Comparaison de motifs sur des nuages de points 3D et application sur des monnaies et objets celtiques

Sofiane Horache

PSL UNIVERSITÉ PARIS

THÈSE DE DOCTORAT
DE L'UNIVERSITÉ PSL

Préparée à MINES ParisTech

Comparaison de motifs sur des nuages de points 3D et application sur des monnaies et objets celtiques

Soutenue par
Sofiane HORACHE
Le 07 06 2022

École doctorale n°621
ISMME

Spécialité
Informatique temps réel, robotique et automatique - Paris

Composition du jury :

Géraldine MORIN Professeure, ENSEEIHT	*Présidente*
Sylvain MARCHAND Professeur, Université de la Rochelle	*Rapporteur*
Mathieu AUBRY Chargé de Recherche, École des Ponts Paristech	*Rapporteur*
David WIGG-WOLF Docteur, German Archaeological Institute	*Examinateur*
François GOULETTE Professeur, Mines Paris, Université PSL	*Directeur de thèse*
Jean-Emmanuel DESCHAUD Chargé de recherche, Mines Paris, Université PSL	*Examinateur*
Thierry LEJARS Directeur de recherche, CNRS - ENS Ulm, Université PSL	*Codirecteur*
Katherine GRUEL Directrice de recherche émérite, CNRS - ENS Ulm, Université PSL	*Examinatrice*

MINES ParisTech | PSL

Remerciements

Je voudrais tout d'abord remercier les membres du jury de thèse à savoir Géraldine Morin, David Wigg-Wolf, Mathieu Aubry et Sylvain Marchand pour leurs critiques et leurs commentaires sur le manuscrit et la présentation.

Je tiens à remercier également mes directeurs de thèse Katherine Gruel, François Goulette, Jean-Emmanuel Deschaud, Thierry Lejars pour leur accompagnement et leur soutien scientifique tout au long de l'élaboration de cette thèse. Il a été très enrichissant d'avoir pu être encadré par des experts en nuages de points 3D et en archéologie celtique. Je m'estime chanceux d'avoir participé à faire voir le jour à cette thèse pluridisciplinaire grâce à eux. Merci notamment à Katherine de m'avoir fait découvrir ces monnaies Riedones.

Je remercie les partenaires institutionnels, en particulier l'école PSL dont le programme IRIS SDDS a financé ma thèse. Je souhaite remercier également le Musée de Bretagne qui a autorisé la publication de ses monnaies dans le cadre de cette thèse.

Olivier, Mattéo, Caroline, Benjamin, et Jean-Baptiste, collaborateurs du laboratoire AOROC de l'ENS m'ont aidé en particulier pour la constitution du jeu de données. Sans eux, je n'aurai pas eu la base de travail nécessaire à l'éclosion de cette thèse.

I would like to say thank you to the members of the torch-points3d team especially Nicolas, Thomas, Loic, Chris. This collaboration was amazing, it was a fantastic experience! So Thank you very much!

Je voudrais remercier les membres du Discord PhD Student. Merci aux doctorants et doctorantes d'avoir partagé tous ces conseils, ces victoires, ces coups de blues, ces actualités liés à la thèse ou non.

J'exprime ma gratitude aux membres permanents du Centre de Robotique de Mines Paris : Fabien, Arnaud, Cyril, Bogdan, Alexis, Amaury, Christine, Christophe, David et Jacky (sans eux l'ambiance du laboratoire ne serait pas la même!) qui ont permis ce cadre chaleureux dans lequel j'ai pu travailler au quotidien.

Je tiens aussi à témoigner ma reconnaissance aux docteurs et ingénieurs qui sont passés par ce laboratoire avant moi, en particulier Daniele, Michelle, Hughes, Xavier, Paul, Grégoire, Guillaume, Martin D., Martin B et Emmanuel qui ont contribué à m'initier aux rouages du doctorat et avec lesquels j'ai eu le plaisir d'échanger.

Merci aussi aux doctorants, Sami, Colin, Agapius, aux doctorants de la V026, Joseph, Thomas, Camille, Arthur, Jésus, Raphaël, ainsi qu'aux doctorants de l'équipe NPM3D, Jean-Pierre, Pierre, Jules, Louis, Nathan pour leur soutien et cette super ambiance. J'ai pu avoir des discussions passionnantes sur une multitude de sujets qui m'ont fait grandir.

Je tiens à remercier mes amis de Télécom, en particulier Aloïs (merci de m'avoir aidé dans ma recherche d'emploi!), Wissem, Ramzi, Benjamin, Astrid, Xavier, Paul et Vincent qui ont participé à me soutenir durant ces années de thèse.

Un très grand merci à ma famille. Je souhaite remercier ma grand-mère, mes oncles et tantes, mes cousins, qui ont toujours souhaité le meilleur pour moi. Je remercie également Anass, mon grand frère, et Ryan, mon petit frère, qui sont toujours présents à mes côtés dans les meilleurs et les moins bons moments.

Enfin, last but not least, je rends grâce à mes parents pour tout ce qu'ils ont fait pour moi, pour tous les brillants conseils qu'ils m'ont apporté et pour tout l'amour qu'ils me témoignent. Je leur dédie ma thèse.

Table des matières

Glossaire

charactéroscopie Méthode visant à déterminer les caractéristiques propre à chaque coin. 20

coin outil permettant d'imprimer un motif sur une monnaie. 18

CPD *Coherent Point Drift*. Algorithme de recalage local probabiliste basé sur l'algorithme *Expectation Maximisation* pour conjointement estimer la densité de probabilité des nuages de points et la transformation. 38

descripteurs Un descripteur est un vecteur de dimension fixe qui représente la géométrie locale autour d'un point. Le descripteur peut être calculé à partir de formules pré-définies ou par des méthodes basées sur l'apprentissage profond. 41

FGR *Fast Global Registration*. Algorithme robuste d'estimation de transformation plus rapide que RANSAC mais moins efficace. 43

flan morceau de métal avant d'être frappé par un coin monétaire. 18

FMR *Feature Match Ratio*. Métrique pour évaluer le résultat de la mise en correspondance de points dans un nuage de points. 99

ICP *Iterative Closest Point*. Algorithme de recalage local et itératif se basant sur les plus proches voisins pour obtenir des correspondances qui permettent de calculer une transformation. 34

KPConv *Kernel Point Convolution*. Convolution sur des nuages de points basée sur des noyaux. 46

Minkowski Engine Bibliothèque python pour faire de l'apprentissage profond sur des nuages de points voxélisés (comme la bibliothèque torchsparse). 46

MLP *Multi-Layer Perceptron*. Réseau de neurones entièrement connectés et multi-couche.. 94

permudroèdres polygone à $d!$ arêtes et $\frac{(d-1)d!}{2}$ sommets où d est la dimension de l'espace. Un treillis en permudroèdres est un ensemble de points qui sont les sommets de permudroèdres qui quadrillent l'espace. 38

RANSAC *Random Sampling Consensus*. Algorithme robuste d'estimation de transformation. 43

SRE *Scaled Registration Error*. Métrique pour évaluer le résultat du recalage. Contrairement à l'erreur de rotation et de translation, le SRE ne varie pas lors de la translation des nuages de points. 58

TEASER Algorithme robuste d'estimation de transformation découplant l'estimation de la rotation et la translation. 44

torchsparse Bibliothèque python pour faire de l'apprentissage profond sur des nuages de points voxélisés (comme *Minkowski Engine*). 98

vérité terrain La vérité terrain sont les informations obtenues et vérifiées par les experts qui permettent l'évaluation et la comparaison quantitative de différents algorithmes. Dans le cas du recalage, la vérité terrain entre une paire de nuage de points est la transformation (rotation et translation) qui aligne les deux modèles 3D. Pour le *clustering*, il s'agit des labels de chaque image (par exemple, les coins de chaque monnaie). 84, 85

Liste des tableaux

Table des figures

Chapitre 1

Introduction générale

La vision par ordinateur est une discipline de l'informatique qui étudie les algorithmes de traitements d'images 2D ou 3D. Les sujets de cette discipline vont de l'acquisition physique jusqu'à la reconnaissance d'objet en passant par les traitements intermédiaires (comme la super-résolution ou la reconstruction 3D) [Szeliski, 2010]. Ce domaine a connu récemment un essor avec notamment l'arrivée de l'apprentissage profond. De nombreuses applications ont vu le jour dans l'industrie, l'imagerie médicale, la robotique ou le véhicule autonome. Aujourd'hui, certains questionnements archéologiques et informatiques coïncident et nécessitent le développement d'outils modernes basés sur des algorithmes de vision par ordinateur ou d'intelligence artificielle [Koppe, 2022, Cherner, 2021].

L'archéologie s'intéresse à l'étude d'objets trouvés lors de fouilles ou de collections de musées. Les archéologues collectent, analysent et interprètent principalement des données visuelles. Par exemple, il peut s'agir des objets eux même, de photographies, d'acquisitions 3D par photogrammétrie ou lumière structurée, d'acquisitions par rayon X, etc. Cependant, l'analyse de ces données visuelles est très longue, fastidieuse et requiert une très grande expertise. Lorsque la base de données devient trop volumineuse, l'analyse manuelle de celle-ci peut prendre des mois, voire des années. De plus, les résultats de cet analyse restent subjectifs sans méthode statistique ou géométrique.

L'objectif de cette thèse est de concevoir de nouveaux algorithmes de vision par ordinateur pour traiter automatiquement ces bases de données visuelles et fournir aux spécialistes de nouveaux outils d'analyse.

Lors de ces travaux, nous nous sommes principalement intéressé à la numismatique qui est l'étude des monnaies et des médailles sous tous ces aspects. L'analyse du coin monétaire qui est l'outil permettant d'imprimer un motif sur la monnaie ouvre de nombreuses pistes de recherches importantes en histoire de l'art ou en histoire économique. En effet, pour faciliter les échanges, solder les armées, payer les tributs, des séries monétaires en or, argent ou bronze, ont été produites en grandes quantités. Les images figurées sur chaque face signalent l'identité du pouvoir émetteur. Pour trier ces séries, dater les différents types et évaluer les quantités frappées, les spécialistes les classent en fonction des coins.

Pour certaines civilisations, peu de textes nous sont parvenus. Nous savons donc souvent très peu de choses sur les infrastructures réelles qui sous-tendent la conception, la production et l'émission des séries monétaires de ces civilisations. Il ne reste que les monnaies pour étudier leurs échanges monétaires. De plus, les monnaies font partie des objets qui ont été produits en masse. Les bases de données de monnaies sont donc suffisamment volumineuses pour pouvoir évaluer rigoureusement les algorithmes développés et utiliser des méthodes d'apprentissage. Grâce aux outils informatiques, il est possible de proposer un classement automatique des images imprimées sur les monnaies dans la mesure où ces images sont théoriquement reproduites à l'identique par impression d'une même forme sur de nombreux exemplaires (Shen *et al.* [Shen et al., 2019] ont proposé un outil informatique pour retrouver des copies de dessins dans une base de données d'oeuvres d'art). Le défi pour les algorithmes automatiques est que deux monnaies de

(a) Coin 1 (b) Coin 1

(c) Coin 1 (d) Coin 2

FIGURE 1.1 – Variabilité et similarité de quatre monnaies Riedones (du droit) frappées avec deux coins différents. (a), (b), (c) sont des photographies de monnaies d'un même coin alors que (d) est la photographie une monnaie venant d'un coin différent. Crédit : Katherine Gruel

même coin peuvent avoir des états de conservation très différents (usure, cassure). De plus, le coin peut s'émousser avec le temps ce qui affecte l'image de la monnaie. Finalement, des coins faits par un même graveur sont très similaires. La figure 1.1 montre qu'il est très difficile pour un œil non entraîné de faire la différence entre deux coins. Un outil automatique d'identification de coin basé sur des algorithmes de vision par ordinateur pourrait beaucoup apporter. Si ces outils s'avèrent efficaces pour les études monétaires, ils devraient l'être également pour l'analyse de tout motif réalisé à partir de poinçons ou de moules. En effet, la reconnaissance de coin est finalement un cas particulier de la reconnaissance de motifs estampés sur d'autres types d'objets. L'objectif est de déterminer pour deux motifs semblables s'ils viennent de la même matrice d'impression ou s'ils ont été gravés à la main.

Afin de répondre aux objectifs, nous avons tout d'abord constituer un jeu de données appelé Riedones3D. Ce jeu de données nous a servi à évaluer des algorithmes et à entraîner des méthodes basées sur l'apprentissage profond. Grâce à Riedones3D, nous avons pu proposer une chaîne de traitement pour regrouper les monnaies selon leurs coins. Nos contributions scientifiques sont :

— Nous proposons Riedones3D : une base de données annotée et publique avec la collaboration de Katherine Gruel du laboratoire AOROC-ENS et avec l'accord du musée de Bretagne de Rennes [Horache et al., 2021b]

— Nous établissons un bilan des travaux les plus récents à la pointe de l'état de l'art en recalage sur Riedones3D [Horache et al., 2021b]

— Nous proposons une méthode qui permet de classer plusieurs monnaies selon leur coin [Horache et al., 2021b, Horache et al., 2020].

— Nous proposons une méthode de recalage basée sur l'apprentissage profond, capable de fonctionner sur un jeu de données 3D venant de différents capteurs [Horache et al., 2021a].

Ce manuscrit s'articule en quatre chapitres. Le chapitre 2 montre comment les numismates reconnaissent les coins. Une analyse des approches et de leur évolution jusqu'aux méthodes utili-

sant l'imagerie 3D permet de voir l'intérêt du recalage de données. Nous présentons Riedones3D, un jeu de données d'un millier d'acquisitions 3D de monnaies celtiques datant du IIème siècle avant notre ère.

Le chapitre 3 aborde la question du recalage, problème essentiel pour savoir si deux monnaies ont été frappées à partir du même coin. Nous présentons un état de l'art des méthodes basées sur le recalage. Nous proposons une méthode de recalage qui fonctionne sur Riedones3D. Nous comparons également différentes méthodes, dont des méthodes basées sur l'apprentissage profond.

Le chapitre 4 aborde la chaîne de traitement complète qui utilise le recalage. Nous évaluons la chaîne de traitement proposée et nous produisons une analyse des cas extrêmes. Nous testons la méthode de recalage sur les motifs du casque d'Agris (grottes des Perrats, Charente) et du fourreau d'épée de Moscano di Fabriano (Marches, Italie). Nous abordons des pistes de réflexion pour rendre la méthode simple d'emploi pour des utilisateurs non spécialistes, archéologues et autres.

Le chapitre 5 aborde le problème de la généralisation des techniques de recalage et propose un nouvel algorithme pour généraliser le recalage à d'autres types de capteurs. La méthode proposée basée sur une nouvelle architecture et une nouvelle méthode d'apprentissage par transfert non supervisé a des performances supérieures comparées aux autres méthodes à la pointe de l'état de l'art.

Cette thèse a été réalisée au centre de Robotique de Mines Paris de l'université Paris Sciences et Lettres (PSL) en co-tutelle avec le laboratoire d'Archéologie et de Philologie d'Orient et d'Occident (AOROC) de l'ENS Ulm de l'université PSL. Cette thèse a été financée par le programme IRIS Science des données et données de la science de l'université PSL qui vise à promouvoir la collaboration interdisciplinaire entre les différents laboratoires de l'université PSL.

Ces contributions ont donné lieu à des articles publiés dans des conférences internationales :

- Horache, Sofiane, Jean-Emmanuel Deschaud, François Goulette, Katherine Gruel, Thierry Lejars, et Olivier Masson. Riedones3D : A Celtic Coin Dataset for Registration and Fine-Grained Clustering. *Eurographics Workshop on Graphics and Cultural Heritage*, 2021.
- Horache, Sofiane, Jean-Emmanuel Deschaud, et François Goulette. 3D Point Cloud Registration with Multi-Scale Architecture and Unsupervised Transfer Learning. In *2021 International Conference on 3D Vision (3DV)*, 1351-61, 2021.
- Chaton, Thomas, Nicolas Chaulet, Sofiane Horache, et Loic Landrieu. Torch-Points3D : A Modular Multi-Task Framework for Reproducible Deep Learning on 3D Point Clouds. In *2020 International Conference on 3D Vision (3DV)*, 1-10. Fukuoka, Japan : IEEE, 2020.
- Horache, S., F. Goulette, J.-E. Deschaud, T. Lejars, et K. Gruel. Automatic Clustering of celtic coins based on 3D point cloud pattern analysis. In *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, V-2-2020 :973-80. Copernicus GmbH, 2020.

Chapitre 2

Étude de coins monétaires

Sommaire

2.1 Introduction

Les monnaies sont des témoignages des anciennes civilisations et l'étude des coins est cruciale pour la compréhension de l'histoire économique, surtout lorsque peu de traces écrites sont disponibles. C'est le cas des nombreux peuples celtes du II-I siècle avant notre ère. Les monnaies sont un reflet de la réalité politique et économique puisqu'ils sont la marque du pouvoir émetteur. Les monnaies proviennent de contextes archéologiques (habitats, sanctuaires, tombes, etc) ou peuvent être amassées sous forme de dépôts ou de trésors (par exemple, le trésor Le Catillon II de Jersey avec environ 69 000 monnaies celtiques [de Jersey, 2018]). Beaucoup de méthodes peuvent être utilisées pour étudier les monnaies. Nous pouvons citer les outils physico-chimiques pour déterminer la teneur des différents métaux ou alliages [Gruel et al., 2017]. Grâce à des dépôts de matières organiques, il est parfois possible de dater ces objets [Gruel et al., 2003].

Une méthode intéressante pour les numismates est l'étude des coins. D'après Sir George Mac Donald, « l'étude des coins est l'un des instruments les plus utiles que le numismate ait à sa disposition » (de Callataÿ présente un historique de l'étude des liaisons de coins [De Callatay, 2007]). Un coin est un outil permettant d'imprimer une image sur un bout de métal (un flan). On distingue le coin droit, fixe qui est encastré sur une enclume et le coin de revers, mobile, que l'artisan tient à la main. L'artisan frappe avec le coin de revers sur la pièce pour marquer simultanément les motifs sur les deux faces du flan (figure 2.1). En général, la matrice du coin est plus grande que la monnaie elle-même. Par conséquent, l'image du coin n'est présente que partiellement.

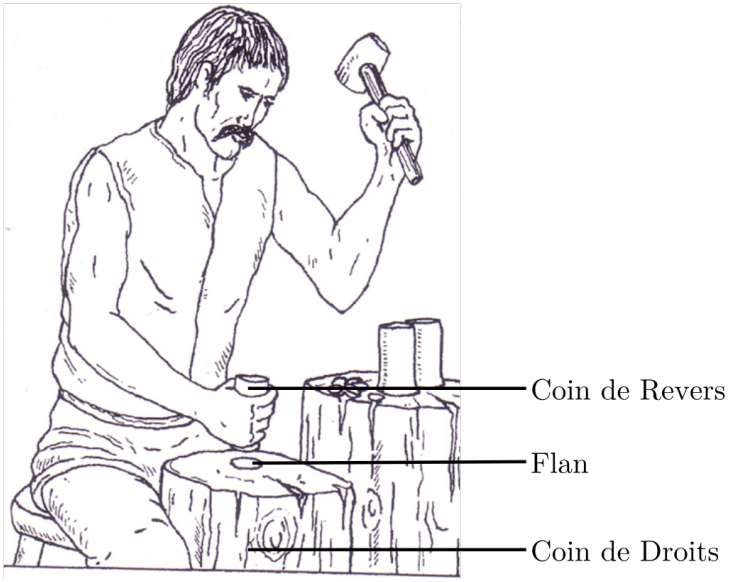


FIGURE 2.1 – Schéma montrant le procédé de fabrication d'une monnaie. Crédit : AOROC CNRS PSL

Le regroupement de monnaies selon leur coin peut grandement aider les spécialistes pour l'établissement d'une chronologie relative entre les coins et donc entre les monnaies. Cette chronologie est utile pour étudier l'évolution des différents coins. Les coins peuvent être regroupés en classe. L'évolution de ces classes permet d'étudier l'évolution de style des artisans. Le coin est aussi la marque du pouvoir émetteur. La figure 2.2 montre une vue d'ensemble du trésor de Liffré (monnaies Riedones) et la figure 2.3 une photographie d'une monnaie de ce trésor.

Si nous regardons l'étude [Tolksdorf et al., 2017] sur les monnaies romaines des régions rhénanes, l'étude de coin a permis d'établir une chronologie plus fine de la circulation des troupes romaines de Varus en l'an 7 de notre ère. De plus, avec ce regroupement, il est possible d'estimer le nombre de coins qui ont été fabriqués. Par conséquent, on peut proposer une estimation du nombre de monnaies ayant circulé dans une région à un moment donné (Esty présente différentes méthodes pour estimer les nombres de monnaies à partir des coins [Esty, 1986]). Cette estimation du nombre de monnaies permet de savoir à quel point, une zone est très monétarisée

FIGURE 2.2 – Le trésor des Riedones. Nous pouvons voir que les monnaies peuvent être très usées par le temps. Crédit : AOROC CNRS PSL

FIGURE 2.3 – Exemple des deux faces d'une monnaie riedones. À gauche, le droit représenté par une tête de profil et à droite le revers avec un cheval et une roue. Crédit : Katherine Gruel

ou non. Une autre application est la reconstitution de coins virtuels. Non seulement la reconstitution de ces matrices virtuelles va aider les spécialistes à mieux classer les différents coins, car ils auront une vision globale du coin, mais cette reconstitution a également des applications dans la communication et la vulgarisation scientifique, les musées peuvent être fortement intéressés par cette reconstitution pour mieux transmettre au public les prouesses des artisans celtes. D'autant plus, que pendant longtemps, d'après JB Colbert de Beaulieu, la numismatique celtique a longtemps été délaissée par rapport à la numismatique grecque ou romaine [Colbert De Beaulieu, 1949]. Pour la reconstitution d'un coin virtuel, nous avons besoin de l'imagerie 3D, afin de regrouper les monnaies selon leurs coins.

L'étude de coins la plus ambitieuse à ce jour est celle de Wolfgang Fischer-Bossert, qui est basée sur 8 000 monnaies (des drachmes d'argent de 510-200 avant notre ère) [Fischer-Bossert, 1999]. Cette étude a nécessité près d'une dizaine d'années [American Numismatic Society, 2020]. Une étude sur un millier de monnaies peut prendre un an, même avec l'aide d'un tube à dessin au microscope (microscope équipé d'un outil de dessin permettant l'identification manuelle de coin), selon Aagard *et al.* [Aagaard and Märcher, 2015]. De plus, l'étude de coin demande un œil aguerri

pour surmonter les difficultés occasionnées par les fissures, l'usure, ou la corrosion. Certaines pièces sont trop usées pour que l'on perçoive la différence avec d'autres monnaies. Par conséquent, des méthodes automatiques de regroupement de monnaies selon leur coin sont devenues essentielles. Mais pour pouvoir implémenter et évaluer des méthodes automatiques, il nous faut des données annotées.

Notre contribution scientifique dans ce chapitre est :

— La proposition d'une base de données annotée et publique avec la collaboration de Katherine Gruel du laboratoire AOROC-ENS [Horache et al., 2021b].

Ce chapitre s'organise en trois parties. Nous abordons les méthodes utilisées en numismatique. Plus spécifiquement, nous reprenons la notion de charactéroscopie, développée par JB Colbert de Beaulieu. Puis nous présentons l'utilisation de l'imagerie 3D pour la reconnaissance de coin par Katherine Gruel [Gruel and Tangue, 2020]. Enfin, nous présentons le jeu de données Riedones3D.

2.2 Charactéroscopie sur des photographies

Avant de se plonger dans les techniques plus récentes, il est important de faire un pas de côté et de comprendre comment un spécialiste détermine que deux monnaies sont du même coin. La charactéroscopie a pour objectif de déterminer les caractéristiques propres à chaque coin. Cette méthode vise à répondre à la question suivante : qu'est-ce qui distingue un coin d'un autre? La charactéroscopie va utiliser des éléments qualitatifs (quel type de motif, quel style de roue, quel aurige, etc.) ainsi que des éléments quantitatifs (mesures d'angles, distances, alignement, etc.). Les numismates détectent des points d'intérêts pour chaque monnaie, font des mesures d'angles et de distances précises autour de ces points caractéristiques [Gruel, 1981].

Ces mesures quantitatives permettent de rendre l'identification de coin plus objective et plus reproductible. JB Colbert de Beaulieu (1949) mesurait avec un compas les distances directement sur les monnaies [Gruel, 1981, Colbert De Beaulieu, 1949]. Si cette méthode structurée présente l'avantage de limiter le nombre d'erreurs, elle est néanmoins très chronophage dans la mesure où elle nécessite de comparer les monnaies par paires. Pour un trésor de x monnaies, il y a $\frac{x(x-1)}{2}$ comparaisons à réaliser. Pour 1000 monnaies, il faut réaliser 499 500 comparaisons ce qui prendrait des années. Une solution pour réduire le nombre de comparaisons est de détecter les accidents de coin. En effet, un coin est un outil fragile, qui peut se casser lors de la frappe et marquer la monnaie (voir figure 2.4). Ces accidents de coins sont visibles sur les pièces et permettent rapidement d'isoler les monnaies issues d'un même coin.

FIGURE 2.4 – Monnaies réalisées à partir d'un même coin endommagé. L'accident de coin est matérialisé par la fissure qui traverse en oblique l'effigie des trois figures.

Un autre inconvénient de la charactéroscopie est que les mesures sont plus difficilement réalisables sur des photos. Si la lumière change de direction, si les photos ne sont pas prises dans

les mêmes conditions, si l'appareil photo n'a pas été correctement calibré, il n'est pas possible de prendre les mesures directement sur celles-ci [Gruel, 1981]. Il faut aussi un œil bien entraîné pour repérer ces points d'intérêts. Fragile, le coin peut également s'émousser, les traits s'épaississent et les frappes ne sont pas toujours bonnes. Il faut donc pouvoir identifier les points caractéristiques même si la monnaie est usée. Nous pouvons le voir, la charactéroscopie pour l'identification de coins présente de nombreuses limites. Le problème est donc double, il faut pouvoir faire des mesures précises sur la monnaie et trouver une méthode pour faciliter la reconnaissance de coin. La solution proposée par les archéologues est d'utiliser l'imagerie 3D [Gruel, 1981]. Nous le verrons, non seulement l'imagerie 3D sert à faciliter la reconnaissance de coin, mais de plus, elle permet aussi une reconstitution d'un coin virtuel afin de mieux caractériser les différentes classes. Ces images 3D de monnaies et de coins peuvent aussi servir à la communication scientifique dans les musées.

2.3 Utilisation de l'imagerie 3D

Cette section présente comment savoir si deux monnaies sont du même coin avec des acquisitions 3D. La première étape est de scanner les faces des monnaies. La seconde étape est d'aligner les motifs par du recalage manuel fait par des logiciels comme CloudCompare [Cloud Compare, 2013]. Après avoir recalé les motifs des monnaies, la troisième étape consiste à vérifier que les motifs se superposent correctement. La figure 2.5 résume ces différentes étapes. Nous allons revenir

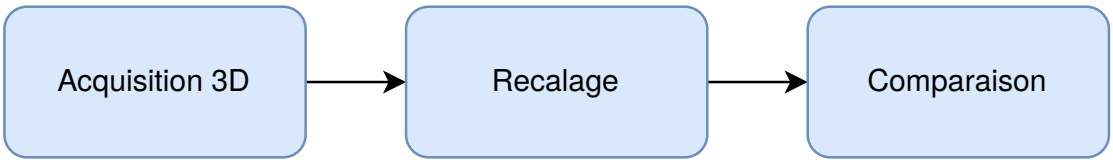


FIGURE 2.5 – Schéma résumant la méthode de classement de coins par recalage manuel.

sur l'acquisition 3D et sur le recalage manuel. Mais tout d'abord, nous allons expliquer les limites de la photographie pour l'analyse des monnaies.

