Reconstruction 3D et mosaïques d'images : analyse des données pour la cartographie ou la navigation

Flavien Briendo
(supervisor: L. Avantley & L. Beaudoin)

Rapport technique n°0, July 20, 2024

On pourra s'intéresser dans ce sujet aux méthodes de reconstruction 3D dense ou éparse, online (SLAM) ou offline, à la comparaison et à l'analyse des résultats obtenus, à leur certification, et au traitement des données brutes recueillies pour permettre leur utilisation. Des outils de traitement d'images classiques ou basés IA seront mis à profit pour ces travaux.

On pourra s'intéresser dans ce sujet aux méthodes de reconstruction 3D dense ou éparse, online (SLAM) ou offline, à la comparaison et à l'analyse des résultats obtenus, à leur certification, et au traitement des données brutes recueillies pour permettre leur utilisation. Des outils de traitement d'images classiques ou basés IA seront mis à profit pour ces travaux.

Keywords
Reconstruction 3D, SLAM, Analyse de données, Cartographie, Navigation

Laboratoire de Recherche de l'EPITA
14-16, rue Voltaire – FR-94276 Le Kremlin-Bicêtre CEDEX – France
Tél. +33 1 53 14 59 22 – Fax. +33 1 53 14 59 13
flavien.briendo@lre.epita.fr – http://www.lre.epita.fr/

Copying this document

Contents

Chapter 1

Introduction

Dans un problème de construction spatiale des données, de validation des résultats et de la précision de ceux-ci, nous allons nous intéresser à plusieurs technologies utilisant le traitement d'images par des algorithmes classiques, ou par IA.

Mais comment introduire ce sujet sans évoquer les utilisations pratiques? En effet, cette étude révèle plusieurs aspects du traitement d'images qui ne sont pas évoqués dans le résumé, comme l'importance de trouver des *points d'intérêt* d'une image, ou comment faire "matcher" les points entre deux images. Ces aspects sont nécessaires à la reconstruction 3D, qui est utilisée dans de nombreux systèmes comme la cartographie.

Ainsi, deux axes ont été abordés dans les travaux menés:

1. Localisation de cibles à partir des images
2. Detection de points d'intérêt basée sur les IA

Le premier axe possède plusieurs applications. Par exemple, des robots peuvent être porteurs des marqueur cibles et nous voulons pouvoir connaître leurs positions 3D par rapport à une caméra. L'autre cas d'application est l'estimation de la position et des distances entre des marqueurs posés sur le sol depuis une caméra qui possède une vue d'ensemble partielle de la scène. Ces marqueurs, si les estimations de la distance/positions sont assez précises, pourraient alors servir de réseau microgéodésique pour contrôler la qualité géométrique des nuages reconstruits à partir d'images.

Chapter 2

État de l'Art

Comme le montre la bibliographie, les études traitant de près ou de loin de ce sujet sont nombreuses, et nous allons nous appuyer sur celle-ci pour nous aider dans ces recherches.
Nous pouvons scinder cette partie en 2:

- Les outils de traitement d'images
- La place des IA dans le traitement d'image

Cette coupure est nécessaire car l'IA a permis de résoudre de nombreux problèmes. L'exemple sur le matching de points montre bien l'importance des IA, car elles ont permis une meilleure précision et une meilleure flexibilité.

2.1 Le traitements d'images

Il y a de nombreux outils de traitement d'images dans différents domaines, et il existe beaucoup d'outils comme openCV ou Pillow. Dans le cadre de l'embarqué, il nous faut 2 critères:

- Flexibilité
- Efficacité

Or openCV a certaines caractéristiques:

- Open source
- Utilisable en temps réel
- Dédié à l'IA

Par tous ces avantages, nous allons nous focaliser sur openCV[5] pour la partie *computer vision* car c'est la référence dans ce domaine. Créée initialement par la société Intel, cette bibliothèque n'a cessé de se développer.
Cette bibliothèque est composée de modules indépendants, qui permettent une grande liberté. Le module arUco [6] va nous intéresser.

