depth V2 étant des vidéos à l'intérieur, ETH3D correspond mieux au traitement de ce projet.

Les modèles suivants ont été testés :

- Distill Any Depth <https://github.com/Westlake-AGI-Lab/Distill-Any-Depth>
- Depth Anything <https://github.com/DepthAnything/Depth-Anything-V2>
- MiDas https://pytorch.org/hub/intelisl_midas_v2/
- Depth Pro <https://github.com/apple/ml-depth-pro> (ce modèle a été conseillé par Adrien Gressin)

Tous ces modèles ont besoin uniquement d'une image en entrée et d'un modèle pré-entraîné. Ils retournent une image sous forme de carte de profondeur en sortie. Il est conseillé d'avoir une carte graphique NVIDIA qui permet de faire les calculs avec la GPU au lieu de la CPU (plus lent). L'ordinateur utilisé pour ce travail ayant une carte graphique Intel UHD Graphics, les durées des calculs seront plus longues.

Les différents modèles ont été testés sur 2 images (DSC6987 et DSC7050) et les résultats visuels de chaque modèle sont représentés sur la Figure 30

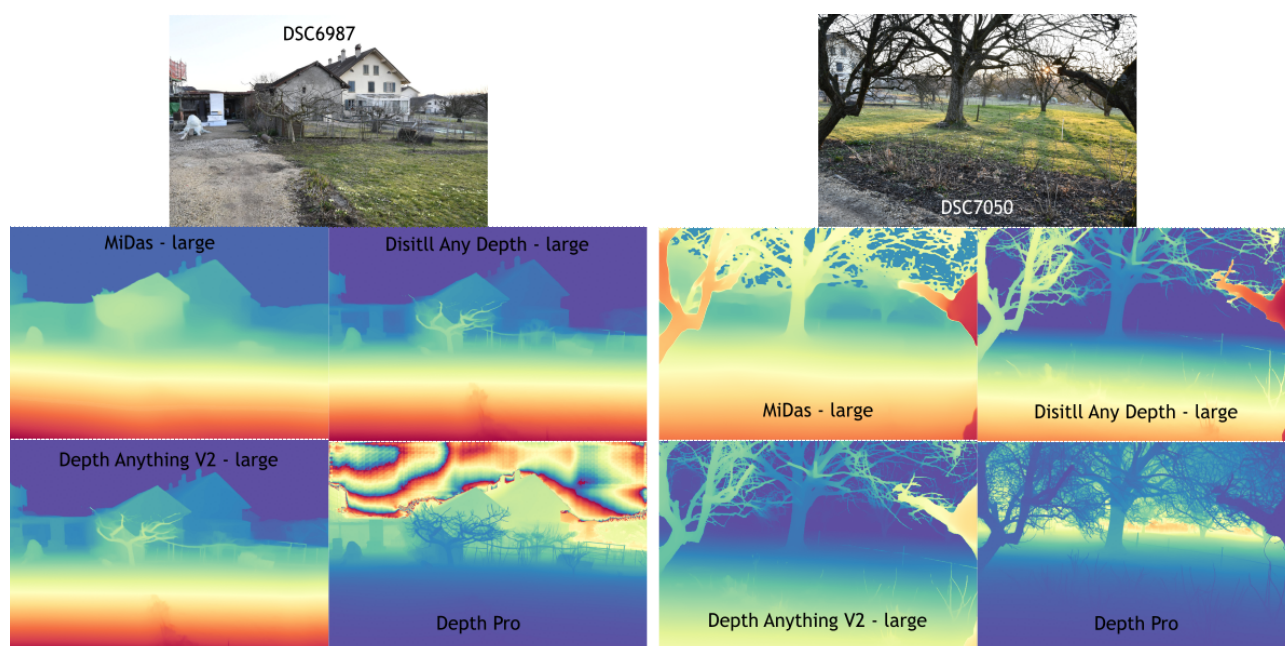


FIGURE 30 – Carte de profondeur avec différents modèles IA

Les différents modèles sont comparés uniquement sur le visuel dans la Table 4.

Modèle IA	Temps de calcul sur CPU (Intel Core i3 3GHz et de 8 Go de RAM)	Précision visuelle des détails	Détection du ciel
MiDas	DSC6987 : 20s DSC6987 : 40s	-	+
Depth Anything V2	DSC6987 : 40s DSC7050 : 60s	+	++
Distill Any Depth	DSC6987 : 40s DSC7050 : 90s	+	+
Depth Pro	DSC6987 : 420s DSC7050 : 620s	++	+

TABLE 4 – Comparaison des modèles de carte de profondeurs

Au vu des résultats, les meilleures solutions sont Depth Anything V2 et Depth Pro. Les deux images testées ayant un ciel de couleur unie, les deux modèles ont été évalués sur deux images supplémentaires (Figure 31 ci-après) comprenant un ciel nuageux et un lac afin d'analyser leur gestion des reflets et des nuages.

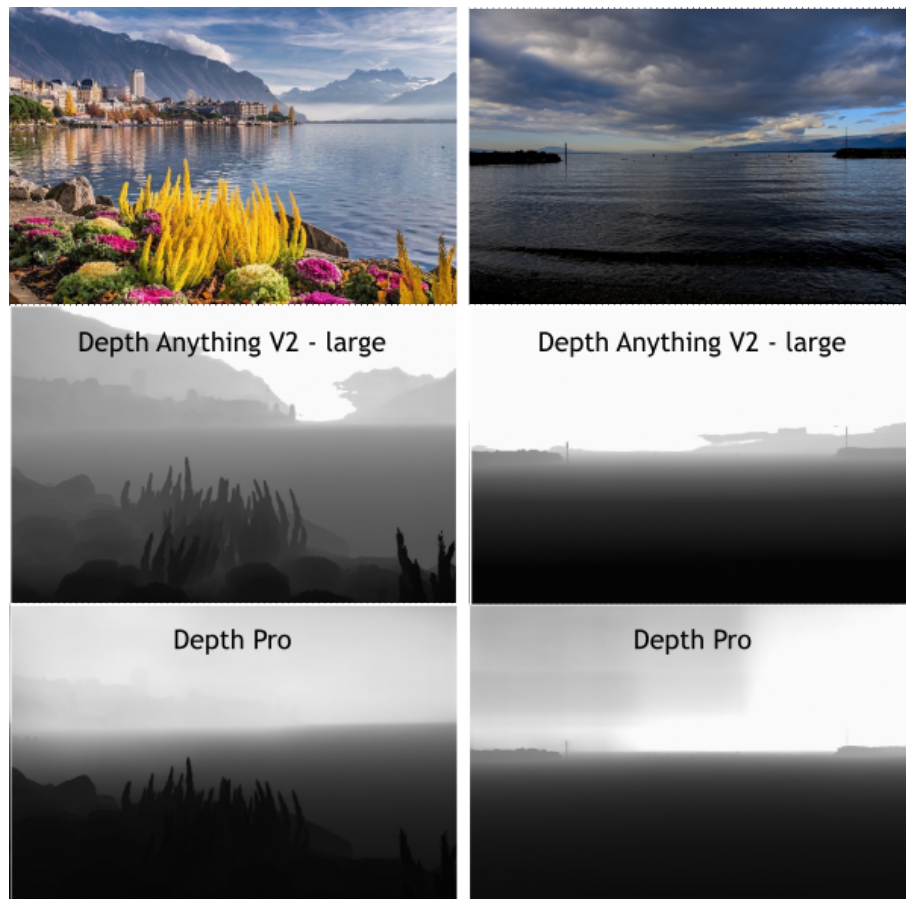


FIGURE 31 – Comparaison entre Depth Anything et Depth Pro

Les reflets du lac sont correctement gérés par les deux modèles. Cependant, Depth Pro présente différentes valeurs dans le ciel et a tendance à ne pas distinguer clairement le ciel de l'arrière-plan éloigné. Le fait qu'il ne détecte pas l'arrière-plan n'est pas un problème, car les projets de construction devraient se situer devant les montagnes. L'un des avantages de Depth Anything est que le ciel est attribué à une valeur de 0 dans la carte de profondeur.

La combinaison des deux solutions représente l'option idéale, bien qu'elle nécessite un traitement plus complexe. Par combinaison, il est entendu d'utiliser la carte de profondeur générée « Depth Pro », complétée avec les valeurs du ciel détectées avec « Depth Anything V2 ». Ce dernier se présente comme la solution intermédiaire la plus simple à mettre en œuvre. Toutefois, ce modèle IA ne prend pas en compte certains détails fins, comme les petites branches. Ces éléments, bien que manquants, ne sont pas essentiels dans le cadre de la visualisation du projet qui se concentre principalement sur les éléments structurels et les grandes caractéristiques du paysage.

2.5.4 Conclusion

La Table 5 comprend une comparaison synthétique entre les différentes méthodes, évoquées précédemment, pour générer une carte de profondeur d'une image. La méthode par Deep Learning est la plus simple pour avoir des valeurs sur l'entier de la carte de profondeur. Cependant, les valeurs produites ne sont pas exprimées en unités métriques. Une transformation est donc nécessaire afin de les convertir. Cette conversion peut être réalisée à l'aide d'une carte partielle de profondeur métrique. À l'inverse, la méthode du Monoplotting constitue l'approche la plus directe pour produire une carte de profondeur métrique.

La solution retenue pour ce projet serait de créer une carte partielle de profondeur métrique par Monoplotting d'un nuage peu dense et une carte de profondeur par Deep Learning. Au moyen d'un algorithme de transformation, la carte de profondeur par Deep Learning serait alors transformée dans un système métrique.

Type de carte de profondeur	Unité	Donnée sur l'ensemble de l'image	Durée du calcul	Complexité d'obtention
Stéréo-image	mètre	Dépendant des images acquises	Plusieurs minutes à quelques heures selon le nombre et la taille des images	Compétence nécessaire pour l'acquisition des images
Monoplotting de nuage de points	mètre	Dépendant du nuage de points	Plusieurs minutes selon la taille de l'image	Simple si le nuage de points est déjà disponible
Deep learning	Aucune unité	Oui mais qualité non déterminée	Quelques secondes avec une carte graphique NVIDIA et 1 à 5 minutes selon le modèle IA avec un calcul sur CPU	Simple, uniquement une image nécessaire

TABLE 5 – Comparaison des calculs de carte de profondeurs

2.6 Récapitulatif de l'état de l'art

L'état de l'art a permis d'explorer les différentes méthodes utilisées pour créer des photomontages réalistes de projets de construction, en les évaluant chacune selon leur précision, accessibilité, possibilité d'automatisation et pertinence pour une visualisation compréhensible par tous.

En reprenant les différentes étapes pour la création d'un photomontage de projet, une stratégie de développement pour obtenir un résultat précis et automatique peut être défini.

2.6.1 Géoréférencement et orientation d'un image

La solution retenue pour d'orienter et géoréférencer une image acquise par un utilisateur lambda est d'utiliser un modèle photogrammétrique de la situation du projet de construction en effectuant l'appariement en détectant des points homologues par IA.

2.6.2 Maquette de projet

La solution la plus simple est de reprendre la maquette géoréférencée créée par un architecte pour le dossier de mise à l'enquête. Une exportation de la modélisation au format Wavefront OBJ permet d'avoir une maquette lisible par la majorité des logiciels 3D en préservant la texture définie en amont.

Afin de diminuer la quantité de maillage de la maquette, il est possible de supprimer les éléments intérieurs.

2.6.3 Position du projet dans l'image

La position du projet peut être déterminée avec l'équation de colinéarité (1).

2.6.4 Rendu visuel du projet

Le rendu visuel du projet peut être déterminé de deux manières :

- Illumination de Phong pour un rendu rapide si le réalisme du projet n'est pas important pour l'utilisateur.
- Illumination via le lancer de rayon pour un rendu réaliste mais avec une durée de traitement plus long.

2.6.5 Détermination de la visibilité via carte de profondeur

Pour déterminer la visibilité du projet dans l'image, il est décidé d'utiliser une carte de profondeur déterminée par IA et de l'adapter par un algorithme de transformation utilisant le monoplottage d'un nuage de points.

Le nuage de points peut être acquis en amont soit :

- En reprenant le modèle photogrammétrique utilisé pour le géoréférencement pour créer un nuage de points
- En l'acquérant avec un Laser scanner
- En téléchargeant des données LIDAR opensource, par exemple sur Swisstopo⁹

Pour ce projet, il est utilisé le nuage de points du modèle photogrammétrique. Il a l'avantage de ne pas nécessiter une intervention supplémentaire et d'être à jour.

9. <https://www.swisstopo.admin.ch/fr/modele-altimetrique-swissurface3d>

3 MÉTHODE PROPOSÉE

Ce projet a pour but d'automatiser les traitements de création d'un photomontage de projet. L'ordre des tâches est identique à un photomontage manuel :

1. Calibration interne et externe de la photographie
2. Position du projet dans l'image
3. Création d'un rendu visuel du projet
4. Détermination de la visibilité du projet dans l'image

Les ressources décrites à l'état de l'art permettent d'établir un diagramme permettant de résumer l'ensemble des démarches nécessaires pour faire fonctionner cette application de photomontage de projet automatique.

La méthode proposée est séparée en 3 groupes comme le montre la Figure 32 :

- Traitement en amont de l'application : Acquisition des données de base
- Traitement externe : Importation d'une image utilisateur par une personne lambda
- Programmation du calcul du photomontage : Définition des fonctions utilisées dans l'application

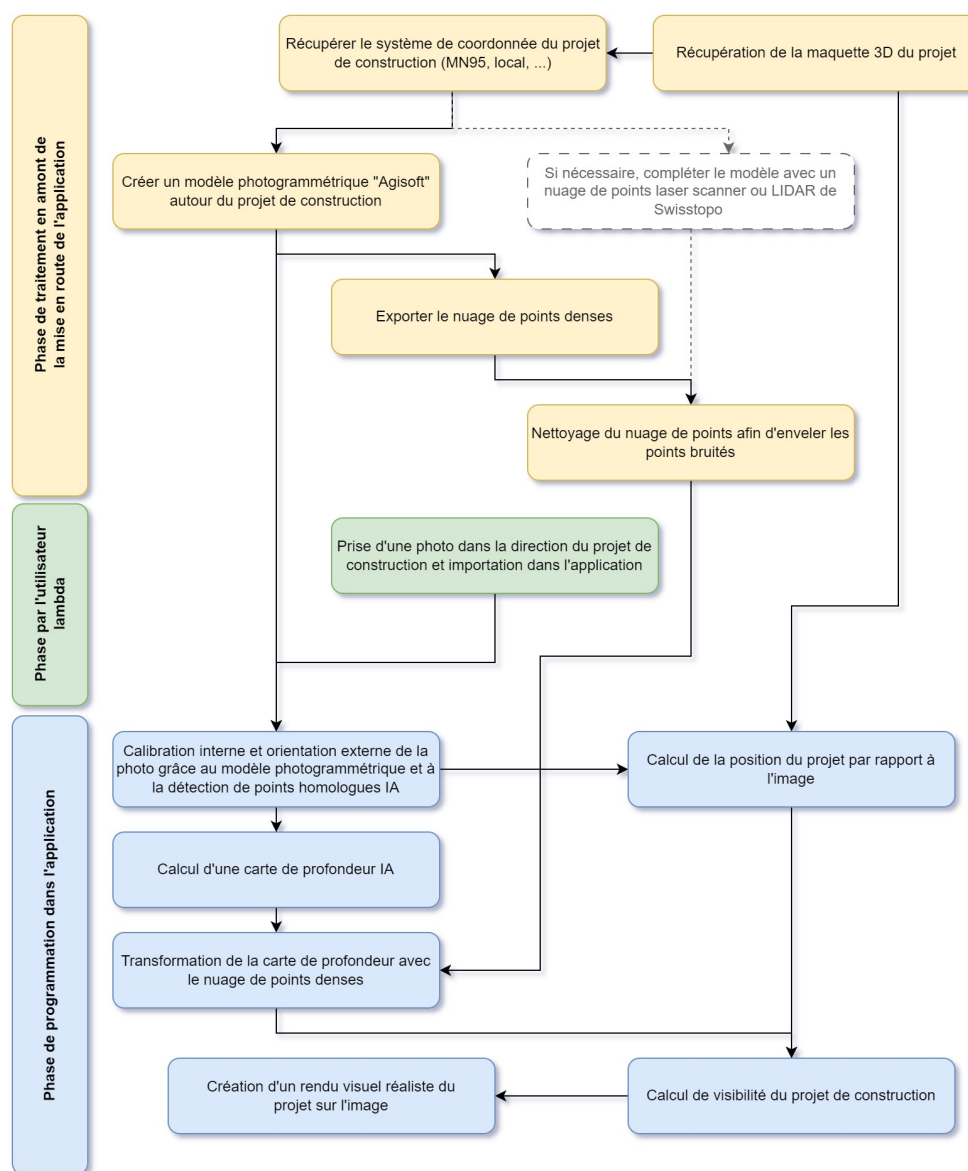


FIGURE 32 – Diagramme de synthèse des phases de traitement

La suite de cette partie sera principalement consacrée au développement réalisé pour la phase de programmation. Il sera séparé en 5 sections :

- Calibration interne et orientation externe de la photographie
- Position du projet dans l'image
- Nuage de points et carte de profondeur
- Rendu du projet dans l'image
- Calcul de visibilité du projet dans l'image

3.1 Calibration interne et orientation externe de la photographie

La détermination des paramètres interne et externe de notre image cible est un point clé de notre application. Les éléments à déterminer sont :

- Orientation externe
 - La position de l'image cible $S_{cible} = [X_{cible}, Y_{cible}, Z_{cible}]$
 - L'orientation de l'image $R_\omega, R_\phi, R_\kappa$
- Calibration interne (modèle « Frame » de Agisoft)
 - Le point principal d'autocollimation $PPA = [x_c, y_c]$
 - La focale f
 - Les coefficients de distorsions radiale (k_{1-4}), tangentielle (p_{1-2}) et affine (b_{1-2})

Afin de résoudre l'ensemble de ces paramètres, la méthodologie suivante a été appliquée :

1. Modèle photogrammétrique du lieu (calcul par aérotriangulation avec Agisoft)
2. Recherche des points homologues entre une image utilisateur¹⁰ et les images du modèle de base
3. Détermination approximative des paramètres de calcul
 - Détermination des coordonnées 3D des points homologues et filtre par reprojection
 - Détermination de la position et de l'orientation approchée de l'image utilisateur avec la DLT et filtre des valeurs aberrantes avec les écarts aux observations
4. Détermination précise de l'orientation externe et de la calibration interne
 - Calcul de la position de l'orientation et de la focale de l'image utilisateur par moindre carré non linéaire grâce à la détermination approximative et filtre des valeurs aberrantes avec les résidus normés.
 - Calcul de la position de l'orientation, de la focale, du PPA et des distorsions par moindre carré non linéaire.

3.1.1 Modèle photogrammétrique

Deux modèles photogrammétriques, détaillé à l'annexe B, ont été acquis. Ils servent de base au développement de la méthode.

3.1.2 Recherche des points homologues

Dans un premier temps, il est nécessaire d'identifier les images de référence qui serviront à détecter les points homologues. Pour simplifier les calculs, il a été décidé d'utiliser la position approximative de l'image cible qui est reprise, si possible, du capteur GNSS interne de l'appareil et sinon d'une indication par l'utilisateur sur une carte interactive. À partir de cette localisation, les cinq images les plus proches sont sélectionnées pour la recherche initiale de points homologues.

Le modèle IA « LightGlue » a été retenu en raison de sa rapidité et de sa compatibilité avec des configurations

10. Image acquise par un utilisateur dont les calibrations ne sont pas connues

matérielles modestes, ne nécessitant pas de carte graphique dédiée (voir section 2.1.3). L'autre avantage de LightGlue est qu'il évite la détection surabondante de points dans une même zone de l'image. En effet, une concentration excessive de points dans un secteur restreint n'améliore pas significativement la qualité du calcul de calibration ou d'orientation externe. En cas d'échec manifeste de LightGlue (détectations aberrantes), le modèle « GIM-ROMA » peut être utilisé comme alternative. Un filtrage épipolaire est ensuite appliqué à l'aide de la fonction « `cv2.findFundamentalMat` » de la bibliothèque OpenCV¹¹ afin d'éliminer une partie des correspondances incorrectes.