2.3.1 Limites de l'utilisation de photographies

La photographie est une solution relativement peu coûteuse, rapide et très répandue chez les numismates pour la reconnaissance de coins. Or, les monnaies sont des objets spéculaires, l'illumination doit rester constante, une mauvaise illumination nous empêche de bien voir la géométrie du motif surtout si celui-ci est érodé. Il y a aussi le problème de la couleur, des ombres portées, ainsi que des taches de rouilles qui peuvent affecter la reconnaissance des motifs. Certains experts préfèrent donc travailler avec des clichés en noir et blanc. La prise de mesure sur des photos est compliquée. Il faut calibrer l'appareil photo, connaître la distance précise entre l'appareil photo et l'objet, prendre les images dans les mêmes conditions. Les monnaies sont des petits objets (environ 2 cm de diamètre). L'équipe AOROC a essayé d'utiliser des méthodes de reconstructions 3D comme la photogrammétrie ou la RTI sur des monnaies. Cependant, avec ces méthodes il n'a pas été possible d'obtenir une reconstruction fine de la géométrie du motif. Pour pouvoir bien distinguer les détails, il nous faut une précision infra-millimétrique.

2.3.2 Acquisition 3D

L'objectif est de comparer des motifs entre eux et de réaliser des mesures précises. L'acquisition 3D semble indispensable. Contrairement aux photos, les acquisitions 3D de haute définition peuvent mettre en valeur les détails très fins du motif ce qui peut être crucial pour l'étude des

coins. Pour une monnaie de 2 cm, une résolution d'un ordre de grandeur de 10^{-1} mm est nécessaire pour percevoir ces détails. Avec les acquisitions 3D (voir figure 2.6), la géométrie du motif

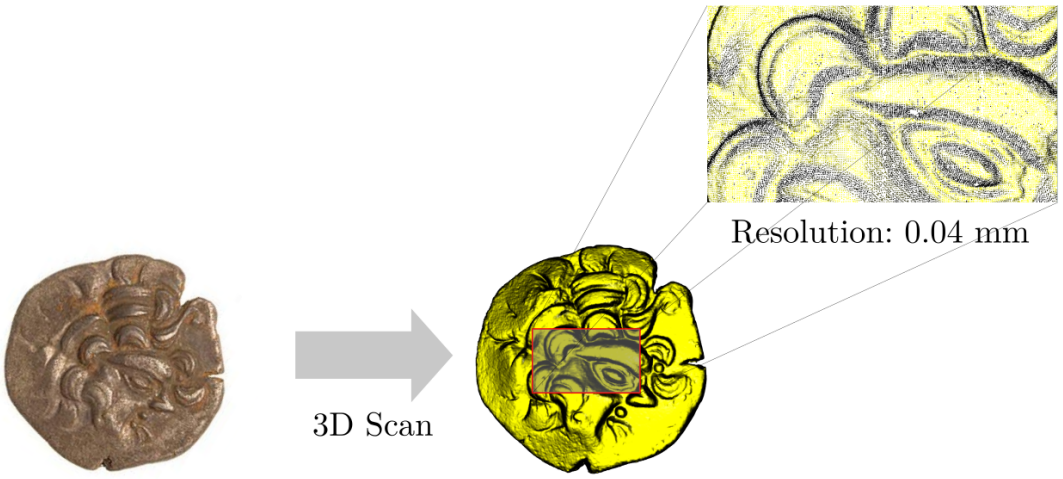


FIGURE 2.6 – Acquisition 3D d'une face d'une monnaie. Le modèle 3D est représenté par un nuage de points 3D

apparaît plus nettement qu'avec de simples clichés photographiques et sans que les taches de corrosions ne viennent entraver la lecture. L'équipe menée par Katherine Gruel a opté pour une acquisition avec un scanner de très haute précision (voir figure 2.7).

Les faces des monnaies ont pu être acquises grâce au scanner Hexagon SmartScan (anciennement AICON). Le scanner peut scanner des objets avec une résolution de 0,04 mm. De plus, il faut scanner les deux faces de la monnaie (le droit et le revers). Pour être plus rapide, les monnaies ont été scannées par 4 (voir figure 2.7). Avec le logiciel Optocat du scanner, il faut réaliser un post-traitement (boucher les trous quand il y en a, améliorer l'alignement des parties, supprimer les données aberrantes). En sortie, nous avons un maillage de la géométrie de chaque face des monnaies (au format STL). Nous pouvons extraire un nuage de points dense avec les sommets du maillage. Il est ainsi possible de faire des mesures précises sur ce modèle 3D et de comparer les motifs. La 3D présente néanmoins quelques inconvénients. Contrairement aux images, les maillages ou les nuages de points 3D sont plus difficiles à manipuler. Un problème classique est par exemple le calcul des plus proches voisins. Pour les images, le calcul des plus proches voisins est immédiat puisque les données sont structurées dans un tableau de pixels. Mais ce problème n'est pas insurmontable comme nous le verrons dans les prochains chapitres. Un autre inconvénient est le temps d'acquisition, il faut 10 min pour scanner et traiter 4 monnaies soit 8 faces. Avant l'acquisition, il y a aussi une étape de calibration de l'appareil cruciale pour une bonne acquisition. Avec ces nuages de points 3D, il est également possible d'appliquer la charactéroscopie ou encore d'opter pour une approche optimisée, mais toujours manuelle, des donnée 3D (cf. supra) [Gruel and Tangue, 2020].

2.3.3 Méthode par recalage manuel

Afin de comparer deux monnaies, l'utilisateur va prendre deux modèles 3D de monnaies et il va sélectionner manuellement plusieurs points d'intérêts pour les deux monnaies. Ensuite, il va aligner les modèles 3D grâce aux points d'intérêt sélectionnés. Il s'agit d'une méthode de recalage par sélection de points. Ensuite, il y a une première vérification à l'œil où l'utilisateur va contrôler

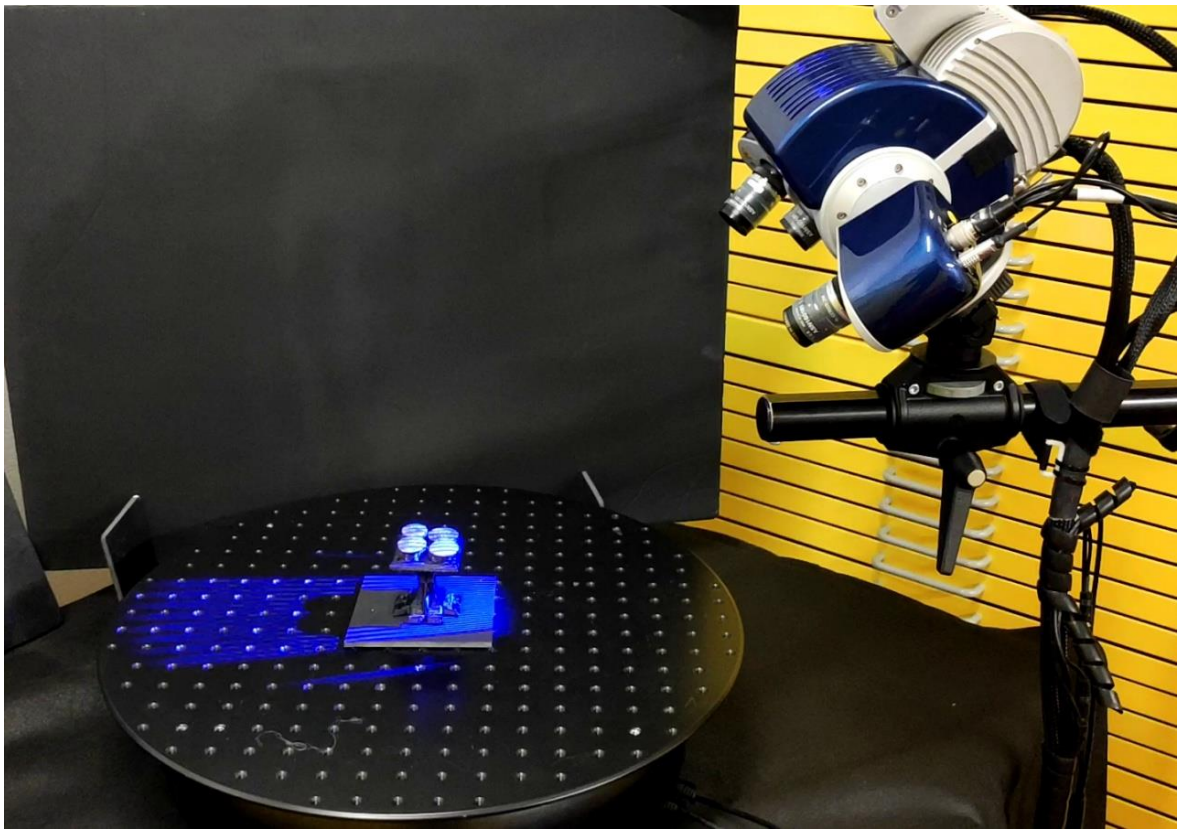


FIGURE 2.7 – Photo du scanner 3D. Le scanner est un Smartscan AICON, qui utilise la lumière bleue structurée pour acquérir la géométrie 3D de l'objet. Crédit : AOROC CNRS PSL

si les motifs se superposent correctement. S'il a un doute, une deuxième vérification va être faite en calculant la distance point à point entre les modèles 3D et en comparant les histogrammes des distances.

Cette méthode de recalage manuel a été appliquée sur 82 monnaies. Il s'agit de la classe des « barbus ». Cette classe se caractérise par une tête de profil représentant un homme barbu tourné à droite de style naturaliste. Pour une illustration du type, on se reportera à la figure 2.8.

Toutes ces actions peuvent se faire avec un logiciel comme Cloud Compare [Cloud Compare, 2013] ou 3D Reshaper [Hexagon, 2021]. La méthode présente plusieurs avantages par rapport à la charactéroscopie. Premièrement, elle est plus rapide. Deuxièmement, une personne non experte en numismatique peut classer une grande partie du lot de monnaies avec cette méthode de recalage. L'expertise est cependant nécessaire si le doute subsiste encore après les deux vérifications. L'avantage aussi est qu'il est possible à partir de cette méthode de procéder à une reconstitution virtuelle du coin . Cependant, même si c'est plus rapide que de la charactéroscopie, cette méthode serait très longue et fastidieuse pour classer un millier de monnaies (il a fallu plusieurs semaines de travail pour classer 82 monnaies [Gruel and Tangue, 2020]).

Cette méthode est manuelle, mais l'avantage est qu'elle nous donne des pistes pour pouvoir automatiser la reconnaissance de coins. L'utilisateur aligne les motifs. Ce problème d'alignement peut se résoudre par des algorithmes de recalage. Ensuite, dès que les motifs sont alignés, il faut réaliser un calcul de similarité pour prendre la décision de classer la paire de monnaies. Il est également possible de calculer automatiquement cette similarité (nous le verrons dans le chapitre 4).

2.4 Jeu de données Riedones3D

Si nous voulons développer des algorithmes de comparaison de coins, il est important de correctement définir le jeu de données sur lequel nous allons travailler. En effet, un jeu de données va

nous servir à évaluer les différents algorithmes de comparaisons de coins. Un jeu de données 3D va permettre aux numismates de faire des études plus approfondies sur des objets acquis avec une grande précision. C'est la raison pour laquelle nous proposons Riedones3D, un corpus de données d'un millier de monnaies celtiques.

Notre objectif est avant tout de pouvoir évaluer des méthodes automatiques pour regrouper des monnaies selon leur coins à partir du jeu de données Riedones3D. L'objectif est également de pouvoir entraîner et évaluer des méthodes basées sur l'apprentissage. Nous présentons deux tâches à évaluer sur le jeu de données proposé. La première consiste à regrouper les monnaies selon leurs coins. La seconde est le recalage rigide entre des paires de monnaies. Le recalage est essentielle pour le regroupement de monnaies selon leurs coins.

Ce jeu de données s'adresse principalement à trois groupes d'utilisateurs :

— Les archéologues (particulièrement les numismates)

— Les chercheurs en vision par ordinateur

— Les musées

Cette base de données sert aux archéologues pour l'étude des monnaies avec des données de grande qualité. Il peut apporter également à la communauté de chercheurs en vision par ordinateur, car les chercheurs de cette communauté peuvent développer de nouveaux algorithmes de reconnaissance de motifs fins. Riedones3D s'adresse également dans une certaine mesure aux musées. En effet, ce jeu de données permet une reconstitution 3D des coins, ce qui pourrait faciliter la transmission des particularités de l'art celtique au grand public. Riedones3D est disponible à cette adresse : `https://npm3d.fr/coins-riedones3d` ou cette addresse : `https://www.chronocarto.eu/spip.php?article84&lang=fr`.

2.4.1 Travaux connexes

Avant de présenter le jeu de données Riedones3D, nous présentons un état de l'art des travaux dédiés à la reconnaissance des monnaies ainsi que des jeux de données pour le recalage et la reconnaissance de motifs.

2.4.1.1 Reconnaissance de monnaies

D'après [Arandjelović and Zachariou, 2020], la plupart des travaux de vision par ordinateur appliqués à la numismatique sont dans des environnements contrôlés et ne peuvent pas être appliqués à des conditions réelles. Nous pensons que la cause vient du manque de jeux de données publics et le code n'est pas toujours en logiciel libre. L'acquisition de données prend du temps et les scanners sont chers. Par conséquent, peu de chercheurs travaillent sur des données 3D [Tolksdorf et al., 2017, Zambanini et al., 2009]. Leurs jeux de données se limitent le plus souvent à quelques dizaines d'occurrences.

Ainsi, peu de méthodes ont été développées sur des données 3D. La plupart des méthodes dans la littérature utilisent des images (nous pouvons citer [Salgado, 2016,Schlag and Arandjelovic, 2017,Zambanini and Kampel, 2013,Zambanini et al., 2013,Anwar et al., 2021]), plus facile à obtenir. Ces bases peuvent contenir jusqu'à 24 000 images de monnaies. Ces photographies viennent en général des sites web de musées ou d'enchères [Anwar et al., 2021, Zambanini and Kampel, 2013, Schlag and Arandjelovic, 2017]. Néanmoins, ces monnaies ne sont pas labélisées par coin parce que comme nous l'avons vu, la labélisation est plus ardue. Lorsque les monnaies sont usées, il est très difficile de voir correctement les détails surtout avec des photos. Pourtant, certains détails sont essentiels pour la reconnaissance de coin. Ainsi, la plupart de ces jeux de données d'image de monnaies ne sont pas adaptés pour la reconnaissance de coin. La plupart de ces jeux de données cherchent à évaluer des méthodes qui peuvent classer les monnaies selon des propriétés de haut niveau (quelle classe, quel style, quel personnage, etc.). Le problème est différent du nôtre où nous cherchons à réaliser un classement encore plus fin (quel coin).

D'après [Arandjelović and Zachariou, 2020], les méthodes automatiques de reconnaissance de coin sont rares, voire inexistantes. La plupart des méthodes sont semi-automatiques [Lista, 2019, Tolksdorf et al., 2017]. Nous pouvons mentionner le projet IBISA [Marchand, 2014] (*Image Based Identification Search for Archaeology*). Le principe est de savoir si deux monnaies sont du même coin en appliquant des techniques de recalage d'image. Mais Marchand [Marchand, 2014] a souligné que les problématiques liées à l'image sont les variations de lumières qui peuvent perturber le recalage d'image (d'où la contribution de [Marchand, 2014] pour une meilleure prise en compte de ces variations). Une modélisation 3D est d'une grande aide pour pallier cette contrainte. Un autre exemple nous est donné par Tolksdorf *et al.* [Tolksdorf et al., 2017] qui ont proposé le classement de coins de monnaies romaines numérisées en 3D en recourant à des méthodes semi-automatiques, mais avec un corpus limité à 37 monnaies seulement. Un autre travail intéressant est celui proposé par Heinecke *et al.* [Heinecke et al., 2021]. Leur approche est basée sur les Processus Gaussiens pour calculer des points d'intérêt et le *microclustering* bayésien [Natarajan et al., 2021] pour regrouper les monnaies de même coin. Leur approche a été testée uniquement sur des monnaies romaines. Il serait intéressant d'évaluer leur approche sur Riedones3D. Enfin, on peut mentionner le projet ClareNet [ClaReNet, 2021], qui vise à classer le trésor Le Catillon II de Jersey de 69 000 monnaies [de Jersey, 2018] grâce à des méthodes basées sur l'apprentissage profond. Deux approches ont été proposées, une approche non supervisée basée sur *Deep Cluster* [Caron et al., 2018] et une approche supervisée basée sur un réseau de neurones convolutionnels et l'algorithme des K-Moyennes. D'après leur présentation, leur approche nécessite un jeu de données d'entraînement volumineux. De plus, cette approche ne semble pas fonctionner pour les coins ayant peu de monnaies représentantes.

2.4.1.2 Jeux de données existants pour la reconnaissance de motifs

Le regroupement de motifs similaires peut-être vu comme un problème de définition de similarité entre deux motifs. L'objectif est de pouvoir avoir une fonction de similarité qui prend en entrée deux motifs et indique s'ils sont similaires ou non malgré les distorsions qu'ils peuvent subir. Une méthode pour définir une métrique est d'utiliser les méthodes d'apprentissage de métrique. Cet apprentissage peut être utilisé pour le regroupement de motifs similaires, mais il peut avoir d'autres applications comme la reconnaissance faciale [Liu et al., 2017], la recherche d'images [Cakir et al., 2019], ou l'apprentissage à partir de peu d'exemples ou *few-shot learning* [Vinyals et al., 2016]. Il existe beaucoup de jeux de données pour ces tâches surtout des jeux de données d'images 2D [Wah et al., 2011, Zheng et al., 2015, Song et al., 2016, Krause et al., 2013]. En 3D, nous pouvons citer les jeux de données de la compétition SHREC, avec SCHREC 2018 et SHREC 2021 pour le patrimoine culturel (cultural heritage). SHREC 2021 [Sipiran et al., 2021] est un jeu de données 3D texturé d'objets anciens (jarres, bols, figurines, etc.). L'objectif est de retrouver les objets qui ont la même forme ou viennent de la même culture. Ce jeu de données est différent du jeu de données Riedones3D. Le jeu de données Riedones3D et SHREC2021 ont la même taille, mais Riedones3D a plus de classes, et les classes peuvent être très déséquilibrées. De plus, Riedones3D est un jeu de données de monnaies avec des motifs très fins où l'objectif est d'identifier les coins des monnaies, ce qui est très différent de faire la différence entre des bols et des jarres. Nous pouvons citer également [Shen et al., 2021]. Shen *et al.* [Shen et al., 2021] propose un jeu de données de filigranes historiques ainsi qu'une méthode pour identifier automatiquement les documents portant le même filigrane. Cette méthode est basée sur l'apprentissage de métrique. Le principe de cette approche est d'identifier les sceaux de manuscrits similaires grâce à des méthodes basées sur l'apprentissage profond. Le problème a quelques similarités avec l'identification de coin, il faut pouvoir identifier si un même sceau a été utilisé dans différents documents. Les différences sont dans la nature des données. Shen *et al.* [Shen et al., 2021], travaillent avec des photos 2D de documents historiques où il faut retrouver le filigrane. Dans notre cas, il faut pouvoir identifier le coin malgré les usures et les cassures des monnaies.

2.4.1.3 Jeux de données existants pour le recalage de nuages de points

Comme nous l'avons déjà souligné, le regroupement d'acquisitions 3D de monnaies selon leur coin peut se décomposer en deux étapes : une première étape pour aligner les motifs, puis une seconde pour évaluer la similarité des motifs. Nous proposons donc d'évaluer le recalage sur Riedones3D parce que c'est une étape importante pour la reconnaissance de coins. Le recalage est présent dans de nombreux domaines, mais nous pouvons diviser les jeux de données pour le recalage en deux types :

— Recalage d'objets

— Recalage de scènes d'intérieur ou d'extérieur

Pour le recalage d'objet, un jeu de données très répandu et populaire est le *Stanford 3D scanning repository* [Curless and Levoy, 1996], avec le célèbre Bunny qui est très utilisée pour les expériences en informatique graphique. Le problème principal de ce jeu de données est qu'il est assez petit pour l'évaluation. Les chercheurs ajoutent du bruit synthétique ainsi que des données aberrantes pour tester leurs algorithmes de recalage, mais ce n'est pas toujours réaliste. Récemment, beaucoup de méthodes basées sur l'apprentissage profond ont émergé pour améliorer le recalage de nuages de points. Les chercheurs utilisent principalement le jeu de données *ModelNet* [Wu et al., 2015], alors qu'il a été conçu pour la classification de nuage de points. *ModelNet* est un jeu de données synthétique d'objets CAO du quotidien (chaise, table, pots, etc.) : il contient 40 classes et 12 311 nuages de points. Parce que c'est un jeu de données synthétique, il n'est pas représentatif de données réelles. La géométrie est trop simple comparée à la finesse des motifs de Riedones3D. De plus, les nuages de points de *ModelNet* sont petits (quelques milliers de points). Les méthodes développées sur *ModelNet* ne vont pas forcément fonctionner sur un jeu de données issu du monde réel comme les acquisitions de scènes d'intérieur (voir la section expériences de [Choy et al., 2020]).

Avec la généralisation de l'usage de scanners 3D, on constate une augmentation significative des jeux de données comme les acquisitions d'intérieur représenté par des images RGBD, par exemple 3DMatch [Zeng et al., 2017] ou TUM [Sturm et al., 2012], ou des acquisitions d'extérieurs représentés par des nuages de points issus de capteurs LiDAR comme ETH [Pomerleau et al., 2012], KITTI [Geiger et al., 2012] ou WHU-TLS [Dong et al., 2017]. Ces jeux de données peuvent être volumineux avec des nuages de points de grande taille. Le recalage est plus difficile sur ces jeux de données. Il sert dans ce cas à la reconstruction 3D ou au *Simultaneous Localisation And Mapping* (SLAM). Mais, ils sont très différents de Riedones3D et les défis ne sont pas les mêmes. Les monnaies sont des structures plates avec des déformations (bordure) et altérations (fissures, corrosion) qui peuvent perturber le recalage.

2.4.2 Présentation de Riedones3D

Les Riedones sont un peuple gaulois installé dans la région rennaise. Peu de choses sont connues des Riedones, à cause du manque de sources écrites. Dans le livre *Commentarii de Bello Gallico* (commentaire de la guerre des Gaules) de Jules César, les Riedones sont brièvement mentionnés. Autour de la ville de Rennes, beaucoup de mobiliers et d'artefacts attribués aux Riedones ont été trouvés, dont des trésors. Les monnaies du jeu de données proposé ont été trouvées dans la forêt de Liffré, à une dizaine de kilomètres au nord-est de Rennes, et sont conservées dans le Musée de Bretagne à Rennes. Ce trésor (voir figure 2.2) est exceptionnel, probablement parce que ces monnaies ont peu circulé ; beaucoup de monnaies de cet ensemble sont de mêmes coins.

Le trésor de Liffré est composé de 1121 monnaies. Il y a deux faces par monnaies. Suite à l'acquisition et aux pré-traitements des monnaies du trésor réalisés par Katherine Gruel et son équipe, on dispose de 1106 acquisitions de droits et 1107 acquisitions de revers (quelques faces n'ont pas pu être acquises). Ensuite, nous avons implémenté un algorithme pour regrouper les monnaies selon leur coin. Nous décrirons cet algorithme dans les prochains chapitres. Après vérification, nous avons constitué un jeu de données avec ces monnaies (nous n'avons pas utilisé toutes les monnaies, certaines classées plus tard n'ont pas pu être intégrées à la base de données).

Le jeu de données Riedones3D est constitué ainsi de 2213 acquisitions 3D de faces (1106 droits et 1107 revers) de monnaies. On dispose de 1024 droits sans barbe et 82 droits avec barbes.

FIGURE 2.8 – En haut, des photos de monnaies. En bas, des acquisitions 3D de ces mêmes monnaies. De gauche à droite, nous pouvons voir un exemple de revers, de droit avec barbe, droit sans barbe.

La figure 2.8 illustre différents droits et revers. Les droits représentent la tête d'un homme tournée vers la droite. Le revers représente un cheval, avec une roue en dessous, et parfois un aurige qui n'est pas toujours visible ou clairement identifiable. Nous faisons une distinction entre les droits barbus et non barbus parce qu'ils sont de classes différentes et que l'on sait qu'ils ne sont pas du même coin. Il n'y a pas besoin d'expertise pour arriver à cette conclusion. Les droits barbus et autres droits ne sont donc pas évalués ensemble. Chaque acquisition correspond à une seule face (droit ou revers) d'une monnaie : une acquisition a un identifiant unique qui est LxD ou LxR. L signifie Liffré, x est un identifiant de la monnaie représenté avec un nombre à quatre chiffres et D veut dire droit et R revers (par exemple L0081D ou L0145R). Les barbus correspondent aux 81 premiers numéros, les suivants désignent les imberbes. Pour chaque acquisition, il existe trois représentations, un maillage au format STL (ce sont les données brutes), un nuage de points colorisé extrait de ce maillage au format PLY avec les normales, les couleurs sont calculées artificiellement à partir des normales. ces dernières sont calculées à partir du maillage. Des rendus 2D des maillages sont disponibles au format PNG.

Les monnaies de ce jeu de données sont issues de 284 coins : 83 droits (75 droits sans barbe, 8 droits avec barbes) et 201 revers. Il est normal que pour les revers, il y ait plus de coins, car ils s'usent plus vite que les coins de droit. Le tableau 2.1 résume le nombre d'acquisitions et le nombre de coins de Riedones3D. La figure 2.9 montre une vue synthétique d'une partie des monnaies Riedones3D regroupées par coin.

	Revers	Barbus	Droits	Total
Nombre d'acquisitions	1107	82	1024	2213
Nombre de coins	201	8	75	284

TABLEAU 2.1 – Nombre d'acquisitions et nombre de coins pour chaque face

Pour le recalage et le regroupement de monnaies selon leur coin, le jeu de données a été divisé en un jeu de données d'entraînement, un jeu de données de validation et un jeu de données test. Les monnaies de mêmes coins ont déjà été toutes recalées entre elles. Les monnaies de coins différents ne sont pas alignées. D'autres méta-données sont disponibles comme le poids, le lieu de découverte, la classe, un numéro d'inventaire et d'autres informations qualitatives (le style de

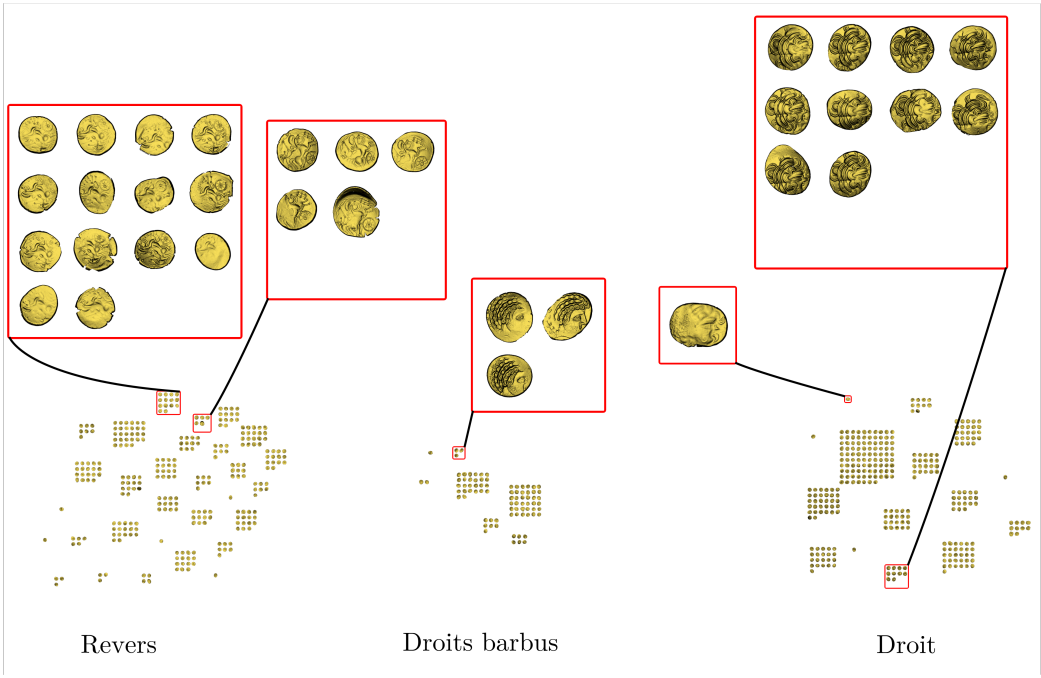


FIGURE 2.9 – Vue synthétique du jeu de données test de Riedones3D pour le regroupement de monnaies par coins (30 coins pour les revers, 7 pour les barbus et 14 pour les droits).

l'aurige ou de la roue, par exemple).