2.1.1 OpenCV

OpenCV est composé de modules indépendants, dont les principaux:

- **core**: le module qui contient tous les objets de base.
- **imgproc**: dédié au traitement d'image de base.
- **video**: dédié à la vidéo
- **calib3d**: dédié à la calibration des caméras

2.1.2 Marqueurs cibles: ArUco

ArUco[6] est un module d'openCV[5] qui permet la création et la détection de marqueurs. Ces marqueurs ont certaines caractéristiques:

- des bords noirs
- une matrice déterminant son ID
- supporté par openCV
- permet la détection et la correction d'erreurs

Ces marqueurs peuvent être possiblement détectés dans une image, et leur position relative peut être estimée par rapport à une caméra, si nous savons la taille du marqueur et ses paramètres de calibration de la caméra.
Voici un exemple de marqueur généré par arUco.

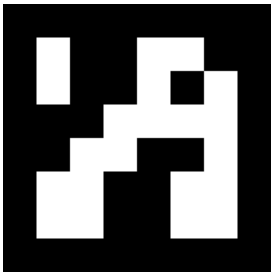

Figure 2.1: Exemple de marqueur arUco[6]

2.2 Détecteur de points basés sur les IA

Pour les IA, il y a de nombreuses manières d'aborder les problèmes, mais deux IA sortent du lot:

- SuperPoint : Permet de donner les points d'intérêt d'une image
- SuperGlue : Permet d'apparier des points entre 2 images, à noter que SuperGlue ne fait qu'utiliser les points qui lui sont donnés

Ces deux IA ont montré leur efficacité sur leurs articles respectifs (SuperPoint[7] et SuperGlue[3]). De plus, elles sont une évolution des IA précédentes et se démarquent par l'utilisation d'un réseau de convolution.

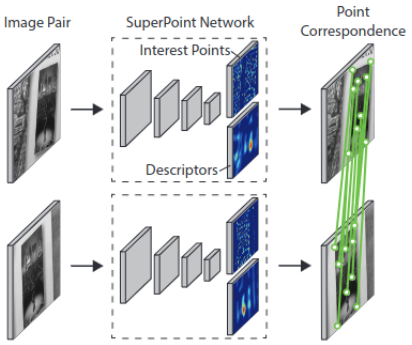

Figure 2.2: Illustration SuperPoint - extrait de l'article SuperPoint[7]

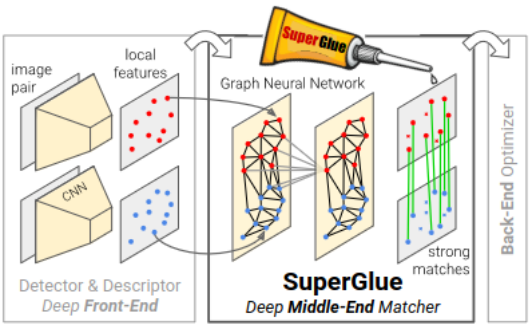

Figure 2.3: Illustration SuperGlue - extrait de l'article SuperGlue[3]

2.2.1 SuperPoint

Présentation

SuperPoint est une IA développée par les chercheurs Daniel DeTone, Tomasz Malisiewicz et Andrew Rabinovich. Ils mettent à disposition un GitHub[8] qui peut être utilisé pour tester l'IA, et le rapport[7] explique son fonctionnement.
L'objectif de SuperPoint est de détécter les *key-points* ou les *landmarks* qui peut se traduire par les points caractéristiques d'une image. Pour illustrer cela, une image vaut mieux que mille mots.