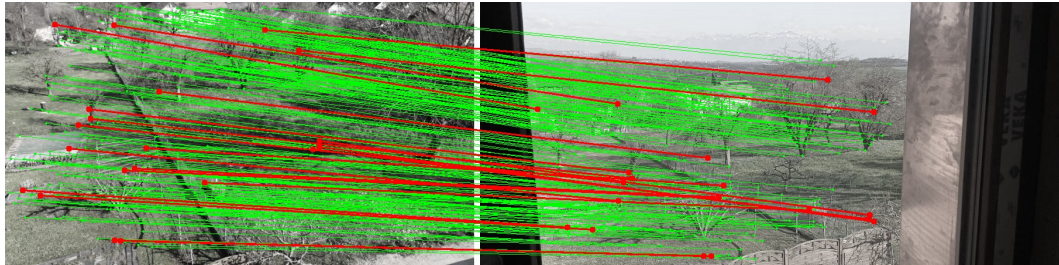


FIGURE 33 – Filtrage des points homologues avec « `cv2.findFundamentalMat` »

Le modèle IA « LightGlue » fournit également un score de fiabilité pour chaque point homologue, compris entre 0 (non fiable) et 1 (très fiable). En calculant la médiane des scores de fiabilité pour chaque paire d'images, il est possible d'identifier les correspondances peu probables. Par expérience, une médiane inférieure à 0.3 (pour des images redimensionnées à 680 pixels de large) indique généralement que les deux images n'ont aucune correspondances.

Après cette première phase, les résultats de la détection sur l'image cible sont évalués en examinant :

- le nombre total de points homologues,
- leur distribution spatiale sur l'image,
- le nombre d'images de référence dans lesquelles ces points apparaissent.

Pour déterminer le nombre de points de liaison nécessaires, il convient de rappeler qu'un minimum de six points de contrôle est requis pour estimer de manière approximative la position d'une image (voir les équations 3 de DLT à la page 8). Un point de contrôle 3D est obtenu par triangulation à partir d'au moins deux images de référence. Ainsi, un minimum de deux images est indispensable pour définir les coordonnées 3D d'un point homologue. Afin d'assurer une redondance suffisante, chaque point devrait idéalement être observé dans trois images. Par sécurité, un seuil minimal de 25 points homologues visibles dans deux images minium a été fixé.

Si cette première détection sur les cinq images les plus proches ne satisfait pas les critères ci-dessus, une seconde recherche est effectuée. Pour cela, une image principale est définie. Il s'agit de l'image présentant le plus grand nombre de points homologues avec l'image cible, et dont la médiane des scores de fiabilité est supérieure à 0.3, voir 0.4. Cette image est normalement, en termes de position et d'orientation, la plus proche de l'image cible.

À partir de cette image principale, on cherche les images de référence les plus susceptibles de présenter des correspondances. Pour cela, on analyse la position et l'orientation relative de chaque image par rapport à l'image principale. Un poids de correspondance est attribué à chaque image, basé sur deux facteurs :

- un facteur de proximité spatiale, défini par la formule :

$$\text{poids}_{\text{distance}} = \begin{cases} 1 - \frac{\text{dist}}{70}, & \text{si dist} < 70 \\ 0.0000001, & \text{sinon} \end{cases}$$

La valeur déterminante de 70m a été choisie de manière empirique.

- un facteur d'orientation, calculé à l'aide du produit scalaire entre les vecteurs d'orientation (résultant en une valeur entre -1 et 1, soit le cosinus de l'angle).

Le poids final est le produit de ces deux facteurs. La Figure 34 illustre l'évolution des poids en fonction de la distance à l'image principale, pour trois écarts d'orientation : 20°, 45° et 75°. Les cinq images ayant le poids de

11. <https://docs.opencv.org/4.5.1/>

correspondance le plus élevé sont sélectionnées pour une nouvelle tentative de détection de points homologues. Si cette tentative échoue également, une nouvelle image principale peut être choisie et le processus est répété avec un nouveau lot d'images.

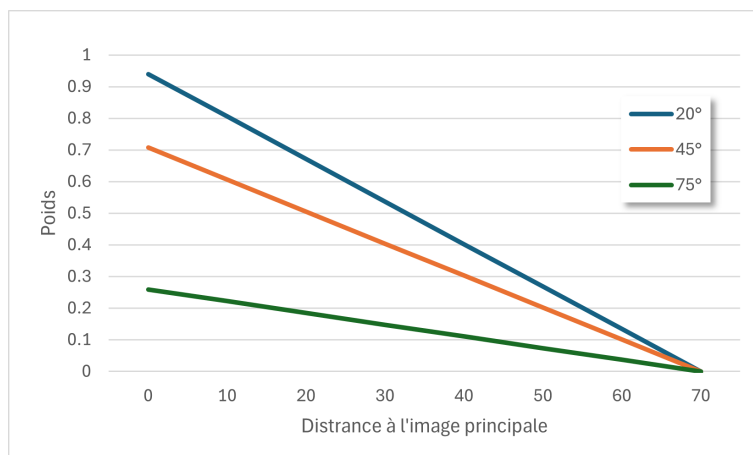


FIGURE 34 – Évolution des poids de correspondance relatifs à l'image principale selon trois différences d'orientation

La Figure 35 récapitule les itérations de détection de points homologues par rapport à notre image utilisateur. En pratique, ce processus se termine généralement après une seule itération. Toutefois, si l'ensemble des images du modèle photogrammétrique a été exploité sans permettre la détection d'un nombre suffisant de points homologues, l'image utilisateur ne pourra pas être correctement référencée.

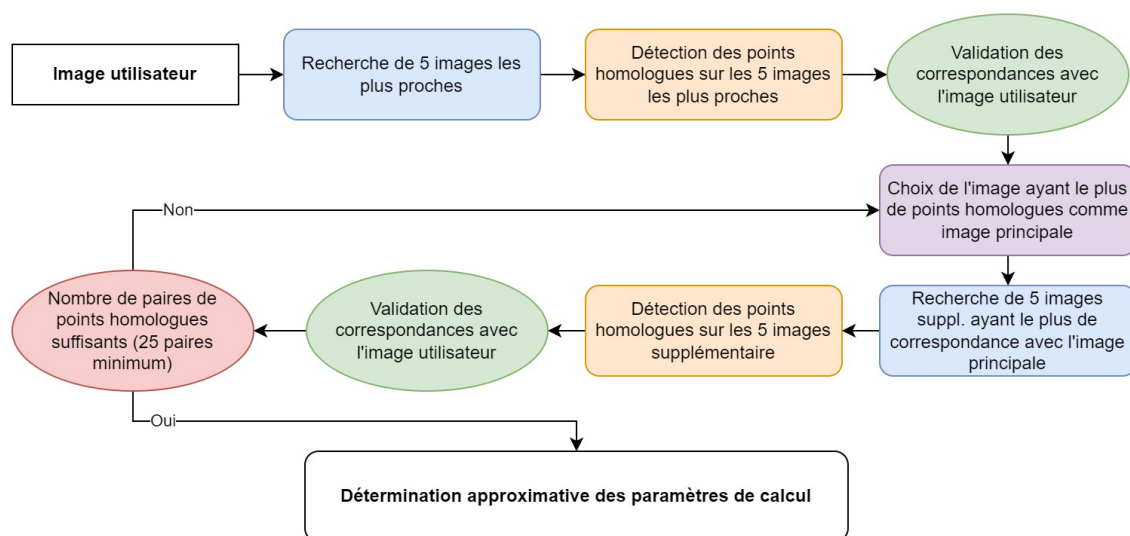


FIGURE 35 – Liste des étapes de la détection des points homologues par rapport à l'image utilisateur

La Figure 36 illustre les deux modèles de recherche d'image à traiter. Celle de gauche correspond à une recherche par proximité, tandis que celle de droite s'appuie sur une image principale. La partie inférieure représente, pour chaque point homologue, le nombre d'images dans lesquelles il a été identifié.

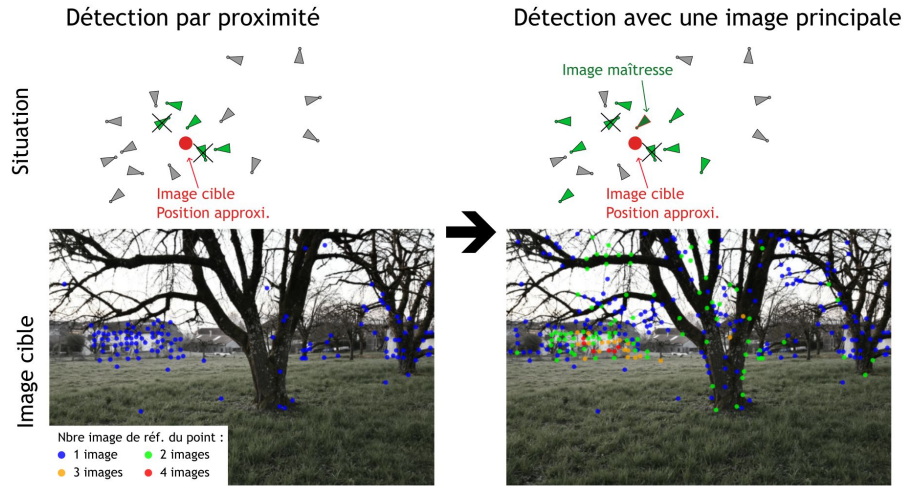


FIGURE 36 – Schéma de la détection des images de référence

3.1.3 Détermination approximative des paramètres de calcul

Comme expliqué au point 2.1.2, il convient d'appliquer une DLT (équation 3 à la page 8) pour trouver l'orientation externe approximative de l'image utilisateur.

L'utilisation de cette fonction linéaire repose sur la connaissance de points 3D déterminés selon la partie gauche de la Figure 37 et de leurs correspondances 2D sur l'image cible, représentées dans la partie droite. Les mesures 2D ont été acquises lors de la détection des points homologues.

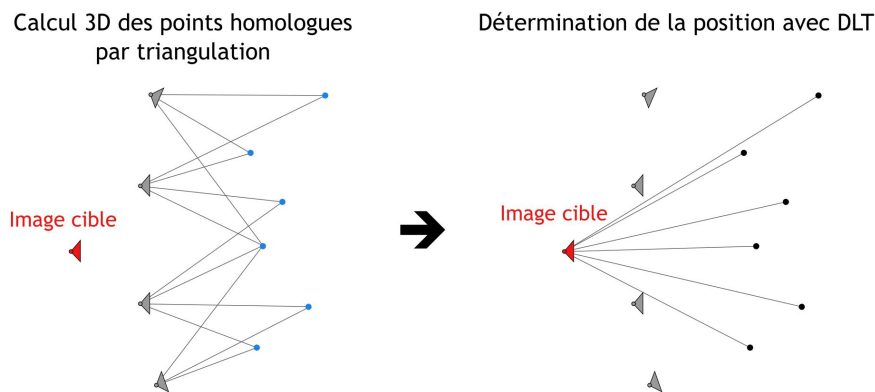


FIGURE 37 – Schéma des déterminations approximatives des inconnues

Pour déterminer la coordonnée 3D d'un point homologue (M), il est nécessaire d'obtenir les vecteurs de direction depuis les images de référence ($\vec{V}$). Ils sont déterminés grâce à l'équation de colinéarité en n'oubliant pas d'appliquer les paramètres de calibration. Il est alors possible de déterminer le point 3D avec les moindres carrés (voir annexe C.1) en se basant sur les équations suivantes qui ajoutent un facteur d'échelle comme inconnue à chaque vecteur de direction. Il faut au minimum deux équations pour déterminer l'ensemble des inconnues :

$$\begin{cases} S_1 = M - \lambda_1 \cdot \vec{V}_1 & \text{Depuis l'image n°1} \\ S_2 = M - \lambda_2 \cdot \vec{V}_2 & \text{Depuis l'image n°2} \\ \vdots & \end{cases}$$

Cette approche consiste à linéariser l'équation de colinéarité afin de résoudre le système de manière approximative. Il ne s'agit pas de la méthode rigoureuse de triangulation habituellement employée en photogrammétrie. En principe, la procédure correcte consiste à utiliser les résultats de cette estimation comme valeurs initiales, puis à affiner la position 3D du point à l'aide d'une résolution complète selon l'équation de colinéarité. Néanmoins,

dans le cadre de ce projet, seule l'estimation initiale a été retenue, car des tests sur des points théoriques ont montré que l'erreur de reprojection restait proche de 1 pixel, ce qui a été jugé acceptable à ce stade.

Une fois la coordonnée 3D d'un point homologue déterminée, il est possible d'appliquer un filtrage par reprojection. Cette méthode consiste à reprojeter le point 3D dans les images d'origine afin d'obtenir ses coordonnées 2D théoriques, puis à les comparer aux mesures réelles issues de la détection. L'écart entre les deux, appelé erreur de reprojection, permet d'évaluer la précision de la reconstruction.

Si cette erreur dépasse un seuil prédéfini, par exemple de 25 pixels, le point est considéré comme incorrect ou imprécis et est supprimé. Le seuil dépend de la précision de notre modèle photogrammétrique de référence. Il doit être défini de manière contextuelle, en fonction de la qualité des données et des exigences de l'application. Un exemple de détection de points faux est illustré en rouge sur la Figure 38.

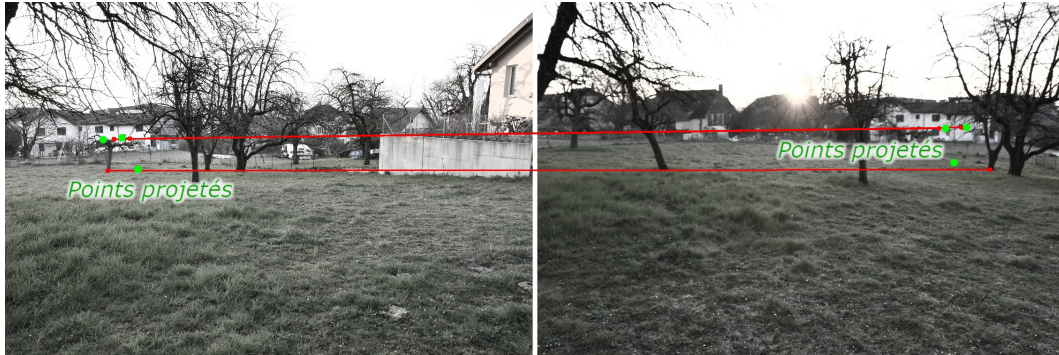


FIGURE 38 – Exemple de détection de points faux par reprojection

Il est maintenant possible de déterminer la coordonnée approchée de notre image cible en utilisant la DLT. L'équation linéaire (Équation n°3 à la page 8) est rappelée ci-dessous :

$$x = \frac{L_1X + L_2Y + L_3Z + L_4}{L_9X + L_{10}Y + L_{11}Z + 1}$$

$$y = \frac{L_5X + L_6Y + L_7Z + L_8}{L_9X + L_{10}Y + L_{11}Z + 1}$$

La détermination des paramètres inconnus se fait à l'aide des moindres carrés (voir annexe C.1).

À partir des inconnues L_{1-11} , il est déterminé approximativement la position, l'orientation, la focale et le PPA de l'image cible (voir équation 4 à la page 8).

Les moindres carrés sont fortement sensibles aux valeurs aberrantes. Il est possible que les observations sur l'image cible ne correspondent pas aux points homologues déterminés précédemment.

Pour pallier ce problème, la méthode de RANSAC (RANDOM SAMPLE CONSENSUS) (voir annexe C.2) permet de résoudre des équations entachées de fautes grossières.

En effectuant la résolution de l'algorithme RANSAC de la DLT sur une image test dont la position et les mesures des points homologues 3D sont réputés juste, il est possible de remarquer que les erreurs sur les observations avec la DLT sont en moyenne entre 1 et 2 pixels avec une erreur maximum à 4 pour un appareil reflex avec une focale de 17mm. Les paramètres utilisés pour la résolution sont les suivants :

- Le nombre minimum de données nécessaire pour ajuster le modèle a été fixé à 6¹²
- Le nombre maximal d'itérations a été fixé à 200
- Le seuil des observations compatibles avec le modèle a été fixé à 5 pixels.
- Le nombre minimum d'observations compatibles avec le modèle a été fixé à 15.

Un test complet a été effectué en effectuant la détection de points homologues sur un jeu d'image dont l'ensemble des paramètres sont connus. Il est obtenu un résultat pour une image principale avec une précision de positionnement de ± 3 m comme le montre la Figure 39.

Pour obtenir un résultat cohérent, la justesse des points homologues doit être garantie. La détection automatique de points homologues génère fréquemment des erreurs. Ces fautes ne sont pas forcément détectées avec

12. Voir Point 2.1.2

un RANSAC d'une DLT car le modèle ayant le moins d'erreur pourrait être identifié en attribuant des valeurs d'orientation et de PPA aberrantes. Une prochaine détection des erreurs sera effectuée ultérieurement.

Élément	Valeur réelle	Valeur calculée depuis DLT
Position	2528509.797 / 1159642.110 / 508.26	2528512.24 / 1159644.12 / 508.43
Angle $\omega\phi\kappa$	-98.458 / 34.817 / -3.027	-102.04 / 30.94 / -3.97
$x_c y_c$	105.21 / -122.601	-17.1 / 320.03
f	4249.59	3785.43

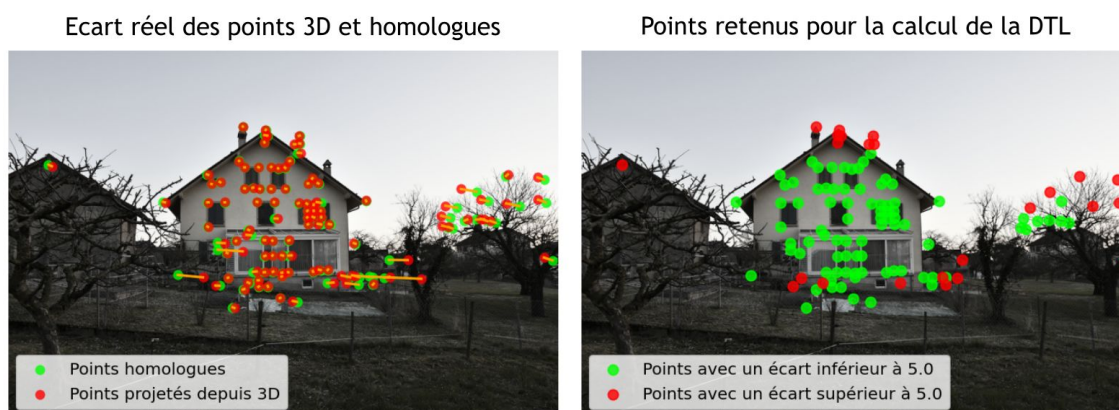


FIGURE 39 – Analyse de la DLT par RANSAC avec des valeurs fausses et peu précise

3.1.4 Détermination précise de l'orientation externe et de la calibration interne

Dans un premier temps, une analyse des paramètres de calibration interne est effectuée. Le modèle utilisé, « Frame » de Agisoft¹³, comprend 11 inconnues. Il est possible que toutes ces inconnues n'aient pas la même influence sur la précision du modèle. Il est donc essentiel d'identifier ceux ayant le plus d'impact. Pour cela, un banc de calibration (voir Figure 40) a été utilisé avec 19 points de contrôle (GCP) dont les coordonnées ont été déterminées avec une précision de l'ordre du millimètre. Afin de définir correctement la longueur de focale, il est nécessaire que les points de contrôle ne soient pas tous situés dans un même plan. Sur ce banc de calibration, les points sont positionnés sur deux plans.