2.5 Conclusion

Dans ce chapitre nous avons présenté les méthodes de travail des numismates pour l'identification de coins monétaires. Nous avons vu une approche manuelle à partir des acquisitions 3D. Même si l'acquisition 3D des monnaies prend du temps, cette approche est plus efficace, car elle nécessite moins d'expertise et est plus rapide pour l'annotation des monnaies. Nous avons présenté également notre contribution, la préparation du jeu de données Riedones3D pour le recalage et le classement automatique de monnaies par coin en collaboration avec Katherine Gruel et son équipe de l'ENS-AOROC [Horache et al., 2021b]. Nous avons vu que ce jeu de données peut être utilisé par les archéologues ou les musées, mais aussi par les chercheurs en vision par ordinateur. Dans les prochains chapitres, nous présentons des algorithmes d'identification de coins testés principalement sur Riedones3D. Mais d'abord, il nous faut aborder un problème fondamental pour l'identification de coins, celui du recalage.

Chapitre 3

Recalage de nuages de points

Sommaire

3.1 Introduction

Le recalage consiste à trouver la transformation qui permet d'aligner un nuage de points 3D par rapport à un autre nuage de points pris comme référence. Plus précisément, l'objectif est de trouver la rotation et la translation 3D qui permettent d'aligner au mieux deux nuages de points 3D. Le recalage est un problème qui a des applications en cartographie et reconstruction 3D de scènes d'intérieures [Choi et al., 2015], véhicule autonome, pour le SLAM [Dellenbach et al., 2021] et en réalité augmentée (voir kinect fusion [Newcombe et al., 2011]). Il présente de la même façon, comme on le verra, un intérêt pour la reconnaissance de motifs 3D en archéologie.

Dans ce chapitre, nous nous attardons sur ce problème qui va nous permettre de reconnaître des motifs sur des modèles 3D d'objets archéologiques comme des pièces de monnaie, ou des ornements de casques. Particulièrement, le recalage se révèle déterminant pour la reconnaissance de coins. Cependant, les algorithmes de recalages sont souvent appliqués à des données synthétiques, des scènes d'intérieurs ou d'extérieurs. Ces jeux de données sont très différents de Riedones3D et à priori, il n'y a aucune garantie que ces algorithmes fonctionnent sur Riedones3D. L'objectif de ce chapitre est justement de voir quels sont les algorithmes qui peuvent être adaptés à ce cas de figure. Nos contributions scientifiques sont :

— Basé sur l'algorithme ICP [Besl and McKay, 1992], nous proposons un algorithme de recalage local et global. Nous expliciterons comment modifier l'algorithme ICP pour qu'il fonctionne sur Riedones3D [Horache et al., 2020]

— Nous établissons une base de référence des dernières méthodes à la pointe de l'état de l'art sur le jeu de données Riedones3D [Horache et al., 2021b]

Ce chapitre est organisé de la manière suivante. Tout d'abord, nous présentons un état de l'art du recalage sur des nuages de points 3D. Nous montrons dans quelles conditions des méthodes sans apprentissage peuvent fonctionner sur Riedones3D. Ensuite, nous comparons les méthodes basées sur l'apprentissage profond à d'autres méthodes classiques.

3.2 État de l'art en recalage

L'objectif de cette section est de présenter les méthodes de recalage manuelles, les méthodes automatiques de recalage locales, puis les méthodes de recalage globales. Nous présentons les méthodes basées sur l'apprentissage profond dans une section à part. La figure 3.1 récapitule les différentes méthodes possibles.

3.2.1 Définition du recalage

Avant de présenter les différentes méthodes de recalage, nous formalisons ce problème. Soit $\mathbf{X}$ et $\mathbf{Y}$ deux nuages de points 3D. Mathématiquement, $\mathbf{X}$ et $\mathbf{Y}$ sont des ensembles de points de $\mathbb{R}^3$, $\mathbf{X} = \{x_1 \in \mathbb{R}^3, x_2 \in \mathbb{R}^3 \dots x_n \in \mathbb{R}^3\}$ et $\mathbf{Y} = \{y_1 \in \mathbb{R}^3, y_2 \in \mathbb{R}^3, \dots y_{n'} \in \mathbb{R}^3\}$. n est la taille du nuage de points $\mathbf{X}$ et n' est la taille du nuage de points $\mathbf{Y}$. Dans le cadre de notre problème, $\mathbf{X}$ et $\mathbf{Y}$ représentent chacun la surface d'une pièce de monnaie. Dans d'autres cadres, ces nuages de points peuvent être issus de capteurs LiDAR [Geiger et al., 2012] ou de caméras RGBD [Zeng et al., 2017]. Dans notre cas, nous avons également accès aux normales de $\mathbf{X}$ (respectivement $\mathbf{Y}$) appelé $\mathrm{N}_\mathbf{X}$ (respectivement $\mathrm{N}_\mathbf{Y}$). $\mathrm{N}_\mathbf{X}$ (respectivement $\mathrm{N}_\mathbf{Y}$) est une fonction de $\mathbb{R}^3$ dans $\mathbb{S}^3$ qui prend en argument un point $x \in \mathbf{X}$ et en sortie sort un vecteur de $\mathbb{S}^3 = \{n | n \in \mathbb{R}^3, ||n||_2 = 1\}$, qui représente la normale de $\mathbf{X}$ au point x. L'objectif du recalage est d'aligner les motifs présents sur les nuages de points $\mathbf{X}$ et $\mathbf{Y}$. Pour aligner ces deux nuage de points, l'algorithme de recalage doit trouver la «bonne» transformation T qui aligne correctement les motifs. Pour ce problème, on suppose que les motifs ne subissent pas de déformation. Par conséquent, une transformation rigide suffit à aligner les motifs, c'est-à-dire qu'on se restreint aux translations et rotations. On représente donc T par une matrice de rotation $\mathrm{R} \in \mathrm{SO3}(\mathbb{R})$ et un vecteur de translation $t \in \mathbb{R}^3$. L'objectif du recalage est de minimiser cette fonction

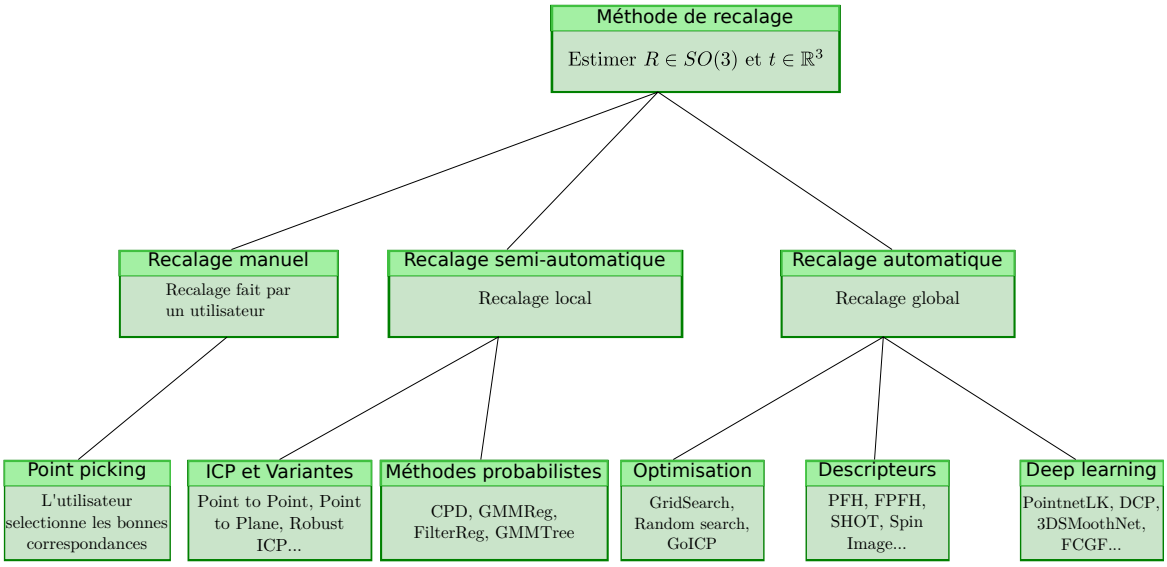


FIGURE 3.1 – Organisation des différentes méthodes de recalage

d'erreur :

$$\underset{\mathrm{R}\in\mathrm{SO}(3),t\in\mathbb{R}^3,\mathbf{M}}{\arg\min} \sum_{(i,j)\in\mathbf{M}} \mathrm{L}(\mathrm{R}x_i + t, y_j) \tag{3.1}$$

M est un ensemble de correspondances de points entre **X** et **Y**. L est une distance. Le plus souvent, L est la distance euclidienne au carré. Mais L peut désigner une norme robuste, ou une distance entre un point et un plan. Optimiser selon R, t, **M** est difficile. En effet, **M** n'est pas dans le domaine continu, contrairement à R ou t. L'enjeu des méthodes de recalage est de pouvoir trouver une bonne correspondance. Et pour cela, les méthodes vont chercher à estimer des correspondances [Salti et al., 2014] valides et trouver une fonction de coût adaptée selon les correspondances.

3.2.2 Recalage par sélection de points

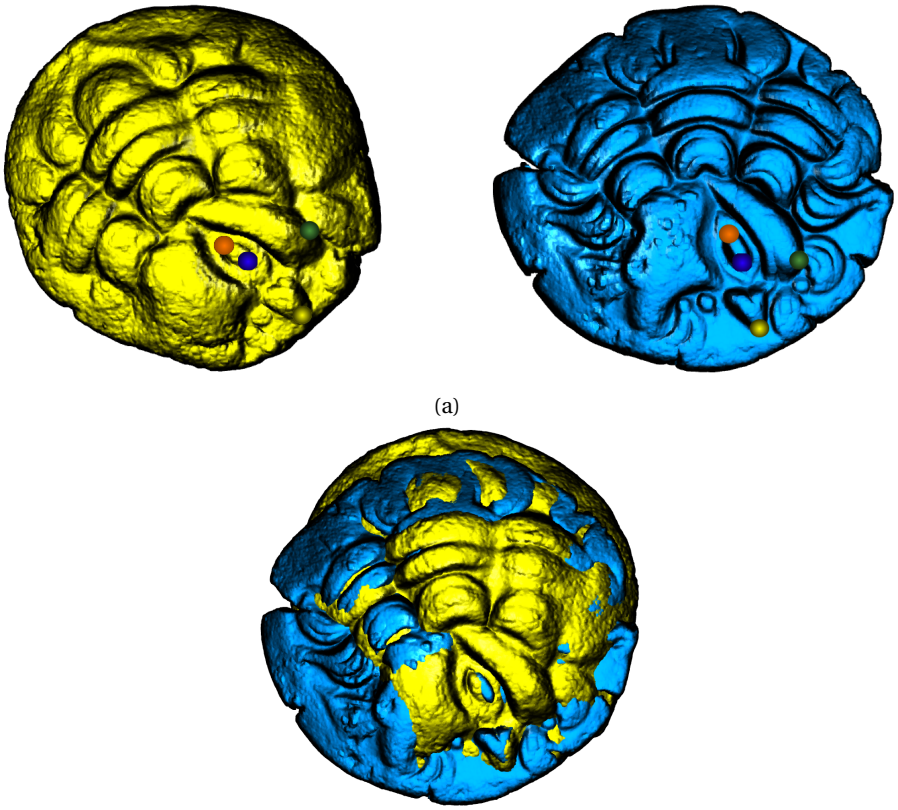
Même si à terme, nous voulons une méthode 100% automatique, les méthodes manuelles sont essentielles. En effet, grâce à ces méthodes, les experts peuvent plus rapidement obtenir la transformation vérité terrain [Gruel and Tangue, 2020]. Le principe des méthodes manuelles est de mettre en correspondance les points à la main (voir équation 3.1). Pour cela, l'utilisateur va sélectionner manuellement des points d'intérêt dans les deux modèles (voir figure 3.2b). Nous avons donc accès à une liste de points **M** qui est une liste d'indices. Le problème de l'équation 3.1 revient à ce problème :

$$\underset{\mathrm{R}\in\mathrm{SO}(3),t\in\mathbb{R}^3}{\arg\min} \sum_{(i,j)\in\mathbf{M}} ||\mathrm{R}x_i + t - y_j||_2^2 \tag{3.2}$$

Ici, la norme L2 ne pose pas de problème, car l'utilisateur a lui-même sélectionné les points, les données ne sont donc pas aberrantes, par principe. Ce problème peut être résolu de manière analytique grâce à l'algorithme de Kabsch [Kabsch, 1978]. L'avantage de cette méthode est que l'on peut obtenir le recalage souhaité avec quasi-certitude. Mais l'inconvénient est double, non seulement, il faut une intervention humaine pour obtenir la transformation, mais en plus, le recalage n'est pas forcément précis (plusieurs recalages manuels vont donner des résultats légèrement différents).

3.2.3 Recalage local

Le problème du recalage est compliqué parce que nous ne connaissons pas les correspondances entre deux nuages de points. Cependant, si nous faisons l'hypothèse que les nuages de



(a)

(b)

FIGURE 3.2 – Étapes pour le recalage manuel. En (a), on sélectionne des points identiques dans les deux images. En (b), on a le résultat du recalage manuel

points sont «presque alignés», alors il existe des algorithmes efficaces pour recaler les nuages de points. Sans doute, l'algorithme le plus célèbre pour du recalage sur des nuages de points est l'algorithme *Iterative Closest Point* [Besl and McKay, 1992].

3.2.4 Iterative Closest Point

3.2.4.1 ICP point à point

Nous allons donc ici présenter l'un des algorithmes les plus populaires pour résoudre ce problème. L'idée principale est de partir de l'équation 3.2. Cette équation présente une solution analytique, mais le problème est qu'ici nous n'avons pas de correspondance de points entre deux modèles 3D comme pour le recalage manuel. La question est de savoir comment trouver des correspondances. L'idée de L'ICP (*Iterative Closest Point*) [Besl and McKay, 1992] est de prendre les plus proches voisins de x_i dans $\mathbf{Y}$, et ensuite de calculer la meilleure transformation (en résolvant 3.2). Si l'on prend $x_i \in \mathbf{X}$, on peut définir le point $\mathrm{NN}_{\mathbf{Y}}(x_i)$ qui correspond dans $\mathbf{Y}$ par :

$$\mathrm{NN}_{\mathbf{Y}}(x_i) = \underset{y \in \mathbf{Y}}{\arg\min} ||x_i - y||^2 \tag{3.3}$$

Pour résumer, l'ICP [Besl and McKay, 1992] consiste en chaque étape :

1. établir une correspondance entre $\mathbf{X}$ et $\mathbf{Y}$, en calculant le plus proche voisin de chaque point x dans $\mathbf{Y}$
2. calculer R et t en utilisant l'algorithme de Kabsch, comme dans le cas du recalage manuel (résoudre l'équation 3.2)

On itère ces deux étapes jusqu'à convergence. ICP [Besl and McKay, 1992] est un algorithme très utilisé [Newcombe et al., 2011, Dellenbach et al., 2021], car simple à implémenter et efficace dans

certains cas, notamment si la solution initiale est suffisamment proche de la solution optimale (il reste à évaluer à quel point il faut que la solution initiale soit proche de la solution optimale). L'autre avantage d'ICP est que c'est un algorithme très modulaire et beaucoup de variantes ont été proposées pour l'adapter à certains cas particuliers et essayer de corriger certains défauts [Bouaziz et al., 2013, Gelfand et al., 2003, Rusinkiewicz and Levoy, 2001, Babin et al., 2019, Pavlov et al., 2018]. Le problème d'ICP est que c'est un algorithme de recalage local. Par conséquent, ICP ne trouve pas la solution optimale si la solution initiale n'est pas suffisamment proche de la solution optimale. De plus, si des correspondances sont fausses ou aberrantes, le minimum optimal d'ICP n'est pas celui qui aligne au mieux les nuages de points. Heureusement, le deuxième problème peut être partiellement corrigé par quelques modifications de l'algorithme.

3.2.4.2 ICP point à plan

Une autre variante incontournable est la version d'ICP point à plan [Rusinkiewicz and Levoy, 2001, Chen and Medioni, 1992, Gelfand et al., 2003]. L'équation 3.2 est une minimisation de la distance euclidienne au carré entre deux points. Mais au lieu de minimiser la distance entre x et y, deux points correspondants, il est possible de minimiser la distance entre x et le plan passant par y de normale $\mathrm{N}(y)$. L'équation à résoudre devient :

$$\underset{\mathrm{R},t}{\arg\min} \sum_{(i,j)\in\mathbf{M}} [(\mathrm{R}x_i + t - y_j)^{\mathrm{T}} \mathrm{N}(y_j)]^2 \tag{3.4}$$

L'intuition derrière cette fonction coût est qu'elle est plus robuste face aux correspondances bruitées. Si la correspondance entre x et y n'est pas bonne (la bonne correspondance est y'), la fonction de coût point à plan va diminuer l'impact surtout si $\mathrm{N}(y)$ et $\mathrm{N}(y')$ sont similaires. L'équation 3.4 n'admet pas de solution analytique comme l'équation 3.2. Il faut donc faire appel à des algorithmes d'optimisation de style Gauss-Newton pour estimer la solution de manière itérative. Mais cette variante converge plus rapidement que la version point à point. De plus, cette variante est très intéressante pour notre problème, car il a été montré que la version point à plan fonctionne très bien sur des structures plates [Rusinkiewicz and Levoy, 2001]. Or, nous sommes intéressés par le recalage de motifs sur des structures plates, comme les monnaies. Il existe une autre version appelée plan à plan, décrite par ICP généralisé [Segal et al., 2009]. ICP généralisé [Segal et al., 2009] propose une formulation générale d'ICP en prenant en compte les matrices de covariance de chaque point correspondant. La formulation d'ICP généralisé permet de déduire la version point à point et point à plan, mais permet aussi d'en déduire une version plan à plan.

3.2.4.3 Variantes d'ICP

ICP est un algorithme modulaire. De nombreuses variantes ont été testées et conçues. Le schéma de la figure 3.3 montre les différents modules d'ICP (inspiré de [Pomerleau et al., 2015]). Pour les variantes d'ICP, il y a trois modules importants que nous pouvons modifier :

1. Sélection de points
2. Mise en correspondance
3. Rejet des données aberrantes

Sélection de points Dans l'ICP classique, on essaye de mettre en correspondance tous les points du nuage **X** avec les points du nuage **Y**. Le problème est que pour les pièces de monnaie, les nuages de points peuvent être volumineux (jusqu'à 100k points). La mise en correspondance peut prendre beaucoup de temps, cette étape de sélection est importante pour réduire le temps de calcul. Une solution est de sélectionner k points aléatoirement dans **X**. La sélection aléatoire de points a pour avantage d'accélérer ICP, mais l'inconvénient est que la résolution perd en stabilité. Gelfand *et al.* [Gelfand et al., 2003] propose un échantillonnage qui permet de rendre plus stable la résolution de l'ICP point à plan. Il est possible aussi de sélectionner les points selon la densité du nuage de point [Pomerleau et al., 2015].

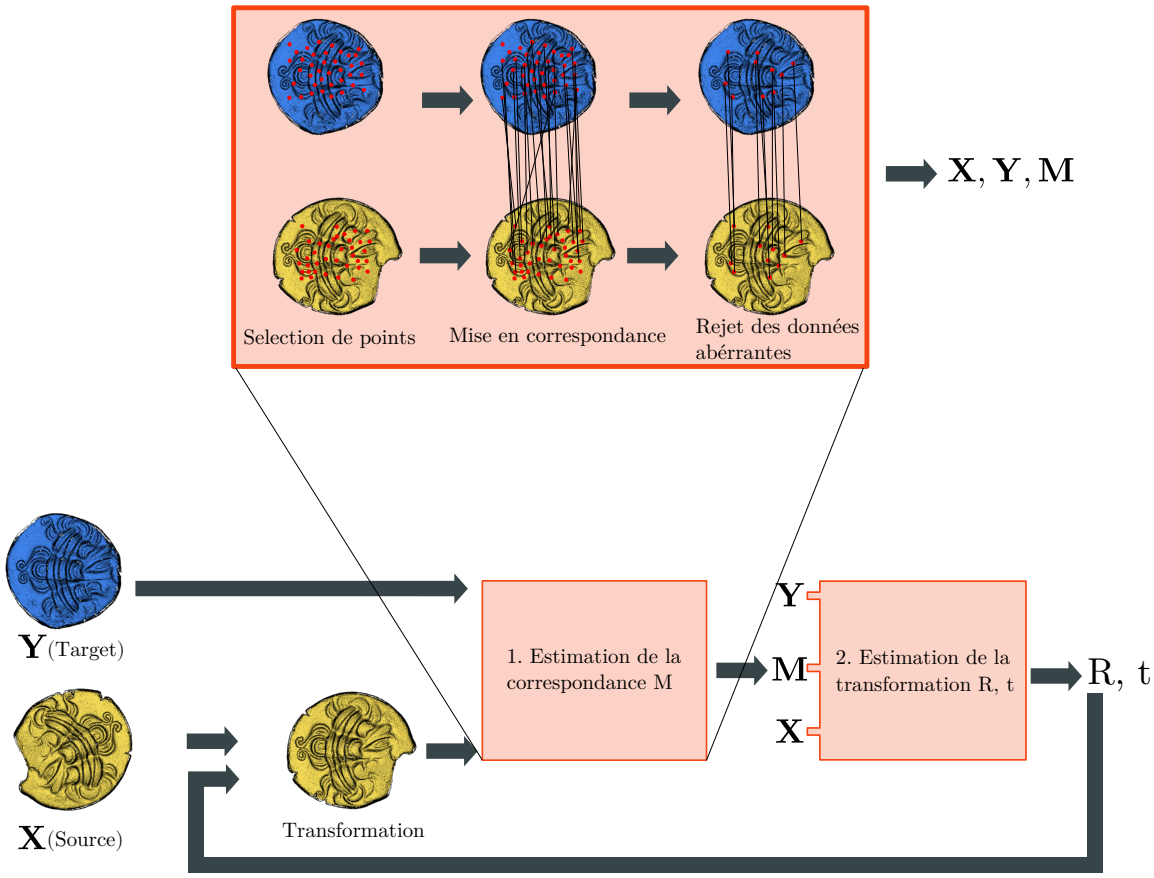


FIGURE 3.3 – Schéma résumant l'algorithme ICP. Chaque module d'ICP peut donner lieu à une variante différente

Mise en correspondance La mise en correspondance entre **X** et **Y** consiste à chercher le plus proche voisin des points $x \in \mathbf{X}$ dans **Y**. Pour des nuages de points, c'est un problème assez compliqué, une implémentation naïve peut prendre beaucoup de temps. Une structure de données comme les KD-Tree [Rusinkiewicz and Levoy, 2001, Pomerleau et al., 2015] peut être utilisée pour avoir un gain de rapidité. Il est également possible de prendre les k plus proches voisins pour chacun des points.

Rejet des données aberrantes Le rejet des données aberrantes peut se faire de plusieurs manières [Bouaziz et al., 2013, Chetverikov et al., 2002, Babin et al., 2019]. Une idée répandue consiste à privilégier une norme robuste. Les normes robustes sont moins sensibles aux données aberrantes, ces normes sont largement utilisées dans les méthodes de recalage.

Accélération de la convergence Une technique pour accélérer la convergence d'ICP est d'utiliser l'accélération d'Anderson [Pavlov et al., 2018]. Dans la plupart des variantes d'ICP proposées, seulement la solution trouvée précédente importe pour estimer la nouvelle solution. L'idée de AA-ICP est de mettre à contribution toutes les solutions précédentes.

3.2.4.4 ICP plus robuste

Pour l'ICP point à point, la distance euclidienne au carré est utilisée. Cette norme n'est pas robuste aux données aberrantes. Or, lors du calcul de correspondances entre deux nuages de points, si le recouvrement entre les nuages de points est faible, ou si il y a des points aberrants, la fonction de coût va y être sensible. Pour limiter l'impact de ces fausses correspondances, de nombreux travaux ont proposé de modifier la fonction de coût et de la rendre plus robuste [Bouaziz et al., 2013, Chetverikov et al., 2002].

$$\sum_{(i,j)\in\mathbf{M}} \rho\left(\frac{\|\mathrm{R}x_i + t - y_j\|}{s}\right) \tag{3.5}$$

Où s est un hyper paramètre d'échelle. ICP point à point utilise $\rho : x \rightarrow x^2$. Sparse ICP [Bouaziz et al., 2013] propose d'utiliser $\rho_p : x \rightarrow x^p$ où $p \in]0,1[$. Babin *et al.* [Babin et al., 2019] proposent d'évaluer différentes fonctions de coût robustes pour ρ comme la fonction carré, la fonction valeur absolue, la fonction de Geman Mc Clure ou bien Trimmed ICP [Chetverikov et al., 2002]. Mais selon la fonction de coût, il faut choisir correctement les hyper paramètres. La conclusion de Babin *et al.* [Babin et al., 2019] est qu'une fonction de coût L1 permet déjà d'être robuste à différentes correspondances aberrantes et il n'y a pas d'hyper paramètre à choisir. Pour les autres fonctions de coût, il faut faire attention au choix des hyper paramètres qui peuvent donner des résultats moins bons que pour la norme L2 selon les fonctions de coût robustes.

3.2.5 Méthodes probabilistes

3.2.5.1 Nuages de points comme densité de probabilité

À cause de l'acquisition, les nuages de points peuvent être bruités. Pour prendre en compte ce bruit, il est possible de modéliser un nuage de points par une densité de probabilité. Tsin et Kanade [Tsin and Kanade, 2004] proposent de voir l'alignement de motifs comme une corrélation par noyau. La corrélation par noyau est définie entre deux points, puis entre deux nuages de points. Le principe est de trouver la transformation qui minimise la corrélation entre deux nuages. La méthode de Tsin et Kanade [Tsin and Kanade, 2004] revient à estimer la densité de probabilité des nuages de points avec une estimation par noyau, puis de trouver la transformation qui va maximiser la corrélation. Le principe de NDT [Biber and Strasser, 2003] est de voxéliser un nuage de point. Pour **X** et **Y**, **Y** va être voxélisé. Chaque point de chaque voxel va servir à estimer la moyenne et l'écart-type. Ensuite, on estime la transformation en minimisant la vraisemblance négative entre les points de **X** et les densités de probabilité. Jian et Vemuri [Jian and Vemuri, 2011] proposent de modéliser **X** et **Y** grâce à des mélanges de gaussiennes. Ensuite, l'estimation de la transformation

se fait par alignement de mélanges de gaussiennes. Jian et Vemuri montrent que l'algorithme ICP, la corrélation par noyau de Tsin et Kanade sont des cas particuliers d'alignements de mélanges de gaussiennes.

3.2.5.2 ICP probabilistes

Si les nuages de points sont bruités et avec des données aberrantes, alors la plupart des correspondances calculées par ICP seront incorrectes. Par conséquent, on perd en stabilité dans la solution. Certaines méthodes ont été proposées pour prendre en compte l'aspect bruité d'ICP. La différence par rapport à ICP consiste à assigner des poids aux correspondances. Ainsi non seulement il faut trouver la bonne transformation, mais il faut aussi optimiser des poids pour chaque correspondance. Soft assign [Tamaki et al., 2010] et Mean ICP [Liu, 2007] proposent de rajouter des termes de régularisations (maximiser la somme des poids ou le produit des poids). L'idée de ces poids est de pouvoir sélectionner les correspondances à garder. La régularisation consiste à pousser les poids à être non nuls et égaux. EM-ICP [Tamaki et al., 2010] est différent. Au lieu de considérer juste des paires de correspondances calculées avec des algorithmes de plus proches voisins, EM-ICP [Tamaki et al., 2010] va considérer tous les points et va assigner des poids à chacune des correspondances (les poids sont calculés avec la fonction *softmax*). EM-ICP [Tamaki et al., 2010] est une version *soft* d'ICP point à point. Ces méthodes peuvent se résumer ainsi :

— Évaluation des poids de chaque correspondance

— Estimation de la rotation et translation

En réalité, il existe un formalisme statistique pour ces deux étapes. Il s'agit de l'algorithme *Expectation Maximization* (EM).