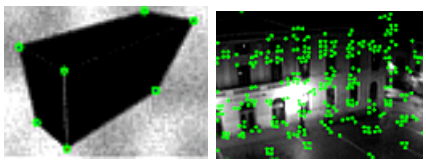

Figure 2.4: SuperPoint - extrait de l'article SuperPoint[7]

Ces images ont été extraites du rapport fait pour SuperPoint, d'où la taille. Malgré tout, la notion de points d'intérêt ou *landmarks* peut se deviner facilement.
A noter que les points d'intérêt/*landmarks* ne sont pas simples à identifier, comme le montre une autre IA sur le marché FAST.

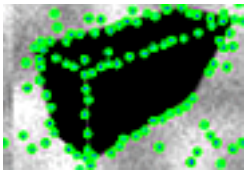

Figure 2.5: FAST - extrait du rapport[7]

En comparant les deux résultats, on remarque une nette différence en faveur de SuperPoint. Mais comment l'expliquer ? SuperPoint innove par l'utilisation d'un réseau de covolution que nous détaillerons dans la prochaine sous-partie.

2.2.2 Fonctionnement

Pour comprendre comment SuperPoint marche, il faut commencer par comprendre comment elle a été faite et comment elle a été entraînée.

Architecture

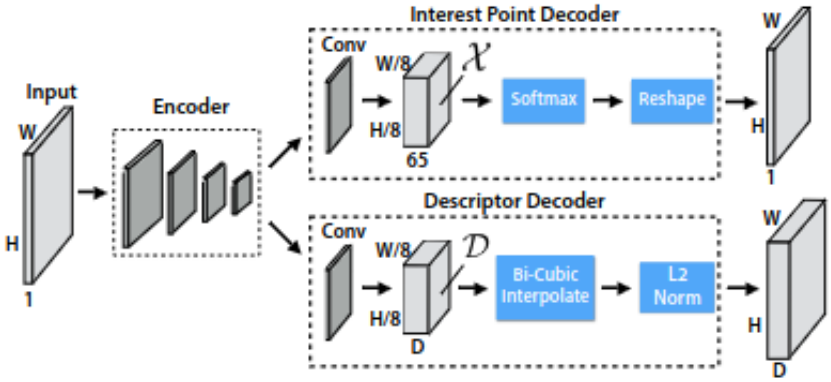

Figure 2.6: Architecture SuperPoint - extrait de l'article SuperPoint[7]

Pour commencer, SuperPoint va transformer l'image en l'encodant et en utilisant une covolution. Puis, l'image va être dupliquée afin de passer par deux traitements. En effet, il y aura une partie qui détectera les *key-points/landmarks* (nomé *Interest Point Decoder*) puis un autre qui détecte les points communs entre deux images grâce à une homothécie (nomé *Descriptor Decoder*). Cette dernière transformation va permettre d'identifier les repères malgré que l'objet auquel il est attaché ait bougé ou effectué une rotation. Il faut bien comprendre que ce traitement est une aide pour le modèle.

Entraînement

D'après le rapport, il n'y a pas de grandes bases de données labélisées avec des points d'intérêts (à noter que cet article a été publié en 2018). Ainsi, le modèle a été entraîné sur la base d'un autre modèle MagicPoint qui a été entraîné sur des formes synthétiques (cube, cylindre, ...). MagicPoint est une version simplifiée de SuperPoint. Finalement, MagicPoint a été entraîné plus de 200 000 itérations de données synthétiques.
SuperPoint a aussi été entraîné sur des données générées par MS-COCO 2014[10] avec 80 000 itérations.

Définition 1 (MagicPoint[7]) *The MagicPoint architecture is the SuperPoint architecture without the descriptor head.*

2.2.3 SuperGlue

Présentation

SuperGlue est un modèle qui évalue la correspondance des points entre deux images. Tout comme SuperPoint, il y a un GitHub [2] pour tester l'IA et un article[3] qui explique comment est créée cette IA. SuperGlue permettra d'aider à effectuer une estimation de structures 3D et de la position de la caméra.

Fonctionnement

SuperGlue fait matcher les correspondances *locales* des points entre plusieurs images.