FIGURE 40 – Banc de calibration¹⁴

13. Voir équation 2 à la page 7

Quatre jeux d'images (environ 6 photos chacun) ont été acquis avec les dispositifs suivants :

- Nikon D7500 avec un objectif Sigma de 17mm de focale
- Smartphone Samsung Galaxy A15 avec une focale de 3.98mm
- Smartphone POCO M4 Pro avec une focale de 4.12mm
- Smartphone POCO M4 Pro avec une focale de 1.65mm

Dans un premier temps, l'ensemble des paramètres de calibration a été estimé dans Agisoft pour chaque jeu d'images. Cela constitue notre modèle de référence, considéré comme le plus complet et précis. La position des images est alors connue avec précision, et les écarts aux GCP ainsi que les erreurs de reprojection en pixels sont calculés. L'erreur la plus rigoureuse à analyser est la différence en pixel. En effet, l'erreur en mm sur le GCP est dépendante de la résolution du pixel (mm/px). La Table 6 présente les erreurs de reprojection moyenne, minimum et maximum sur les points de contrôle. Une erreur moyenne de 0.86 pixel et de 1.0 mm pour les coordonnées GCP a été observée avec le Smartphone Samsung Galaxy A15.

Points de contrôle (GCP)	Erreur [mm]	Erreur [px]
Moyenne	1.0	0.86
Maximum	1.6	1.53
Minimum	0.6	0.54

TABLE 6 – Erreur des points de contrôle sur le banc de calibration

À partir de ce modèle de référence, il a été testé différentes combinaisons restreintes de paramètres internes, afin d'évaluer leur influence sur l'erreur de reprojection et sur la précision de positionnement des caméras.

Pour commencer, il a été analysé uniquement les écarts aux GCP pour les combinaisons suivantes dans le but de trouver les paramètres ayant le plus d'influence sur le résultat :

1. f, cx, cy
2. f, k_1
3. f, p_1
4. f, b_1

Le paramètre f est inclus systématiquement, car il représente le facteur d'échelle entre le système caméra et le terrain. Pour le Nikon D7500, le paramètre k_1 permet de réduire significativement l'erreur sur les GCP, contrairement aux autres. En revanche, pour les smartphones, les différences entre les combinaisons sont négligeables, comme le montre la Figure 41.

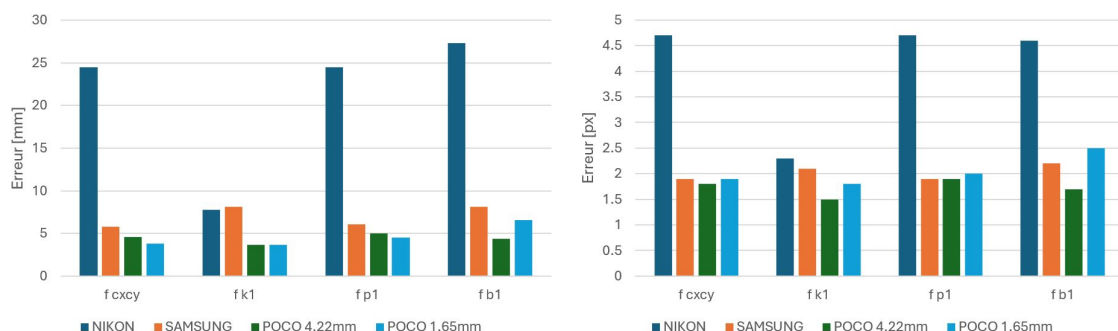


FIGURE 41 – Erreurs selon des combinaisons de deux paramètres de calibration

Une seconde analyse est effectuée en évaluant des combinaisons plus complètes en gardant toujours la distorsion radiale k_1 :

1. f, cx, cy, k_1, k_2

14. source : Banc de calibration de la HEIG-VD à Yverdon

2. f, cx, cy, k_1, k_2, b_1
3. f, cx, cy, k_1, k_2, k_3
4. f, cx, cy, k_1, k_2, p_1
5. f, k_1, k_2, k_3, p_1

Pour chacune, il a été comparé la position des caméras obtenue à celle du modèle de référence, ainsi que l'erreur de reprojection. Les résultats sont illustrés dans la Figure 42. La combinaison la plus satisfaisante est la 3^{ème} avec une erreur moyenne de reprojection inférieure à 0.8 pixel pour les smartphones et une erreur de positionnement proche de 10 mm pour l'ensemble des caméras testées. La première combinaison est la plus simple et permet d'obtenir un résultat dont la qualité est largement suffisante avec une erreur de reprojection moyenne de maximum 1.2 pixel. Pour l'ensemble des combinaisons l'erreur moyenne de la position des caméras est inférieure à 20 mm et les coordonnées des points de contrôle ont majoritairement une erreur inférieure à 2 mm. Pour rappel, la précision nécessaire à ce projet a été définie à ± 20 cm dans l'état de l'art au point 2.1 à la page 4. La première combinaison est alors largement suffisante.

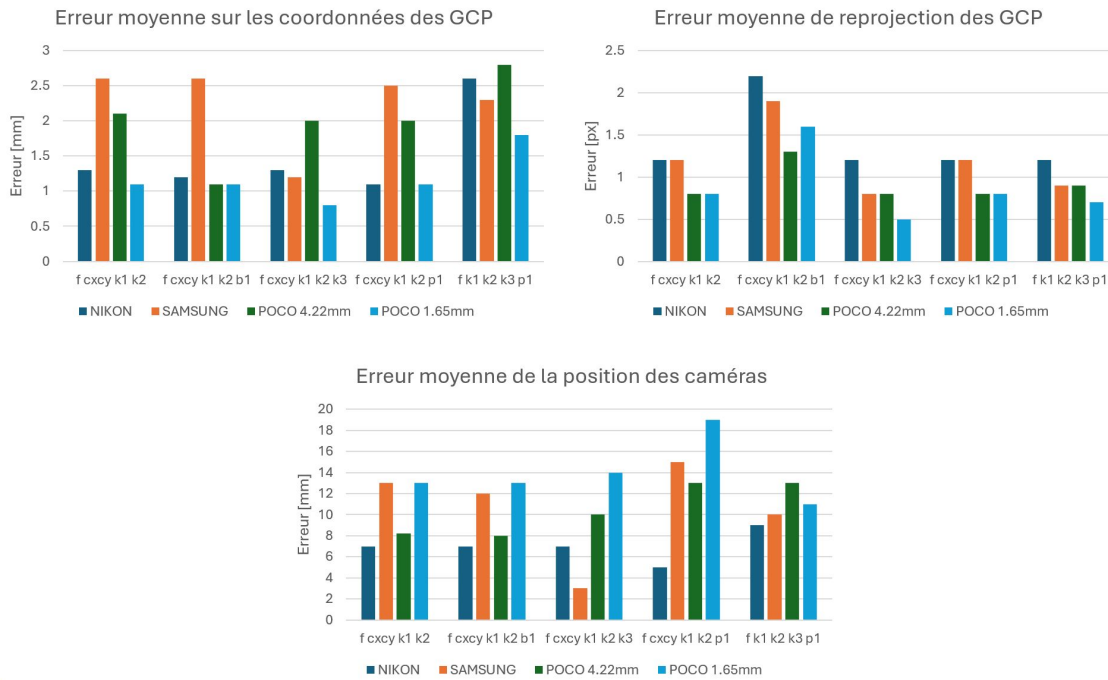


FIGURE 42 – Erreurs selon différentes combinaisons de paramètres de calibration

La modélisation de la distorsion radiale k est cruciale pour ce type de capteurs. Pour des optiques particulières, comme les objectifs fisheye, une nouvelle analyse serait nécessaire.

Ayant déterminé les paramètres importants, il est maintenant possible de déterminer l'orientation externe et la calibration interne avec la méthode des moindres carrés non linéaires (voir annexe C.1). Dans ce travail, il a été choisi d'utiliser directement les coordonnées 3D des points homologues afin de réduire le nombre d'équations d'observation. En effet, les équations rigoureuses nécessiteraient de prendre en compte toutes les mesures pixel des points homologues pour chaque image du modèle photogrammétrique en plus de notre image à traiter. Cette approche impliquerait également l'ajout d'une inconnue 3D par point homologue. Or, si le modèle photogrammétrique est suffisamment précis, les coordonnées 3D déjà calculées des points homologues fournissent une information fiable, rendant inutile la considération explicite des mesures pixels dans le système d'équations. Il est maintenant possible de définir notre matrice A en fonction.

La matrice jacobienne A est complexe à déterminer manuellement, car elle nécessite le calcul des dérivées partielles de chaque équation d'observation par rapport à chaque paramètre inconnu. Cela s'avère particulièrement difficile dans le cas d'équations non linéaires, comme celles impliquant la colinéarité avec correction de la distorsion. L'utilisation de la bibliothèque Python JAX¹⁵ (librairie Python) simplifie ce processus. Elle permet d'automatiser le calcul de la jacobienne à partir d'une fonction décrivant le modèle. Il suffit de spécifier les paramètres inconnus, les constantes, ainsi que les variables qui changent à chaque équation d'observation.

15. <https://docs.jax.dev/en/latest/quickstart.html>

Par exemple, les coordonnées 3D des points homologues sont différentes pour chaque équation, tandis que certains paramètres comme la distorsion tangentielle p_2 , que l'on peut fixer à zéro ($p_2 = 0$), sont considérés comme des constantes.

Afin d'obtenir une matrice A , JAX a besoin de trois éléments :

- Une fonction f correspondant à une seule équation d'observation. Comme exemple, il est possible de prendre une fonction linéaire $f(a, x, b) = a \cdot x + b$.
- Une variable $grad_f$ permettant de calculer les dérivées des inconnues. Elle est obtenue en spécifiant le ou les argument(s) de f à dériver grâce à la fonction $grad$. Dans l'exemple, l'inconnue est le 2^{ème} paramètre de la fonction $f(a, x, b)$.
- Une variable $grad_{batched}$ vectorisant $grad_f$ en lui permettant d'indiquer des tableaux aux arguments de f au lieu de scalaires. Elle est obtenue en indiquant quels arguments doivent être « batchés » grâce au paramètre in_axes de la fonction $vmap$. Dans l'exemple, a et b doivent être « batchés ».

Si des tableaux doivent être utilisés dans la fonction, il faut les créer avec JAX et pas avec numpy.

L'exemple $f(a, x, b) = a \cdot x + b$ est utilisé pour effectuer un traitement JAX dans un code python :

```
#Importation de jax
import jax.numpy as jnp #Pour la création de tableau
from jax import grad, vmap

def f(a, x, b):
    return a*x+b

grad_f = grad(f, argnums=([1])) #x est l'argument à dériver
grad_batched = vmap(grad_f, in_axes=(0,None,0)) #a et b sont des vecteurs tandis
que x est une constante scalaire

#Tableau de valeur de a et b pour 2 équations d'observation
a=jnp.array([
    1.0,
    -0.5
])
b=jnp.array([
    2.0,
    5.0
])

x=0.0 #Valeur approximative de l'inconnue

Agrad=grad_batched(a, x, b)
A=np.array(Agrad).T
```

Il est maintenant possible d'appliquer cette façon de déterminer la matrice jacobienne à ce projet. Il a fallu créer 2 sous-détermination de matrice A car notre modèle comprend deux types d'équation d'observation, une pour l'observation de la largeur pixel de l'image (u) et une pour l'observation de la hauteur pixel de l'image (v). Les deux fonctions sont les suivantes :

$$photogra_u(X_{cible}, Y_{cible}, Z_{cible}, M, R_\omega, R_\phi, R_K, f, cx, cy, k_1, k_2, k_3, k_4, p_1, p_2, b_1, b_2)$$

$$photogra_v(X_{cible}, Y_{cible}, Z_{cible}, M, R_\omega, R_\phi, R_K, f, cx, cy, k_1, k_2, k_3, k_4, p_1, p_2, b_1, b_2)$$

Il est maintenant possible de définir les inconnues et les valeurs changeantes. Dans ce cas, uniquement les coordonnées des points homologues M sont changeantes. Pour les inconnues, il faut choisir les paramètres de calibration interne à résoudre. Lors de la détermination approximative de l'orientation externe par DLT, il peut arriver que la détermination ne soit que peu précise avec des écarts de $\pm 15m$ sur la position et de $\pm 20^\circ$ d'orientation. En calculant directement par moindres carrés non linéaires sur l'ensemble des données inconnues (orientation externe et calibration interne), le résultat est fortement dépendant de la qualité du résultat de la DLT. Pour obtenir un résultat cohérent avec une position, une orientation et une focale totalement aberrantes en entrée, il faut effectuer une première détermination par moindres carrés avec uniquement l'orientation externe

$(X_{cible}, Y_{cible}, Z_{cible}, R_{\omega}, R_{\phi}, R_{\kappa})$ et la focale (f) comme inconnue. Les autres paramètres de calibration interne sont fixés à 0.0. Il est alors possible de créer les gradients :

$$grad_u = grad(photogra_u, argnums([0, 1, 2, 4, 5, 6, 7]))$$

$$grad_v = grad(photogra_v, argnums([0, 1, 2, 4, 5, 6, 7]))$$

$$gradbatched_u = vmap(grad_u, in_axes = (None, None, None, 0, None, \dots, None))$$

$$gradbatched_v = vmap(grad_v, in_axes = (None, None, None, 0, None, \dots, None))$$

Il suffit à présent d'indiquer les valeurs aux fonctions *gradbatched* et d'ensuite concaténer les sous-matrices pour obtenir la matrice A :

$$A_u = gradbatched_u(X_{cible}^{DLT}, Y_{cible}^{DLT}, Z_{cible}^{DLT}, M, f^{DLT}, 0.0, \dots, 0.0)$$

$$A_v = gradbatched_v(X_{cible}^{DLT}, Y_{cible}^{DLT}, Z_{cible}^{DLT}, M, f^{DLT}, 0.0, \dots, 0.0)$$

$$A = np.concatenate((np.array(A_u)^T, np.array(A_v)^T), axis = 0)$$

Avec

M : coordonnée 3D de l'ensemble des points homologues

La matrice des observations est la suivante :

$$l = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ v_1 \\ v_2 \\ \vdots \end{bmatrix}$$

Les matrices jacobienne et d'observation étant posées, il est maintenant possible d'effectuer la résolution par moindres carrés non linéaire. Cette première résolution permet également de détecter des faux points en effectuant une analyse de précision (voir Figure 43). Pour ce faire, il faut analyser la précision a posteriori des observations en utilisant les résidus normés w décrits dans l'annexe C.1.

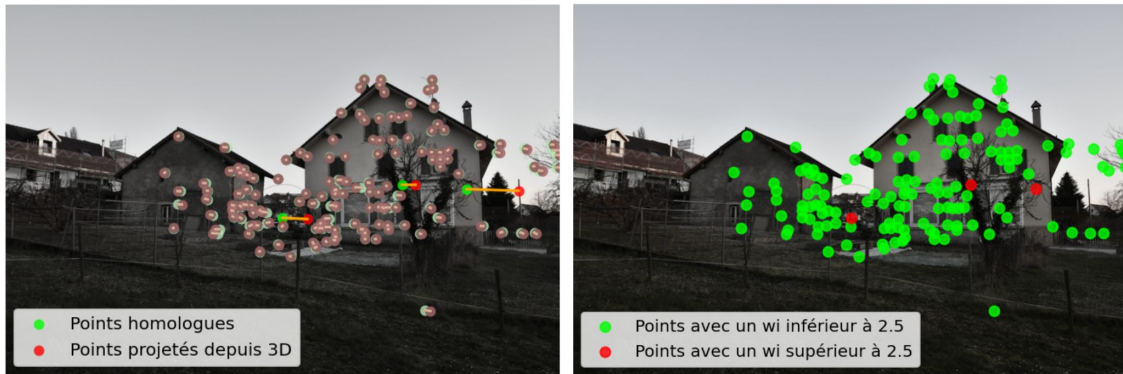


FIGURE 43 – Illustration des points faux détectés par moindres carrés

Il est alors possible de résoudre l'équation avec plus d'inconnues, en rajoutant au minimum c_x, c_y, k_1, k_2 d'après l'analyse effectuée sur la Figure 42, par rapport à la première détermination. Les fautes grossières sont enlevées des observations et les éléments suivants sont utilisés comme valeurs approchées des inconnues :

- Position reprise depuis le premier calcul de moindres carrés non linéaire
- Orientation reprise depuis le premier calcul de moindres carrés non linéaire
- Focale reprise depuis le premier calcul de moindres carrés non linéaire
- Distorsions radiales et transversales égales à 0
- Les valeurs x et y du PPA égales à 0

3.1.5 Validation de la méthode

En utilisant une image dont la position a été déterminée depuis le banc de calibration de l'annexe B.1, il est possible d'estimer la précision de notre traitement de calcul.

Dans un premier temps, il a été utilisé des points homologues répartis sur l'ensemble de l'image comme le montre la Figure 44.

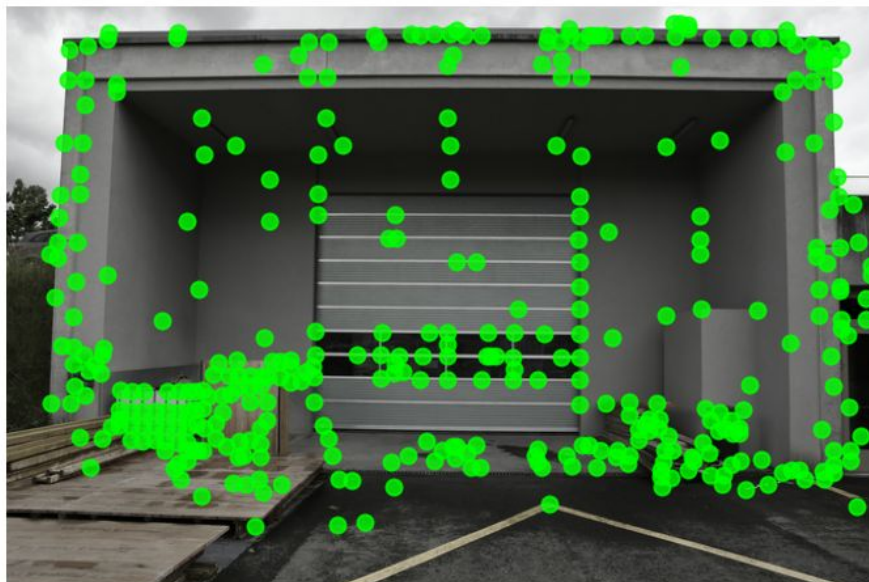


FIGURE 44 – Points homologues pour déterminer la précision du modèle

La Table 7 présente une comparaison entre les valeurs réelles obtenues sur le banc de calibration à l'aide du logiciel Agisoft et les valeurs calculées selon la méthode décrite précédemment. Les coordonnées de position montrent des écarts inférieurs à 5 cm, tandis que les différences d'orientation restent en dessous de $0,1^\circ$. Concernant les paramètres intrinsèques, la focale présente un écart inférieur à un pixel. En revanche, le point principal d'autocollimation (PPA) diffère de plus de 10 pixels. Cette différence est toutefois négligeable, car elle est fortement corrélée aux paramètres de distorsion. En effet, la calibration d'Agisoft prend en compte l'ensemble des paramètres de distorsion, tandis que le modèle calculé ne considère que les coefficients radiaux k_1 et k_2 .

Élément	Valeur réelle	Valeur calculée
Position [m]	2540583.277 / 1181278.085 / 446.095	2540583.256 / 1181278.101 / 446.099
Angle [$^\circ$] $\omega\phi\kappa$	95.4391 / -50.0664 / 175.16956615	95.4927 / -50.0405 / 175.1441
$x_c y_c$ [px]	-12.0 / 22.0	0.1 / 18.4
f [px]	4227.62	4227.21

TABLE 7 – Comparaison des résultats entre les valeurs réelles et calculées

Les valeurs les plus importantes à contrôler sont les différences entre les points de contrôle projetés dans l'image calculée et la valeur mesurée des points de contrôle sur l'image. Il a été analysé 7 points repartis sur l'ensemble de l'image. L'ensemble des points ont une différence entre 2 et 0 pixels comme le montre la Table 8. Notre modèle est alors validé car un pixel correspond à environ 2mm à 10m avec l'image testée.