3.2.5.3 *Coherent Point Drift*

Coherent Point Drift ou CPD [Myronenko and Song, 2010] est une formalisation de l'ICP probabiliste avec l'algorithme *expectation maximisation*. Les densités de probabilité pour modéliser les nuages de points sont des mélanges de gaussiennes (GMM) [Myronenko and Song, 2010, Jian and Vemuri, 2011, Evangelidis et al., 2014, Danelljan et al., 2016a, Danelljan et al., 2016b, Gao and Tedrake, 2019]. Dans le cas de CPD [Myronenko and Song, 2010], nous cherchons à recaler un nuage de point avec une densité de probabilité. Cependant, nous pouvons voir qu'il faut aussi estimer la densité de probabilité. Comme pour SoftAssign ou EM-ICP [Tamaki et al., 2010], CPD [Myronenko and Song, 2010] et variantes [Gao and Tedrake, 2019, Gao and Tedrake, 2019, Evangelidis et al., 2014] utilisent toutes les paires de points comme correspondance et pondèrent les correspondances avec des poids. CPD modélise conjointement le problème de recalage et l'estimation de densité de probabilité. CPD utilise l'algorithme *d'Expectation Maximisation* pour estimer les paramètres de la mélange de gaussiennes, ainsi que la transformation. Le principal défaut de CPD [Myronenko and Song, 2010] est que c'est un algorithme très lent pour des nuages de points volumineux. Considérer toutes les correspondances a une complexité de $\mathcal{O}(|X||Y|)$, c'est à dire quadratique. GMM-Tree [Eckart et al., 2018] et FilterReg [Gao and Tedrake, 2019] proposent d'accélérer CPD. GMM-Tree utilise une représentation hiérarchique des mélanges de gaussiennes (GMM) pour modéliser la densité de probabilité. Le nombre de correspondances à traiter diminue ainsi que la complexité. Une implémentation GPU est possible. FilterReg [Gao and Tedrake, 2019] modélise *l'Expectation* comme une opération de filtrage grâce à un noyau gaussien. Le filtrage est réalisé grâce à une représentation avec des treillis en permudroèdres. FilterReg [Gao and Tedrake, 2019] comme GMM-Tree [Eckart et al., 2018] ont une variante point à plan qui améliore les résultats.

3.2.6 Recalage global

Les méthodes de recalage global sont des méthodes qui vont recaler la paire de nuages de points quelle que soit la transformation initiale. Le problème est bien plus difficile que le recalage

local. Il existe plusieurs familles de méthodes. Ces méthodes ne visent pas forcément à remplacer ICP. Elles sont très souvent complémentaires à ICP [Zhou et al., 2016] et elles permettent d'estimer une solution approximative. ICP permet ensuite d'affiner la solution. Nous traiterons les méthodes basées sur l'apprentissage profond dans une autre section.

3.2.6.1 ICP avec différentes initialisations

Nous avons détaillé différentes méthodes de recalage local, une solution pour le recalage global qui en découle est d'utiliser ces méthodes avec plusieurs initialisations. On cherche à minimiser cette énergie :

$$\mathrm{R}^*, t^* = \arg\min_{\mathrm{R},t} \mathrm{E}(\mathrm{R}, t) = \arg\min_{\mathrm{R},t} \sum_{i=1}^{\mathrm{N}} \|\mathrm{R}x_i + t - \mathrm{NN}_{\mathbf{Y}}(\mathrm{R}x_i + t)\|^2 \tag{3.6}$$

Où $\mathrm{NN}_{\mathbf{Y}}(.)$ calcule le plus proche voisin dans $\mathbf{Y}$ et est défini à l'équation 3.3. Ici, nous cherchons à minimiser cette énergie dans un cadre global. ICP peut trouver un minimum de E mais il sera local. Minimiser E dans l'équation 3.6 pose plusieurs difficultés :

— La fonction de coût n'est pas convexe. Il y a plusieurs minima locaux, atteignables grâce à l'algorithme d'ICP

— L'évaluation de cette fonction de coût est coûteux. Il faut environ 30 ms pour évaluer la fonction de coût pour deux nuages de points de 100k points.

— La fonction de coût est sensible aux données aberrantes.

Pour le deuxième point, la fonction de coût est coûteuse à évaluer dans notre cas, car nous avons des nuages de points volumineux (environ, 100k points), et le calcul de voisinage prend du temps. Pour le dernier point, c'est le même défaut que l'algorithme ICP. Nous pouvons voir que le problème d'optimisation a six degrés de liberté (3 pour les translations, trois pour les rotations). Mais dans le cas des pièces de monnaie, nous sommes dans un cas particulier : les pièces de monnaie sont des structures plates. Nous pouvons donc grossièrement aligner les pièces sur le même plan. Ensuite, nous pouvons chercher uniquement les transformations dans le plan. Il n'y a plus que 3 degrés de liberté. Ce qui peut grandement accélérer les calculs.

Recherche par grille Une première idée consiste à discrétiser l'espace de recherche [Horache et al., 2020]. Si nous autorisons six degrés de liberté et si nous testons 10 valeurs par degré de liberté, il y a 10^6 solutions candidates. Ce n'est pas tenable en pratique. Avec un degré de liberté de 3, le nombre de solutions candidates est de 10^3. Un autre problème est qu'une discrétisation grossière va mener à une solution trop approximative. Il faut donc utiliser l'ICP pour converger vers une solution convenable. Mais si pour chaque essai, nous utilisons ICP, l'algorithme risque d'être trop long. Une idée est donc de chercher une solution et si la solution est suffisamment bonne (si la fonctionnelle E définie à l'équation 3.6 est sous un seuil donné), nous pouvons appliquer ICP. La méthode par grille a comme problème d'être très lente. Heureusement, si les motifs sont similaires, la recherche par grille peut trouver une solution avant et donc il n'est pas nécessaire de tester toutes les solutions.

Recherche aléatoire Une autre idée revient à faire de la recherche aléatoire [Horache et al., 2020]. Nous allons choisir des transformations aléatoires au lieu de toutes les tester par grille. L'avantage de cette solution est que pour la recherche aléatoire, nous n'avons pas de pas à fixer, mais il faut fixer le nombre d'essais. De plus, nous pouvons choisir des solutions candidates qui ne sont pas observées par la méthode de recherche par grille si le pas est trop grossier. Mais l'inconvénient est que cette méthode n'est pas déterministe (nous n'avons pas de garantie de trouver la solution optimale) contrairement à la méthode de recherche par grille. La recherche aléatoire a le même problème que pour la recherche par grille : lorsque les pièces ne sont pas du même coin, la méthode est lente.

3.2.6.2 Approches *branch and bound*

Les méthodes décrites ci-dessus sont trop lentes. Il faut trouver une méthode pour exclure certaines solutions. Au lieu de faire une recherche par grille ou une recherche aléatoire, nous pouvons faire une recherche par *octree*. L'idée peut donc être d'avoir une grille adaptative qui a été introduite par GO-ICP [Yang et al., 2016].

L'optimisation peut-être vue comme du *branch and bound*. On divise l'espace des entrées en hypercube. Puis à chaque centre d'hypercube, on estime une borne supérieure de la fonction de coût. Si la borne supérieure est inférieure à l'erreur optimale, on a trouvé une meilleure solution, on actualise la solution avec l'algorithme d'ICP. Sinon, on calcule la borne inférieure et si la borne inférieure est inférieure à la solution optimale, on subdivise l'hypercube en $2^6 = 64$ sous hypercube ($2^3 = 8$ dans le cas des pièces de monnaie). Nous pouvons voir que l'algorithme est efficace si nous arrivons à estimer une bonne borne supérieure et une bonne borne inférieure pour exclure certains espaces. Sinon, cela revient à faire de la recherche dans une grille. La question est de savoir comment calculer ces bornes. Une rotation peut avoir une représentation matricielle, mais peut avoir aussi une représentation appelée axe/angle. Pour subdiviser l'espace des rotations, il vaut mieux passer par une représentation axe/angle. L'espace des rotations est une boule de rayon π et chaque élément $r \in \mathbb{R}^3$ (R est sa représentation matricielle) de la boule représente une rotation $\frac{1}{||r||}r$ est l'axe de la rotation et $||r||$ est l'angle (en radian) compris entre 0 et π. Si nous prenons une rotation r, comprise dans le cube de centre r_0 (R_0 est sa représentation matricielle) et de côté σ_r, et nous prenons une translation comprise dans le cube de centre t_0 et de rayon σ_t, alors, nous pouvons estimer une borne inférieure et supérieure de la fonction de coût E (définie dans l'équation 3.6). On appelle la borne inférieure $E_{inf}(R, t)$ et la borne supérieure $E_{sup}(R, t)$. Pour GO-ICP [Yang et al., 2016], le problème vient justement du choix de la borne. La borne inférieure et supérieure sont difficiles à choisir et dépendent finalement des nuages de points.

GOGMA [Campbell and Petersson, 2016] utilise la formulation de Jian et Vemuri [Jian and Vemuri, 2011] pour modéliser les nuages de points comme des mélanges de gaussiennes. Ensuite, GOGMA utilise la même fonction objectif. L'optimisation est ensuite résolue par alignement de mélanges de gaussiennes comme pour Jian et Vemuri [Jian and Vemuri, 2011] et le *branch and bound* exclut les translations et les rotations candidates comme pour GO-ICP [Yang et al., 2016]. GOGMA [Campbell and Petersson, 2016] s'est révélé être plus rapide que GO-ICP, car les nuages de points peuvent être réduits à des mélanges de gaussiennes contenant moins de points.

Le problème avec GOGMA et GO-ICP est que l'espace de recherche est de dimension 6. Straub *et al.* [Straub et al., 2017] proposent de réduire la dimension de recherche en découplant la recherche de translation et de rotation. Straub *et al.* [Straub et al., 2017] utilisent une distribution de Von Mises Fisher, et proposent de chercher la rotation optimale puis la translation optimale. Liu *et al.* [Liu et al., 2018] font l'inverse. D'abord, ils calculent des mesures invariantes par rotation puis cherchent la translation optimale grâce à ces mesures, et enfin la rotation optimale. Malgré ces accélérations, les méthodes basées *branch and bound* restent très lentes. De plus, le choix des bornes est crucial et dépend beaucoup des données.

3.2.6.3 Approche par mise en correspondances de points d'intérêt

Lors du recalage manuel, l'utilisateur met en correspondance des parties du nuage de points qui sont similaires (dans le cas des monnaies celtiques, il met en correspondance les yeux avec les yeux). Comment mettre en correspondance automatiquement les parties du motif qui sont similaires ? ICP ou certaines méthodes probabilistes [Tamaki et al., 2010] mettent en correspondance le(s) plus proche(s) voisin(s). Mais ces approches sont sensibles à l'initialisation. Cette méthode de correspondance n'est donc pas satisfaisante pour du recalage global.

4PCS 4PCS [Aiger et al., 2008] ne va pas mettre en correspondance des paires de points, mais ce qu'on appelle des points congruents. Dans un nuage de points $\mathbf{X}$, nous prenons 4 paires de points 3D (a, b, c, d) coplanaires (nous allons l'appeler la base coplanaire). Nous pouvons ainsi

calculer deux valeurs $r_1 = \frac{\|e-a\|}{\|b-a\|}$ et $r_2 = \frac{\|e-c\|}{\|d-c\|}$ où e est l'intersection du segment $\overline{ab}$ et $\overline{cd}$. Alors, dans le nuage de points $\mathbf{Y}$, pour chaque paire de points (y_i, y_j) nous pouvons calculer les 4 points congruents $e_{ij}^{(1)}, e_{ij}^{(2)}, e_{ij}^{(3)}, e_{ij}^{(4)}$ avec :

$$e_{ij}^{(1)} = y_i + r_1(y_j - y_i) \tag{3.7}$$

$$e_{ij}^{(2)} = y_j + r_1(y_i - y_j) \tag{3.8}$$

$$e_{ij}^{(3)} = y_i + r_2(y_j - y_i) \tag{3.9}$$

$$e_{ij}^{(4)} = y_j + r_2(y_i - y_j) \tag{3.10}$$

Ici, on ne prend pas toutes les paires de points, mais les paires de points qui ont la même distance que les quadruplets. Ces points congruents nous donnent des quadruplets de points candidats qu'il est possible de mettre en correspondance avec (a, b, c, d). En effet après translation et rotation, les quadruplets de points auront la même intersection. Le problème de 4PCS est le calcul des congruences qui prend beaucoup de temps (complexité en $\mathcal{O}(|\mathbf{Y}|^2)$). Super4PCS [Mellado et al., 2014] propose une complexité plus faible grâce à une indexation des points dans une grille de voxels ce qui permet de chercher les paires de points qui ont la même distance que la base coplanaire plus rapidement. Il est également possible d'accélérer la recherche de points congruents identique ce qui permet d'obtenir une complexité de $\mathcal{O}(|\mathbf{Y}| + k_1 + k_2)$ où k_1 est le nombre de paires dans $\mathbf{Y}$ qui ont la même distance que les segments de la base coplanaire et k_2 est la taille des points congruents.

Descripteurs Une approche assez populaire est une approche par descripteur [Rusu et al., 2009, Salti et al., 2014, Zeng et al., 2017, Choy et al., 2019, Bai et al., 2020, Ao et al., 2020, Poiesi and Poiesi, 2021, Han et al., 2018]. Le principe est de calculer pour certains points des descripteurs, c'est-à-dire des vecteurs de dimension d. Ensuite, on calcule les points correspondants en cherchant les points qui ont le descripteur le plus proche. Ensuite, grâce à la paire de points correspondants, on calcule la transformation. Le schéma de la figure 3.3 montre les similarités avec ICP. Il y a quelques différences. Ce ne sont plus les plus proches voisins en matière de points 3D, mais en matière de descripteurs. La deuxième différence est qu'il n'y a besoin que d'une itération pour ces approches descripteurs contrairement à ICP. La question qu'on peut se poser est :

1. Où calcule-t-on les descripteurs ?
2. Comment sont-ils calculés ?

Où calcule-t-on les descripteurs ? Pour certaines méthodes, le calcul de descripteurs est coûteux à cause de l'extraction de zones locales. Mais comme pour la vision 2D, beaucoup de méthodes ont été proposées pour extraire des points d'intérêts sur des nuages de points. Nous appellerons ces méthodes des détecteurs. Un point d'intérêt est un point saillant du nuage de points, un point comportant une courbure assez forte. Ces points sont intéressants pour compresser un nuage de points ou pour le recalage. Une propriété que devrait avoir un détecteur est la répétabilité, c'est à dire pour certaines transformations (rotations, translations, petits bruits, changements d'échelle, occlusions, légères déformations), les points d'intérêts doivent rester les mêmes. Un détecteur simple et efficace est simplement d'extraire un nombre de points aléatoire. Il est moins répétable que d'autres détecteurs et il faut en sélectionner un certain nombre pour avoir de la répétabilité, mais a l'avantage d'être beaucoup plus rapide que les autres détecteurs. En effet, les détecteurs se basent souvent sur l'extraction de zones locales qui prend du temps. Une version des détecteurs de coins de Harris existe en 3D [Sipiran and Bustos, 2011]. Pour le faire sur des nuages de points 3D, Harris3D [Sipiran and Bustos, 2011] extrait une zone locale, puis régresse un paraboloïde. Les paramètres du paraboloïde vont déterminer si le centre de la zone locale est saillant ou non. La signature de forme intrinsèque (*Intrinsic Shape Signature* ou ISS) proposée par Zhong [Zhong, 2009] consiste à extraire une zone locale autour d'un point p que l'on va nommer $\mathcal{N}(p)$, ensuite,

on calcule la matrice de covariance :

$$C_p = \sum_{x \in \mathcal{N}(p)} (x-p)(x-p)^T \tag{3.11}$$

On diagonalise cette matrice de covariance, et en faisant le test suivant, on peut détecter les points saillants :

$$\frac{\lambda_2}{\lambda_1} < \gamma_{12} \wedge \frac{\lambda_3}{\lambda_2} < \gamma_{23} \tag{3.12}$$

Gelfand *et al.* [Gelfand et al., 2005] proposent de calculer des descripteurs de volume intégraux (*integral volume descriptors*). Il s'agit du volume sous une courbe locale. Ensuite, le principe est de calculer un histogramme. L'histogramme permet de sélectionner les points d'intérêt. Les points dans les compartiments de l'histogramme les moins peuplés sont choisis comme des points d'intérêts.

Comment sont calculés les descripteurs? Un descripteur est un vecteur de dimension d qui représente localement une forme 3D. Voici les propriétés intéressantes pour les descripteurs :

- Invariance par isométrie : pour faire de la correspondance, il faut être invariant par rotation et translation.
- Robustesse au bruit : même si du bruit est ajouté, il faut que les descripteurs restent proches au sens de la distance euclidienne.
- Robustesse à l'occlusion. Les nuages de points peuvent être acquis à des prises de vue différentes et peuvent être incomplets. Malgré les occlusions, les descripteurs doivent rester proches au sens de la distance euclidienne.

Comme pour en image, beaucoup d'approches ont été proposées. Une revue de la littérature est proposée par Xian *et al.* [Han et al., 2018].

Spin Image Johnson *et al.* [Johnson and Hebert, 1999] ont introduit les descripteurs locaux *Spin image* [Johnson and Hebert, 1999]. Le principe est d'extraire une zone locale (un *patch*) autour d'un point p qu'on appelle $\mathbf{P}_p$ puis pour chaque paire de points de calculer deux valeurs $\alpha(p,q) = N(q)\cdot(p-q)$ et $\beta(p,q) = \sqrt{\|p-q\|^2 - \alpha}$. $N(q)$ est la normale au point q. Les *Spin Image* sont des histogrammes 2D où les valeurs α, β sont accumulées dans une grille discrétisée 2D. Par définition, ce descripteur est invariant par rotation, mais il est rapporté que ce descripteur est sensible au bruit [Han et al., 2018].

3D Shape context *3D Shape context* [Körtgen et al., 2003] est une extension des *2D shape contexts* [Belongie et al., 2001]. Le principe est d'avoir une zone locale (*patch*) orientée selon la direction de la normale. Les points sont convertis en coordonnée sphérique et les compartiments de l'histogramme sont formés en divisant également l'azimut et l'élévation. Les dimensions spatiales sont espacées de manière logarithmique. Le descripteur est l'histogramme du nombre de points tombants dans les compartiments. Cet histogramme décrit la forme de la surface, mais dépend beaucoup de l'estimation des normales. Si les normales sont bruitées, le descripteur n'est pas vraiment invariant par rotation. *Unique Shape Context* [Tombari et al., 2010] est une extension de *3D shape context* [Körtgen et al., 2003]. Le principe est de réorienter la zone locale selon un repère calculé localement avec une analyse de composante principale.

Histogramme de caractéristiques de paires de points (PFH et FPFH) Rusu *et al.* [Rusu et al., 2008] ont exploité le fait que les angles entre les paires de points et la normale donnent des descripteurs invariants par rotation. Les histogrammes de caractéristiques de paires de points (*Point Feature Histogram* ou PFH) sont des descripteurs où le principe est de calculer pour chaque paire de points, un repère local (u, v, w). Ensuite, plusieurs valeurs sont calculées comme le cosinus de l'angle entre la normale d'un point de la paire et de la zone locale et certains des axes du repère

local, ou bien la distance entre paires. L'inconvénient de ce descripteur est qu'il est très lent à calculer, car il faut prendre en compte toutes les paires de points dans la zone locale.

SHOT Signature of histogram of Orientation (SHOT) ou la signature des histogrammes des orientations a été introduite par Salti *et al.* [Salti et al., 2014]. L'idée de ce descripteur est de se baser sur des histogrammes sur les angles entre les normales. Comme tous les descripteurs locaux, il faut d'abord extraire un *patch*, $\mathbf{P}_x$ pour chaque point de x où nous voulons calculer un descripteur. Pour obtenir l'invariance par rotation du descripteur, nous calculons d'abord le système de coordonnée locale (*Local Reference Frame* ou LRF). Ensuite, on oriente le *patch* selon le LRF et nous pouvons ensuite calculer le descripteur. Pour cela, on définit une grille sphérique, qui va partitionner l'espace selon l'axe radial, azimutal et l'élévation. Pour chaque axe, nous allons accumuler les points dans le même secteur pour avoir un histogramme. SHOT juxtapose ces histogrammes pour obtenir le descripteur final. Il existe une version de SHOT appelée B-SHOT [Prakhya et al., 2015] où le descripteur n'est composé que de 0 et de 1. C'est l'histogramme SHOT binarisée selon des conditions définies par Prakhya *et al.*. L'avantage d'un descripteur binaire est que la mise en correspondance peut être très rapide (la mise en correspondance peut se faire avec une distance de Hamming).

3.2.6.4 Estimateur robuste

Cette partie est importante et à associer avec la sous-section sur le recalage basé descripteur. Pendant la mise en correspondance des descripteurs, il est très probable qu'il y ait des correspondances aberrantes (appelées *outliers* en anglais) [Rusu et al., 2009, Gojcic et al., 2019]. Si nous appliquons la méthode de Kabsch naïvement comme pour ICP [Besl and McKay, 1992], la transformation va être perturbée par les correspondances aberrantes. Nous avons besoin d'un estimateur robuste qui calcule la bonne transformation, même si la majorité des correspondances est aberrante (*outliers*).

RANSAC *Random Sampling Consensus* (RANSAC) [Fischler and Bolles, 1981] est un estimateur robuste très populaire en vision par ordinateur [Szeliski, 2010], car c'est une méthode facile à implémenter et très efficace. Le principe est de prendre aléatoirement des correspondances, calculer une transformation candidate, puis compter le nombre *d'inliers*. La transformation finale est la transformation avec le plus *d'inliers*. Même si RANSAC [Fischler and Bolles, 1981] est très efficace avec peu d'*inliers*, RANSAC souffre de quelques défauts. Le choix du seuil pour séparer les *inliers* des *outliers* peut être compliqué dans certains cas. Dans notre cas, on peut prendre un seuil plus élevé, car même si le recalage n'est pas très précis, il est toujours possible de faire un ICP pour corriger le recalage (on se ramène à un recalage local). Le deuxième défaut de RANSAC est que c'est un algorithme non déterministe (nous n'avons que peu de garanties de trouver la solution optimale de manière certaine). Il faut que le nombre d'itérations de RANSAC soit grand pour avoir une faible probabilité d'erreur. Par conséquent, c'est un algorithme assez lent, surtout si le nombre de correspondances correct est faible. Il existe de nombreuses variantes de RANSAC. C'est encore un domaine très actif de recherche avec PROSAC [Chum and Matas, 2005], MAGSAC [Barath et al., 2019], SDRSAC [Le et al., 2019] ou GRAPH-CUT RANSAC [Barath and Matas, 2018]. L'idée des variantes vise à accélérer la recherche de données valides en faisant différentes hypothèses (par exemple, si une correspondance est bonne alors, il y a de bonnes chances que les correspondances proches soient bonnes aussi).

Recalage global rapide (FGR) Une solution plus rapide consiste à poser le problème comme un problème d'optimisation robuste. C'est le principe de *Fast Global Registration* (FGR) proposée par Zhou *et al.* [Zhou et al., 2016]. Nous avions vu que pour le recalage manuel, nous résolvons l'équation 3.2, qui utilise une norme L2. Pour du recalage manuel, il peut y avoir une erreur de rotation ou de translation. Cependant, il n'y a pas d'*outliers* puisque c'est l'utilisateur qui fixe les correspondances. Pour les approches basées sur les descripteurs, les correspondances aberrantes peuvent

même être majoritaire [Zhou et al., 2016]. Le principe de FGR est de remplacer dans l'équation 3.2 la norme L2 par une norme robuste :

$$\underset{\mathrm{R},t}{\arg\min} \sum_{(i,j)\in\mathbf{M}} \rho(\|\mathrm{R}x_i + t - y_j\|) \tag{3.13}$$

Dans l'article original, la fonction robuste ρ est la suivante :

$$\rho : x \rightarrow \frac{\mu x^2}{\mu + x^2} \tag{3.14}$$

Il existe d'autres possibilités comme la quadratique tronquée [Yang et al., 2020]. L'idée de la fonction robuste est de pénaliser les valeurs élevées. Il n'existe pas de solution analytique à l'équation 3.13. De plus, la fonction de coût n'est pas convexe. Il y a des minima locaux. Mais nous pouvons remarquer que plus μ est petit, plus ρ pénalise les valeurs élevées. Pour un μ élevé, la fonction de coût est localement convexe sur une grande région. Nous pouvons résoudre le problème pour un μ élevé, puis baisser la valeur de μ petit à petit. C'est ce qu'on appelle la non-convexité graduelle (Graduated Non Convexity [Yang et al., 2020, Black and Rangarajan, 1996]). FGR ressemble à l'approche de Babin *et al.* [Babin et al., 2019] pour rendre ICP robuste aux données aberrantes. La différence réside dans l'utilisation de la non-convexité graduelle pour FGR [Zhou et al., 2016].

TEASER++ TEASER++ [Yang et al., 2020] améliore FGR en plusieurs points. TEASER++ va découpler l'estimation de la translation et de la rotation, en estimant des mesures invariantes par translation appelée *Translation Invariant Measurement* (TIM). La translation et la rotation peuvent être estimées par des méthodes différentes. TEASER++ estime la translation avec un algorithme de vote adaptatif et la rotation avec un algorithme de Graduation non convexe.

Tout d'abord, on estime les mesures invariantes par translation (*Translation invariant Measurement* ou TIM). Les TIM sont simplement définis comme :

$$x_{kl} = x_k - x_l \tag{3.15}$$

Ensuite, on filtre les TIM associés aux correspondances aberrantes. Ces mesures vont permettre de découpler l'estimation de la translation et de la rotation. Ensuite, on va optimiser uniquement la rotation R grâce à de la Convexité non graduelle (*Graduated Non-Convexity*) à partir des TIM, comme pour FGR [Zhou et al., 2016]. Ensuite, nous allons estimer la translation avec un vote adaptatif. Chaque coordonnée de la translation sera estimée indépendamment avec cet algorithme. Le vote adaptatif cherche le minimum de cet ensemble :

$$\underset{t_k\in\mathbb{R}}{\arg\min} \sum_{i,j\in\mathbf{M}} f_k(t_k) = \underset{t_k\in\mathbb{R}}{\arg\min} \sum_{i,j\in\mathbf{M}} \min\left(\frac{(t_k - (\mathrm{R}x_i - y_j)_k)^2}{\delta^2}, \bar{c}\right) \tag{3.16}$$

Où t_k est la coordonnée k de la translation. Chaque coordonnée de la translation va être estimée séparément avec l'algorithme du vote adaptatif. L'idée est de :

- Constituer des intervalles pour chaque correspondance qui seront des votants.
- Constituer des translations candidates. Si la translation candidate est dans un intervalle d'une correspondance, cela veut dire que la correspondance «vote» pour la translation candidate.
- Les correspondances qui ont voté pour le candidat gagnant estiment la translation finale grâce à une régression linéaire.

Choix de l'estimateur robuste Le choix de l'estimateur robuste dépend beaucoup de notre situation. Si nous voulons un algorithme qui trouve rapidement la solution, il vaut mieux privilégier FGR [Zhou et al., 2016]. Si nous avons beaucoup d'*outliers*, des algorithmes comme RANSAC [Fischler and Bolles, 1981] ou TEASER++ [Yang et al., 2020] sont plus adaptés. Dans le cas des pièces de monnaie, nous n'avons pas de contrainte temps réel, la robustesse aux *outliers* est plus importante. Nous allons donc privilégier RANSAC et TEASER++.

3.2.7 Recalage basé sur l'apprentissage profond

Comme nous avons pu le voir, beaucoup de méthodes basées descripteurs ont été proposées. Cependant, ces méthodes reposent sur l'extraction d'une zone locale. Et extraire une zone locale peut être très lent pour des nuages de points volumineux. De plus, ces méthodes sont souvent inefficaces sur des données réelles [Zeng et al., 2017, Choy et al., 2020]. C'est pour cela que les méthodes modernes se tournent vers l'apprentissage profond.