Architecture

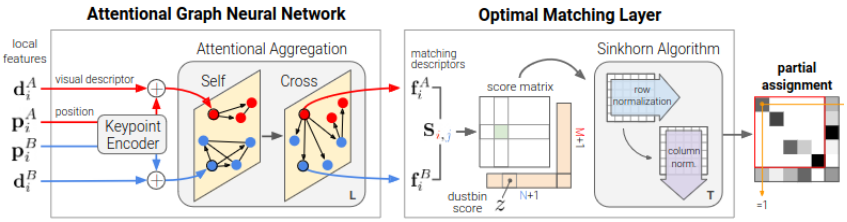

Figure 2.7: Architecture SuperGlue - extrait de l'article SuperGlue[3]

Sans s'attarder sur l'aspect technique, SuperGlue utilise un réseau de neurones pour détecter les correspondances *locales* et effectuer un traitement d'images. Il y a aussi un traitement effectué pour l'optimisation.

Je mets la définition trouvée dans l'article SuperPoint[7] au sujet de l'algorithme Sinkhorn si vous voulez aller plus loin.

Définition 2 (Sinkhorn Algorithm[3]) *The solution of the above optimization problem corresponds to the optimal transport between discrete distributions a and b with scores* $\bar{}S$*. Its entropy-regularized formulation naturally results in the desired soft assignment, and can be efficiently solved on GPU*

Entraînement

SuperGlue a été entraîné avec ScanNet[4], qui est une immense base de données contenant des vidéos annotées avec des positions 3D.

2.2.4 SuperPoint avec SuperGlue

Intérêts

SuperGlue a besoin d'un *matcher* pour récupérer la corrélation entre différentes images, et l'article SuperGlue[3] a montré que la performance de SuperPoint est améliorée par rapport à d'autres *matchers*, comme SIFT[1] qui récupère aussi les *points d'intérêt*.

Local features	Matcher	Pose estimation AUC @5°	@10°	@20°	P	MS
ORB	NN + GMS	5.21	13.65	25.36	72.0	5.7
D2-Net	NN + mutual	5.25	14.53	27.96	46.7	12.0
ContextDesc	NN + ratio test	6.64	15.01	25.75	51.2	9.2
SIFT	NN + ratio test	5.83	13.06	22.47	40.3	1.0
	NN + NG-RANSAC	6.19	13.80	23.73	61.9	0.7
	NN + OANet	6.00	14.33	25.90	38.6	4.2
	SuperGlue	**6.71**	**15.70**	**28.67**	**74.2**	**9.8**
SuperPoint	NN + mutual	9.43	21.53	36.40	50.4	18.8
	NN + distance + mutual	9.82	22.42	36.83	63.9	14.6
	NN + GMS	8.39	18.96	31.56	50.3	19.0
	NN + PointCN	11.40	25.47	41.41	71.8	25.5
	NN + OANet	11.76	26.90	43.85	74.0	25.7
	SuperGlue	**16.16**	**33.81**	**51.84**	**84.4**	**31.5**

Figure 2.8: Comparaison efficacité entre plusieurs matchers - extrait de l'article SuperGlue[3]

Lors de la comparaison entre SuperGlue et SIFT, il y a eu une différence de *matching score* de 20% en faveur de SuperGlue.
C'est la raison pour laquelle il est intéressant de combiner SuperPoint et SuperGlue.

Chapter 3

Réalisation du projet

Pour la reconstruction 3D, nous devons tout d'abord avoir une méthode pour caractériser les mesures. Pour cela, nous allons utiliser ArUco en réalisant des tests préliminaires afin de déterminer les précisions que nous pouvons obtenir.
En ce qui concerne la détection de points caractéristiques basée sur l'IA, nous nous intéresserons, notamment SuperPoint [7] et SuperGlue [3]. Nous les testerons sur une base de données afin de les évaluer dans un environnement aquatique.