Points	u mesuré [px]	v mesuré [px]	u calculé [px]	v calculé [px]	Différences [px]
pt_10	342	2159	344	2159	2
pt_13	443	349	443	350	1
pt_20	1642	2107	1642	2107	0
pt_33	2810	378	2810	377	1
pt_40	3867	2112	3866	2112	1
pt_50	5267	2228	5267	2228	0
pt_53	5217	377	5217	377	0

TABLE 8 – Analyse de la précision sur les points de contrôle

La précédente analyse a été effectuée avec des points repartis de manière cohérente sur l'ensemble de l'image. Dans certains cas, il est probable que les points homologues soient positionnés uniquement sur une partie de l'image. Cela peut arriver s'il y a par exemple un champ et seulement un seul bâtiment dans un coin de l'image. Afin de simuler cette possibilité, il a été gardé uniquement une partie des points selon la Figure 45.



FIGURE 45 – Points homologues réduits pour déterminer la précision du modèle

En ayant qu'une partie des points homologues, les coordonnées de position montrent des écarts inférieurs à 5 cm, tandis que les différences d'orientation restent en dessous de 0.2° comme le montre la Table 9.

Élément	Valeur réelle	Valeur calculée
Position [m]	2540583.277 / 1181278.085 / 446.095	2540583.293 / 1181278.092 / 446.110
Angle [$^\circ$] $\omega \phi \kappa$	95.4391 / -50.0664 / 175.16956615	95.4400 / -50.2145 / 175.2534
$x_c y_c$ [px]	-12.0 / 22.0	-2.2 / 18.3
f [px]	4227.62	4231.85

TABLE 9 – Comparaison des résultats entre les valeurs réelles et calculées avec des points homologues réduits

Comme précédemment, il a été contrôlé 7 points de contrôle. Les écarts sont similaires à la Table 8.

3.2 Position du projet

L'importation du projet permet de détecter l'emprise du bâtiment sur l'image. Cette étape permet de réduire les temps de calcul en spécifiant la zone de pixel devant être traitée. La méthode effectuée est de récupérer les sommets de chaque face du projet, de les projeter dans l'image grâce à l'équation de colinéarité (1) à la page 5 et ensuite appliquer les paramètres de calibration. Il est alors possible de garder les extrémités des pixels projetés afin d'obtenir un rectangle englobant de l'ensemble des sommets du projet. Cette solution est représentée sur la Figure 46.

Une autre solution serait de créer une boîte englobante 3D des sommets et de récupérer uniquement les huit sommets de la boîte dans le but de réduire le nombre points à projeter sur l'image.

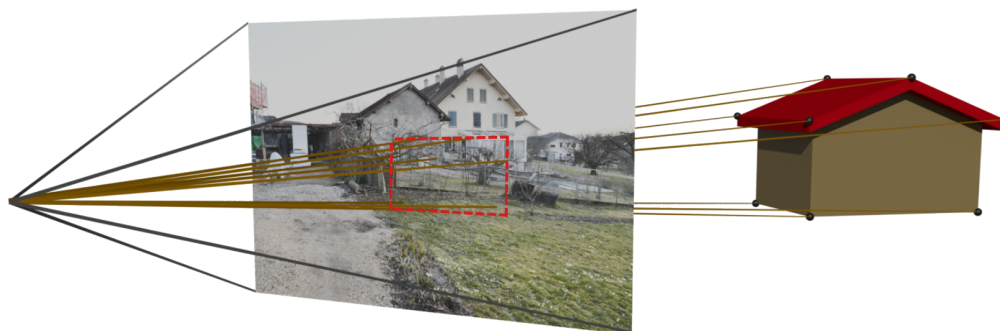


FIGURE 46 – Illustration de la projection du projet sur l'image

3.3 Nuage de points et carte de profondeur

Comme expliqué au point 2.5.4, le calcul de la carte de profondeur est effectué avec du Deep Learning et adapté avec un nuage de points. Le nuage de points est projeté sur la carte de profondeur, ce qui permet d'obtenir un jeu de donnée « valuesdepthmap » comprenant :

- La valeur de la carte de profondeur obtenue par IA (depthmapIA)
- La distance du point à l'image (dproj)
- Les coordonnées pixels de la projection du point sur l'image (UV)
- Les coordonnées terrains du point (XYZ)

Avant de créer ce jeu de donnée, le nuage de points doit être épuré afin de garder uniquement les points qui sont dans le champ de vision de la photographie et dans l'emprise du projet. Pour ce faire, les paramètres d'orientation et de position de la caméra sont récupérés dans le but de créer une pyramide qui permet de garder les points à l'intérieur. La Figure 47 montre la position d'une prise de vue et la séparation du nuage en fonction d'une pyramide. Les points verts à l'intérieur sont gardés.

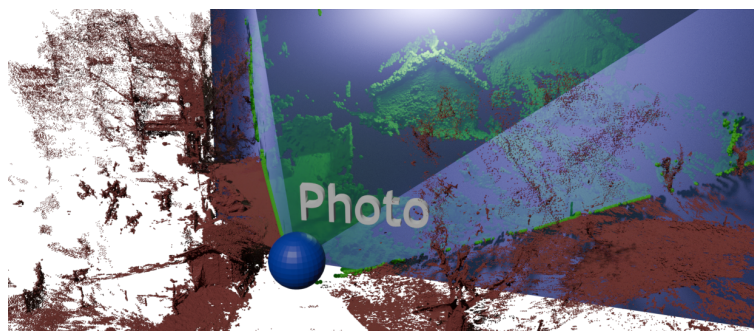


FIGURE 47 – Séparation du nuage de points selon la prise de vue

Il est maintenant possible de faire un monoplottage¹⁶ du nuage de points sur l'image et les comparer par rapport au résultat des cartes de profondeur IA. Une analyse entre les valeurs « depthmapIA » et les valeurs « dproj » montre les disparités entre les cartes de profondeur issues de « Depth Pro », à gauche sur la Figure 48, ou de « Depth Anything V2 », à droite. Pour Depth Anything V2, la valeur « depthmapIA » est inversée et n'est pas linéaire par rapport à la distance de l'image. Les 2 cartes de profondeur n'ont pas de valeur métrique.

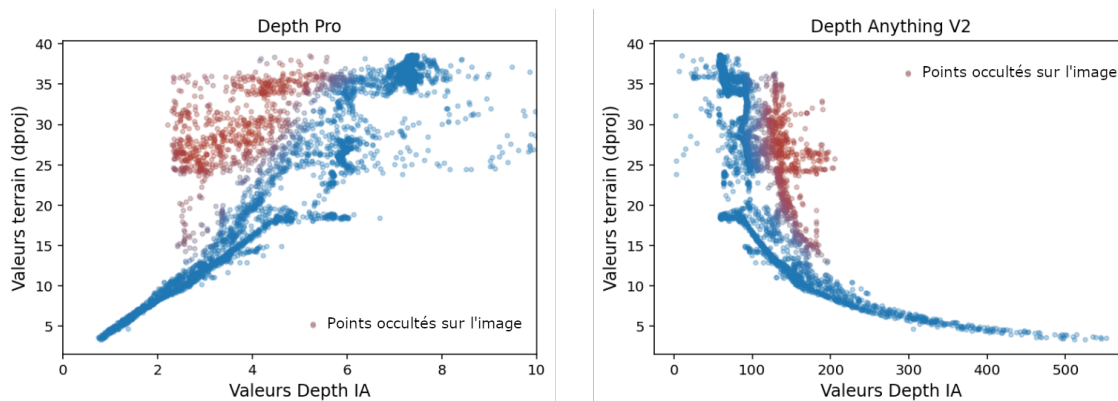


FIGURE 48 – Comparaison des valeurs terrains avec les cartes de profondeur IA

Des valeurs fausses sont présentes et particulièrement visibles avec « Depth Pro » (voir points rouges sur la Figure 48). Ces points sont le fait que certaines zones de nuages peuvent être occultées par des objets situés plus près de la caméra (voir Figure 49).

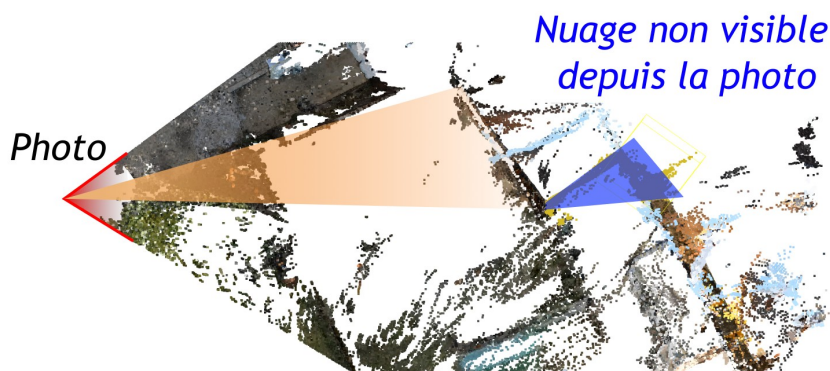


FIGURE 49 – Schéma de visibilité du nuage de points par rapport à la photo

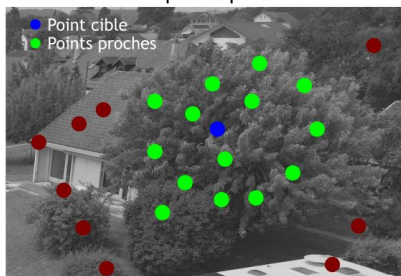
Un contrôle de visibilité permet d'éliminer les points qui ne sont associés à aucun pixel de l'image. La méthode qui a été choisie a été d'analyser chaque point projeté selon le déroulement suivant :

1. Recherche des 20 points les plus proches en regardant uniquement les pixels
2. Recherche des points dont la distance de projection est plus petite que celle de l'image cible
3. Création d'une enveloppe convexe
4. Évaluation de l'inclusion du point cible dans l'enveloppe :
 - 4.1 Point à l'intérieur : il est considéré comme occulté et supprimé
 - 4.2 Point à l'extérieur : il est considéré comme visible et gardé

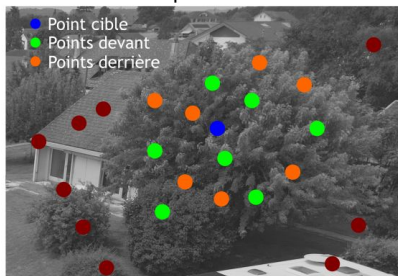
La Figure 50 schématise la détection d'un point occulté.

16. Voir Point 2.5.2 à la page 23

1. Recherche des points proches



2. Recherche des points situés devant



3. Création d'une enveloppe convexe

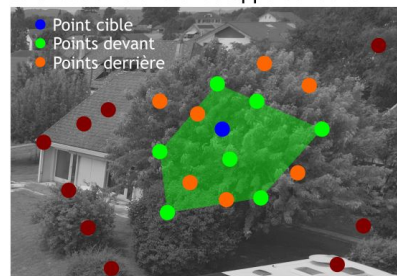


FIGURE 50 – Calcul de visibilité des points

Le calcul de visibilité a été testé. La suppression des points fonctionne correctement comme le montre la Figure 51. Les points situés derrière la haie, le bâtiment et l'arbre sont détectés et supprimés.

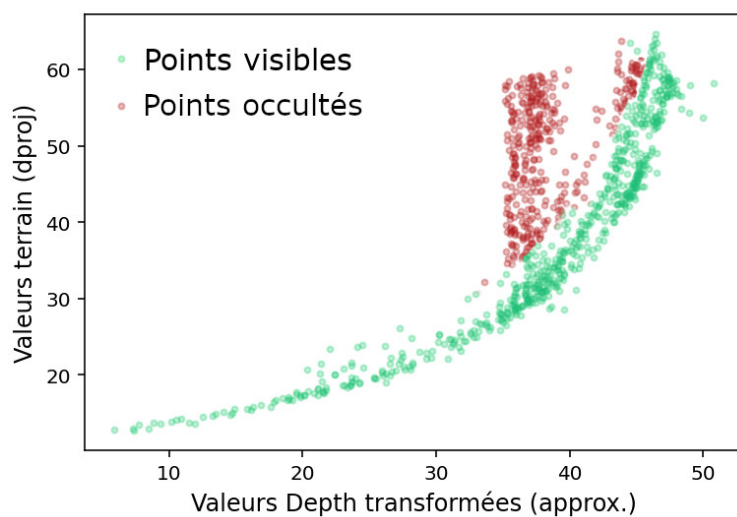
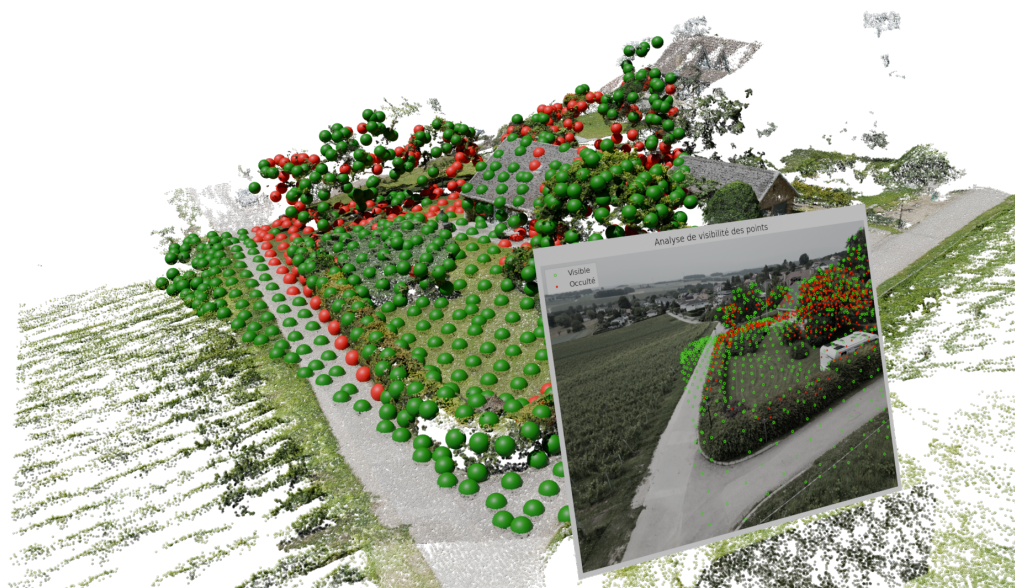


FIGURE 51 – Résultat de l'analyse de visibilité du nuage de points

3.3.1 Transformation simple de la carte de profondeur

Comme indiqué précédemment, les valeurs issues de la carte de profondeur IA n'ont aucune unité. Une transformation est nécessaire pour obtenir une carte approximative de profondeur métrique. Les moindres carrés permettent de minimiser l'erreur d'une fonction linéaire ($ax + c$), modélisé en orange sur le graphique de gauche de la Figure 52, qui épouse au mieux le nuage de points.

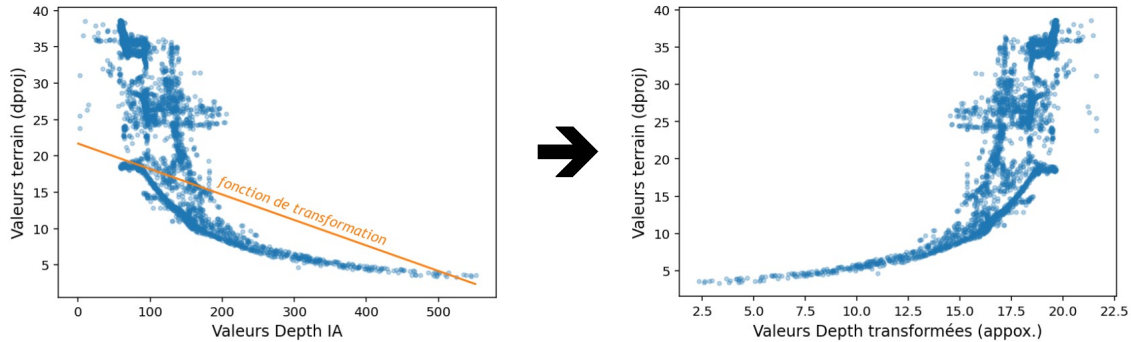


FIGURE 52 – Transformation approximative des valeurs de la carte de profondeur

Cette transformation permet d'avoir grossièrement un ordre d'échelle similaire entre les distances issues du monoplottage du nuage de points sur l'image et celles issues de la carte de profondeur IA. L'analyse de la précision en est facilitée. Sur le graphique de droite de la Figure 52 transformée en mètre, les valeurs « depthmapIA » sont approximatives. En effet, les valeurs ne suivent pas précisément une fonction linéaire telle qu'appliquée précédemment.

En appliquant un algorithme de clustering comme KMeans¹⁷ pour séparer le nuage de points, une transformation linéaire par groupe peut être réalisée pour améliorer la précision du résultat. Les deux graphiques de la Figure 53 montrent la transformation des valeurs « depthmapIA » selon deux fonctions linéaires, rose et verte. Des imprécisions sont toujours présentes mais moins importantes.

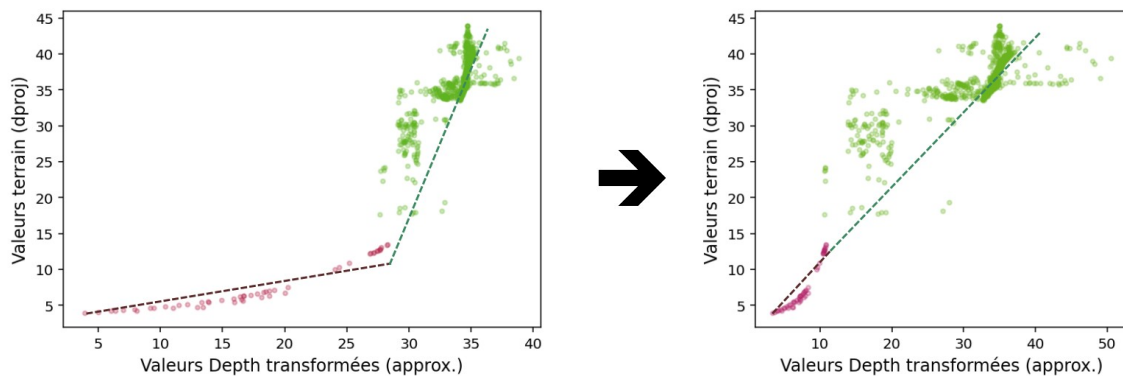


FIGURE 53 – Deuxième transformation approximative des valeurs de la carte de profondeur

3.4 Rendu du projet dans l'image

Il a été choisi d'utiliser le modèle de Phong afin de calculer le rendu de l'image (Voir Point 2.4.1) car il est facilement implémentable dans n'importe quel script. Le calcul est effectué pixel par pixel dans le but d'en définir la texture comme illustré sur la Figure 54.

17. <https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html>

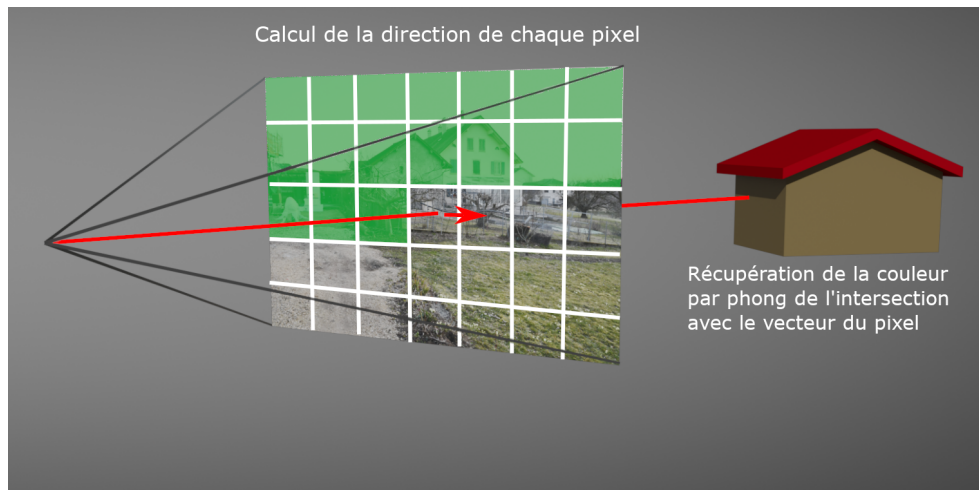


FIGURE 54 – Illustration de calcul pixel par pixel de rendu par phong

3.5 Calcul de visibilité du projet dans l'image

L'emprise du projet sur l'image a déjà été effectuée et est réutilisée afin d'effectuer le calcul de visibilité uniquement sur la portion nécessaire. Les cartes de profondeurs générées précédemment n'étant pas précise, il est effectué une détermination pixel par pixel selon les étapes suivantes :

1. Regarder si l'orientation du pixel intersecte le projet
2. Comparer la distance de projection du projet et la valeur de la carte de profondeurs approximatives.
 - 2.1 Largement devant le projet → Le pixel est situé devant le projet.
 - 2.2 Largement derrière le projet → Le pixel est situé derrière le projet.
 - 2.3 À proximité du projet → Faire un calcul par moindre carré avec les points terrain proche du pixel.