Nous avons vu plusieurs défauts des méthodes présentées précédemment. Dans beaucoup de cas, les méthodes de recalage global sans apprentissage sont lentes et échouent dans des cas difficiles, quand le recouvrement entre les nuages de point est faible [Huang et al., 2021], ou quand les nuages de points sont différents, à cause du bruit, des données aberrantes ou d'un changement de point de vue [Zeng et al., 2017]. Pour ces cas difficiles, l'apprentissage profond s'est rapidement imposé comme une approche offrant des résultats satisfaisants sur des données réelles [Zeng et al., 2017, Choy et al., 2019, Bai et al., 2020]. Cependant, l'apprentissage profond a beaucoup d'inconvénients. Premièrement, la plupart des architectures ne fonctionnent que sur des nuages de points peu volumineux. De plus, les méthodes basées sur l'apprentissage profond dépendent des données d'entraînements, donc ces méthodes sont moins performantes sur d'autres types de données [Poiesi and Poiesi, 2021, Ao et al., 2020, Gojcic et al., 2019].

3.2.7.1 Qu'est-ce que l'apprentissage profond sur des nuages de points ?

L'apprentissage profond (*deep learning*) est une famille de méthodes dans la branche de l'apprentissage automatique (*machine learning*) où le principe est de calculer des représentations intermédiaires grâce à des réseaux de neurones artificiels pour résoudre plus facilement certaines tâches. Les méthodes d'apprentissage automatique ont la particularité de devoir d'abord être entraînés sur un jeu de données d'entraînement. Un jeu de données d'entraînement est constitué d'un corpus labellisé. Les labels dépendent de la tâche que l'ont souhaite résoudre (pour le recalage, les labels sont la transformation entre deux nuages de points). L'entraînement sert à choisir le modèle qui fait le moins d'erreurs, l'erreur est mesurée grâce à une fonction de coût (l'erreur et la fonction de coût sont à définir selon la tâche). Ces modèles sont ensuite évalués et comparés sur des jeux de données test (la méthode d'apprentissage automatique doit deviner le bon label sur ce jeu de données test). Le jeu de données test sert principalement à mesurer la capacité du modèle à généraliser sur des données inconnus. Un modèle généralise lorsqu'il fonctionne sur des données très différentes des données d'entraînement. Par exemple, pour Riedones3D, si le jeu de données d'entraînement n'est composé que des droits, un modèle qui généralise bien est un modèle qui a une erreur faible sur les revers, même s'il n'a été entraîné que sur les droits.

Avant l'apprentissage profond, les travaux se basaient principalement sur des descripteurs calculés à la main [Salti et al., 2014, Rusu et al., 2009] pour avoir des représentations intermédiaires ce qui leur permettait de résoudre certains problèmes (nous avons vu le cas du recalage, mais les descripteurs sont aussi utilisés en segmentation sémantique [Weinmann et al., 2015]). L'apprentissage profond va calculer automatiquement les représentations intermédiaires et va les utiliser pour résoudre les tâches que l'on souhaite. En nuage de points, l'apprentissage profond est appliqué en classification d'objets [Qi et al., 2017, Thomas et al., 2019], en détection d'objet [Qi et al., 2019], en segmentation sémantique de scènes d'intérieur ou d'extérieur [Choy et al., 2019, Tang et al., 2020, Thomas et al., 2019], en reconstruction de scènes [Dai et al., 2020], et particulièrement en recalage [Aoki et al., 2019, Lu et al., 2019, Wang and Solomon, 2019a, Choy et al., 2019, Zeng et al., 2017].

Contrairement à l'apprentissage profond sur des images, l'apprentissage profond sur des nuages de points 3D est difficile, car les nuages de points sont des données non structurées. En image, une convolution est rapide et peu coûteuse en mémoire et en temps, car les pixels sont rangés dans un tableau, la recherche du voisinage est plus facile et rapide. Pour les nuages de points, la recherche de voisinage est longue et coûteuse.

Plusieurs méthodes ont été suggérées comme les réseaux de neurones convolutionnels avec

vues multiples (Multi-View CNN) [Su et al., 2015], ou bien les réseaux de neurones convolutionnels 3D [Roynard et al., 2018, Zhirong Wu et al., 2015, Zeng et al., 2017] mais ces méthodes demandent une grande mémoire même pour de petits nuages de points. Une révolution pour l'apprentissage profond sur des nuages de points 3D a été l'apparition de PointNet [Qi et al., 2017] et PointNet++ [Qi et al., 2017] qui ont permis à de nombreuses applications de voir le jour [Deng et al., 2018, Zhao et al., 2019, Poiesi and Poiesi, 2021].

Le principe de PointNet [Qi et al., 2017] est d'utiliser un Perceptron Multi Couche (*Multilayer Perceptron* ou MLP) sur des vecteurs caractéristiques d'entrées associés à chaque point du nuage de points (il peut s'agir des coordonnées xyz, des normales des points ou de la couleur des points par exemple). Ensuite, un *pooling* global est réalisé pour obtenir un encodage du nuage de points qui peut être utilisé pour de la classification ou de la segmentation. Il a été démontré que PointNet est un approximateur universelle [Qi et al., 2017]. Néanmoins, l'inconvénient de PointNet est qu'il peut difficilement capturer les structures locales comme le ferait une convolution par exemple. PointNet++ [Qi et al., 2017] est une extension de PointNet. Pour PointNet++, il y a une étape d'échantillonnage (dans le papier original, l'algorithme d'échantillonnage est de l'échantillonnage du point le plus loin ou *Farthest Point Sampling*) pour chaque point p échantillonné, on extrait une zone locale par recherche par rayon. Ensuite, PointNet est effectué sur cette zone locale, le vecteur caractéristique résultant est associé à p.

RandLaNet [Hu et al., 2020] est une variante de PointNet++, cependant l'échantillonnage est aléatoire, l'extraction des zones locales se fait avec les k plus proches voisins et le *pooling* se fait grâce à un mécanisme d'attention. Récemment, beaucoup de travaux ont essayé de généraliser la convolution sur des données parcimonieuses (*sparses*) comme les nuages de points.

Kernel Point Convolution (KPConv) [Thomas et al., 2019] propose une convolution sur des nuages de points. Nous prenons un nuage de points $\mathbf{X} = \{x_1 \in \mathbb{R}^3, \dots x_n \in \mathbb{R}^3\}$ avec des vecteurs caractéristiques $\mathbf{F} = \{f_1 \in \mathbb{R}^{d_{in}}, \dots f_n \in \mathbb{R}^{d_{in}}\}$ associé à $\mathbf{X}$. Nous nommons g le noyau de convolution. La convolution est définie comme :

$$(\mathbf{F} * g)(x) = \sum_{x_i \in \mathcal{N}_x} g(x - x_i) f_i \tag{3.17}$$

$$g(x - x_i) = \sum_{k=i}^{\mathrm{K}} h(x - x_i, \tilde{x_k}) \mathrm{W}_k \tag{3.18}$$

$(\mathbf{F} * g)(x)$ correspond aux vecteurs caractéristiques de sorties associées au point x. h est une fonction de corrélation symétrique (h peut être la corrélation gaussienne). Les $\tilde{x_k}$ sont les points du noyau g fixé à l'avance, mais peuvent être appris. Les W_k sont les poids associés à chaque point du noyau $\tilde{x_k}$ qui correspondent aux paramètres de la *Kernel Point Convolution* (KPConv). $\mathcal{N}_x$ est l'ensemble des points de la zone locale. Avec la KPConv, il est possible de créer des architectures d'apprentissages profonds pour la classification, la segmentation [Thomas et al., 2019] ou bien le recalage [Bai et al., 2020]. *Minkowski Engine* [Choy et al., 2019] et Tang *et al.* [Tang et al., 2020] proposent une convolution 3D sur une représentation voxélique parcimonieuse. Contrairement à [Zhirong Wu et al., 2015] qui représente une image comme une grille d'occupation des voxels, une représentation voxélique parcimonieuse signifie que l'image 3D est représentée par un ensemble de voxels. Les voxels nuls ne sont pas stockés. Pour réaliser une recherche des pixels voisins, *Minkowski Engine* [Choy et al., 2019] et Tang *et al.* [Tang et al., 2020] utilisent une table de hachage implémenté sur GPU. De plus, la convolution est aussi parcimonieuse, dans la mesure où elle n'est réalisée que sur les voxels non vides. Une convolution 3D dense [Zhirong Wu et al., 2015] réalise la convolution sur tous les voxels du tableau. Pour l'apprentissage profond sur des nuages de points on se reportera à l'étude très complète de Guo *et al.* [Guo et al., 2020] parue en 2020.

3.2.7.2 Apprentissage profond pour le recalage

Beaucoup de travaux appliquant l'apprentissage profond pour le recalage [Aoki et al., 2019, Lu et al., 2019, Bai et al., 2020, Wang and Solomon, 2019a] ont montré des résultats remarquables sur

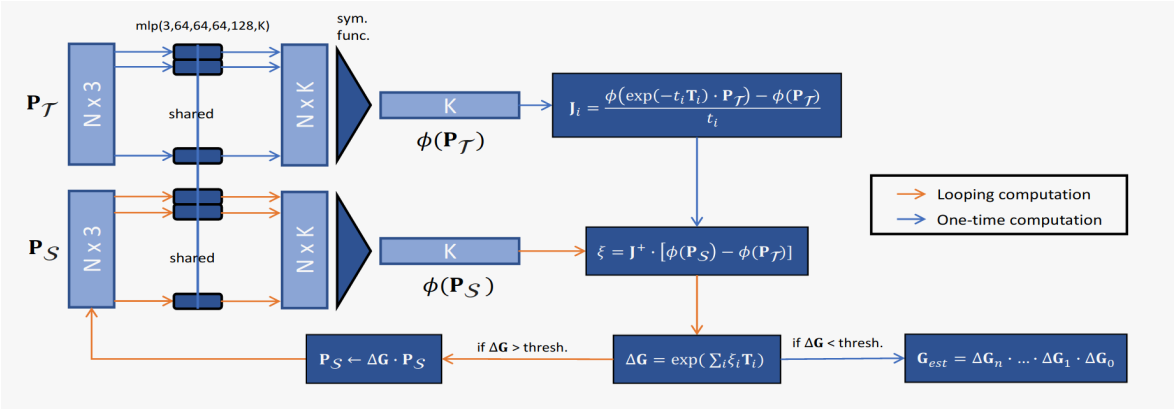


FIGURE 3.4 – Schéma de PointNetLK. PointNet calcule un descripteur global pour chaque nuage de points. Après avoir estimé la jacobienne, le principe est de calculer de manière itérative la transformation qui minimise la distance entre les descripteurs grâce à la jacobienne. Source : Aoki *et al.* [Aoki et al., 2019]

des jeux de données synthétiques comme ModelNet [Wu et al., 2015] ou bien des données réelles comme 3DMatch [Zeng et al., 2017] ou KITTI Odométrie [Geiger et al., 2012]. Nous observons trois tendances pour appliquer l'apprentissage profond pour le recalage :

— Les méthodes dites de bout en bout [Aoki et al., 2019, Yew and Lee, 2020, Wang and Solomon, 2019b, Wang and Solomon, 2019a]. L'objectif est de construire un système entièrement différentiable qui va directement calculer une translation et une rotation (comme PointNetLK [Aoki et al., 2019] et Deep Closest Point [Wu et al., 2015]).

— Les méthodes basées sur des descripteurs profonds locaux qui extraient une zone locale [Gojcic et al., 2019, Zeng et al., 2017, Deng et al., 2018, Ao et al., 2020], puis calculent un descripteur pour cette zone locale avec un modèle basé sur l'apprentissage profond (comme 3DSmoothNet [Gojcic et al., 2019])

— Les méthodes basées sur l'architecture U-Net [Choy et al., 2019, Bai et al., 2020, Huang et al., 2021] : les descripteurs profonds sont calculés simultanément grâce à une architecture semblable à un U-Net (FCGF [Choy et al., 2019])

3.2.7.3 Méthodes dites de bout en bout

Les méthodes sont dites de bout en bout lorsque la méthode prend en entrée deux nuages de points et en sortie estime directement la transformation. Ces méthodes sont complètement différentiables. Beaucoup de méthodes dites de bout en bout ont été proposées pour traiter le problème de recalage. PointNetLK [Aoki et al., 2019] calcule un descripteur global pour les deux nuages de points à recaler grâce à un PointNet [Qi et al., 2017]. PointNetLK [Aoki et al., 2019] cherche la transformation qui minimise la distance euclidienne entre les deux descripteurs globaux. Avec le jacobien (dérivé d'un descripteur global par rapport aux coordonnées de la transformation), il est possible de calculer de manière itérative, une transformation pour minimiser la distance euclidienne entre les deux descripteurs globaux (voir Figure 3.4).

Deep Closest Point [Wang and Solomon, 2019a] utilise un réseau convolutionnel sur graphe [Wang et al., 2019] (PointNet peut aussi être utilisé) pour calculer des descripteurs pour chaque point puis utilise un *Transformer* [Vaswani et al., 2017] afin de mettre en correspondance les descripteurs. Ensuite, la transformation est calculée avec une décomposition en valeurs singulières comme pour l'algorithme de Kabsch. Tout est différentiable et peut être entraîné de bout en bout (voir Figure 3.5).

RPMNet [Yew and Lee, 2020] et PRNet [Wang and Solomon, 2019b] ont le même principe que DCP [Wang and Solomon, 2019a], c'est-à-dire un module qui calcule des descripteurs par point, puis une partie mise en correspondance de points d'intérêt, puis une partie calcul de la transformation. RPMNet [Yew and Lee, 2020] estime une matrice de correspondance stochastique à partir

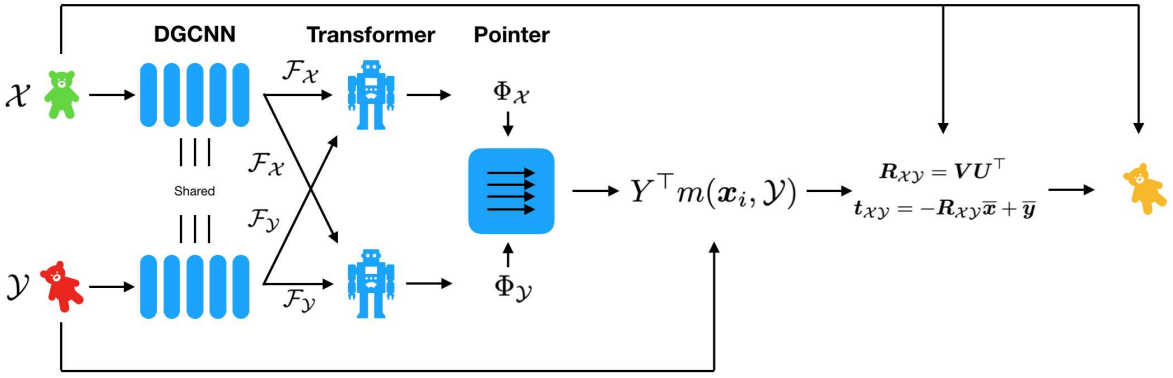


FIGURE 3.5 – Schéma de Deep Closest Point. Le principe est d'estimer des descripteurs pour chaque point puis de les mettre en correspondance avec un Transformers. Source : Wang *et al.* [Wang and Solomon, 2019a]

des descripteurs calculés par un PointNet puis estime la transformation avec une décomposition en valeur singulière. La fonction de coût prend en compte la transformation, mais aussi la matrice de correspondance stochastique. PRNet [Wang and Solomon, 2019b] remarque que la matrice de correspondance de DCP [Wang and Solomon, 2019a] est floue, et propose d'utiliser l'échantillonneur de Gumbel–Softmax pour avoir une matrice de correspondance moins floue. Contrairement à DCP [Wang and Solomon, 2019a], PRNet [Wang and Solomon, 2019b] estime la transformation de manière itérative. La plupart des méthodes bout à bout citées précédemment ont été entraînées de manière auto supervisée sur ModelNet. Cependant, ModelNet [Zhirong Wu et al., 2015] n'est qu'un jeu de données synthétique.

Huang *et al.* [Huang et al., 2020] ont une approche semblable à PointNetLK [Aoki et al., 2019] mais ajoutent un décodeur [Groueix et al., 2018] pour reconstruire la même version du nuage de points transformée et propose une fonction de coût supplémentaire basée sur la distance de Chamfer. Huang *et al.* [Huang et al., 2020] ont évalué leur méthode sur ModelNet, ont testé leur méthode sur 7-Scene [Shotton et al., 2013] qui est aussi une partie du jeu de données 3DMatch [Zeng et al., 2017]. Mais ils n'ont pas évalué leur méthode sur le jeu de données test de 3DMatch.

Deep Global Registration [Choy et al., 2020] est une méthode de recalage bout à bout qui estime des descripteurs [Choy et al., 2019], puis estime pour chaque paire de points une probabilité d'être une correspondance avec un réseau de neurones convolutionnels 6D. La transformation est estimée grâce à une décomposition en valeur singulière. C'est une méthode qui fonctionne bien sur des données réelles, comme 3DMatch [Zeng et al., 2017] ou KITTI Odométrie [Geiger et al., 2012] , mais ils n'ont pas montré de capacité de généralisation. Récemment, PointNetLK a été revisité par Li *et al.* [Li et al., 2021]. Ils proposent d'appliquer PointNetLK sur des voxels plutôt que sur tout le nuage de points. PointNetLK revisité [Li et al., 2021] a montré des résultats prometteurs sur 3DMatch.

3.2.7.4 Descripteurs profonds

Comme en image [Lowe, 2004], il est possible de calculer des descripteurs pour chaque point que nous pouvons mettre en correspondance pour ensuite estimer la transformation de façon robuste [Zhou et al., 2016]. Et comme pour le traitement d'image [Balntas et al., 2017], il est possible d'apprendre des descripteurs pour des nuages de points avec de l'apprentissage profond [Choy et al., 2019, Zeng et al., 2017, Deng et al., 2018, Deng et al., 2018, Gojcic et al., 2019, Bai et al., 2020]. Contrairement aux méthodes bout en bout, l'idée est d'utiliser dans un premier temps l'apprentissage profond pour calculer des descripteurs pertinents, puis dans un deuxième temps, mettre en correspondance les points qui ont le même descripteur, puis dans un troisième temps, utiliser un estimateur robuste (comme RANSAC [Fischler and Bolles, 1981]) pour estimer la rotation et la translation. Pour estimer les descripteurs, soit on se base sur des zones locales du nuage de points,

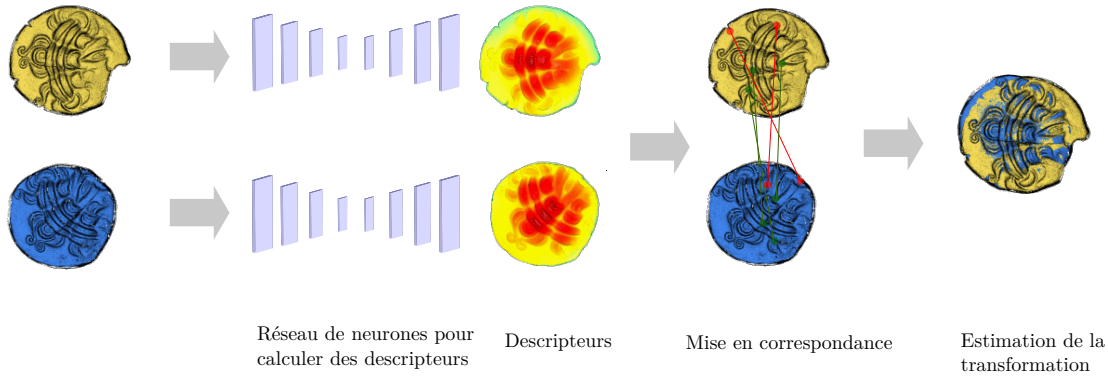


FIGURE 3.6 – Schéma représentant le calcul de descripteurs sur les droits sur Riedones3D. Les descripteurs calculés par le réseau de neurones sont ici représentés par des couleurs.

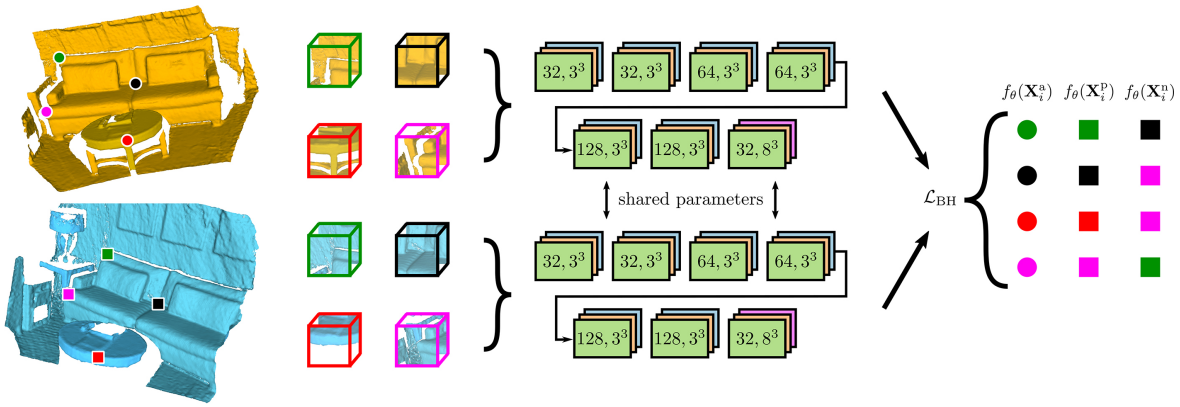


FIGURE 3.7 – Schéma de 3DSmoothnet qui représente bien le principe des méthodes basées *patch*. La première étape est l'extraction de *patch* (ici des cubes), puis une étape de calcul de descripteurs pour chacun des patchs. Les descripteurs pertinents sont appris grâce à de l'apprentissage de métrique. Source : Gojcic *et al.* [Gojcic et al., 2019]

ou bien nous estimons des descripteurs pour tous les nuages de points en utilisant un réseau U-Net.

Descripteurs profonds locaux par *patch* Depuis que 3DMatch [Zeng et al., 2017] a été introduit, beaucoup d'améliorations ont été faites sur les méthodes basées *patch*. Zeng *et al.*ont proposé le jeu de données 3DMatch [Zeng et al., 2017], mais également un 3DCNN pour calculer un descripteur par *patch*. PPFNet [Deng et al., 2018] utilise le réseau PointNet pour calculer des descripteurs locaux sur des *patches*, et avec un *pooling* global sur les descripteurs locaux, calcule un descripteur global pour apporter plus de contexte aux descripteurs locaux. PPF-Foldnet [Deng et al., 2018] et CapsuleNet [Zhao et al., 2019] utilisent un auto-encodeur pour calculer les descripteurs sur des zones locales sans supervision. 3DSmoothNet [Gojcic et al., 2019] a beaucoup contribué à améliorer les résultats sur 3DMatch [Zeng et al., 2017]. De plus, 3DSMoothNet [Gojcic et al., 2019] a montré de bonnes capacités à généraliser sur des données venant de différents capteurs comme les données venant du jeu de données ETH [Pomerleau et al., 2012]. 3DSmoothNet [Gojcic et al., 2019] calcule un système de coordonnées locales de référence ou LRF (comme pour SHOT [Salti et al., 2014]), pour orienter les *patches* et ensuite, utilise un réseau de neurones convolutionnels 3D sur les *patches* voxélisé et lissé avec un noyau gaussien pour calculer les descripteurs.

Suivant le succès de 3DSmoothNet [Gojcic et al., 2019], MultiView [Li et al., 2020], DIP [Poiesi and Poiesi, 2021], GeDI [Poiesi and Boscaini, 2021], et SpinNet [Ao et al., 2020] ont aussi amélioré

les capacités de généralisation sur le jeu de données ETH [Pomerleau et al., 2012] (voir Figure 3.7 pour voir). MultiView [Li et al., 2020] utilise un moteur de rendu différentiable pour générer des vues locales (les positions des caméras sont apprises) puis utilise un CNN 2D pour calculer un descripteur. DIP utilise un réseau PointNet [Qi et al., 2017] pour calculer un descripteur, mais contrairement à PPFNet [Deng et al., 2018], le *patch* est réorienté avec un LRF, comme pour 3DSmoothNet [Gojcic et al., 2019]. GeDI [Poiesi and Boscaini, 2021] est une extension de DIP qui utilise un réseau PointNet++ [Qi et al., 2017] pour calculer le descripteur. SpinNet [Ao et al., 2020] aligne le *patch* selon z, voxélise le *patch* dans une sphère, oriente chaque voxel dans le plan xy et projette les points dans un volume cylindrique. Ce prétraitement permet d'avoir un *patch* local invariant par rotation puis utilise une convolution cylindrique pour obtenir un descripteur. Néanmoins, même si ces méthodes sont invariantes par rotation, les méthodes par *patch* sont très lentes. Dans des conditions réelles, les descripteurs locaux profonds ne sont pas applicables, à cause de l'extraction de *patch*. De plus, chaque *patch* est traité individuellement, leur design n'est pas assez flexible pour ajouter de nouvelles capacités (comme un détecteur de points d'intérêt [Bai et al., 2020], un module pour estimer la transformation directement [Choy et al., 2020], ou un pré entraînement pour de la segmentation sémantique [Xie et al., 2020]...).

Méthodes basées sur l'architecture U-Net FCGF [Choy et al., 2019] est une des premières méthodes U-Net qui a été appliquée pour du recalage de nuages de points. Grâce à l'architecture U-Net et aux convolutions parcimonieuses sur des voxels, cette méthode fonctionne sur des données réelles comme 3DMatch [Zeng et al., 2017]. FCGF [Choy et al., 2019] est beaucoup plus rapide en inférence que les méthodes basées sur *patch*. D3Feat [Bai et al., 2020] utilise KPConv au lieu des convolutions parcimonieuses, et calcule un détecteur et un descripteur. Mais, FCGF [Choy et al., 2019] et D3Feat [Bai et al., 2020] ont de mauvaises capacités à généraliser [Ao et al., 2020]. PREDATOR [Huang et al., 2021] utilise D3Feat [Bai et al., 2020] et vient ajouter un réseau de neurones sur graphe ainsi qu'un module d'attention, pour calculer des descripteurs pertinents, même si le recouvrement entre les nuages de points est faible. *You Only Hypothesis Once* (YOHO) [Wang et al., 2021] va calculer des descripteurs pour 60 versions pivotées du nuage de points. YOHO ajoute un module pour estimer une rotation à partir d'une correspondance, ce qui permet de grandement accélérer RANSAC [Fischler and Bolles, 1981]. Liu *et al.* [Liu et al., 2021] proposent d'ajouter aux convolutions parcimonieuses une normalisation du voisinage. Cela permet d'être plus robuste aux variations de densité. Ces travaux montrent que les méthodes basées U-Net sont plus flexibles que les méthodes basées *patch*. De nouvelles capacités peuvent plus facilement être introduites comme un détecteur de points d'intérêt, ou peuvent servir pour construire des méthodes bout à bout (comme DGR [Choy et al., 2020]). Mais l'inconvénient de ces méthodes est qu'elles généralisent très mal sur des jeux de données différents [Ao et al., 2020, Poiesi and Poiesi, 2021, Poiesi and Boscaini, 2021].

3.2.7.5 Conclusions

Nous avons vu les différents algorithmes de recalage basés sur l'apprentissage profond. Les méthodes basées U-Net offrent de nombreuses possibilités pour un recalage rapide et efficace. Mais elles nécessitent un entraînement supervisé et généralisent mal quand on change de type de données [Ao et al., 2020]. Un modèle U-Net entraîné sur des données RGBD va généralement mal généraliser. De plus, nous pouvons nous demander si ces méthodes vont fonctionner sur le jeu de données Riedones3D : ce sont des données complètement différentes des données LiDAR ou RGBD et nous disposons d'un petit jeu de données d'entraînement.