3.1 ArUco

Pour commencer, nous allons explorer la partie *computer vision*. L'objectif principal est d'utiliser OpenCV et ArUco pour qualifier les résultats que les IA fourniront. Pour cela, nous devons connaître l'erreur d'ArUco.
Nous allons donc utiliser ArUco afin de déterminer la position d'un marqueur dans un environnement donné.

3.1.1 Le protocole

Cette partie présente le protocole à suivre pour reproduire l'expérience.
Avant de commencer le protocole, il faut respecter trois prérequis :

1. Calibrer la caméra
2. Produire deux marqueurs de même taille et du même dictionnaire
3. Mesurer la taille des marqueurs

Nous allons réaliser une série de mesures en respectant le protocole suivant :

1. Fixer les marqueurs de telle sorte qu'ils ne bougent pas
2. Déterminer la distance entre les deux marqueurs, cela servira de référence
3. Placer la caméra à une certaine distance

4. Prendre une capture
5. Déterminer la position relative des marqueurs
6. Pour chaque capture, déterminer la distance entre les deux marqueurs
7. Si d'autres mesures sont nécessaires, retourner à l'étape 3
8. Comparer l'évolution de la distance entre les deux marqueurs par rapport à la référence

3.1.2 Les résultats attendus

Une variation de distance entre les deux marqueurs est à prévoir. Il faut également noter que la précision diminue lorsque la caméra s'éloigne des marqueurs.

3.1.3 Réalisation

Figure 3.1: Exemple de deux photos prises lors de l'expérimentation

Lors de la réalisation, nous allons utiliser la distance de la caméra par rapport au mur pour qualifier plus précisément les résultats.

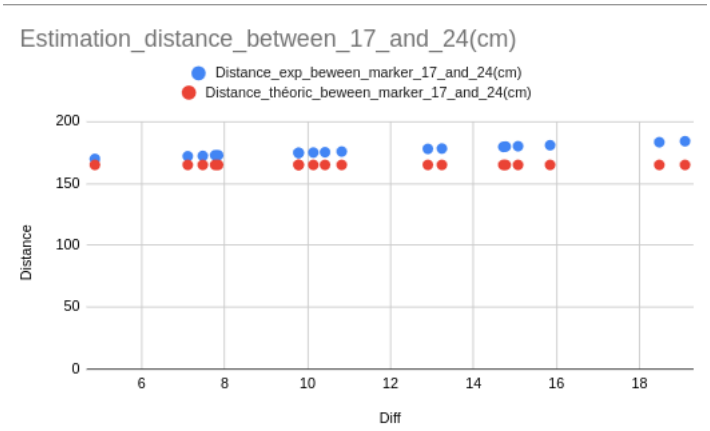

Figure 3.2: Distance entre les deux marqueurs en fonction de l'éloignement de la caméra

Comme prévu, plus la caméra s'éloigne, moins les mesures sont précises. Cependant, en procédant de manière qualitative, la distance semble beaucoup varier. Pour confirmer ce doute, nous allons commencer par mesurer la différence entre le maxima et le minima (pour la distance expérimentale) de la distance entre les deux marqueurs trouvés. Nous trouvons ainsi une valeur

de 0,75 m, ce qui est énorme si l'on considère que, dans notre expérimentation, la distance maximale entre la caméra et le mur est de 3 mètres.
Cependant, l'écart-type des valeurs est de 0,17 mètres. Les mesures peuvent donc présenter des écarts significatifs entre elles.

Ainsi, les mesures ne sont pas forcément satisfaisantes malgré un protocole rigoureux. Il faut refaire des expérimentations afin de comprendre l'origine de cet écart et améliorer la précision des mesures.
Cette imprécision peut être causée par la détection du marqueur par arUco. Pour en être sûre, les points que arUco devrait détecter avec chaque marqueur ont été sélectionnés à la main, cela a donné une solution plus satisfaisante, comme le montre le graphique ci-dessous.