L'estimation de la précision (σ) des valeurs de la carte de profondeur permet de fixer une valeur au terme « largement » indiqué auparavant. Par simplicité, la différence la plus grande entre la valeur terrain et la carte de profondeurs (20m sur la partie droite de la Figure 53) a été utilisée. Une autre possibilité sera de déterminer des précisions variables (σ) en fonction de plage de valeurs (Exemple : de 0 à 15m la précision est de 2m, de 15 à 30m la précision est de 20m, ...).

- 2.1 $\text{depthmapIA}_{\text{transf}} + \sigma < \text{dproj} \Rightarrow$ Le pixel est devant le projet, il ne doit pas être modifié.
- 2.2 $\text{depthmapIA}_{\text{transf}} - \sigma > \text{dproj} \Rightarrow$ Le pixel est derrière le projet, il doit être modifié.
- 2.3 *sinon* $\Rightarrow$ Le pixel doit faire l'objet d'une détermination plus précise par moindre carré

Dans le cas (2.3) où il est nécessaire de faire un traitement plus précis, la méthode retenue est de prendre les points les plus proches du jeu de donnée « valuesdepthmap » en regardant principalement la valeur $\text{depthmapIA}_{\text{transf}}$ et la position du pixel. Avec ces points, il est déterminé une droite par moindre carré (en orange sur la Figure 55). Il suffit alors de prendre la valeur terrain correspond à notre pixel sur la droite (point rouge sur la Figure 55).

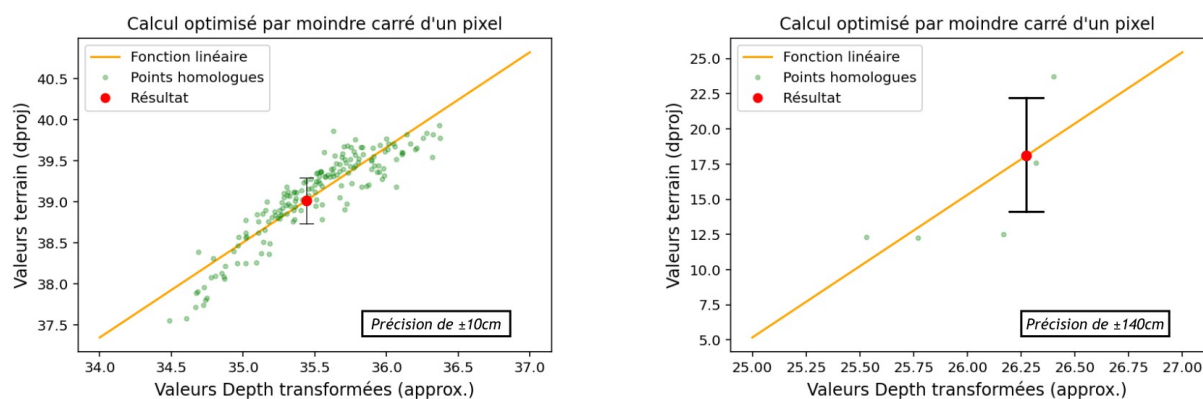


FIGURE 55 – Détermination de la valeur terrain du pixel

Les deux graphiques ci-dessus montrent le calcul moindres carrés d'une droite pour obtenir la valeur d'un pixel. Dans les 2 cas, il y a la possibilité de déterminer une valeur mais avec un ordre de précision différent. Cette précision peut être utilisée pour déterminer la visibilité de projet et ainsi permettre de rajouter un facteur de transparence pour le rendu du projet en fonction de l'intervalle de confiance. Cette étape n'a pas été effectuée dans le code.

Si aucune valeur terrain n'est disponible, la $\text{depthmapIA}_{\text{transf}}$ est utilisée. Cela ne devrait normalement pas être le cas car le nuage de points est censé être homogène à la position du projet.

4 APPLICATION

Afin de permettre à tout utilisateur d'importer une image dans notre programme, une interface graphique a été développée sous la forme d'un site web à l'aide du framework Flask¹⁸, basé sur Python. Ce framework permet d'intégrer les différents modules python nécessaires à la création d'un photomontage. Cette interface est actuellement composée de 5 pages web :

- Page d'accueil
- Liste des projets intégrés
 - Le projet pour créer le photomontage
- Les références utilisées pour effectuer le photomontage
- Tutoriel d'utilisation

La mise en page a été générée avec l'aide de Bootstrap¹⁹. Bootstrap permet de récupérer des styles et des fonctionnements, notamment sur les menus de manière simple et rapide.

Le site a également utilisé la librairie jQuery²⁰ qui permet de simplifier les commandes à effectuer pour les interventions JavaScript avec la partie front-end du site internet.

L'Annexe A donne plus d'information sur l'installation de l'application ainsi que sur son fonctionnement.

4.1 Page d'accueil

La page d'accueil permet d'avoir un descriptif du projet (voir Figure 56).

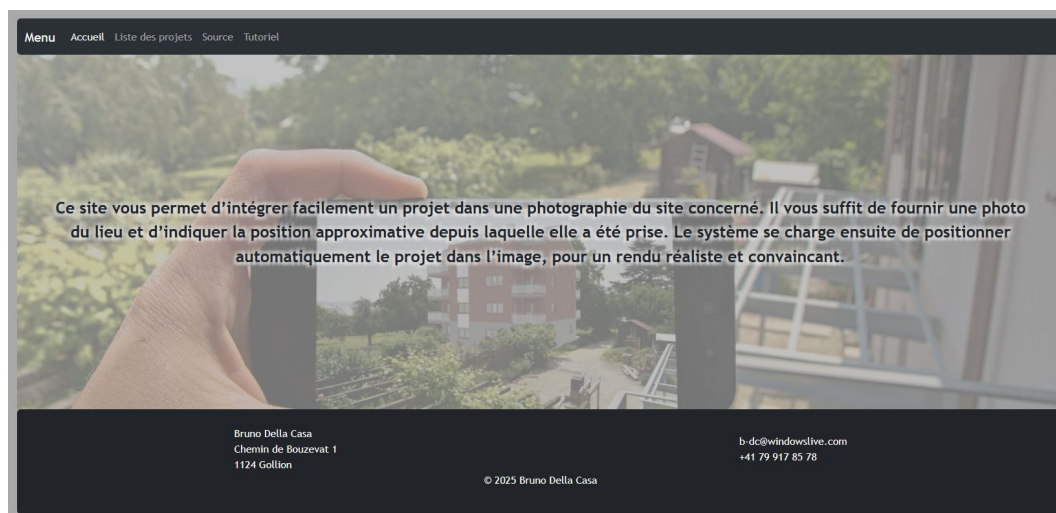


FIGURE 56 – Page d'accueil de l'application

4.2 Les projets

4.2.1 Liste des projets

La liste permet de choisir un projet, soit en cliquant sur le projet sur la carte, soit en cliquant sur la liste. Dès la sélection, il est possible d'ouvrir le projet. La Figure 57 est une capture d'écran de la sélection du projet de Bremblens. Une carte de l'ensemble des projets intégrés à l'application a été créée grâce à l'outil OpenLayers²¹.

La librairie proj4.js²² permet de modifier le système de projection de la carte. Dans ce projet, le système de

18. <https://flask.palletsprojects.com/en/stable/>

19. <https://getbootstrap.com/>

20. <https://jquery.com/>

21. <https://openlayers.org/>

22. <https://github.com/openlayers/openlayers/blob/main/src/ol/proj/proj4.js>

coordonnées correspond à l'EPSG :2056²³.

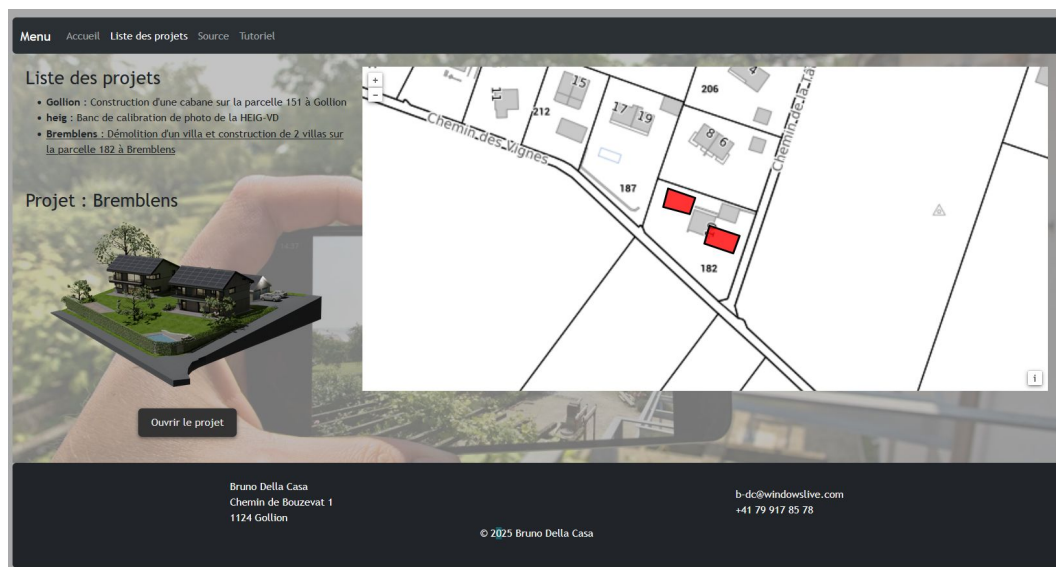


FIGURE 57 – Page de la liste de projet

4.2.2 Calcul de photomontage

Dès l'ouverture d'un projet, il est possible d'importer une image. Une position de l'image doit être indiquée, soit par l'utilisateur en cliquant sur la carte, soit en la récupérant des métadonnées de l'image.

Lors de la récupération des métadonnées, les coordonnées sont dans le système WGS84. Il est nécessaire de les transformer vers le système suisse MN95 (EPSG :2056). L'API de SwissTopo²⁴ permet d'effectuer la transformation entre les deux systèmes de manière rigoureuse.

L'utilisateur doit également choisir un mode de calcul qui définit la taille de l'image du photomontage.

La Figure 58 montre une capture d'écran lors de l'importation d'une image. Il est maintenant possible d'effectuer le calcul de photomontage.

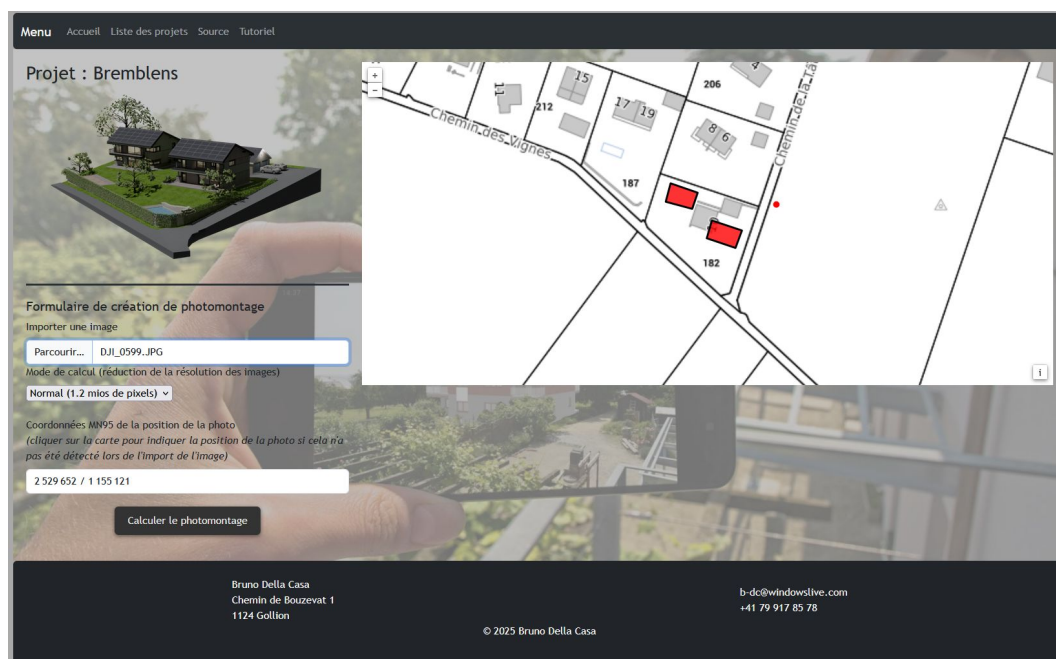


FIGURE 58 – Exemple de page de projet

23. <https://epsg.io/2056.js>

24. <https://www.swisstopo.admin.ch/fr/rest-api-geodesique-reframe-web>

5 MISE EN ŒUVRE SUR UN PROJET EXISTANT

Afin de tester les différents développements, l'entreprise « atelier pat Architectes SA » a fourni un projet qui comprend la démolition d'un bâtiment existant, la construction de deux villas, l'extension d'un garage existant et la création d'une piscine. La Figure 59 montre la maquette 3D du projet.

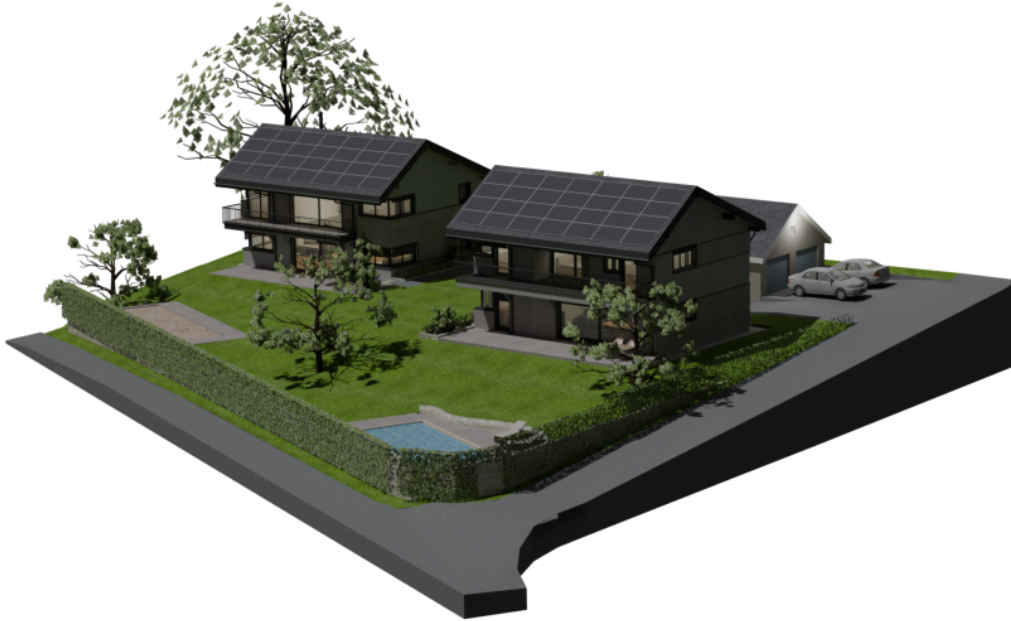


FIGURE 59 – Projet de construction

Le projet se situe en bordure de village. Il est entouré au nord et à l'ouest par des parcelles bâties et par de champ et des vignes à l'est et au sud comme le montre la photo drone sur la Figure 60.



FIGURE 60 – Photographie drone de la parcelle du projet

5.1 Acquisition du modèle de base photogrammétrique

5.1.1 Acquisition terrain

L'acquisition de photographie a été effectuée avec un drone DJI mini 2. Le drone a l'avantage de pouvoir acquérir facilement une multitude de photographie avec des points de vue différents. Il est également possible d'automatiser l'acquisition grâce à des applications tel que pix4dCapture²⁵. Les photos sont acquises sous forme de cercle avec des hauteurs différentes autour de la parcelle comme illustré sur la Figure 61. Grâce à cette méthode, il est possible d'obtenir un nuage de points sur l'ensemble de la parcelle.



FIGURE 61 – Photographie drone de la parcelle du projet

Pour effectuer le géoréférencement, 5 points de contrôle ont été levés avec un système GNSS de l'appareil « Leica GS 18 ». Cet appareil permet de déterminer des coordonnées 3D avec une précision de $\pm 5\text{cm}$ dans le cadre de référence suisse MN95. L'utilisation d'un drone équipé d'un capteur GNSS RTK aurait permis de simplifier davantage cette étape. La position des points de contrôle est illustrée dans la Figure 62. Afin de couvrir l'ensemble du chantier, un point supplémentaire aurait été nécessaire dans la partie nord-ouest de la parcelle. Toutefois, cette zone se trouvant sur une propriété privée voisine, le relevé n'a pas été réalisé, d'autant plus qu'il s'agit ici d'un exemple.



FIGURE 62 – Photographie drone de la parcelle du projet

5.1.2 Traitement Agisoft

Les écarts mesurés sur les points de contrôle après le calcul d'aérotriangulation sont en moyenne de $\pm 3\text{ cm}$, avec des erreurs de reprojection inférieures à un pixel.

25. <https://www.pix4d.com/product/pix4dcapture/>

5.1.3 Préparation des données

Caméra

Il est désormais possible d'exporter les orientations externes des images ainsi que la calibration interne du drone. Les coordonnées utilisées sont exprimées dans le système de référence MN95, caractérisé par de grandes valeurs numériques (par exemple : 2'600'000 / 1'200'000). Afin d'éviter certains problèmes informatiques présents dans certains logiciels, un décalage a été appliqué pour recentrer le système. Cette translation doit également être appliquée à la maquette 3D et au nuage de points.

Le script pour calculer la position d'une image utilisateur reprend directement les coordonnées exportées. Le système MN95 repose sur la double projection conforme cylindrique à axe oblique, ce qui engendre des différences entre les distances terrains et projetées. En toute rigueur, les coordonnées projetées dans MN95 devraient être transformées en coordonnées cartésiennes 3D pour garantir la précision des calculs métriques. Toutefois, étant donné la taille réduite du chantier (environ 50 m) et la précision des points de contrôle (± 5 cm), les corrections métriques induites par la projection peuvent être considérées comme négligeables. Il est donc acceptable, dans ce contexte, de conserver les coordonnées projetées MN95 pour les traitements photogrammétriques.

La correction de distance liée à la projection cartographique est généralement exprimée à l'aide d'un facteur d'échelle. Par exemple, dans le canton de Vaud, ce facteur atteint environ 220 ppm au niveau de La Dôle, ce qui correspond à une correction d'environ 11 mm sur une distance de 50 m. À Morges, ce facteur d'échelle est nettement plus faible, autour de 30 ppm, soit moins de 2 mm sur 50 m, ce qui confirme l'inutilité d'utiliser un système cartésien pour ce cas-ci. La formule de correction de distance pour le système MN95 est décrite ci-dessous :

$$Dist_{projMN95} = Dist_{mes} \cdot \left(1 + \frac{x^2}{2R^2} - \frac{H}{R}\right)$$

R : Rayon terrestre moyen sur la Suisse $R = 6'378'800$

H : Altitude moyenne du lieu

x : Différence sur la coordonnée Nord par rapport au point fondamental ($x = N - 1'200'000$)

Nuage de points

La photogrammétrie permet d'obtenir un nuage de points par reconstruction Multi-Vues. Le nuage de points obtenus contient plus de 4 millions de points. Il a été décidé de le réduire pour en diminuer la taille mémoire (voir Figure 63). En effet, il n'est pas nécessaire d'avoir un nuage de points très dense pour effectuer la transformation de la carte de profondeur décrite au Point 3.3. Lors de cette réduction, un contrôle est également effectué pour enlever le possible bruit du nuage de points.