3.3 Analyse des algorithmes de recalage sur Riedones3D

L'objectif de cette section est d'évaluer différents algorithmes de recalage sur Riedones3D. Pour pouvoir évaluer correctement les différents algorithmes nous avons besoin d'un jeu de don-

nées test. Pour les méthodes basées sur l'apprentissage profond, nous avons également besoin d'un jeu de données d'entraînement.

3.3.1 Présentation du jeu de données test

Dans le chapitre 2, nous avons présenté le jeu de données Riedones3D. Ici, nous détaillons le jeu de données test pour évaluer différents algorithmes de recalage. Le jeu de données test est composé de 160 droits (80 droits avec barbe et 80 droits sans barbe) et 79 revers. Pour l'évaluation du recalage, le jeu de données test est composé de 2158 paires de monnaies avec des transformations aléatoires fixées à l'avance (les rotations autour de l'axe x et y sont comprises entre -25 et +25 degrés et les rotations autour de l'axe z sont comprises entre -180 et +180 degrés). Pour toutes les méthodes, nous sous-échantillonnons le nuage de points avec une taille de voxel de 0,1 mm. Pour les méthodes basées sur l'apprentissage, nous utilisons une partie de Riedones3D comme jeu de données d'entraînement composée de 200 droits (pas de revers et pas de droit barbu pour l'entraînement). Pendant l'évaluation nous utilisons les droits et les revers pour voir si les méthodes basées sur l'apprentissage profond sont capables de généraliser à d'autres motifs. Cette évaluation de la généralisation est très utile pour nous indiquer si les méthodes basées sur l'apprentissage profond pourront être appliquées directement sur d'autres collections sans ré-entraînement. Un modèle qui généralise bien devrait fonctionner sur d'autres séries de motifs que Riedones3D.

3.3.2 Spécificité de Riedones3D par rapport aux autres jeux de données

Dans la section précédente nous avons procédé à un état de l'art non exhaustif des algorithmes de recalage. Les algorithmes sont souvent testés sur des nuages de points 3D d'objets volumétriques comme ModelNet ou le *Stanford 3D scanning repository* [Curless and Levoy, 1996] ([Yang et al., 2016, Gelfand et al., 2005, Wang and Solomon, 2019a, Aoki et al., 2019, Rusu et al., 2009, Mellado et al., 2014, Aiger et al., 2008]). D'autres méthodes sont évaluées sur des données réelles comme les données LiDAR ou RGBD [Choy et al., 2019, Mellado et al., 2014, Choy et al., 2020, Gojcic et al., 2019, Zhou et al., 2016]. Nous pouvons lister certaines difficultés que Riedones3D partage avec d'autres jeux de données tandis que d'autres encore lui sont propres :

— Recalage de motifs différent du recalage de forme (voir Figure 3.8)

— Nuage de points volumineux

— Motifs effacés, fissurés ou incomplets

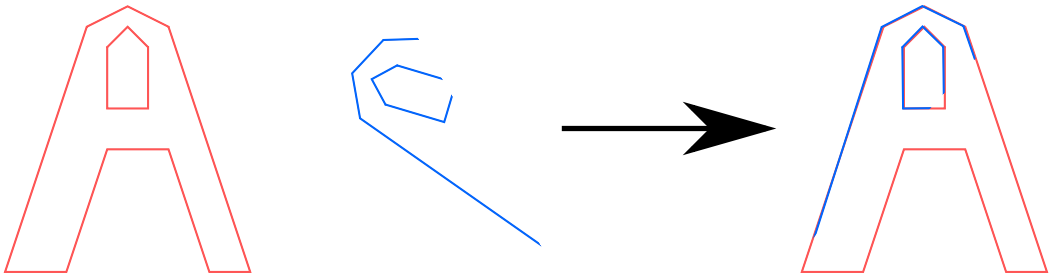
— Petit jeu d'entraînement

— Formes plates

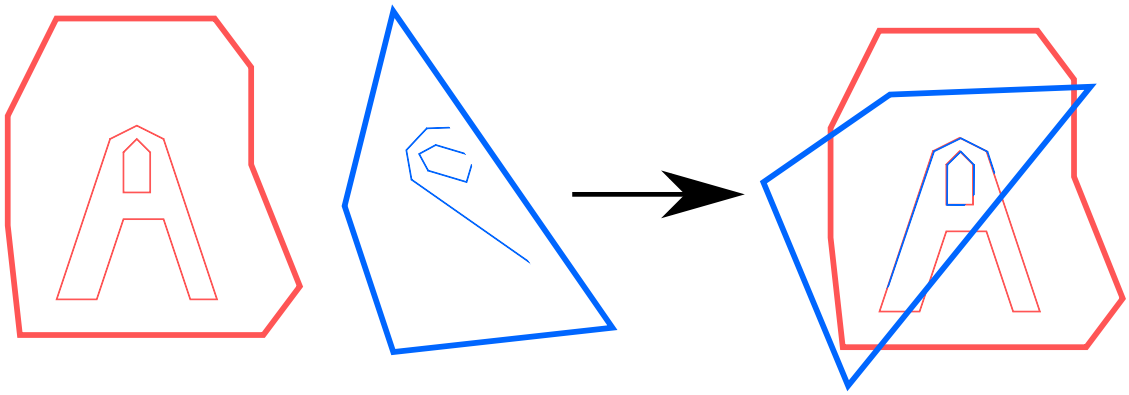
Le premier point est sans doute la plus grande différence par rapport aux autres nuages de points

Recalage de motifs Les jeux de données comme ModelNet [Wu et al., 2015], 3DMatch [Zeng et al., 2017] ou KITTI [Geiger et al., 2012] évalue le recalage pour de la reconstruction 3D d'objets ou de scènes. Ils évaluent donc le recalage de forme. Dans le cas de Riedones3D, l'objectif est de recaler des motifs. La figure 3.8 montre la différence entre Riedones3D et les autres jeux de données. Dans le cas de Riedones3D, le motif est inclus dans une forme. Il faut être prudent sur le choix des algorithmes à ce sujet.

Nuage de points volumineux Cette problématique est importante à considérer pour le choix des algorithmes. Certains algorithmes ne sont pas adaptés aux nuages de points sont volumineux. Les nuages de points de Riedones3D ont une taille d'environ 100k points. C'est semblable aux acquisitions de KITTI [Geiger et al., 2012] mais c'est très différent de ModelNet où les nuages de points ont une taille de 4k. De plus, dans le cas de Riedones3D les motifs sont très fins. Par conséquent, nous ne pouvons pas trop sous-échantillonner au risque de détruire le motif. Cela veut dire que certaines méthodes [Myronenko and Song, 2010, Aoki et al., 2019] ne sont pas applicables, car



(a) Recalage de formes



(b) Recalage de motifs

FIGURE 3.8 – Recalage de motifs par rapport au recalage de formes

ces algorithmes sont trop lents ou prennent trop de mémoire. La plupart des méthodes basées sur l'apprentissage profond de bout en bout ne vont donc pas fonctionner, car elles ne sont pas conçues pour des nuages de points volumineux.

Formes quasi-plates Les acquisitions 3D des monnaies ou d'autres motifs sont des structures quasi-plates, ce qui va impacter les résultats de beaucoup d'algorithmes. ICP point à point ne fonctionne pas sur des structures plates d'après [Rusinkiewicz and Levoy, 2001]. Pour des algorithmes comme 4PCS, comme quasiment tous les points sont coplanaires, il y a beaucoup de candidats congruents. Par conséquent, la recherche de la transformation n'est pas faisable dans ce cas.

3.3.3 Méthodes testées

Nous présentons les méthodes testées pour recaler des motifs sur Riedones3D. À priori, beaucoup de méthodes ne sont pas adaptées pour le recalage sur Riedones3D. Pour les méthodes de recalage local, nous nous sommes surtout concentrés sur les variantes d'ICP. Pour les méthodes de recalage global, nous nous sommes concentrés sur les méthodes basées descripteurs. En effet, ce sont des méthodes où le principe est de mettre en correspondance des *patches*. Ces méthodes sont donc adaptées pour le recalage de motifs.

3.3.3.1 Recalage de motifs avec ICP

Nous avons vu que pour l'ICP [Besl and McKay, 1992], il existe plusieurs variantes qui sont pertinentes selon le contexte. Si nous appliquons ICP [Besl and McKay, 1992] sur Riedones3D, les motifs ne vont pas être correctement recalés. En effet, les bords de la monnaie vont perturber le recalage. ICP cherche la transformation qui minimise la somme des distances au carré des cor-

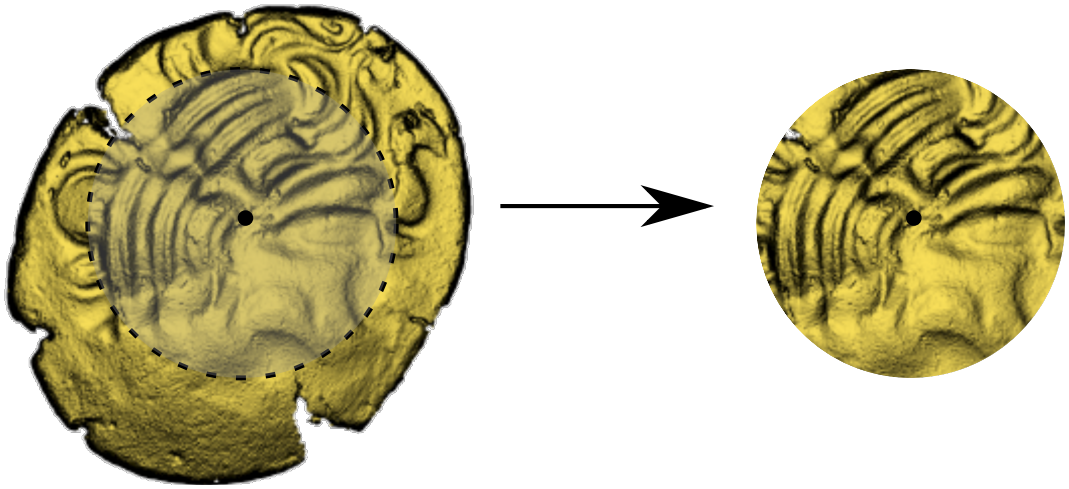


FIGURE 3.9 – Illustration de l'exclusion des bords

respondances, et même les correspondances près des bords qui ne contiennent pas de motif. Dit autrement, ICP recale les supports du motif et non les motifs eux-mêmes. Selon la fonction de coût, les algorithmes de recalage peuvent être sensibles sensibles aux données aberrantes [Babin et al., 2019]. Mais dans notre cas, nous ne pouvons pas faire l'hypothèse que les données aberrantes sont des points aléatoires (hypothèses souvent faites par d'autres travaux [Myronenko and Song, 2010, Gao and Tedrake, 2019]). Une fonction robuste seule risque toujours de prendre en compte les parties de la monnaie ne contenant pas de motif. Une solution simple est de supprimer les bords de la pièce [Horache et al., 2020]. En effet, le motif est en général centré. Pour supprimer les bords, nous calculons le centre de la pièce. Puis nous prenons une boule de rayon r. Si les points ne sont pas dans la boule, nous excluons ces points (voir figure 3.9). Cette variante permet d'obtenir de bons résultats sur les droits en général dans la mesure où le motif est centré. Cependant, quand le motif est excentré et qu'il vient se placer sur la bordure, la méthode devient moins efficace.

Nous avons vu dans l'état de l'art que la variante ICP point à plan, où le principe est de minimiser la distance entre les points et les plans représentés par les normales. Comme indiqué par Rusinkiewicz *et al.* [Rusinkiewicz and Levoy, 2001], la version originale d'ICP (ICP point à point), ne fonctionne pas sur les nuages de points plutôt plats. Seul l'ICP point à plan permet la convergence. Cette affirmation sera vérifiée lors de nos expérimentations. Une autre possibilité est d'appliquer ICP après une méthode basée descripteur [Horache et al., 2021b]. Après la mise en correspondance des parties similaires du nuages de points, nous extrayons une zone locale autour de chaque *inlier*, puis nous appliquons ICP uniquement autour de ces *inliers* (voir figure 3.10).

L'avantage est que nous connaissons déjà les *inliers*. Par conséquent, si les descripteurs ont mis en correspondance le motif, nous pouvons localiser le motif et appliquer ICP uniquement sur le motif. Le principe est qu'autour de chaque point correspondant, nous extrayons une zone locale puis nous effectuons le recalage sur ces zones locales. L'inconvénient de cette approche est que le succès de l'ICP est conditionné au succès de la mise en correspondance des parties de motifs similaires.

3.3.3.2 FPFH

Point Feature Histogram (PFH) et *Fast Point Feature Histogram* (FPFH) [Rusu et al., 2009] sont des descripteurs locaux assez populaires pour le recalage de scènes [Zhou et al., 2018, Rusu et al., 2009, Choy et al., 2020, Yang et al., 2020, Zhou et al., 2016]. Il y a donc une étape d'extraction de voisinage comme pour SHOT [Salti et al., 2014] et Spin image [Johnson and Hebert, 1999]. Pour PFH, nous allons utiliser les paires de points. $\mathbf{P}_x = \{p_i, \dots p_k\}$ est le voisinage du point x. Pour

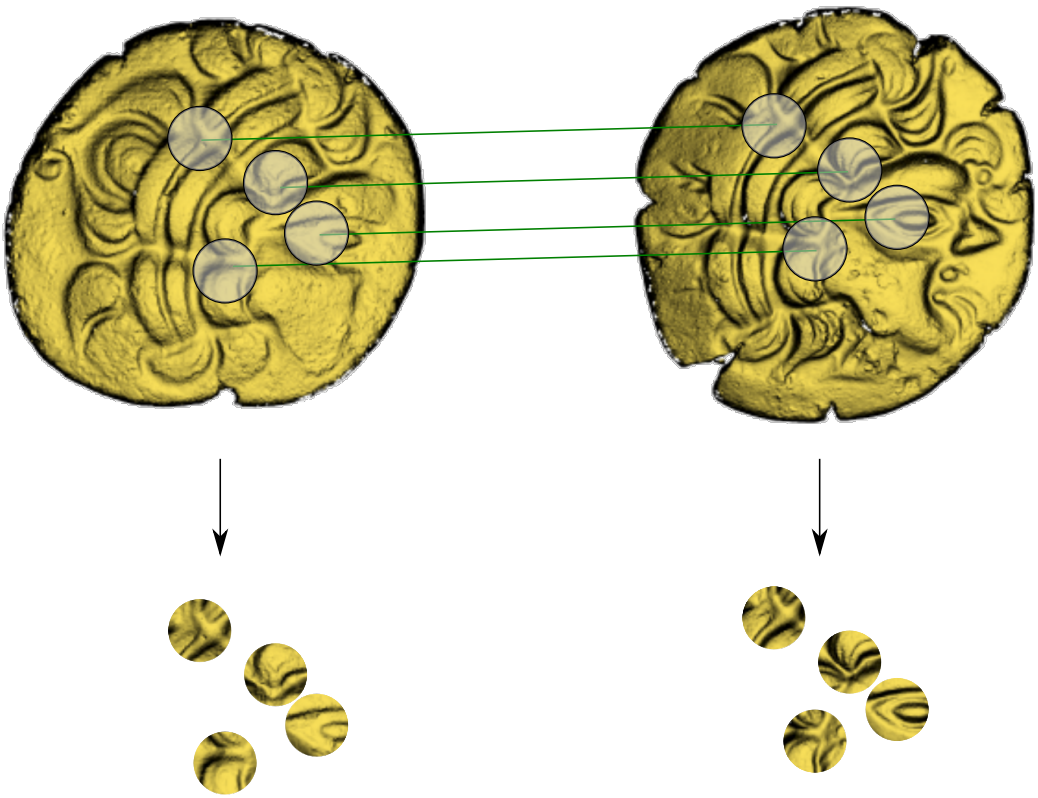


FIGURE 3.10 – Illustration de l'extraction locale uniquement autour des *inliers*

chaque paire de points dans le voisinage, nous allons :

1. Construire un repère local (u_{ij}, v_{ij}, w_{ij}) pour chaque paire de points i, j

$$u_{ij} = \mathrm{N}(p_i) \tag{3.19}$$
$$v_{ij} = u_{ij} \times \frac{p_j - p_i}{\|p_j - p_i\|} \tag{3.20}$$
$$w_{ij} = u_{ij} \times v_{ij} \tag{3.21}$$

2. On calcule les valeurs suivantes

$$\alpha_{ij} = v_{ij} \cdot \mathrm{N}(p_j) \tag{3.22}$$
$$d_{ij} = \|p_j - p_i\| \tag{3.23}$$
$$\beta_{ij} = u_{ij} \cdot (p_j - p_i) \tag{3.24}$$
$$\theta_{ij} = \arctan\left(\frac{w_{ij} \cdot \mathrm{N}(p_j)}{u_{ij} \cdot \mathrm{N}_(p_j)}\right) \tag{3.25}$$

Ensuite, on calcule un histogramme sur l'ensemble des valeurs $\alpha_{ij}, d_{ij}, \beta_{ij}, \theta_{ij}$. On divise l'espace de chaque paramètre en intervalle et nous comptons le nombre de paramètres dans chaque intervalle de l'histogramme. Mais PFH est long à calculer puisqu'il faut considérer pour chaque voisinage chaque paire de points (la complexité est de $\mathcal{O}(nk^2)$). FPFH est une variante plus rapide que PFH. Intuitivement, on compare plusieurs valeurs comme l'angle entre les normales des paires de points, la distance entre les points, l'angle entre les paires de points et la normale d'un point. L'astuce de FPFH est de calculer l'histogramme uniquement par rapport au centre du voisinage. Nous appelons SPFH(x) (Simple Point Feature Histogram), le PFH, mais uniquement pour les paires

$\{(x,p)|p \in \mathcal{N}_x\}$. Pour PFH décrit précédemment, on considérait toutes les paires du voisinage. Ensuite pour calculer FPFH(x), nous utilisons la formule suivante.

$$\text{FPFH}(x) = \text{SPFH}(x) + \frac{1}{|\mathbf{P}_x|} \sum_{p \in \mathbf{P}_x} \frac{1}{\|x-p\|} \text{SPFH}(p) \tag{3.26}$$

On passe ainsi à une complexité de $\mathcal{O}(nk)$

3.3.3.3 DIP

DIP [Poiesi and Poiesi, 2021] est une méthode pour calculer des descripteurs grâce à de l'apprentissage profond sur des *patches*, le principe est d'extraire un *patch* (zone locale autour d'un point), $\mathbf{P}_x = \{p_i, \dots p_k\}$ (x est le centre du *patch*). Ensuite, l'idée est de centrer les points par rapport à x. Ensuite, les *patches* sont orientés grâce à un repère de référence locale comme pour [Ao et al., 2020] ou 3DSmoothNet [Gojcic et al., 2019]. Le *patch* centré et orienté va être l'entrée d'un PointNet [Qi et al., 2017] composé d'un module TNet qui estime une transformation affine A (avec une régularisation du type $\|\mathrm{A}^{\mathrm{T}}\mathrm{A}-\mathrm{I}\|_{\mathrm{F}}$). La sortie du PointNet donne un descripteur de sortie qui est ensuite normalisé. La sortie de PointNet peut aussi être utilisée pour calculer ρ qui est un indicateur de la qualité du descripteur. En entraînement, DIP reçoit un *batch* de paires de *patches* locaux. DIP calcule un descripteur en sortie pour chaque *patch*. Ensuite, nous appliquons la fonction de coût suivante pour optimiser les descripteurs (appelée *Hardest contrastive loss*) :

$$\mathrm{L}(\theta) = \sum_{(i,j)\in\mathbf{M}^+} \{[\|f_{x_i} - f_{y_j}\| - m_+]_+^2 \tag{3.27}$$

$$+ \frac{1}{2}[m_- - \min_{k|(i,k)\in\mathbf{M}^-} \|f_{x_i} - f_{y_k}\|]_+^2 \tag{3.28}$$

$$+ \frac{1}{2}[m_- - \min_{k|(k,j)\in\mathbf{M}^-} \|f_{x_k} - f_{y_j}\|]_+^2\}, \tag{3.29}$$

θ sont les paramètres du réseau U-Net, f_{x_i} est le descripteur de sortie (il dépend de θ) associé au point x_i, où $[.]_+ = \max(.,0)$, $\mathbf{M}^+$ est l'ensemble des correspondances positives, et $\mathbf{M}^-$ est l'ensemble des correspondances négatives. $m+$ est un hyper paramètre appelé la marge positive, et $m-$ est appelé la marge négative. Dit autrement, cette fonction de coût incite les descripteurs décrivant une même partie similaire à avoir une faible distance euclidienne et des descripteurs décrivant des parties différentes à avoir une distance plus grande. Nous appliquons également une distance de Chamfer :

$$\mathrm{L}_c(\theta) = \sum_{p\in\mathcal{N}_x} \min_{q\in\mathcal{N}_y} \|\mathrm{A}p - \mathrm{A}q\|_2 + \sum_{q\in\mathcal{N}_y} \min_{p\in\mathcal{N}_x} \|\mathrm{A}p - \mathrm{A}q\|_2 \tag{3.30}$$

Cette distance de Chamfer est une régularisation du module TNet qui calcule la transformation affine du *patch* avant d'être traité par le réseau PointNet [Qi et al., 2017].

L'avantage des méthodes par *patch* est qu'il est possible de traiter des nuages de points volumineux, car seule une zone locale du nuage de points est traitée. DIP a montré des bonnes capacités de généralisations sur des capteurs différents. Cependant, DIP est très lent à cause de l'extraction de *patch* comme la plupart des méthodes basées *patch*.

3.3.3.4 FCGF

Fully Convolutional Geometric Feature ou FCGF est un réseau U-Net qui calcule pour chaque nuage de points, des descripteurs. Tout d'abord, le nuage de points $\mathbf{X}$ est voxélisé et est représenté par deux ensembles $\mathbf{V_X} = \{(c_1^{(1)}, c_2^{(1)}, c_3^{(1)}, b^{(1)}), \dots, (c_1^{(m)}, c_2^{(m)}, c_3^{(m)}, b^{(m)})\}$ et $\mathbf{F_X} = \{1, \dots 1\}$ qui sont respectivement l'ensemble des voxels et les *features* d'entrée. $(c_1^{(i)}, c_2^{(i)}, c_3^{(i)})$ sont les coordonnées des voxels et $b^{(i)}$ est un indice pour indiquer le *batch*. m est la taille du nuage de points voxélisé. L'avantage de la représentation voxélisée parcimonieuse est qu'il est possible de traiter

des nuages de points volumineux. L'architecture U-Net est composée de quatre blocs encodeurs puis de 4 blocs décodeurs. Chaque bloc est résiduel [He et al., 2016] avec une convolution 3D parcimonieuse [Choy et al., 2019], une fonction d'activation ReLU et une *Batch normalization* [Ioffe and Szegedy, 2015]. Une convolution 3D parcimonieuse consiste à une recherche des voxels voisins parcimonieux dans le nuage de points voxélisé ce qui correspond à une recherche dans une table de hachage ce qui peut être implémenté sur le GPU [Choy et al., 2019, Tang et al., 2020]. En sortie du U-Net, il y a un descripteur qui est un vecteur de taille d (nous avons choisi $d = 32$). Pour entraîner les descripteurs de sortie à être pertinents, l'idée est d'utiliser un apprentissage de métrique sur chaque descripteur, comme pour DIP [Poiesi and Poiesi, 2021]. L'idée est de prendre un *batch* de paires de nuages de points. On les nomme $\mathbf{X}$ et $\mathbf{Y}$. Ils sont ensuite voxélisés. Le réseau calcule des descripteurs pour chacun des nuages de points voxélisés. Ensuite, nous appliquons la fonction de coût "*Hardest contrastive loss*" comme pour DIP [Poiesi and Poiesi, 2021]. En inférence, le réseau peut être utilisé pour calculer des descripteurs sur un ou plusieurs nuages de points voxélisés. Enfin, il y a une mise en correspondance des descripteurs puis une estimation de la transformation grâce à un estimateur robuste comme RANSAC (comme pour DIP [Poiesi and Poiesi, 2021], FPFH [Rusu et al., 2009] ou d'autres descripteurs).

3.3.4 Choix de la meilleure variante d'ICP

L'objectif de cette partie est d'évaluer différentes variantes d'ICP sur Riedones3D. Nous allons comparer la variante point à point avec la variante point à plan. Nous allons également déterminer l'influence de l'exclusion des bords sur le résultat du recalage. Nous allons également évaluer l'influence de l'initialisation pour des méthodes globales. Pour cela, nous allons comparer l'initialisation aléatoire avec l'initialisation par grille. Nous allons également comparer avec GO-ICP.

3.3.4.1 Protocole expérimental

Pour les expériences qui vont suivre, nous allons utiliser comme métrique l'erreur de translation (en mm) et l'erreur de rotation (en degré) qui sont définies ainsi :

$$\delta t_{12} = ||t_2 - t_1||_2 \tag{3.31}$$

$$\delta \mathrm{R}_{12} = ||\log_{\mathrm{SO3}}(\mathrm{R}_1 \mathrm{R}_2^{\mathrm{T}})||_2 \tag{3.32}$$

Pour savoir si le recalage est un succès ou non, nous pouvons effectuer un test sur l'erreur de translation et de rotation. Si l'erreur de translation $\delta t_{12} < \delta t_{manual}$ et l'erreur de rotation $\delta \mathrm{R}_{12} < \delta \mathrm{R}_{manual}$, nous pouvons considérer que le recalage est un succès. t_{manual} et R_{manual} sont des valeurs à fixer. Pour les fixer, nous pouvons regarder l'écart-type de la variation de la translation et de la rotation lorsqu'on recale manuellement les acquisitions 3D. Nous avons mesuré expérimentalement $\delta \mathrm{R}_{manual}$ à 5 degrés et δt_{manual} à 0,5 mm. Ces métriques ont plusieurs défauts (nous les verrons ultérieurement dans la sous-sous section 3.3.5.1), mais sont couramment utilisées pour évaluer le recalage [Choy et al., 2019, Rusinkiewicz and Levoy, 2001, Myronenko and Song, 2010]. Pour le jeu de données, nous utilisons pour cette section une partie de Riedones3D. Cette partie est constituée de 66 *scans* de droits classés par coin. Pour évaluer le recalage, nous évaluons sur les paires de pièces qui sont du même coin. Ici, nous n'évaluons que sur un type de motif. Mais ce petit jeu de données donne quand même un très bon aperçu du succès ou de l'échec d'une méthode.

3.3.4.2 Évaluation de l'ICP local

Nous avons évalué les différentes variantes d'ICP proposées afin voir quels sont les paramètres importants :

— L'influence des bords

— L'influence de la fonction de coût (point à point vs point à plan)

La Figure 3.11 montre l'influence de l'exclusion des bords. Elle montre également que ICP point à plan est meilleur que ICP point à point. On peut même en conclure que ICP point à point échoue à converger vers la solution désirée. Rusinkiewicz *et al.* [Rusinkiewicz and Levoy, 2001] avaient déjà remarqué que ICP point à plan marchait mieux sur les structures plates que ICP point à point. En supprimant les bords des pièces, on obtient une erreur plus faible. Mais si nous ne gardons que les points dans une boule de rayon de 4 mm, nous remarquons que l'erreur est plus grande. Par conséquent, nous pouvons conclure que si nous excluons trop de bords en prenant un faible rayon, l'erreur de rotation et translation est grande.

3.3.4.3 Influence de l'initialisation

Nous avons évalué l'influence de l'initialisation, pour cela nous avons calculé le bassin de convergence d'ICP. La figure 3.12 montre le bassin de convergence d'ICP en faisant varier deux paramètres. Pour la translation, nous avons pris une direction aléatoire sur le plan de la pièce de monnaie puis nous avons fait varier la norme de la translation. Pour la rotation, nous avons fait varier l'angle de rotation autour de la normale sur le plan de la pièce.

Pour une paire de pièces, avec une distance entre l'initialisation et le minimum global de moins de 10 degrés et 3 mm en translation, on peut admettre qu'ICP va converger pour des monnaies similaires. Ces résultats ne sont pas généralisables sur toutes les paires de monnaies. Mais cette information est utile pour le recalage global.