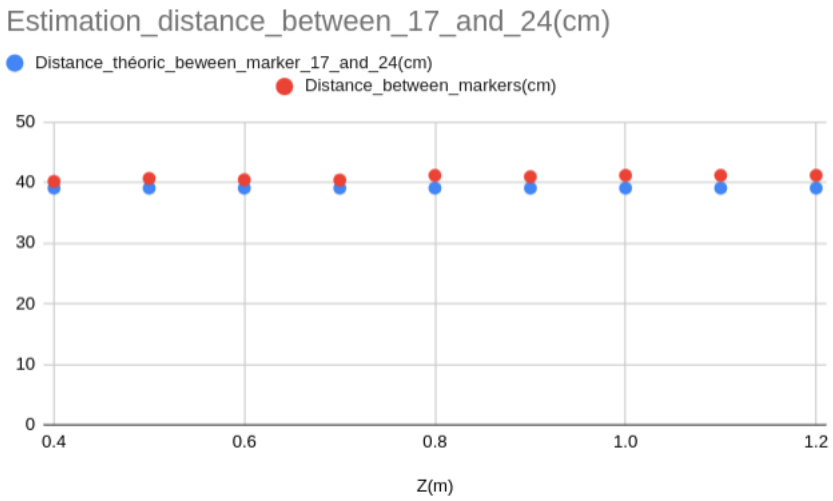

Figure 3.3: Distance entre les deux marqueurs en fonction de l'éloignement de la caméra

Dans ce cas, nous avons une variation de 1 cm entre le maxima et le minima du graphique, ce qui est déjà plus acceptable. Ainsi, nous remarquons que arUco rencontre des difficultés à détecter correctement les marqueurs avec la distance. A noter que le problème a été résolu sur ces points particuliers, mais la même résolution appliquée sur d'autres mesures donne le graphe ci-dessous.

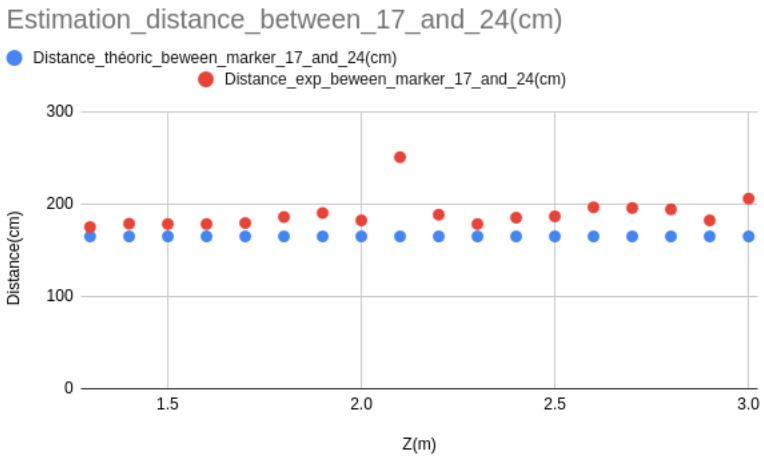

Figure 3.4: Distance entre les deux marqueurs en fonction de l'éloignement de la caméra

Ces résultats sont déjà beaucoup moins satisfaisants. Ainsi, la détection des marqueurs par arUco n'est pas la seule erreur. Une autre cause d'erreur pourrait être l'optique de déformation de fisheye, qui n'a pas encore été testée.

3.2 SuperPoint et SuperGlue

L'objectif de cette partie est de déterminer *qualitativement* l'efficacité de SuperPoint [7] et SuperGlue [3] sur des images sous-marines. Ces résultats pourront ensuite être utilisés comme référence pour qualifier d'autres expérimentations. Nous allons utiliser SuperPoint et SuperGlue sur la base de données de la Sirène [9] afin de récupérer le nombre de points caractéristiques et le nombre d'appariements obtenus dans une paire d'images. La base de données de la Sirène [9] contient une série de photos du fond marin prises lors d'une plongée aux alentours de Saint-Raphaël, sous 20 mètres de profondeur et couvrant une distance de 150 m².