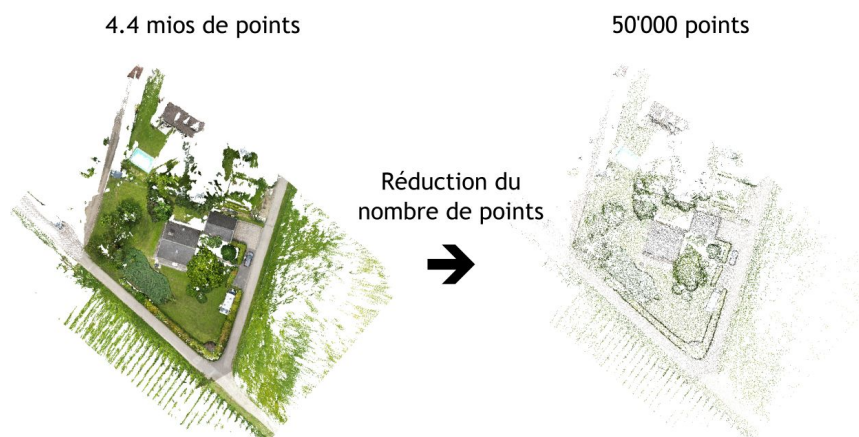


FIGURE 63 – Réduction du nuage de points

Maquette 3D

La maquette architecturale initiale comportait de nombreux éléments secondaires tels que du mobilier, des stores ou encore de la végétation, qui n'étaient pas essentiels à la compréhension du projet. Afin d'optimiser les performances et de réduire les temps de calcul lors des phases de traitement, une opération d'épuration de la scène a été réalisée. Seuls les éléments architecturaux structurants et pertinents pour la lecture du projet ont été conservés.

Dans un souci de clarté et de lisibilité, la végétation a été entièrement retirée. Cette suppression vise à ne pas masquer le bâtiment ou ses volumes, tout en allégeant la scène pour une meilleure intégration dans l'environnement photographique.

Par ailleurs, les textures basées sur des images n'ont pas pu être conservées, en raison d'une limitation technique propre à notre outil de traitement. En effet, le code Python actuellement utilisé ne prend en charge que des matériaux définis par des couleurs unies. Cette contrainte a conduit à remplacer les textures complexes par des teintes simples, ce qui permet néanmoins de maintenir une lisibilité satisfaisante des formes architecturales dans le photomontage final.

La Figure 64 montre la différence entre la maquette originale et réduite.

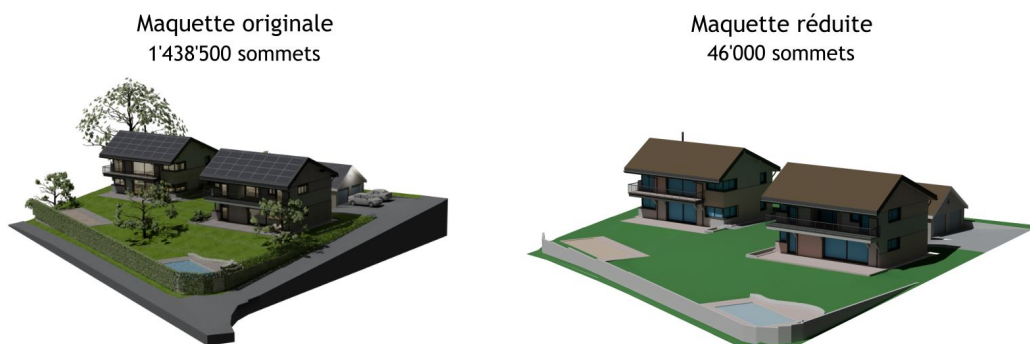


FIGURE 64 – Maquette du projet originale et réduite

5.2 Spécialité du projet



FIGURE 65 – Maquette du projet avec le nuage de points

Ce projet de construction inclut la démolition d'un bâtiment ainsi que la suppression de la végétation existante. Ces éléments doivent donc être pris en compte lors de la création du photomontage. Une contrainte particulière

est ajoutée : si un objet existant (bâtiment ou végétation) se trouve à l'intérieur de l'emprise du projet, les pixels correspondants doivent être remplacés par ceux du projet, même si un autre élément, normalement visible devant, les masquerait. La Figure 65 illustre la maquette 3D combinée avec le nuage de points de l'existant. On peut y observer que le bâtiment destiné à être démoli, ainsi qu'une partie de la végétation, se trouvent devant le volume du projet. Grâce à une emprise définie pour le projet, il est possible d'indiquer que les objets situés à l'intérieur de cette zone ne doivent pas occulter la visualisation du projet. Cette emprise est représentée en rouge sur la Figure 66.

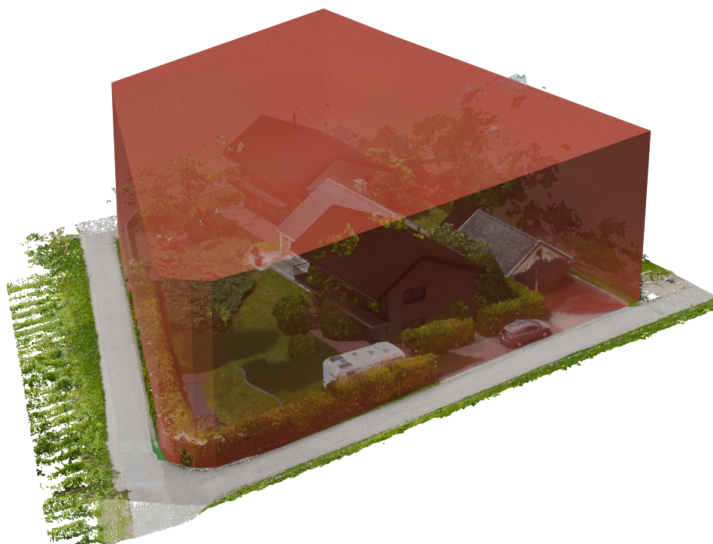


FIGURE 66 – Emprise du projet

5.3 Résultat

5.3.1 Résultat automatique

Une fois l'ensemble des données rassemblées, il est possible de générer automatiquement le photomontage du projet. La Figure 67 en présente le résultat. On observe que certaines zones en bordure du projet présentent de légères incohérences, notamment la présence résiduelle de haies qui devraient être supprimées. Malgré ces imprécisions locales, le photomontage permet une bonne compréhension de l'insertion du projet dans son environnement.



FIGURE 67 – Résultat automatique du photomontage

5.3.2 Résultat semi-automatiquement

À partir du photomontage automatique, il est facile de l'améliorer manuellement en utilisant une image de la maquette originale générée par lancer de rayon avec Blender. La Figure 68 en présente le résultat et a pu être créée en moins de 10 minutes.



FIGURE 68 – Résultat semi-automatique du photomontage

6 CONCLUSION

L'objectif de ce travail était de tirer parti des outils numériques modernes afin de permettre à tout un chacun de réaliser un photomontage réaliste et précis d'un projet de construction, sans nécessiter de compétences techniques avancées. L'enjeu principal résidait dans la capacité à garantir une précision suffisante pour éviter toute discordance entre la visualisation proposée et la réalisation effective du projet.

Dans un premier temps, les étapes classiques de création d'un photomontage ont été analysées :

- Obtention d'une photographie géoréférencée et orientée
- Modélisation 3D du bâtiment
- Calcul de la position du projet dans l'image
- Définition manuelle d'un masque de transparence dans un logiciel de retouche, afin d'ajuster la visibilité du projet selon sa position relative aux éléments de l'image.

Sur cette base, un processus automatisé a été défini, en s'appuyant sur les technologies actuelles :

- Utilisation d'une base photogrammétrique du site comme référence, afin de géoréférencer automatiquement d'autres images.
- Estimation de la position et des paramètres internes de l'appareil photo par détection automatique de points homologues. Cette détection, réalisée via des modèles d'intelligence artificielle, s'est révélée plus robuste que les approches classiques (SIFT) face aux variations temporelles et aux différences de conditions de prise de vue.
- Projection du projet depuis une maquette 3D géoréférencée fournie par l'architecte. La visibilité du projet est ensuite analysée à l'aide d'une carte de profondeur, obtenue par IA et recalibrée via un nuage de points issu du modèle photogrammétrique.

L'approche choisie, reposant sur une base photogrammétrique, permet d'assurer une précision compatible avec les attentes réglementaires et juridiques, tout en restant accessible. Les autres méthodes existantes, bien que valides, nécessitent souvent du matériel spécialisé ou des connaissances techniques avancées, ce qui en limite l'adoption par le grand public.

Pour la projection du projet dans l'image, l'équation de colinéarité a été utilisée. Quant à l'analyse de visibilité, l'utilisation de cartes de profondeur générées par IA s'est avérée être une solution souple et efficace, permettant d'obtenir facilement une information sur toute l'image, contrairement aux solutions traditionnelles. Un algorithme de transformation a toutefois été nécessaire pour convertir ces données dans un référentiel métrique, en s'appuyant sur la projection d'un nuage de points de faible densité.

Enfin, le rendu visuel du projet a été généré à l'aide du modèle d'illumination de Phong, aisément implémentable en Python. Une amélioration future envisageable consisterait à recourir à un moteur de lancer de rayons pour un rendu plus photoréaliste, bien que plus coûteux en ressources de calcul.

Actuellement, le traitement complet d'un photomontage peut durer 15 minutes jusqu'à 8h selon la complexité du projet. Le traitement qui prend le plus de temps est l'intersection entre les maillages du projet et les directions des pixels de l'image. Afin de diminuer les temps de traitement, une réduction de la taille de l'image est conseillé.

6.1 Utilisation

Son utilité varie selon le type, l'échelle et la complexité du projet. Dans certaines situations, il constitue une alternative particulièrement efficace aux gabarits traditionnels, tandis que dans d'autres, il peut être utilisé comme outil complémentaire, voire rester optionnel.

L'outil prend tout son sens dans les cas suivants :

- Projets de grande envergure (immeubles, ensembles résidentiels, zones à densifier) : la complexité et la volumétrie rendent les gabarits insuffisants pour une compréhension globale de l'impact spatial.
- Projets avec terrassements ou surélévations importantes : par exemple, lors du comblement de gravières ou de projets industriels nécessitant des mouvements de terre massifs où il est crucial de visualiser la différence avec l'état naturel du terrain.
- Situation difficile : lorsque l'installation de gabarits est techniquement contraignante (ex. : sur toitures,

terrains étroits, zones urbaines denses, hauteur de la construction trop grande pour des gabarits, etc).

- Intégration dans des démarches participatives de projet de construction : les autorités et les riverains peuvent mieux se représenter les impacts visuels réels, réduisant les incompréhensions ou oppositions.

Dans certains contextes, le recours à l'outil n'est pas forcément pertinent :

- Petits projets individuels (ex. : villas de faible emprise, annexes, garages) : les gabarits classiques restent souvent suffisants, simples à mettre en place et bien compris du grand public.
- Projets très éloignés de zones habitées et sans impact important sur le paysage : l'intérêt de produire un visuel réaliste est moindre en l'absence d'enjeux de cohabitation paysagère.

Ce développement pourrait également être utilisé partiellement par des professionnels pour faciliter la création de photomontage en diminuant la possibilité d'erreur humaine. Cela pourrait être le cas pour des concours d'architecture afin de produire rapidement des visuels convaincants à moindre coût.

6.2 Perspectives d'amélioration

Ce travail constitue une première étape prometteuse vers une automatisation rigoureuse du photomontage dans le cadre de projets de construction. Plusieurs pistes d'amélioration sont envisageables tant au niveau technique qu'en termes d'expérience utilisateur ou d'intégration réglementaire.

6.2.1 Optimisation des traitements de calcul

Le développement réalisé dans le cadre de ce mémoire a été soumis à des contraintes de temps et de ressources, limitant l'optimisation de certains traitements. Des changements pourraient réduire les temps de calcul notamment sur les calculs d'intersection des vecteurs de direction des pixels et du projet de construction.

Ces développements sont pour l'instant théorique et ont été testés uniquement sur un projet. Des phases de tests plus complets sur d'autres projets pourraient engendrer des modifications du code pour obtenir des résultats corrects en particulier sur la gestion des cartes de profondeur IA et le géoréférencement des images.

6.2.2 Amélioration du rendu visuel du projet

Le rendu actuel utilise l'illumination de Phong, suffisante pour une première visualisation, mais limitée en réalisme :

- L'intégration d'un moteur de lancer de rayons permettrait de simuler avec précision les ombres, les réflexions et l'occlusion ambiante, pour un rendu plus crédible.
- Il serait également pertinent d'intégrer des données environnementales contextuelles, telles que l'ensoleillement, la météo ou les heures de prise de vue, pour adapter dynamiquement la lumière et les ombrages.
- Une calibration colorimétrique automatique basée sur la photographie réelle pourrait améliorer la cohérence entre la scène et le projet simulé.

6.2.3 Évolution de l'interface utilisateur

L'interface actuelle, simple et fonctionnelle, peut être enrichie :

- Affichage des étapes intermédiaires de traitement (géoréférencement, projection, visibilité) pour améliorer la transparence du processus et permettre un contrôle qualité à chaque étape.
- Ajout d'une barre de progression interactive ou de messages pédagogiques durant les calculs.
- Amélioration du design graphique et de l'ergonomie pour une navigation plus intuitive.

Le développement d'une application smartphone permettrait de lancer le calcul du photomontage dès l'acquisition sans devoir passer par un site web.

6.2.4 Intégration d'informations réglementaires et contextuelles

Pour renforcer l'utilité du photomontage dans un contexte légal et administratif, plusieurs éléments peuvent être ajoutés :

- Affichage du Cadastre RDPPF (restrictions de droit public à la propriété foncière) sur la scène, afin de démontrer visuellement que le projet respecte les contraintes légales (zones de protection, distances aux limites, etc.).
- Affichage d'une barre de précision estimée concernant la localisation du projet dans l'image (en mètres), pour informer les utilisateurs du niveau de confiance de la simulation.

6.2.5 Modification des éléments supprimés

Lors de certains projets, la démolition de certains bâtiments sont prévus. Actuellement, ces bâtiments peuvent encore apparaître dans les photomontages s'ils ne sont pas superposés avec le projet. Une amélioration serait de supprimer le bâtiment en le remplaçant via un traitement par intelligence artificielle.

ANNEXE

A MANUEL

Le code a été développé avec la version 3.10 du langage de programmation python. Il est disponible sur github à l'adresse suivante : https://github.com/Brounst176/photomontage_mdt/.

Les codes suivants ont été repris pour les traitements avec l'intelligence artificielle :

- <https://github.com/DepthAnything/Depth-Anything-V2> [Yang *et al.*, 2024]
- <https://github.com/cvg/LightGlue> [Lindenberg *et al.*, 2023]

A.1 Installation

La première étape est de cloner le code github :

```
git clone https://github.com/cvg/LightGlue.git
```

Il est conseillé d'installer un serveur virtuel et d'installer les librairies nécessaire grâce au fichier « requirements.txt » :

```
python -m venv virtual
virtual\Scripts\activate
python pip install -r requirements.txt
```

Il faut ensuite télécharger le modèle pré-entraîné²⁶ pour la création IA des cartes de profondeur et le mettre dans le dossier « checkpoints ».

A.2 Manuel utilisateur

A.2.1 Démarrer l'application web

Pour lancer l'application web, il convient d'exécuter la commande suivante depuis l'invite de commande, après avoir activé l'environnement virtuel Python :

```
python flask_run.py
```

Cette commande devrait démarrer un serveur web à l'adresse <http://127.0.0.1:5000>

A.2.2 Tutoriel

Une vidéo tutorielle de l'utilisation de l'application est disponible, soit directement sur le serveur web, soit en accédant au fichier "`\\appviewer\\static\\image\\tutoriel.mp4`"

A.2.3 Création d'un nouveau projet de photomontage

Des exemples de dossier de projet sont existants dans le dossier "`data_projet`". La structure d'un dossier de projet est indiquée sur la Figure 69 et doit être mise dans "`data_projet`" :

²⁶. https://huggingface.co/depth-anything/Depth-Anything-V2-Large/resolve/main/depth_anything_v2_vitl.pth?download=true


```

ProjetName
|   emprise.obj
|   CalibExemple.xml
|   image.png
|   pointcloud.las
|   position_orientation.txt
|   projet.obj
|
|---image
|       image1.JPG
|       image2.JPG
|       ...
|
|---output
|       ...

```

FIGURE 69 – Structure par arbre des dossiers de projet

Les fichiers et dossiers suivants doivent rigoureusement avoir le même nom :

- Fichier *emprise.obj* : emprise du projet (ce qui est à l'intérieur est considéré comme démolé ou supprimé)
- Fichier *pointcloud.las* : nuage de points de l'existant
- Fichier *image.png* : image visuelle du projet
- Fichier *position_orientation.txt* : fichier des orientations externes exporté depuis Agisoft et légèrement modifié
- Fichier *projet.obj* : maquette 3d du projet
- Dossier *image* : il comprend l'ensemble des images du modèle photogrammétrique.
- Dossier *output* : il comprend les résultats des photomontages.

Le fichier *CalibExemple.xml* comprend la calibration d'un appareil photo exporté depuis Agisoft. Il peut y avoir autant de fichier de calibration que nécessaire.

Le fichier *position_orientation.txt* est obtenu depuis Agisoft en exportant les caméras avec le format Omega Phi Kappa. Des modifications doivent être apportées au fichier. Il faut ajouter le format à la suite du nom de l'image et le nom de la calibration pour l'image. Un exemple est indiqué sur la Figure 70 avec les éléments rajoutés en rouge :

```

# Cameras (35)
# PhotoID, X, Y, Z, Omega, Phi, Kappa, r11, r12, r13, r21, r22, r23, r31, r32, r33
_DSC7918.JPG 585.159 291.145 445.99 -97.12 27.89 -186.07 -0.87 0.45 0.20 0.15 0.51 0.97 0.42 0.87 -0.12 CalibExemple
DJI_1564.JPG 578.179 282.249 446.09 -98.22 26.79 -176.07 -0.89 0.45 0.00 0.06 0.11 0.99 0.45 0.88 -0.25 CalibExemple2
:

```

FIGURE 70 – Exemple de fichier *position_orientation.txt* d'orientation externe

Il faut maintenant indiquer au serveur web le nouveau projet. Pour ce faire, il faut ouvrir le fichier "appviewer\projet.json" et rajouter le projet. L'architecture du fichier est décrite à la Figure 71. Le champ *coordinate* est un point centrale du projet sous la forme de géométrie WKT²⁷. Le champ *calibration* est une liste des calibrations du modèle photogrammétrique. Il doit correspondre aux noms des fichiers créés précédemment. Le fichier projet est une géométrie *MultiPolygon* des constructions en WKT.