3.3.4.4 Optimisation globale

Nous avons vu dans la section précédente que la métrique point à plan ainsi que l'exclusion des bords nous permet de concevoir une variante d'ICP qui fonctionne localement pour le jeu de données Riedones3D. Comme nous l'avons dit dans les sections précédentes, le recalage local fonctionne que si les nuages de points sont au moins alignés approximativement. Si nous voulons une méthode 100% automatique, il faut une méthode de recalage globale. Comme nous l'avons vu lors de la présentation de l'état de l'art sur les méthodes globales, il y a trois possibilités pour rendre un ICP global :

— RandomSearchICP : nous essayons plusieurs initialisations aléatoires

— GridSearchICP : nous pouvons chercher les initialisations sur une grille

— GO-ICP [Yang et al., 2016] : nous pouvons faire une recherche sur un *octree* grâce à un algorithme de *branch and bound*

Nous avons testé les trois méthodes avec les résultats affichés dans le tableau 3.1. Nous voyons que GO-ICP et GridSearchICP sont équivalents. En effet, les bornes calculées par GO-ICP [Yang et al., 2016] sont trop souples, GO-ICP n'arrive pas à exclure les solutions non pertinentes. GO-ICP revient donc à chercher dans tous les éléments d'une grille et devient équivalent à GridSearchICP. Cela montre la particularité de Riedones3D. GO-ICP a de meilleurs résultats sur des jeux de données comme le *Stanford 3D scanning repository* [Curless and Levoy, 1996]. Dans 94% du temps, RandomSearchICP a une erreur inférieure à 5 degrés (par rapport à 84% pour GridSearchICP). Même si ce n'est pas déterministe, une recherche aléatoire semble plus adaptée qu'une recherche par grille (pour un budget temps similaire).

Méthodes	Erreur < 1 deg	Erreur < 5 deg	Erreur < 0,5 mm
GridSearchICP	18%	84%	75%
GO-ICP	18%	84%	75%
RandomSearchICP	**22%**	**94%**	**86%**

TABLEAU 3.1 – Succès du recalage global en %. Les erreurs sont mesurées avec l'équation 3.31 et 3.32. Pour la recherche par grille et la recherche aléatoire ainsi que GO-ICP, nous exécutons le même nombre d'essais (environ une centaine).

3.3.5 Comparaison des méthodes de recalage global

L'objectif des expériences précédentes étaient de sélectionner la meilleure variante d'ICP pour notre problème. Nous avons conclu que RandomSearchICP avait les meilleurs résultats sur les droits des monnaies de Riedones3D. Dans cette sous-section, nous comparons plusieurs méthodes pour le problème plus général du recalage sur les monnaies Riedones3D. Nous comparons RandomSearchICP avec des méthodes basées descripteurs, notamment FPFH [Rusu et al., 2009]) ainsi que FCGF [Choy et al., 2019] et DIP [Poiesi and Poiesi, 2021]. Nous allons cette fois-ci comparer sur les droits, les barbus et les revers. Ainsi, nous pourrons déterminer si les méthodes basées sur l'apprentissage profond peuvent généraliser sur des motifs qui ne sont pas présents dans la base d'entraînement.

3.3.5.1 Métrique

Pour mesurer l'erreur de recalage, nous utilisons l'erreur introduite par Fontana [Fontana et al., 2021]. Nous allons l'appeler l'erreur de recalage normalisé ou *Scaled Registration Error* (SRE). Supposons que nous avons une paire de nuages de points $\mathbf{X}, \mathbf{Y}$ et la transformation vérité terrain $(\mathrm{R}^{(gt)}, t^{(gt)})$. Nous voulons évaluer un algorithme qui produit la transformation (R^*, t^*). Le SRE pour la paire $\mathbf{X}, \mathbf{Y}$ est défini comme :

$$\mathrm{SRE}(\mathbf{X}, \mathbf{Y}) = \frac{1}{|\mathbf{X}|} \sum_{i=1}^{|\mathbf{X}|} \frac{\|(\mathrm{R}^{(gt)} x_i + t^{(gt)}) - (\mathrm{R}^* x_i + t^*)\|}{\|(\mathrm{R}^{(gt)} x_i + t^{(gt)}) - (\mathrm{R}^{(gt)} \bar{x} + t^{(gt)})\|} \tag{3.33}$$

$$\bar{x} = \frac{1}{\mathbf{X}} \sum_{i=1}^{|\mathbf{X}|} x_i. \tag{3.34}$$

Le SRE dépend de $\mathbf{X}$ et $\mathbf{Y}$, parce que nous estimons R^* et t^* à partir de ces deux nuages de points. Pour toutes les paires de nuages de points dans le jeu de données avec N paires, nous calculons la médiane du SRE entre chaque paire :

$$\mathrm{SRE} = \mathrm{median}_{i=1\ldots\mathrm{N}}(\mathrm{SRE}(\mathbf{X}_i, \mathbf{Y}_i)). \tag{3.35}$$

La moyenne est sensible aux SRE aberrants, et n'est pas représentative des résultats comme expliqués par Fontana *et al.* [Fontana et al., 2021]. Pour évaluer la transformation calculée par les méthodes, les métriques utilisées sont l'erreur de rotation et l'erreur de translation (nous avons utilisé ces métriques précédemment). Même si ces métriques sont très utiles, elles souffrent de quelques défauts : il y a deux mesures au lieu d'une seule, dans certains cas, la comparaison n'est pas possible. De plus, ces métriques ne sont pas invariantes par translation. Par exemple, si le nuage de points est à 100 m du centre de rotation, une petite rotation de 2 degrés va translater le nuage de points de 3 m.

3.3.5.2 Méthodes évaluées

Nous évaluons quelques méthodes : des méthodes comme ICP [Besl and McKay, 1992], et des descripteurs 3D comme FPFH [Rusu et al., 2009]. Nous évaluons aussi les méthodes basées sur l'apprentissage profonds comme FCGF [Choy et al., 2019] et DIP [Poiesi and Poiesi, 2021]. Pour chaque coin, nous avons différentes paires de pièces et nous pouvons calculer le SRE pour chaque paire (comme défini dans l'équation 3.33) et calculer la médiane du SRE pour chaque coin (les résultats sont dans le tableau 3.3, 3.4 et 3.2). Comme ICP [Besl and McKay, 1992] est dépendant de l'initialisation, nous utilisons le RandomSearchICP défini au paragraphe 3.2.6.1 pour ne plus dépendre de l'initialisation. FPFH [Rusu et al., 2009] utilise les normales et des vecteurs caractéristiques calculés sur des paires de points pour calculer des descripteurs locaux. FCGF [Choy et al., 2019] est un modèle U-Net qui calcule des descripteurs pour chaque point. DIP [Poiesi and Poiesi, 2021] calcule des descripteurs sur des *patches* locaux en utilisant une architecture PointNet. Les méthodes basées sur l'apprentissage profond sont souvent considérées comme efficaces seulement si le jeu de données d'entraînement est grand. Or, nos résultats sur Riedones3D démontre

clairement le contraire pour DIP et FCGF. FCGF et DIP ont été entraînés uniquement sur un jeu de données constitué de 200 droits. Pourtant, le tableau 3.2 montre que les méthodes basées sur l'apprentissage profond ont de meilleurs résultats que les autres méthodes en matière de SRE. Ces résultats nous permettent de conclure qu'un petit jeu de données est suffisant pour apprendre des descripteurs pertinents pour le recalage de motifs sur des monnaies. Les résultats pour DIP et FCGF sont satisfaisants. FCGF a un SRE de 8,7 pour les barbus, 9,3 pour les droits et 68,2 pour les revers et DIP a un SRE de 73,4 pour les barbus, 9,3 pour les droits et 281,4 pour les revers. On peut considérer qu'un SRE en dessous de 60 signifie que le recalage est un succès (nous pouvons déterminer ce seuil de manière empirique). Cependant, DIP [Poiesi and Poiesi, 2021] a de moins bons résultats que FCGF [Choy et al., 2019] surtout sur R8 et R5 (voir tableau 3.3). DIP [Poiesi and Poiesi, 2021] a donc plus de mal à généraliser sur des motifs qui lui sont inconnus contrairement à FCGF [Choy et al., 2019], alors que sur des jeux de données comme ETH [Pomerleau et al., 2012] ou 3DMatch [Zeng et al., 2017], DIP [Poiesi and Poiesi, 2021] offre de meilleures capacités que FCGF à généraliser. Une explication est que ETH et 3DMatch ont des géométries plutôt simples alors que les motifs des monnaies de Riedones3D sont plus sophistiqués. De plus, le tableau 3.2 montre que DIP est plus lent que FCGF. Pour le résultat des revers, nous pouvons voir que FCGF [Choy et al., 2019] a un SRE de 68,2. Mais si nous regardons les résultats en détail du tableau 3.3, nous observons que pour la plupart des coins, le SRE reste faible (autour de 10) (R1, R10, R12, R16, R2, R3, R4). Pour certains coins, le SRE est très élevé (R11 par exemple). On peut en conclure à partir des résultats du tableau 3.3 que FCGF fonctionne dans une majorité de cas pour les revers, mais pour certaines faces de même coin mais très différentes, FCGF ne fonctionne pas (voir figure 3.13).

Méthodes	**Droits (D)**	**Droits barbus (DB)**	**Revers (R)**	**Total**	**Temps (en s)**
RandomSearchICP	**7.2**	70.7	512.9	302.2	70.7
FPFH (5000) + TEASER	358.6	448.1	492.0	452.6	2.1
FPFH (5000) + TEASER + ICP	616,6	378,1	501,5	499,6	2,3
DIP (5000) + TEASER	21,9	46,1	370,4	220,9	61,2
DIP (5000) + TEASER + ICP	9,3	73.4	281.4	174,7	62,9
FCGF (250) + TEASER	19.3	42.6	126.0	83.8	**0.6**
FCGF (5000) + TEASER	10,3	21,4	96,7	60,8	2,1
FCGF (250) + TEASER + ICP	9.3	11.0	101.0	60.6	1,2
FCGF (5000) + TEASER + ICP	9.3	**8.7**	**68.2**	**41.9**	3.8

TABLEAU 3.2 – SRE (x1000) sur le jeu de données Riedones3D pour le recalage de monnaies (moyenne des médianes pour chaque coin). Pour FCGF, nous avons essayé FCGF (250) et FCGF (5000) : 250 et 5000 sont le nombre de descripteurs utilisés pour l'estimation de la transformation. Nous utilisons TEASER comme estimateur robuste de la transformation. ICP veut dire que nous rajoutons un ICP additionnel pour raffiner la transformation. Pour les méthodes basées sur l'apprentissage profond (FCGF et DIP), l'entraînement est fait sur les droits sans barbe uniquement.

Méthodes	R1	R10	R11	R12	R14	R16	R17	R2	R3	R4	R5	R6	R7	R8	R9
Random Search ICP	115.8	1559.7	1803.5	27.5	54.8	1.5	258.3	101.0	24.1	12.7	399.4	7.2	5.4	1595.3	1727.1
FPFH (5000) + TEASER	66.6	155.4	1591.2	38.2	69.1	48.3	27.9	740.8	1324.0	59.3	1504.7	39.7	23.6	1670.4	20.9
FPFH (5000) + TEASER + ICP	11.2	14.4	1518.5	20.9	42.8	10.1	738.1	10.4	1321.6	21.7	1578.2	18.3	10.0	2194.6	11.2
DIP (5000) + TEASER	29.7	43.0	1395.8	32.6	55.4	30.6	63.0	28.8	33.0	34.3	2027.7	27.6	15.4	1705.3	33.7
DIP (5000) + TEASER + ICP	8.8	11.6	1546.9	24.2	47.3	9.3	24.3	10.4	11.5	9.0	595.3	10.8	9.6	1892.7	8.8
FCGF (250) + TEASER	47.1	59.7	858.6	41.8	72.0	53.8	108.7	77.3	64.2	59.8	133.8	54.0	85.0	102.7	71.6
FCGF (5000) + TEASER	18.5	23.4	1053.9	18.0	49.5	15.8	35.4	18.4	22.3	24.2	41.6	17.4	42.6	53.7	16.3
FCGF (250) + TEASER + ICP	10.6	12.7	869.4	24.3	82.7	8.9	66.8	9.4	10.8	13.7	93.1	9.7	8.4	281.5	12.6
FCGF (5000) + TEASER + ICP	9.8	9.5	731.4	20.6	61.6	10.4	32.1	10.1	9.2	8.7	70.9	8.4	9.6	22.1	8.0

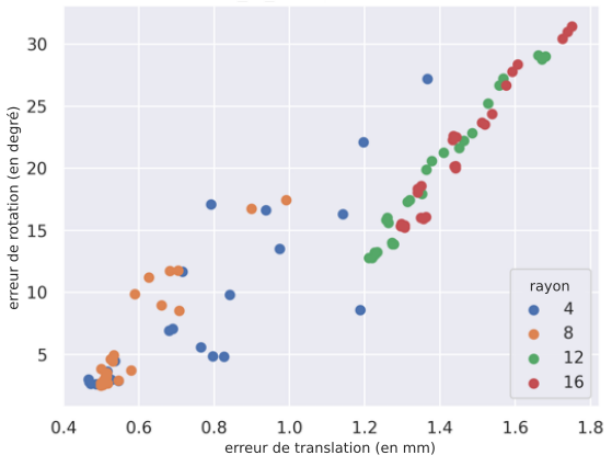
TABLEAU 3.3 – Résultats détaillés avec le SRE (x1000) sur le jeu de données Riedones3D pour le recalage de monnaies. Les résultats sont présentés pour chaque coin séparément (Revers).

Méthodes	DB1a	DB2a	DB3a	DB1b	DB2b	DB3b	D1	D2	D5	D10	D15	D33
Random Search ICP	24.2	337.1	13.5	5.8	22.7	21.1	6.6	1.5	12.6	9.7	7.6	4.9
FPFH (5000) + TEASER	166.1	57.6	25.4	1268.7	1145.9	25.0	1233.1	548.6	103.4	56.0	180.0	30.6
FPFH (5000) + TEASER + ICP	81.4	5.4	7.5	1123.2	1043.8	7.5	1305.2	980.1	100.6	13.5	1295.7	4.6
DIP (5000) + TEASER	30.5	25.4	23.1	140.7	31.1	25.6	22.7	26.2	22.7	21.6	30.5	7.9
DIP (5000) + TEASER + ICP	11.6	10.7	7.2	389.2	14.3	7.6	10.4	8.6	12.2	9.9	11.2	3.3
FCGF (250) + TEASER	53.5	42.3	41.5	48.9	35.5	33.8	22.8	21.7	24.5	22.1	16.6	8.3
FCGF (5000) + TEASER	23.4	29.4	18.8	24.7	17.1	15.0	10.6	7.3	9.8	10.5	12.5	11.0
FCGF (250) + TEASER + ICP	14.9	7.9	11.1	4.9	14.4	12.6	10.5	7.4	12.5	10.0	11.2	4.4
FCGF (5000) + TEASER + ICP	10.7	6.5	9.0	4.8	12.9	8.1	10.5	8.3	11.3	9.7	11.2	5.0

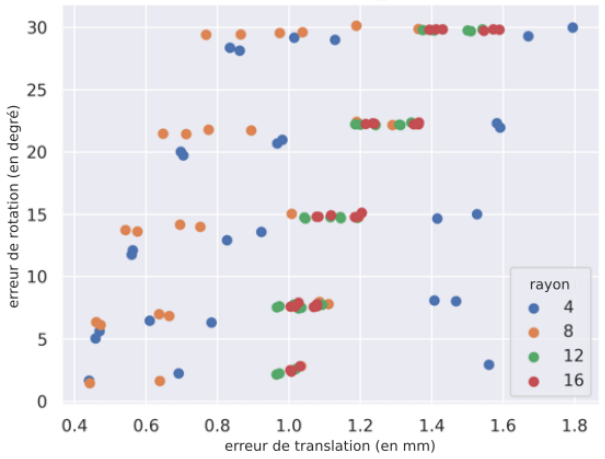
TABLEAU 3.4 – Résultats détaillés avec le SRE (x1000) sur le jeu de données Riedones3D pour le recalage de monnaies. Les résultats sont présentés pour chaque coin séparément (Droits et barbus).

3.3.5.3 Visualisation des descripteurs

Dans notre cas, il est possible de faire une interprétation de la sortie des méthodes basées descripteurs. Pour cela, nous pouvons visualiser les descripteurs en 3 dimensions avec des couleurs en utilisant un algorithme de réduction de la dimension. L'algorithme utilisé dans notre cas est l'Analyse en Composante Principale. En figure 3.14, nous pouvons visualiser les descripteurs pour les droits (en figure 3.15 pour les barbus et en figure 3.16 pour les revers). Pour les trois motifs, nous voyons trois couleurs et une couleur semble indiquer la position du motif dans le nuage de points 3D. Sans avoir indiqué au réseau comment localiser le motif explicitement, il discrimine quand même le motif par rapport aux bords. Cela veut dire que le réseau de neurones doit probablement faire une distinction entre le motif et le bord pour pouvoir identifier les parties de motifs similaires. Pour la figure 3.16, nous remarquons que beaucoup de parties de motifs qui sont similaires n'ont pas la même couleur. En effet, le réseau de neurones n'a pas été entraîné sur les revers, les descripteurs calculés sont donc moins pertinents (cela se voit également sur les résultats quantitatifs). Cependant, en pratique, la méthode proposée a quand même de bons résultats en matière de SRE.



(a) ICP point à Plan



(b) ICP point à point

FIGURE 3.11 – Erreur de translation en mm (abscisse) et erreur de rotation en degré (sur l'axe des ordonnées) avec différentes tailles de rayons (en mm) pour l'exclusion des bords (le rayon est en mm). Chaque point représente la moyenne des erreurs sur des paires de monnaies avec 100 initialisations aléatoires. ICP point à point ne converge pas. ICP point à plan converge seulement si nous excluons les bords, mais si nous excluons trop de zones de la pièce (rayon trop petit), ICP [Besl and McKay, 1992] ne converge pas vers la transformation désirée.

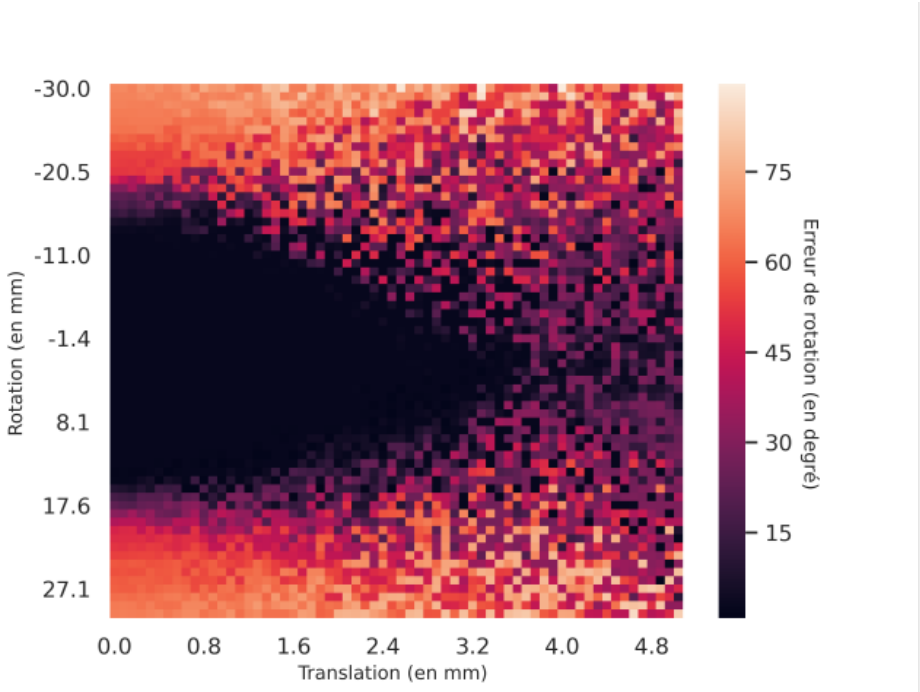
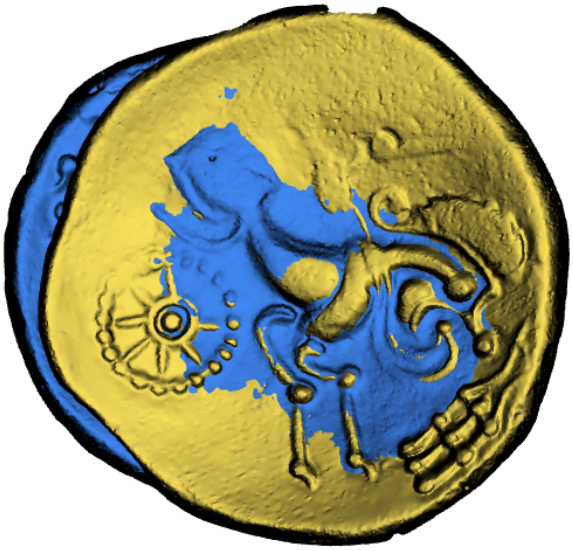
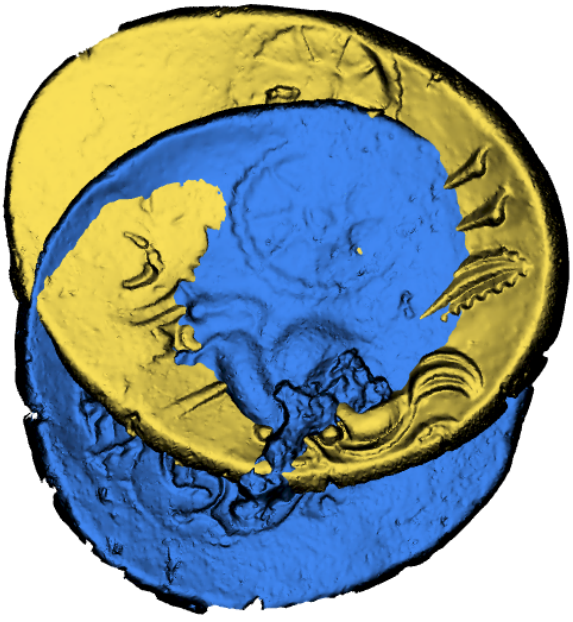


FIGURE 3.12 – Bassin de convergence de l'ICP en fonction de l'initialisation pour une paire de pièces qui vient du même coin. Dans l'axe des abscisses, la norme de la translation varie de 0 à 5 mm. Pour l'axe des ordonnées, la rotation varie de -30 à +30 degrés. La partie sombre de la carte représente le bassin de convergence d'ICP.



(a) Succès du recalage (coin R5).



(b) Exemple d'échec du recalage (coin R11).

FIGURE 3.13 – Recalage de paires de monnaies en utilisant FCGF [Choy et al., 2019] sur le coin R5 entre L0020R et L0061R (succès) et la paire du coin R11 entre L0053R et L0059R (échec).

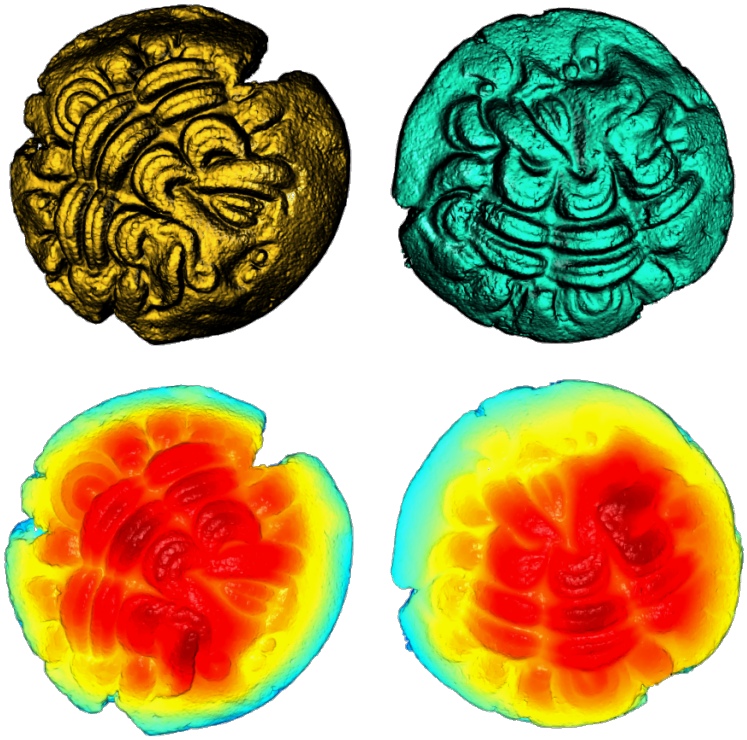


FIGURE 3.14 – Visualisation des descripteurs grâce à une analyse en composante principale, les monnaies sont les droits L0162D et L0163D.

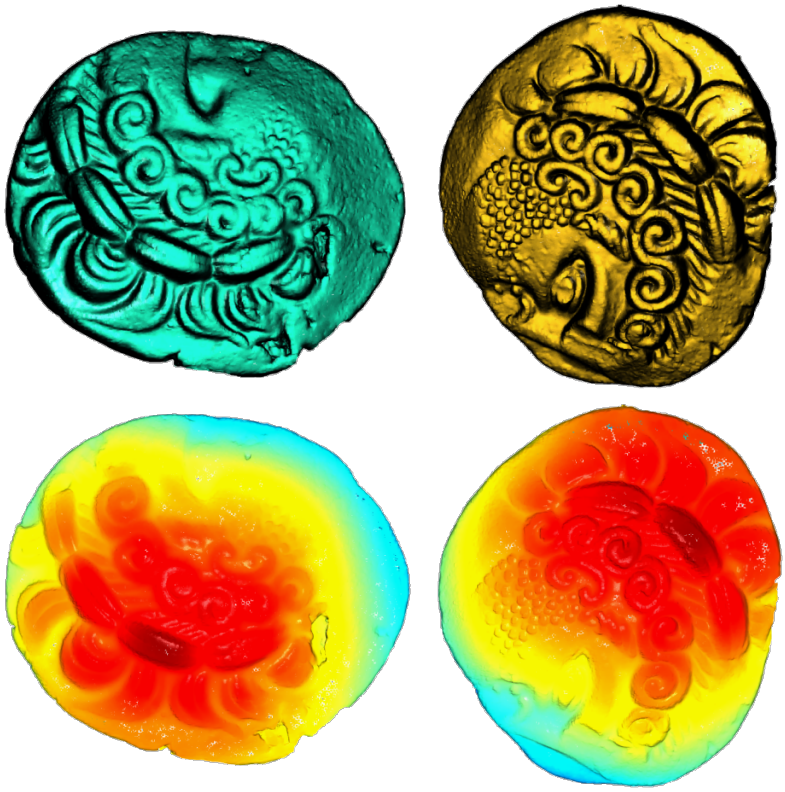


FIGURE 3.15 – Visualisation des descripteurs grâce à une analyse en composante principale, les monnaies sont les droits barbus L0020D et L0061D.

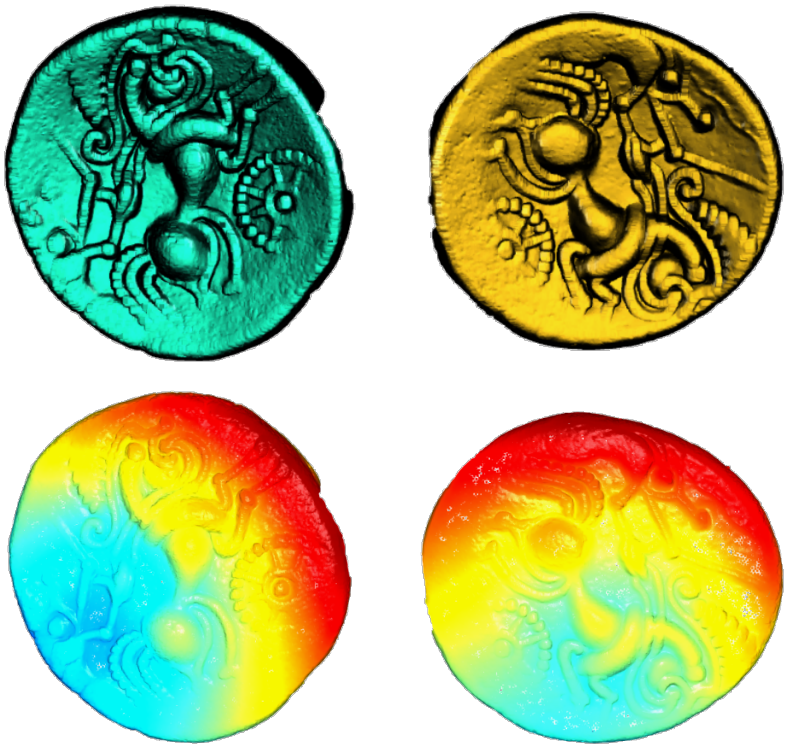


FIGURE 3.16 – Visualisation des descripteurs grâce à une analyse en composante principale, les monnaies sont les revers L0070R et L0077R.