3.2.1 Protocole

Nous allons traiter toutes les images de la base de données en les prenant deux par deux successivement. Ensuite, nous lancerons SuperGlue sur chaque paire.

1. Initialiser une base de données contenant une série d'images
2. Prendre deux images consécutives
3. Récupérer le nombre de points d'intérêt et le nombre de matches

3.2.2 Réalisation

Il faut savoir que SuperGlue a deux modes :

- Indoor: Un mode où SuperGlue a été entraîné sur une base de données contenant des photos d'intérieurs.
- Outdoor: un mode où SuperGlue a été entraîné sur une base de données contenant des photos d'extérieurs.

L'expérience va être réalisée sur les deux modes.

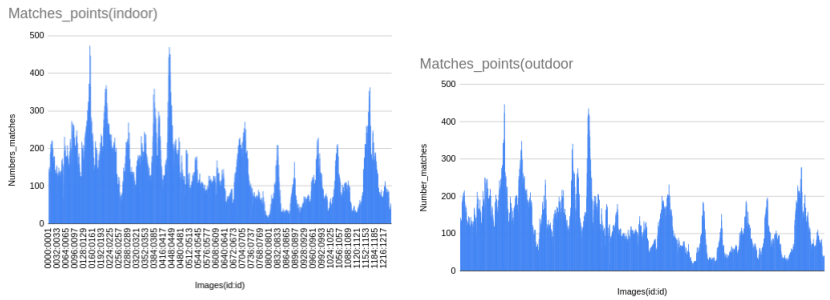

Figure 3.5: Comparaison des matches Indoor et Outdoor SuperGlue [3]

Nous remarquons immédiatement que les pics coïncident. Cela peut être causé par le nombre de points d'intérêt trouvés sur les images. En effet, moins il y a de points d'intérêt, moins il y a de points à faire correspondre.

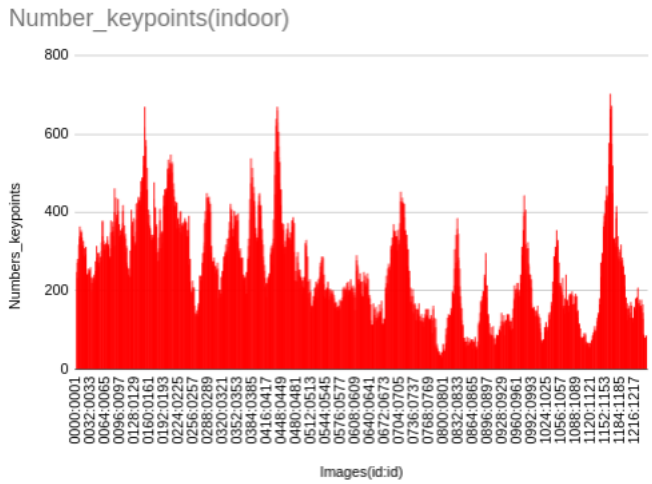

Figure 3.6: Comparaison des keypoints Indoor et Outdoor SuperGlue [3]

Nous devons maintenant comprendre d'où viennent les pics. En y regardant de plus près, les moments où il y a moins de keypoints sont ceux où il n'y a pas d'éléments "remarquables", ici traduits par la présence ou l'absence de rochers et d'algues.

SuperGlue utilise SuperPoint comme matcher, ainsi une différence de correspondance sera nécessairement causée par l'entraînement de SuperGlue.