27. Well-Know Text : <https://libgeos.org/specifications/wkt/>

```

{
  "ProjetName" :{
    "objet" : "Description du projet de construction",
    "coordinate" : "POINT(2528496 1159644)",
    "calibration" : [CalibExemple, CalibExemple2]
    "projet" : "MultiPolygon (((2540574.18 1181272.83, 2540575.59 1181271.19,
      2540577.105 1181272.49, 2540575.71 1181274.11, 2540574.18 1181272.83)))"
  },
  :
}

```

FIGURE 71 – Architecture de *projet.json* comprenant la liste des projets dans *appviewer*

A.3 Manuel développeur

A.3.1 Dossier du code

Le code est séparé en plusieurs dossiers :

- Le dossier « appviewer » comprend l'interface web développée avec la librairie Flask
- Le dossier « data_projet » comprend les données des projets de photomontage
- Le dossier « module_python » comprend les différents modules développés pour établir un photomontage ainsi que le code repris pour le traitement IA.
- Le dossier « checkpoints » comprend les modèles pré-entraînés pour créer des cartes de profondeur IA

A.3.2 Architecture du code

Le code développé utilise 3 *class* python :

- *camera* dans « camera_module.py » qui comprend les calculs photogrammétriques
- *calcul_orientation_photo_homol* dans « ori_photo_cacl_module.py » qui comprend les créations des points homologues et le calcul de l'orientation externe et de la calibration de l'image utilisateur.
- *photomontage_depthmap* dans « photomontage_module.py » qui comprend les traitements des cartes de profondeur et la création du photomontage

Class camera

Variable principale

- ***calibrations*** (dict)

```

{
  CalibrationName (string) : {
    "w" : largeur de l'image (float),
    "h" : largeur de l'image (float),
    "cx" : PPA cx (float),
    "cy" : PPA cy (float),
    "f" : focal (float),
    "k1" : distorsion (float),
    "k2" : distorsion (float),
    "k3" : distorsion (float),
    "k4" : distorsion (float),
    "p1" : distorsion (float),
    "p2" : distorsion (float),
    "b1" : distorsion (float),
    "b2" : distorsion (float),
  },
  ...
}

```

- ***images*** (dict)

```

{
    ImageName (string) : {
        "R" : matrice de rotation (np.array, shape(3,3)),
        "S" : position de l'image (np.array, shape(3,1)),
        "camera" : CalibrationName
    },
    ...
}

```

Fonction principale

- ***import_calib(calibration_name, path_agisoft_calib)***
 - **Descr** : Importation du fichier de calibration exporté depuis Agisoft
 - **Entrée**
 - calibration_name : Nom de la camera (string)
 - path_agisoft_calib : chemin du fichier xml agisoft de calibration (string)
 - **Sortie** : Ajout de la calibration dans la variable *calibrations*
- ***import_from_class_calc_photo_homol(homol)***
 - **Descr** : Importation de la calibration et de l'orientation externe depuis la *class calcul_orientation_photo_homol*
 - **Entrée**
 - homol : *class calcul_orientation_photo_homol*
 - **Sortie** : Ajout de la calibration dans la variable *calibrations* et de l'orientation externe de l'image dans la variable *images*
- ***import_image_from_omega_phi_kappa_file(filepath)***
 - **Descr.** : Importation du fichier des orientations des images exporté depuis Agisoft. Le fichier d'orientation doit être légèrement modifié pour avoir le nom de la calibration et le format des fichiers
 - **Entrée** :
 - filepath : chemin du fichier d'orientation externe omega_phi_kappa
 - **Sortie** : Ajout des images dans la variable *images*
- ***M_to_uv(photname, M)***
 - **Descr.** : Calcul des coordonnées images uv à partir d'un points 3D M
 - **Entrée** :
 - photname : nom du fichier de la photo
 - M : coordonnée 3D d'un point (np.array, shape(3,))
 - **Sortie** :
 - uv : coordonnée en pixels dans l'image (np.array, shape(2,))
- ***calcul_proj_cam(photname, M)***
 - **Descr.** : Calcul des coordonnées 3D de la projection d'un point M sur le plan image et de la distance de projection
 - **Entrée** :
 - photname : nom du fichier de la photo
 - M : coordonnée 3D d'un point (np.array, shape(3,))
 - **Sortie** :
 - H : coordonnées 3D de la projection
 - norm_MH : distance de projection
- ***uv_to_M(photname, uv, distance)***
 - **Descr.** : Calcul des coordonnées 3D M à partir d'une distance à la caméra (S-M) et des coordonnées images uv
 - **Entrée** :
 - photname : nom du fichier de la photo
 - uv : coordonnée en pixels dans l'image (np.array, shape(2,))
 - distance : distance (float)
 - **Sortie** :

– M : coordonnée 3D (np.array, shape(3,))

- ***uv_to_M_by_dist_prof(photname, uv, dist_proj)***

- **Descr.** : Calcul des coordonnées 3D à partir d'une distance projetée sur le plan de l'image et des coordonnées images uv
- **Entrée** :
 - photname : nom du fichier de la photo
 - uv : coordonnée en pixels dans l'image (np.array, shape(2,))
 - dist_proj : distance projetée (float)
- **Sortie** :
 - M : coordonnée 3D (np.array, shape(3,))

Class calcul_orientation_photo_homol

Variable principale

- ***dict_homol*** (dict)

```
{
    u_v : {
        "coord3D" : coordonnée 3D (np.array, shape(3,)),
        "FS" : liste des FS de reprojection (list),
        "homol" : liste contenant une liste des points homologues
                  [imagename, u, v, vect_dir_uv, S] (list),
        "nb_homol" : Nbre de points homologues (int),
        "u" : coord. image largeur,
        "v" : coord. image hauteur
    },
    ...
}
```

- ***homol_array*** (np.array)

```
[
    [
        ImageName (string),
        Nbre de points homologues (int),
        coords_homol_ImageUtilisateur (np.array, shape(nb_homol, 2)) [[u,v], ...],
        coords_homol_ImageName (np.array, shape(nb_homol, 2)) [[u,v], ...],
        Score de justesse du points homologue (np.array, shape(nb_homol,))
    ],
    ...
]
```

Fonction principale

- ***first_feats_matches_calc(resize)***

- **Descr.** : Recherche des points homologues sur les cinq images les plus proches en réduisant la taille des images.
- **Entrée** :
 - resize : taille de la largeur de l'image
- **Sortie** : les points homologues sont rajoutés à la variable homol_array de la class

- ***calcul_iteratif_homol_matches()***

- **Descr.** : Recherche itérative des points homologues sur des images du modèle photogrammétrique jusqu'à obtenir le nombre de points homologues nécessaire.
- **Entrée** : –
- **Sortie** : les points homologues sont rajoutés à la variable homol_array de la class

- ***calcul_approximatif_homol_3D(dict_homol)***

- **Descr.** : Calcul des coordonnées 3D des points homologues à partir du dictionnaire des points de liaison.

- **Entrée :**
 - dict_homol : voir variable dict_homol
- **Sortie :** les coordonnées 3D sont rajoutée à la variable dict_homol de la class
- **RANSAC_DLT(k=200, n_samples=20, threshold=5.0, nb_inliers=15, dict_homol=None)**
 - **Descr. :** Calcul de l'orientation externe de l'image utilisateur avec un RANSAC de la DLT. Les valeurs des observations sont reprises de la class.
 - **Entrée :**
 - k : nbre d'itérations
 - n_samples : nbre de donnée
 - threshold : écart maximum sur les valeurs pour être valide
 - nb_inliers : Nbre de valeurs valide
 - **Sortie :** l'orientation est directement enregistrée dans la class
- **calcul_moindre_carre_non_lineaire(dict_homol, liste_key_none=[], liste_inc=["S", "angles", "f", "k1"])**
 - **Descr. :** Calcul par moindre carré non linéaire de la position d'une image. Les paramètres d'origine sont repris des variables de la class
 - **Entrée :**
 - dict_homol : Voir variable dict_homol
 - liste_key_none : Liste des clés ("u_v") de dict_homol a ne pas traiter
 - liste_inc : Liste des paramètres inconnus
 - **Sortie :**
 - Qxx : Matrice des cofacteurs des inconnues
 - x : Résultat des inconnues
 - A : Matrice des dérivées partielles
 - B : Matrice des observations
 - wi : Matrice des résidus normés
 - vi : Matrice des résidus des observations

Class photomontage_depthmap

Variable principale

- **dict_prof** (dict) : dictionnaire des valeurs de profondeurs du nuage de points et de la carte de profondeur IA

```

{
    u (float) : {
        v (float) : {
            "d" : distance de projection (float),
            "sigma" : 0.3,
            "depthIA" : valeur de la carte de profondeur IA (float)
            "point" : coordonnée du points (np.array, shape(3,))
            "normal" : normal du points (np.array, shape(3,))
        }
    },
    ...
}
```

Fonction principale

- **create_depthmap(pathimage)**
 - **Descr. :** Création d'une carte de profondeur IA
 - **Entrée :**
 - pathimage : chemin de l'image utilisateur
 - **Sortie :** Ajout de la carte de profondeur dans les variables *depthmap_IA* et *depthmap_IA_backup*
- **import_projet_obj(pathobj)**

- **Descr.** : Importation du projet depuis un fichier obj
- **Entrée** :
 - pathobj : chemin de la maquette OBJ du projet
- **Sortie** : Ajout d'un objet trimesh du projet dans la variable *projet_obj* et création de la zone du projet dans l'image dans la variable *boundaries_proj*
- **importer_image_origine(pathimage)**
 - **Descr.** : Importation de l'image utilisateur
 - **Entrée** :
 - pathimage : chemin de l'image utilisateur
 - **Sortie** : Importation de l'image utilisateur dans la variable *image*
- **liste_coord_image_pt_homologue(pathlas, calc_normal=False, inv=False, adapte_basique_prof=False)**
 - **Descr.** : Création de la variable *dict_prof*
 - **Entrée** :
 - pathlas : chemin du nuage de points
 - calc_normal : boolean pour définir s'il y a un filtre du nuage de points avec la normal du points
 - inv : boolean pour dire si il faut inverser le sens de la carte de profondeur IA
 - adapte_basique_prof : boolean pour définir s'il y a une transformation basique de la carte de profondeur IA
 - **Sortie** : Création de la variable *dict_prof*
- **transformation_simple_depthmap_IA()**
 - **Descr.** : Transformation des valeurs de la carte des profondeurs IA par une fonction linéaire entre le champ *d* et *depthIA* de la variable *dict_prof*
 - **Entrée** : Utilisation de la variable *dict_prof*
 - **Sortie** : Transformation du champ *depthIA* la variable *dict_prof* et des valeurs de la variable *depthmap_IA*
- **transformation_seconde_depthmap_IA()**
 - **Descr.** : Transformation des valeurs de la carte des profondeurs IA en séparant par groupe de points
 - **Entrée** : Utilisation de la variable *dict_prof*
 - **Sortie** : Transformation du champ *depthIA* la variable *dict_prof* et des valeurs de la variable *depthmap_IA*
- **calcul_position_projet_sur_images()**
 - **Descr.** : Création du photomontage
 - **Entrée** : Utilisation des variables *dict_prof*, *depthmap_IA* et *projet*
 - **Sortie** :
 - imgphotomontage : image du photomontage

A.3.3 Diagramme des traitements du code

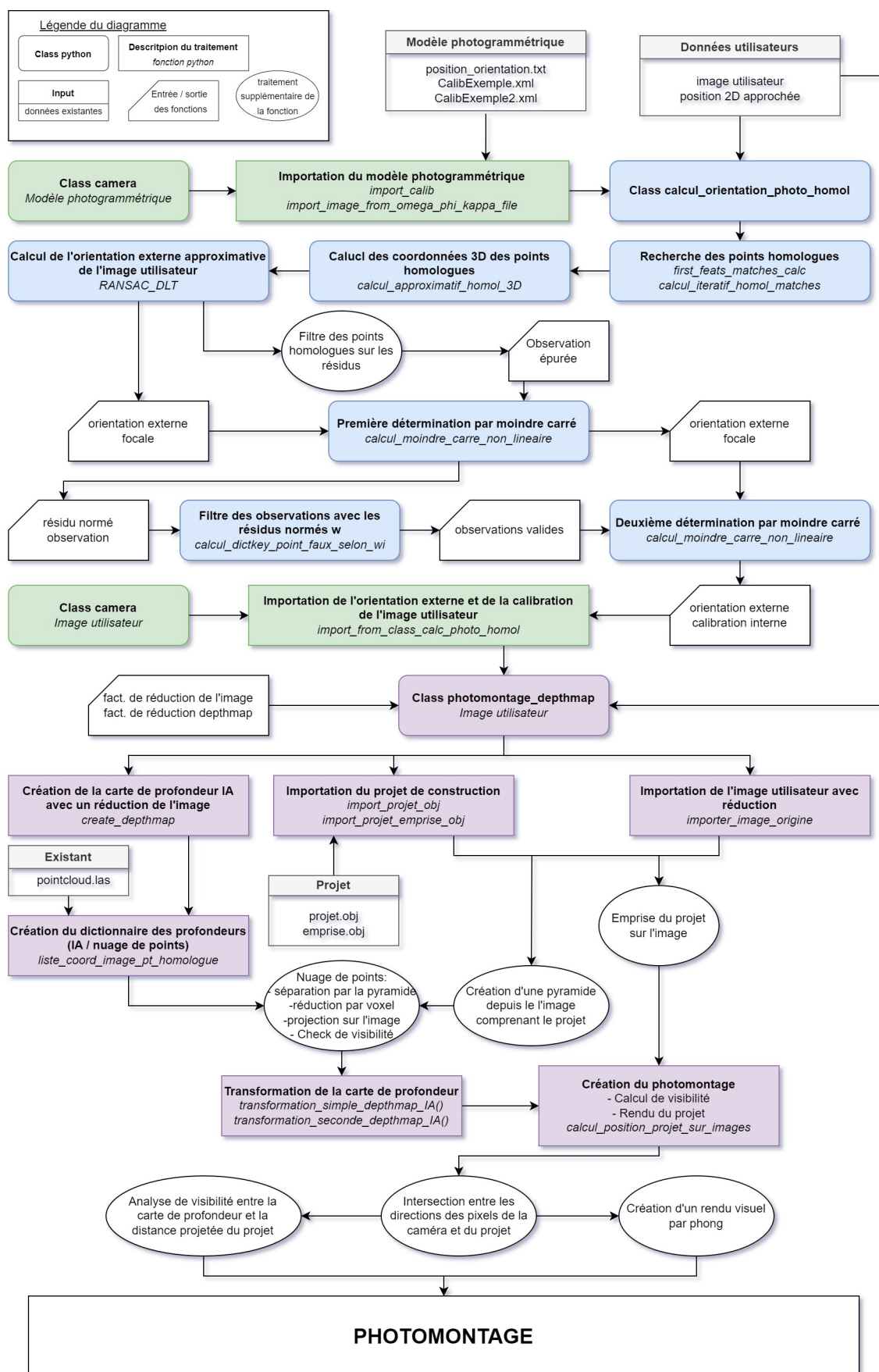


FIGURE 72 – Diagramme des étapes de programmation

B MODÈLE PHOTOGRAMMÉTRIQUE UTILISÉ

B.1 Banc de calibration

L'utilisation de banc de calibration permet de déterminer les paramètres internes d'une caméra de manière précise en utilisant des points de contrôle. La HEIG-VD à Yverdon en a créé un selon l'image sur la Figure 73



FIGURE 73 – Banc de calibration

Les coordonnées dans le système MN95 des différents points ont une précision de 1mm et sont indiquées dans la Table 10.

Numéro	X	Y	Z
pt_10	2'540'579.407	1'181'267.245	445.953
pt_11	2'540'579.398	1'181'267.252	448.116
pt_13	2'540'579.393	1'181'267.248	450.437
pt_23	2'540'578.009	1'181'269.013	450.333
pt_33	2'540'575.755	1'181'271.879	450.290
pt_43	2'540'573.543	1'181'274.691	450.330
pt_53	2'540'572.183	1'181'276.415	450.351
pt_51	2'540'572.171	1'181'276.431	448.142
pt_50	2'540'572.182	1'181'276.41	445.928
pt_20	2'540'575.184	1'181'266.592	446.137
pt_21	2'540'575.157	1'181'266.625	448.567
pt_22	2'540'575.178	1'181'266.598	449.438
pt_32	2'540'572.798	1'181'269.523	449.460
pt_42	2'540'570.757	1'181'272.225	449.464
pt_41	2'540'570.819	1'181'272.147	448.378

Suite page suivante

Numéro	X	Y	Z
pt_40	2'540'570.769	1'181'272.204	446.204
pt_101	2'540'579.467	1'181'267.161	447.300
pt_102	2'540'572.235	1'181'276.268	447.439
pt_103	2'540'572.046	1'181'276.579	447.008

TABLE 10 – Coordonnées des points de contrôle du banc de calibration

Afin de pouvoir effectuer plusieurs tests, un jeu de donnée a été acquis et traité dans Agisoft. La Figure 74 montre les différentes images et les différences par rapport aux points de contrôle.

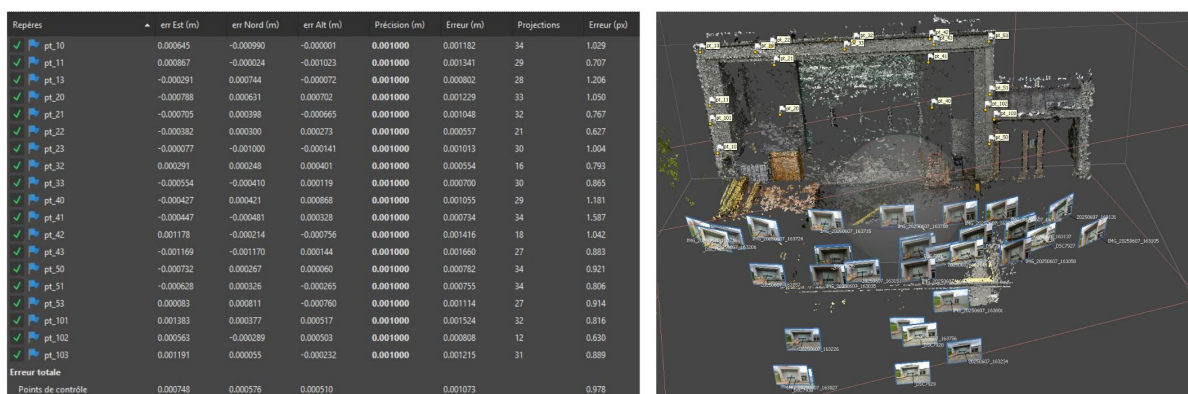


FIGURE 74 – Calcul Agisoft

B.2 Modèle photogrammétrique dans un champ

Afin de pouvoir effectuer les implémentations du code, un exemple d'acquisition de terrain a dû être effectué.

B.2.1 Choix du lieu

Un terrain de base doit être choisi afin de pouvoir développer les différentes solutions et effectuer divers tests d'implémentation. Un verger situé à l'intérieur d'un village a été choisi. Il a comme avantage d'être facilement accessible et d'avoir plusieurs cas concrets à tester avec, par exemple, des barrières et des arbres. Il n'y a également sur ce site aucune contrainte pour effectuer des vols au drone ce qui permettrait d'ajouter cette solution à l'acquisition.



FIGURE 75 – Photos drones du secteur test

B.2.2 Matériel utilisé

Le matériel utilisé n'est pas une contrainte à la bonne démarche du développement. La contrainte est qu'il faut au minimum pouvoir effectuer un référencement de l'ensemble des photos.

GoPro Max 360



FIGURE 76 – GoPro Max 360²⁸

Cette caméra permet de capturer des vidéos 360°. Il a alors été capturé l'ensemble des éléments situés autour de notre site de construction sur l'ensemble des directions.

REDcatch



FIGURE 77 – REDcatch²⁹

Le système REDcatch permet d'obtenir des coordonnées GNSS RTK des photos prises avec un appareil Reflex. Le géoréférencement peut alors être fait sans mettre en place des points GCP. Ce système n'est pas obligatoire, n'importe quel moyen de géoréférencer un projet peut être mis en place pour autant qu'il satisfasse la précision nécessaire.

28. source : <https://maximaloc.com/produit/gopro-hero-max-360/>

29. source : <https://www.redcatch.at/>

DJI mini 2



FIGURE 78 – DJI mini 2³⁰

L'utilisation d'un drone serait nécessaire si des bâtiments de plusieurs étages sont proches de notre projet. Des prises depuis le dessus du projet peuvent ainsi être acquises.

B.2.3 Acquisition terrain

Les photos acquises doivent permettre d'orienter et calibrer les prises de vue qui seront effectuées par le voisinage. Il faut avoir un jeu de donnée avec des images orientées dans l'ensemble des directions autour du projet. Il faut également obtenir un nuage de points le plus complet possible sur l'ensemble des zones du projet afin de pouvoir transformer les cartes de profondeur établies par une IA.

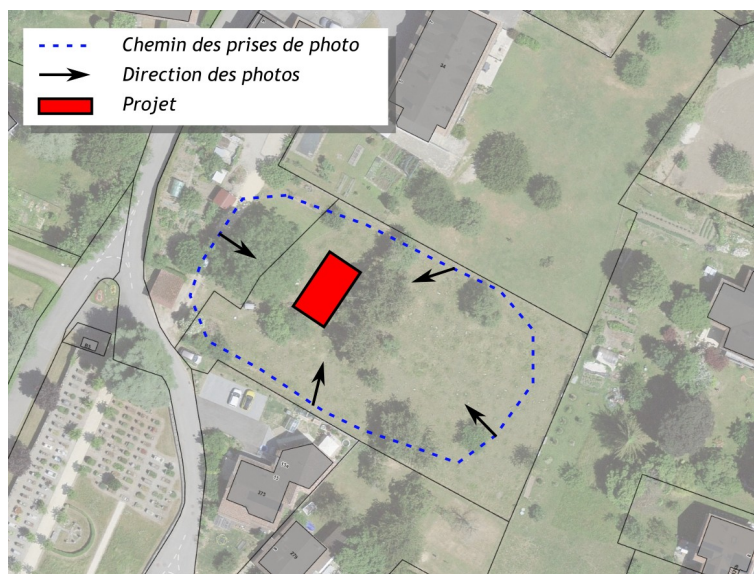


FIGURE 79 – Schéma d'acquisition des photographies

Dans ce cas, uniquement une acquisition photogrammétrique a été faite. En cas de nécessité, un relevé laser scanner permettrait d'obtenir un nuage de points plus homogène sur l'ensemble du secteur.

B.2.4 Traitement bureau

Les traitements bureau doivent obligatoirement être établis avec le logiciel « Agisoft » car les éléments de calibration ont été développés selon le manuel de Agisoft.

Il faudrait que la précision du référencement soit d'environ 20cm maximum afin que les riverains n'aient pas une fausse idée de l'emprise du bâtiment.

30. source : <https://www.digitec.ch/fr/s1/product/dji-mini-2-31-min-249-g-12-mpx-drone-14072748>

C FONCTION MATHÉMATIQUE

C.1 Méthode des moindres carrés

Le calcul par moindre carré consiste à déterminer des valeurs inconnues en minimisant les écarts $\hat{v}$ aux observations pondérées par une matrice des poids P :

$$\hat{v}^T \cdot P \cdot \hat{v} \rightarrow \min$$

L'équation des moindres carrés non linéaire est résolue par itération de la manière suivante :

$\overset{\circ}{x}$: Valeurs approchées des inconnues, x^k : Valeurs déterminées à l'itération k , l : observations

A : Matrice jacobienne ou des coefficients de linéarisation

Q_{ll} : Matrice des cofacteurs des observations dépendante de σ_l

σ_0 : écart type de l'unité de poids, il est fixée à 1 par convention, σ_l : précision des observations

$$P = Q_{ll}^{-1}$$

$$\delta l = l - f(\overset{\circ}{x})$$

$$Q_{xx} = (A^T \cdot P \cdot A)^{-1}$$

$$\delta \hat{x} = Q_{xx} \cdot A^T \cdot P \cdot \delta l$$

$$x^k = \overset{\circ}{x} + \delta \hat{x}$$

Itération du calcul avec $\overset{\circ}{x} = x^k$

L'itération peut s'interrompre selon deux manières :

- Le calcul a convergé. Cela veut dire que la différence $\delta \hat{x}$ est plus petite qu'une valeur seuil ($|\delta x| \leq \varepsilon$)
- Le nombre d'itérations a dépassé une valeur maximum ($k > K$)

Les moindres carrés permettent de déterminer les résidus normés w :

$$v_i = -\delta l$$

$$Q_{vv} = Q_{ll} - A \cdot Q_{xx} \cdot A^T$$

$$s_0 = \sqrt{\frac{v_i^T \cdot P \cdot v_i}{n - u}}$$

$$w = \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{pmatrix} = \begin{pmatrix} \frac{v_1}{s_0 \cdot \sqrt{Q_{vv11}}} \\ \frac{v_2}{s_0 \cdot \sqrt{Q_{vv22}}} \\ \vdots \\ \frac{v_n}{s_0 \cdot \sqrt{Q_{vvnn}}} \end{pmatrix}$$

avec

n : nombre d'observations

u : nombre d'inconnues

v_i : résidu des observations

v_i : matrice des cofacteurs des résidus

s_0 : écart-type de l'unité de poids empirique

L'analyse des résidus normés permet de détecter les fautes grossières. Il faut que le résidu de l'observation w_i soit compris entre une valeur maximale autorisée :

$$w_i \in [-w_{max}, +w_{max}]$$

« Traditionnellement, w_{max} est fixé à 2.5 » [Guillaume, 2023, à la page 133].

Lorsque les équations sont linéaires, aucune itération n'est nécessaire. La détermination est la suivante :

$$\hat{x} = Q_{xx} \cdot A^T \cdot P \cdot l$$

$$\delta l = l - A \cdot \hat{x}$$

C.1.1 Résolution pour le calcul de points homologues 3D

Pour déterminer la coordonnée 3D d'un point homologue (M), il est nécessaire d'obtenir les vecteurs de direction depuis les images de référence ($\vec{V}$). Ils sont déterminés grâce à l'équation de colinéarité en n'oubliant pas d'appliquer les paramètres de calibration. Il est alors possible de déterminer le point 3D avec les moindres carrés en se basant sur les équations suivantes. Il faut au minimum deux équations pour déterminer l'ensemble des inconnues :

$$\begin{cases} S_1 = M - \lambda_1 \cdot \vec{V}_1 & \text{Depuis l'image n°1} \\ S_2 = M - \lambda_2 \cdot \vec{V}_2 & \text{Depuis l'image n°2} \\ \vdots & \end{cases}$$

Dans ce cas, il faut déterminer les inconnues (x) suivantes :

$$M = \begin{bmatrix} X_M \\ Y_M \\ Z_M \end{bmatrix} \quad \text{Coordonnée 3D du point homologue}$$

$$\lambda = \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \end{bmatrix} \quad \text{Facteur d'échelle du vecteur } \vec{V}$$

$$\hat{x} = \begin{bmatrix} X_M \\ Y_M \\ Z_M \\ \lambda_1 \\ \lambda_2 \\ \vdots \end{bmatrix}$$

La matrice des observations l et la matrice jacobienne A sont déterminées comme suit :

$$l = \begin{bmatrix} X_{S_1} \\ Y_{S_1} \\ Z_{S_1} \\ X_{S_2} \\ Y_{S_2} \\ Z_{S_2} \\ \vdots \end{bmatrix} \quad \text{Coordonnées des photos,} \quad A = \begin{bmatrix} 1 & 0 & 0 & -X_{V_1} & 0 & \dots \\ 0 & 1 & 0 & -Y_{V_1} & 0 & \dots \\ 0 & 0 & 1 & -Z_{V_1} & 0 & \dots \\ 1 & 0 & 0 & 0 & -X_{V_2} & \dots \\ 0 & 1 & 0 & 0 & -Y_{V_2} & \dots \\ 0 & 0 & 1 & 0 & -Z_{V_2} & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

C.1.2 Résolution de la DTL

La DLT permet de déterminer approximativement l'orientation externe d'une image à partir de points 3D. L'équation linéaire (Équation n°3 à la page 8) est rappelée ci-dessous :

$$x = \frac{L_1X + L_2Y + L_3Z + L_4}{L_9X + L_{10}Y + L_{11}Z + 1}$$

$$y = \frac{L_5X + L_6Y + L_7Z + L_8}{L_9X + L_{10}Y + L_{11}Z + 1}$$

Les matrices utilisées pour effectuer le calcul par DLT sont les suivantes :

$$\hat{x} = \begin{bmatrix} L_1 \\ L_2 \\ L_3 \\ \vdots \\ L_9 \\ L_{10} \\ L_{11} \end{bmatrix}, \quad l = \begin{bmatrix} x_1 \\ y_1 \\ \vdots \\ x_n \\ y_n \end{bmatrix}, \quad A = \begin{bmatrix} X_1 & Y_1 & Z_1 & 1 & 0 & 0 & 0 & 0 & -x_1X_1 & -x_1Y_1 & -x_1Z_1 \\ 0 & 0 & 0 & 0 & X_1 & Y_1 & Z_1 & 1 & -y_1X_1 & -y_1Y_1 & -y_1Z_1 \\ & & & & & & \vdots & & & & \\ X_n & Y_n & Z_n & 1 & 0 & 0 & 0 & 0 & -x_nX_1 & -x_nY_1 & -x_nZ_1 \\ 0 & 0 & 0 & 0 & X_n & Y_n & Z_n & 1 & -y_nX_n & -y_nY_1 & -y_nZ_n \end{bmatrix}$$

C.2 Algorithme RANSAC

Il s'agit d'une méthode itérative qui consiste à estimer le modèle à partir d'un sous-ensemble aléatoire d'observations, puis à évaluer quelles autres observations s'accordent suffisamment bien avec ce modèle selon un seuil d'erreur prédéfini. Les observations jugées compatibles (inliers) sont alors utilisées pour réestimer le modèle avec une meilleure précision. Ce processus est répété plusieurs fois afin d'identifier la solution qui minimise le plus l'erreur des « inliers ». Les observations non compatibles (outliers) peuvent être considérées comme fausses.

L'algorithme RANSAC prend en entrée les éléments suivants :

- L'ensemble des observations (l)
- Le modèle de détermination des équations d'observation (Dans ce cas, il s'agit des moindres carrés de la DLT)
- Le nombre minimum de données nécessaires pour ajuster le modèle (n)
- Le nombre maximal d'itérations autorisées (k)
- Le seuil permettant de décider si l'observation est compatible avec le modèle (σ)
- Le nombre minimum d'« inliers » requis pour considérer qu'un modèle est valide (d)

La résolution par RANSAC³¹ se détermine comme suit :

l = ensemble des observations

$\text{meilleure_modele} = \text{None}$

$\text{meilleure_inliers} = \text{None}$

$\text{meilleure_erreur} = 999999$

Pour i allant de 0 à k :

$l_i = n$ observations aléatoires de l

$\text{modele}_i =$ Résolution de modèle avec l_i

$\text{inliers}_i = [\quad]$

Pour chaque observation de l :

Si l'observation s'ajuste au modele_i avec une erreur comprise entre $-\sigma$ et $+\sigma$:

Ajouter le point à inliers

Si le nombre d' $\text{inliers}_i \geq d$:

$\text{modele}_i_ajuste =$ Résolution de modèle avec inliers_i

$v_i =$ Erreur moyenne des inliers par rapport au modele_i_ajuste

Si $v_i < \text{meilleure_erreur}$:

$\text{meilleure_modele} = \text{modele}_i_ajuste$

$\text{meilleure_inliers} = \text{inliers}_i$

$\text{meilleure_erreur} = v_i$

Retourne $\text{meilleure_modele}, \text{meilleure_inliers}, \text{meilleure_erreur}$

31. <https://fr.wikipedia.org/wiki/RANSAC>

C.3 Produit scalaire

L'équation de base d'un produit scalaire est :

$$\vec{AB} \cdot \vec{AC} = \|\vec{AB}\| \cdot \|\vec{AC}\| \cdot \cos \alpha$$

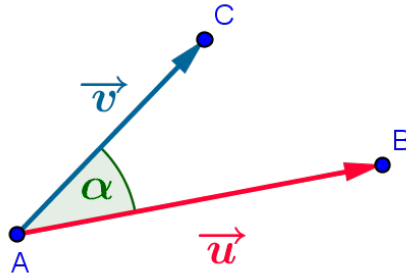


FIGURE 80 – Produit scalaire

À partir de cette équation, nous pouvons déterminer la distance de projection et la coordonnée projetée d'un point P sur un plan dont nous connaissons la normale $\vec{PH}$ et un point S situé sur le plan.

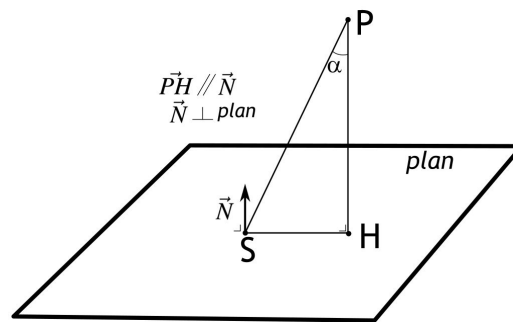


FIGURE 81 – Projection d'un point sur un plan

$$\cos \alpha = \frac{\vec{PS} \cdot \vec{PH}}{\|\vec{PS}\| \cdot \|\vec{PH}\|} = \frac{\vec{PS} \cdot \vec{N}}{\|\vec{PS}\| \cdot \|\vec{N}\|} = \frac{\|\vec{PH}\|}{\|\vec{PS}\|}$$

$$\|\vec{PH}\| = \left| \frac{\vec{PS} \cdot \vec{N}}{\|\vec{PS}\| \cdot \|\vec{N}\|} \cdot \|\vec{PS}\| \right|$$

$$H = P - \frac{\vec{PS} \cdot \vec{N}}{\|\vec{PS}\| \cdot \|\vec{N}\|} \cdot \frac{\|\vec{PS}\|}{\|\vec{N}\|} \cdot \vec{N}$$

C.4 Séparation des données par cluster

Le clustering est une technique qui permet de séparer des données en plusieurs groupes ayant des caractéristiques communes. Il existe de nombreuses méthodes permettant d'effectuer la séparation d'une donnée. Les plus utilisées sont :

- K-Means : K-moyennes est un algorithme qui divise une donnée en un nombre k de groupes transmis par l'utilisateur. L'algorithme se base sur k centroïdes qui sont redéterminés par itération successive sur la donnée.
- DBSCAN : Il sépare la donnée en fonction de leur proximité et d'un nombre minimum de points.

C.4.1 K-Means

Il est important de déterminer de manière correcte le nombre de cluster qui doivent être pris en compte. En effet, nous pouvons voir sur l'image ci-dessous l'importance de la saisie du bon nombre de groupes.

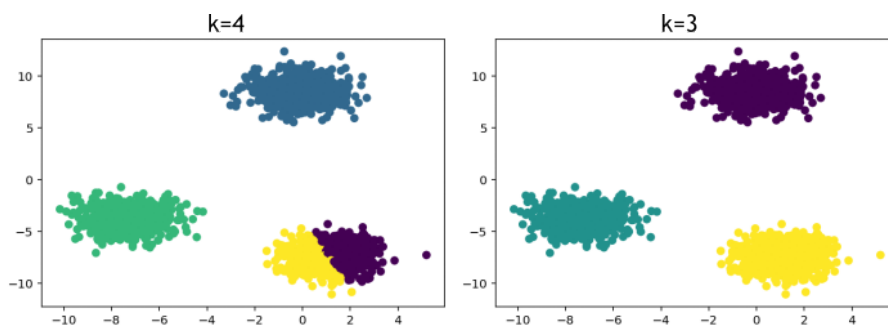


FIGURE 82 – Exemple de cluster avec K-Means

L'un des principaux avantages est le découpage d'une donnée sans séparation claire en un nombre de cluster nécessaire à l'utilisateur.

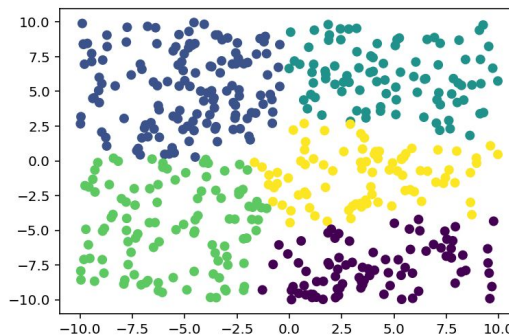


FIGURE 83 – Séparation d'une donnée en 5 groupes avec K-Means

C.4.2 DBSCAN

Il faut indiquer au DBSCAN une distance maximum et un nombre minimum de points pour appartenir à un cluster. Le DBSCAN permet de déterminer lui-même le nombre de cluster qui seront créés et de trouver des points isolés. Il est important de trouver la distance optimale afin de ne pas péjorer le résultat. Une des possibilités est de calculer pour l'entier des points les 2 voisins les plus proches et d'utiliser la moyenne de l'entier des distances pour définir la distance maximum.

BIBLIOGRAPHIE

- [Agisoft, 2025] AGISOFT (2025). *Agisoft Metashape User Manual Professional Edition, Version 2.2*.
- [Albertz et Wignenhagen, 2009] ALBERTZ, J. et WIGGENHAGEN, M. (2009). *Guide for Photogrammetry and Remote Sensing*.
- [Bochkovskii et al., 2024] BOCHKOVSKII, A., DELAUNOY, A., GERMAIN, H., SANTOS, M., ZHOU, Y., RICHTER, S. R. et KOLTUN, V. (2024). Depth pro : Sharp monocular metric depth in less than a second. *arXiv*.
- [Deluzarche, 2024] DELUZARCHE, C. (2024). Deep learning : qu'est-ce que c'est? <https://www.futura-sciences.com/tech/definitions/intelligence-artificielle-deep-learning-17262/>.
- [Edstedt et al., 2024] EDSTEDT, J., SUN, Q., BÖKMAN, G., WADENBÄCK, M. et FELSBURG, M. (2024). RoMa : Robust Dense Feature Matching. *IEEE Conference on Computer Vision and Pattern Recognition*.
- [Gressin, 2023] GRESSIN, A. (2023). *Photogrammétrie*.
- [Grélard, 2018] GRÉLARD, F. (2018). Ombrage de phong. <https://www.labri.fr/perso/fgrelard/ens/webgl/lumiere.pdf>.
- [Guillaume, 2023] GUILLAUME, S. (2023). *Méthode d'estimation et réseaux géodésiques*.
- [He et al., 2025] HE, X., GUO, D., LI, H., LI, R., CUI, Y. et ZHANG, C. (2025). Distill any depth : Distillation creates a stronger monocular depth estimator. *arXiv preprint arXiv : 2502.19204*.
- [IONOS, 2020] IONOS (2020). Quelles sont les différences entre le deep learning et le machine learning? <https://www.ionos.fr/digitalguide/web-marketing/search-engine-marketing/deep-learning-vs-machine-learning/>.
- [Kassel, 2025] KASSEL, R. (2025). K-means : Focus sur cet algorithme de clustering et machine learning. <https://datascientest.com/algorithme-des-k-means>.
- [Kasser et Egels, 2001] KASSER, M. et EGELS, Y. (2001). *Photogrammétrie numérique*. 526. Géodésie, Cartographie. Lavoisier.
- [Kumar, 2025] KUMAR, R. (2025). Guide de l'algorithme de clustering dbscan. <https://www.datacamp.com/fr/tutorial/dbscan-clustering-algorithm>.
- [Le Bris, 2010] LE BRIS, A. (2010). Expériences d'appariement d'images avec SIFT. Technical report, IGN - Laboratoire MATIS. First version of the report in 2007. Final updated version in 2010.
- [Lindenberger et al., 2023] LINDENBERGER, P., SARLIN, P.-E. et POLLEFEYS, M. (2023). Lightglue : Local feature matching at light speed. *In ICCV*.
- [Luhmann et al., 2020] LUHMANN, T., ROBSON, S., KYLE, S. et BOEHM, J. (2020). *Close-range photogrammetry and 3d imaging, 3rd edition*.
- [Pousin et al., 2023] POUSIN, F., URLBERGER, A. et FALBEL, A. (2023). Photomontage et représentation.
- [Shen et al., 2024] SHEN, X., CAI, Z., YIN, W., MÜLLER, M., LI, Z., WANG, K., CHEN, X. et WANG, C. (2024). Gim : Learning generalizable image matcher from internet videos. *In The Twelfth International Conference on Learning Representations*.
- [Starkey,] STARKEY, D. Phong illumination. <https://www.cs.montana.edu/courses/spring2009/425/dslectures/illumination.pdf>.
- [Vaillant, 2018] VAILLANT, R. (2018). Modèle de phong. <https://rodolphe-vaillant.fr/entry/85/phong-illumination-model-cheat-sheet>.
- [Yang et al., 2024] YANG, L., KANG, B., HUANG, Z., ZHAO, Z., XU, X., FENG, J. et ZHAO, H. (2024). Depth anything v2. *arXiv :2406.09414*.
- [Zeroual et Liazid,] ZEROUAL, I. et LIAZID, A. Expérimentation de la transformation linéaire directe pour différentes applications en photogrammétrie.