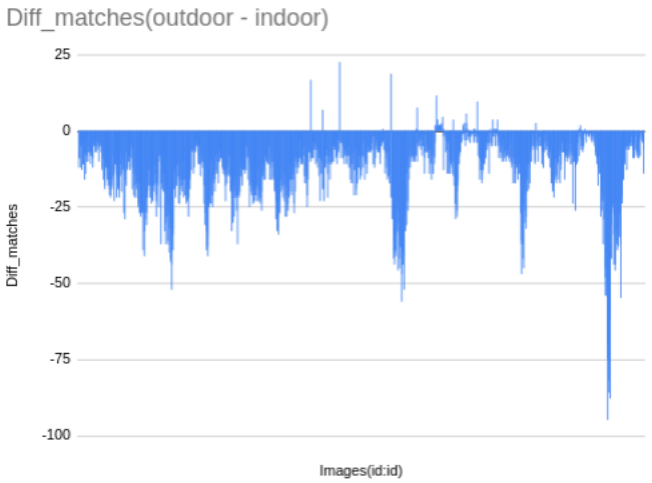

Figure 3.7: Comparaison des matches Indoor et Outdoor SuperGlue [3]

Cela montre que SuperGlue est plus efficace s'il est entraîné sur une base de données d'intérieur contrairement à une base de données d'extérieur, ce qui montre que l'entraînement est vraiment important. Maintenant que nous savons que le mode indoor donne de meilleurs résultats, nous pouvons nous focaliser dessus.

Stats(matches)		Stats(keypoints)		Stats(diff keypoints matches)	
Min	0	Min	25	Min	16
Max	474	Max	703	Max	349
Moy	138,4	Moy	248,3	Moy	109,8
Med	126	Med	232	Med	103

Figure 3.8: Indoor: keypoints matches

D'après le tableau ci-dessus, il y a une grande variation entre les maxima, que ce soit avec les keypoints ou les matches. Pour les keypoints, cela peut s'expliquer par le nombre *d'éléments spéciaux* sur l'image, en effet, s'il n'y a pas de *points d'intérêt* à extraire, alors il n'y aura pas de *matches*.

Chapter 4

Conclusion & Perspectives

Ainsi, les marqueurs arUco montrent des signes d'inprécisions, tandis que SuperPoint et SuperGlue montrent des résultats plutôt satisfaisants.

En ce qui concerne arUco, il faut essayer de trouver d'où proviennent les erreurs anormales, afin de permettre une vraie analyse de la précision sur les résultats acquis dans l'air. Si elle est satisfaisante, des tests seront effectués sur des jeux de données sous-marines.

Pour la détection de points, il faut comparer les résultats de SuperPoint [7] avec le détecteur développé par L. Avantley et L. Beaudoin (Harris adaptatif). Mais pour cela, il faut refaire une nouvelle version du code, qui est actuellement incompatible avec les versions d'openCV actuelles. Il faut aussi tester l'entraînement de SuperPoint [7] avec les points détectés par Harris adaptatif, et comparer les résultats de manière générale.

GlossaryGlossaire

Chapter 5

Bibliography

[1] Lowe', D. G. (2004). Distinctive image features from scale-invariant keypoints. (page 10)

[2] magicleap (2017). Superglue. Dernière visite le 18 mars 2024. (page 10)

[3] Malisiewicz', P.-E. S. D. D. T. (2018). Superglue: Learning feature matching with graph neural networks. *arXiv*. (pages 6, 7, 10, 11, 12, 16, and 17)

[4] of Munich', A. D. A. X. C. M. S. M. H. T. F. M. N. S. U. P. U. T. U. (2017). Scannet: Richly-annotated 3d reconstructions of indoor scenes. *arXiv*. (page 10)

[5] OpenCV Team. Opencv. (pages 5 and 6)

[6] OpenCV Team. Opencv. (pages 5 and 6)

[7] Rabinovich', D. D. T. M. A. (2018). Superpoint: Self-supervised interest point detection and description. *arXiv*. (pages 6, 7, 8, 9, 10, 12, 16, and 19)

[8] rpautrat (2017). Superpoint. Dernière visite le 18 mars 2024. (page 7)

[9] SEAL, LRE, E. (2024). Mermaid underwater dataset. Dernière visite le 26 juin 2024. (page 16)

[10] ´ar", T.-Y. L. M. M. S. B. L. B. R. G. J. H. P. P. D. R. C. L. Z. P. D. (2018). Microsoft coco: Common objects in context. *arXiv*. (page 9)