3.4 Conclusion

Dans cette partie, nous avons effectué un bilan de l'état de l'art en recalage en 2022. Nous avons vu que la plupart des méthodes n'étaient pas adaptées à Riedones3D, car elles étaient testées sur des jeux de données complètement différents. Par conséquent, nous avons proposé deux variantes d'ICP qui fonctionne sur les droits de Riedones3D et nous avons conçu un algorithme de recalage global à partir d'une des variantes d'ICP. Nous avons exposé les limites de la variante proposée sur Riedones3D notamment pour les revers. Nous avons également proposé une évaluation de différentes méthodes de recalage et nous avons montré le succès des méthodes basées sur l'apprentissage profond sur Riedones3D par rapport aux autres méthodes, même si le jeu de données d'entraînement est relativement limité, n'est constitué que de 200 monnaies de droits. Concrètement, cette évaluation nous indique que FCGF a de très bonne chance de fonctionner sur des motifs venant de monnaies différentes. Nous avons vu que FCGF généralise mieux que DIP sur les revers de Riedones3D et est plus rapide. Par conséquent, nous avons choisi d'utiliser FCGF pour le calcul des descripteurs.

Chapitre 4

Applications du recalage aux monnaies et objets celtiques

Sommaire

4.1 Introduction

Nous avons vu dans le chapitre 3 comment réaliser le recalage sur Riedones3D. Le recalage est important pour pouvoir réaliser si deux monnaies ont été frappées avec le même coin (voir figure 2.5 du chapitre 2) Nous avons vu que les approches basées sur l'apprentissage profond (particulièrement FCGF [Choy et al., 2019]) fonctionnent sur Riedone3D, les résultats sont satisfaisant également sur les revers même s'ils sont un peu moins bon que sur les droits. Dans ce chapitre, nous allons voir les applications que permet le recalage pour le regroupement de monnaies selon leur coin.

Notre contribution scientifique dans ce chapitre est que nous proposons une méthode de regroupement de monnaies selon leur coin basé sur le recalage [Horache et al., 2021b].

L'organisation de ce chapitre est le suivant : tout d'abord, nous expliquons la méthode proposée pour regrouper les monnaies selon leur coin. Ensuite, nous voyons comment les experts peuvent exploiter ces résultats. Nous montrons également que la méthode proposée sert à analyser d'autres objets que des monnaies. Finalement, nous allons aborder le sujet de la mise en production de la méthode développée pour que les experts puissent en bénéficier.

4.2 Regroupement des monnaies selon leur coin

Notre objectif est de regrouper les monnaies selon leur coin. Nous avons des nuages de points, c'est-à-dire un ensemble de nuages de points $\mathbf{X}_i, i = 1 \dots N$. Soit c_i un entier qui indique le coin de $\mathbf{X}_i$. On a $c_i \in \mathbb{N}$. Deux monnaies $\mathbf{X}_i, \mathbf{X}_j$ sont frappées par le même coin, si $c_i = c_j$. Notre objectif est donc d'assigner un label c_i pour chaque monnaie i, tel que les monnaies de même coin aient le même label. C'est un problème difficile, car les nuages de points sont volumineux, les motifs sont très fins et la différence de coin est dans certains cas subtile. Une autre difficulté est que comme la base de données ne peut être annotée que par des experts, la quantité de données est limitée. Le regroupement (*clustering*) est différent du problème de classification pour deux raisons. La première est que nous ne connaissons pas le nombre de coins total. La deuxième différence est que le label en soi n'a pas d'importance. Il faut juste que deux monnaies du même coin aient le même label. Si deux monnaies $\mathbf{X}_i$ et $\mathbf{X}_j$ sont de même coin alors, assigner les labels $c_i = c_j = 0$ ou bien $c_i = c_j = 100$ sont tous les deux corrects, tant que $c_i = c_j$. On parle alors de problème de regroupement ou *clustering*. Il faut noter un autre point difficile : le nombre de monnaies par coin peut varier très fortement. La majorité des groupes n'ont qu'une seule monnaie alors que certains groupes comportent des dizaines de monnaies.

Afin de résoudre le problème, nous proposons une chaîne de traitements en trois étapes (voir figure 4.1) :

1. Acquisition des modèles 3D
2. Estimation de similarité pour chaque paire de monnaies
3. Regroupement automatique des pièces de même coin

4.2.1 Acquisition

Nous avions déjà abordé le sujet dans un précédent chapitre. Pour résumer, notre méthode fonctionne avec des modèles 3D de monnaies. Il existe plusieurs technologies pour acquérir un modèle 3D. Nous avons besoin d'un modèle très précis (résolution d'au moins 0,1 mm). Nous avons donc utilisé le scanner SmartScan Hexagon pour obtenir des modèles 3D précis des monnaies.

4.2.2 Estimation de la similarité

Dans cette partie, nous allons voir comment à partir des modèles 3D des monnaies, nous pouvons déterminer si deux monnaies ont été frappées par le même coin. Pour le savoir, nous ali-

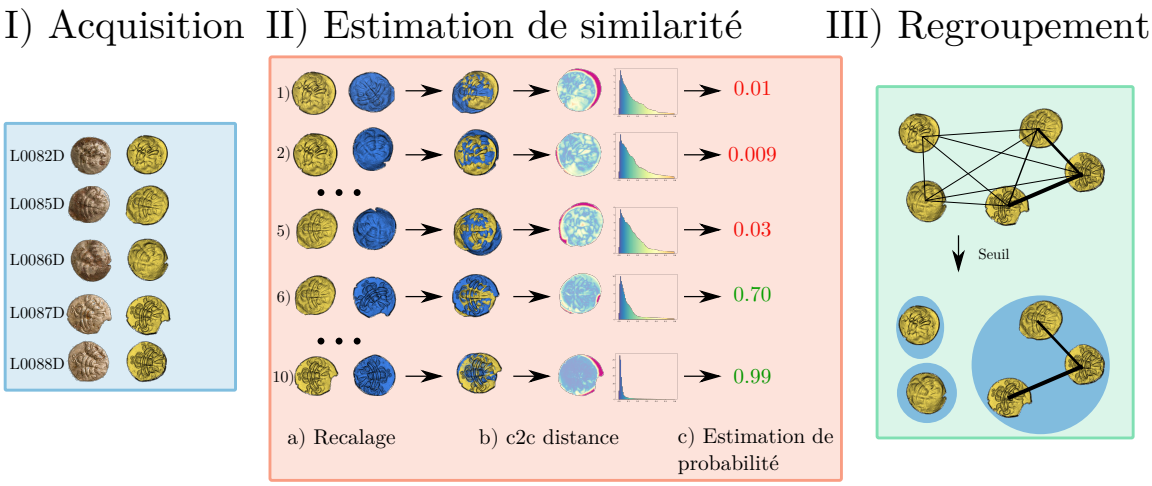


FIGURE 4.1 – Résumé de la méthode proposée : I) Acquisition des données avec un scanner 3D (en bleu). II) estimation de la similarité (en rouge) : l'objectif est d'estimer la probabilité que deux pièces aient été frappées par le même coin. Premièrement, nous alignons les modèles 3D des monnaies avec un algorithme de recalage. Deuxièmement, nous calculons la distance point à point entre les deux nuages de points 3D et l'histogramme des distances. Troisièmement, nous estimons la probabilité que deux pièces soient du même coin à partir de l'histogramme. III) Regroupement (en vert) : nous créons un graphe de correspondances des pièces (le poids des arêtes du graphe correspond à la probabilité que deux pièces soient du même coin), et nous supprimons les arêtes à faible probabilité (représenté par des liens fins). Les groupes (les coins) sont les composantes connexes du graphe après application du seuil.

gnons les motifs des pièces avec un algorithme de recalage, puis nous calculons la distance point à point ainsi que l'histogramme des distances. Ensuite, nous estimons la probabilité que deux pièces soient du même coin à partir de l'histogramme.

4.2.2.1 Alignement des motifs par recalage

Nous avons déjà abordé le problème du recalage dans le chapitre précédent et nous avons expérimenté plusieurs approches. Au début de ce projet, nous n'avions pas de base de données annotée pour pouvoir appliquer l'apprentissage profond. Nous avons donc proposé une variante d'ICP qui fonctionne sur les modèles 3D des monnaies et avec des initialisations aléatoires, nous avons un algorithme de recalage global (appelé RandomSearchICP). Mais nous nous sommes rendu compte que cette approche avait de nombreux défauts. Premièrement, c'est un algorithme lent, surtout lorsque les pièces ne sont pas du même coin (voir tableau 3.2). De plus, cette méthode fonctionne très bien sur les droits et les barbus, mais ne fonctionne pas sur les revers (voir tableau 3.2). Mais cette méthode nous a permis de construire un jeu de données d'entraînement sur les droits. Et avec les données d'entraînements, nous avons pu essayer des méthodes basées sur l'apprentissage profond comme FCGF [Choy et al., 2019] ou DIP [Poiesi and Poiesi, 2021] (voir protocole expérimental dans le chapitre 3). Grâce à ces méthodes qui calculent des descripteurs pertinents et à un estimateur robuste (TEASER [Yang et al., 2020]), nous avons un algorithme de recalage performant sur ces modèles 3D. D'après nos expériences, même si les méthodes basées sur l'apprentissage profond ne sont entraînées que sur les droits, ces méthodes peuvent généraliser sur les revers. DIP généralise moins bien sur les revers que FCGF. Nous avons donc privilégié FCGF dans notre chaîne de traitement. Après le recalage global, nous appliquons un ICP autour des descripteurs (voir le paragraphe 3.3.3.1)

4.2.2.1.1 Et le recalage d'image? Nous l'avons vu dans le chapitre 2, Marchand [Marchand, 2014] propose pour aligner des monnaies d'utiliser le recalage d'image grâce à des méthodes spectrales. Le principe est de transformer les images avec la transformée de Fourier, puis de calculer

l'inter corrélation et enfin d'appliquer la transformée inverse. Cependant, la variation de lumière perturbe le résultat de recalage d'image [Marchand, 2014]. Ces problèmes d'illumination ne sont pas présents dans notre cas, car l'acquisition 3D permet de capturer finement la géométrie. L'approche proposée par Heinecke *et al.* [Heinecke et al., 2021] sur des monnaies romaines est d'appliquer un pré-traitement sur l'image. Tout d'abord, un débruiteur basé sur la variation totale est appliqué, puis une adaptation de contraste par égalisation d'histogramme afin de limiter l'influence de l'illumination et du bruit. Ensuite après application d'un filtre laplacien pour extraire les bords, Heinecke *et al.* [Heinecke et al., 2021] propose de calculer des points d'intérêt basés sur les processus gaussiens. Les descripteurs sont l'histogramme de gradient local. Entre une paire d'images, la distance de Procruste d est calculée pour mesurer la dissimilarité, c'est-à-dire la distance euclidienne minimale entre les points d'intérêts après transformation rigide. Notre approche proposée pour le recalage et l'approche de Heinecke *et al.* [Heinecke et al., 2021] sont toutes les deux des approches basées descripteurs. Cependant, il n'y a pas de garanties que l'approche proposée par Heinecke *et al.* [Heinecke et al., 2021] n'ait pas bénéficiée d'une orientation standard des monnaies. En effet, les monnaies romaines du jeu de données de Heinecke *et al.* [Heinecke et al., 2021] semblent toutes orientées de la même manière. Notre approche avec les *scans* 3D ne suppose pas que les monnaies soient déjà orientées à peu près correctement. Nous nous fixons dans un cadre général où les monnaies sont orientées de manière arbitraire.

4.2.2.2 Distance point à point

Après avoir aligné les motifs, nous voulons connaître un degré de similarité entre les monnaies. Pour cela, nous calculons la distance point à point (ou *cloud to cloud* distance, c2c distance) entre les modèles 3D des monnaies. Si les motifs sont similaires alors la distance point à point entre les nuages de points devrait être faible, même si les motifs sont effacés. Supposons que les nuages de points $\mathbf{X}$ et $\mathbf{Y}$ soient alignés. L'ensemble des distances point à point entre $\mathbf{X}$ et $\mathbf{Y}$ est appelé $\mathbf{D} = \{d_1, d_2 \dots d_{|\mathbf{X}|}\}$ ($\mathbf{D}' = \{d'_1, d'_2 \dots d'_{|\mathbf{Y}|}\}$ pour l'ensemble des distances entre $\mathbf{Y}$ et $\mathbf{X}$) et se calcule ainsi :

$$d_i = \min_{y \in \mathbf{Y}} ||x_i - y||, \quad d'_i = \min_{x \in \mathbf{X}} ||x - y_i|| \tag{4.1}$$

Ensuite, nous calculons l'histogramme $h \in \mathbb{R}^d$ de $\mathbf{D}$ ($h' \in \mathbb{R}^d$ de $\mathbf{D}'$) où d est le nombre d'intervalles (nous l'avons fixé à 70). Les histogrammes sont normalisés (la somme de la surface des rectangles de l'histogramme fait 1). Nous excluons également les distances trop grandes pour le calcul de l'histogramme. Parfois, le recouvrement n'est que partiel et donc même des pièces du même coin vont avoir beaucoup de distances élevées. Pour calculer l'histogramme, nous enlevons donc les distances au-delà de 0,6 mm. Nous faisons finalement la moyenne entre h et h', que nous allons appeler $h^{(moy)}$ afin d'avoir une mesure de similarité symétrique. Si nous comparons l'histogramme pour des pièces de même coin (figure 4.2) et l'histogramme pour des pièces de coins différents (figure 4.3), nous pouvons voir que les histogrammes n'ont pas la même forme. Les spécialistes en numismatique peuvent se baser sur l'histogramme pour savoir si deux pièces sont du même coin. Si l'histogramme montre que l'écart type des distances point à point est faible et que la fréquence des distances point à point faible est majoritaire, alors il est très probable que les monnaies soient du même coin. En revanche, un histogramme montrant une grande variation de distances point à point signifie que, soit les monnaies sont d'un coin différent, soit le recalage a échoué (voir la figure 4.3). Ce dernier point montre l'importance du recalage lorsque les pièces sont du même coin. Mais la question est de savoir s'il est possible automatiquement de déterminer si deux pièces sont du même coin à partir de l'histogramme.

4.2.2.3 Calcul de similarité à partir des histogrammes

Afin de déterminer si deux pièces sont du même coin ou non à partir de l'histogramme, nous proposons de modéliser ce problème comme un problème d'apprentissage automatique. Plus

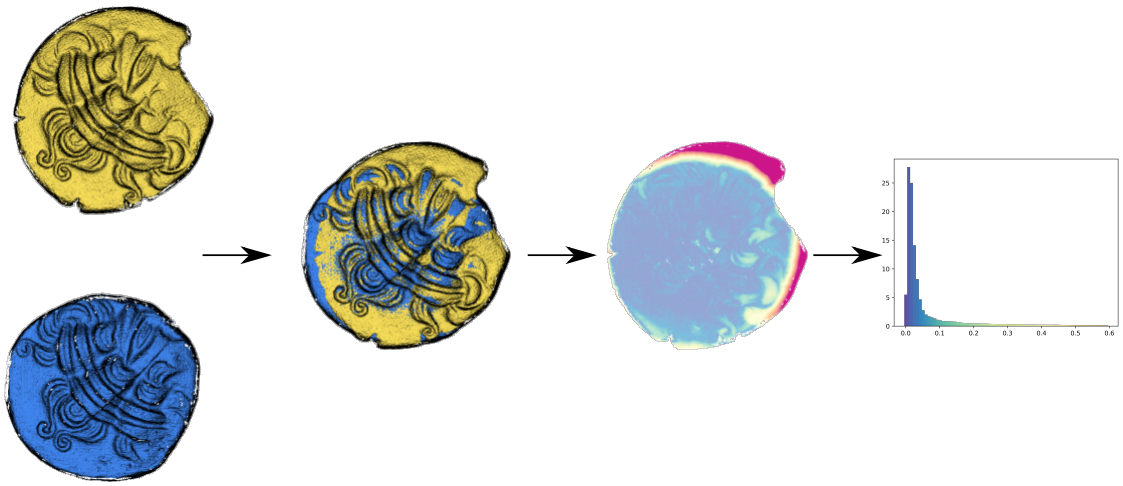


FIGURE 4.2 – Calcul de l'histogramme après alignement des monnaies de même coin. La distance est représentée par une couleur. Proche du bleu, cela veut dire que la distance est faible (environ 0,1 mm). En jaune, la distance est plus grande (environ 0,4 mm) et le rouge veut dire que la distance est très grande (environ 1 mm).

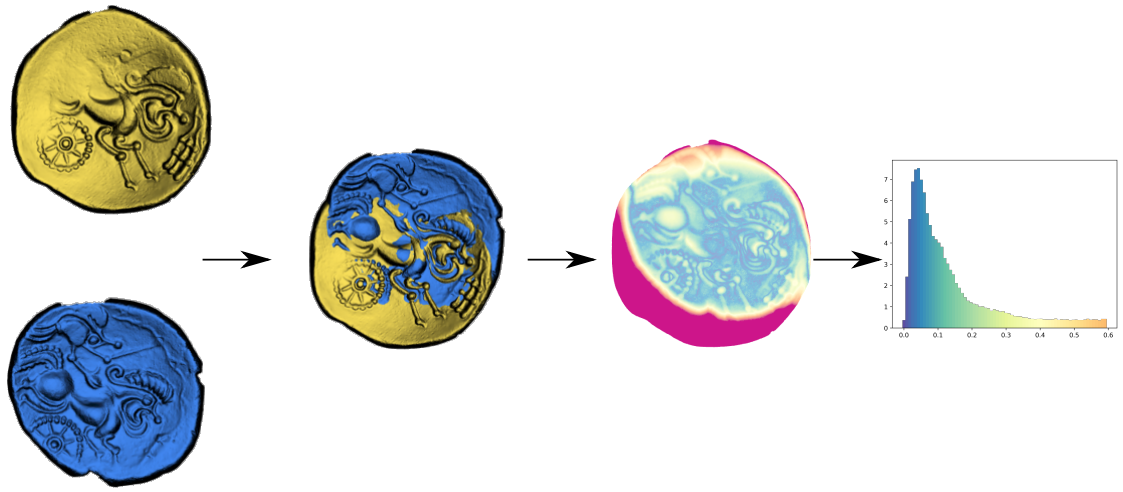


FIGURE 4.3 – Calcul de l'histogramme après alignement des monnaies de coins différents. La distance est représentée par une couleur. Proche du bleu, cela veut dire que la distance est faible (environ 0,1 mm). En jaune, la distance est plus grande (environ 0,4 mm) et le rouge veut dire que la distance est très grande (environ 1 mm).

spécifiquement, nous modélisons le problème comme un problème de classification binaire. L'algorithme d'apprentissage automatique va nous donner en sortie une probabilité que deux motifs soient similaires. L'algorithme utilisé est la régression logistique. Soit $h_1 \dots h_K$ les histogrammes de paires de monnaies (vecteur de dimension d) et soit $l_1 \dots l_K$ les labels (1 si les monnaies sont du même coin, 0 sinon). L'objectif est d'estimer les paramètres $\theta \in \mathbb{R}^d$ et $b \in \mathbb{R}$ tels que :

$$\theta^*, b^* = \underset{\theta \in \mathbb{R}^d, b \in \mathbb{R}}{\arg\min} \sum_{i=1}^{K} (1 - l_i)(\theta^T h_i + b) + \log(1 + \exp(-(\theta^T h_i + b)) + \|\theta\|^2 \tag{4.2}$$

L'objectif peut être minimisé par une descente de gradient ou une méthode de Newton. Après l'évaluation des bons paramètres à partir d'un jeu de données d'entraînement, nous pouvons facilement et rapidement évaluer la probabilité que deux motifs soient similaires. Pour l'entraînement, nous avons utilisé les 200 droits annotés (le même jeu de données qui est utilisé pour apprendre les descripteurs pour le recalage).

Pour chaque paire, l'estimation de la similarité (recalage + distance point à point + estimation de la probabilité) prend environ 4,4s (voir tableau 4.1).

	Recalage	**Estimation de la probabilité**	**Total**
Temps (en s)	3.8	0.6	4.4

TABLEAU 4.1 – Temps (en s) pour chaque opération de l'estimation de la similarité (pour une seule paire). Le recalage utilise FCGF et est calculé sur GPU. Les autres opérations sont calculées sur le processeur. Chaque paire peut être traitée indépendamment.

4.2.3 Regroupement des monnaies selon leur coin

Nous avons décrit une manière de calculer la probabilité que deux pièces soient du même coin. Avec ces comparaisons deux à deux, nous pouvons regrouper les pièces en différents groupes qui correspondent à leur coin. Soit $\mathbf{X}_1, \dots \mathbf{X}_n$ les monnaies scannées. Soit p_{ij} la probabilité que $\mathbf{X}_i$ et $\mathbf{X}_j$ soient du même coin. Cette probabilité est calculée par la méthode décrite précédemment. Nous avons $p_{ij} = p_{ji}$. $P = (p_{ij})_{i=1\dots n, j=1\dots n}$ est la matrice des comparaisons par paire. P est une matrice carrée et symétrique. La matrice P peut être vue comme la représentation matricielle d'un graphe pondéré indirect et complet. Chaque nœud représente une pièce et une arête représente la probabilité que les deux pièces soient du même coin ou non. Pour réaliser le regroupement, nous supprimons les arêtes avec une faible probabilité. Soit A la version du graphe P non pondéré (après application du seuil) représenté sous forme matricielle. A se calcule ainsi :

$$A_{ij} = \mathbb{1}_{p_{ij} \geq \alpha} = \begin{cases} 1 & p_{ij} \geq \alpha \\ 0 & p_{ij} < \alpha \end{cases} \tag{4.3}$$

α est le seuil à fixer.

Pour ensuite avoir les groupes de pièces du même coin, il suffit de calculer les composantes connexes de A. Ce seuil est assez difficile à fixer et nécessite le retour de l'expert. Si l'expert veut limiter le nombre de faux positifs (des monnaies de coins différents, mais des monnaies classées en même coin par l'algorithme), alors il faut un seuil élevé. Au contraire, pour limiter le nombre de faux négatif (monnaies de même coin classées en des coins différents par l'algorithme), il faut un seuil bas. Il peut être intéressant de fixer un seuil bas, même si le nombre de faux positifs va augmenter. En effet, il peut arriver que des coins différents aient des parties similaires. Ces paires peuvent être détectées comme de faux positifs, mais ils nous renseignent sur la manière dont les coins ont été fabriqués (voir paragraphe 4.3.1.4).

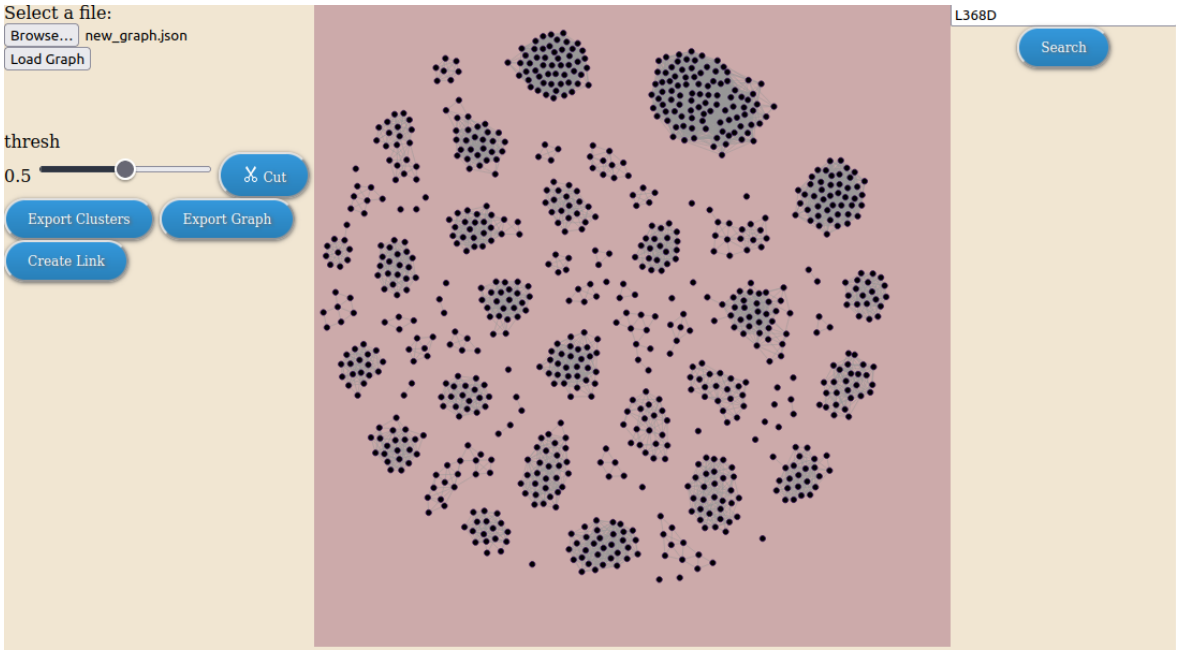


FIGURE 4.4 – Graphe interactif de similarité des monnaies (regroupé par coin). L'utilisateur peut rapidement et sans effort, ajouter une nouvelle arête, couper une mauvaise arête. L'utilisateur peut aussi sauvegarder le nouveau graphe et exporter les groupes.

4.2.4 Amélioration du regroupement par correction manuelle

Une expertise est nécessaire pour vérifier si les monnaies ont correctement été regroupées ou non. Comme nous allons voir dans la sous-section 4.2.5, la méthode automatique de regroupement proposée n'est pas fiable à 100%. Heureusement, le graphe de similarité entre les monnaies est un outil très intéressant pour les numismates. Surtout, le graphe A pour différents seuils α permet de rapidement détecter les paires problématiques. En effet, la vérification est beaucoup plus rapide et facile que la reconnaissance de coin en elle-même. Nous avons donc implémenté un graphe interactif (voir figure 4.4) où l'utilisateur peut rechercher une monnaie dans le graphe, visualiser les liens du graphe, et surtout changer le seuil. Cela permet de visualiser plusieurs seuils en un seul coup d'œil. L'utilisateur peut donc rapidement repérer les paires problématiques. Notre implémentation n'est pas que pour la visualisation. Il est également utile pour l'édition du graphe. L'utilisateur peut ajouter, supprimer des arêtes, sauvegarder le nouveau graphe ou charger un autre graphe. L'expert peut modifier le graphe et donc rapidement corriger une erreur de regroupement.

4.2.5 Expérience

Nous présentons le jeu de données et une première méthode pour le regroupement selon le coin sur Riedones3D. Pour cette tâche, nous utilisons plus de données que le recalage pour l'évaluation. Pour les droits, le jeu de données d'entraînement a 418 monnaies (dont 200 sont utilisées pour le recalage). Le jeu de validation est de 181 pièces et le jeu de données test est de 288 monnaies. Pour les revers, le jeu de données test a 293 monnaies. Les revers et les barbus sont seulement utilisés pour le jeu de données test.

La figure 4.5 montre une vue synthétique du jeu de données test pour le regroupement de monnaies selon leur coin. Ce schéma montre le nombre de monnaies par coin. Il montre que le nombre de monnaies par coin est très déséquilibré. Il y a des coins avec peu de monnaies et des coins avec énormément de monnaies. En effet, certains coins ne donnent pas le même nombre de monnaies, certains sont plus résistants et d'autres sont plus fragiles. Par ailleurs, il faut noter que plus les monnaies ont circulé, plus leur diffusion est grande. Par conséquent, les chances de trouver des monnaies frappées avec le même coin sont plus minces. Pour le jeu de données Rie-

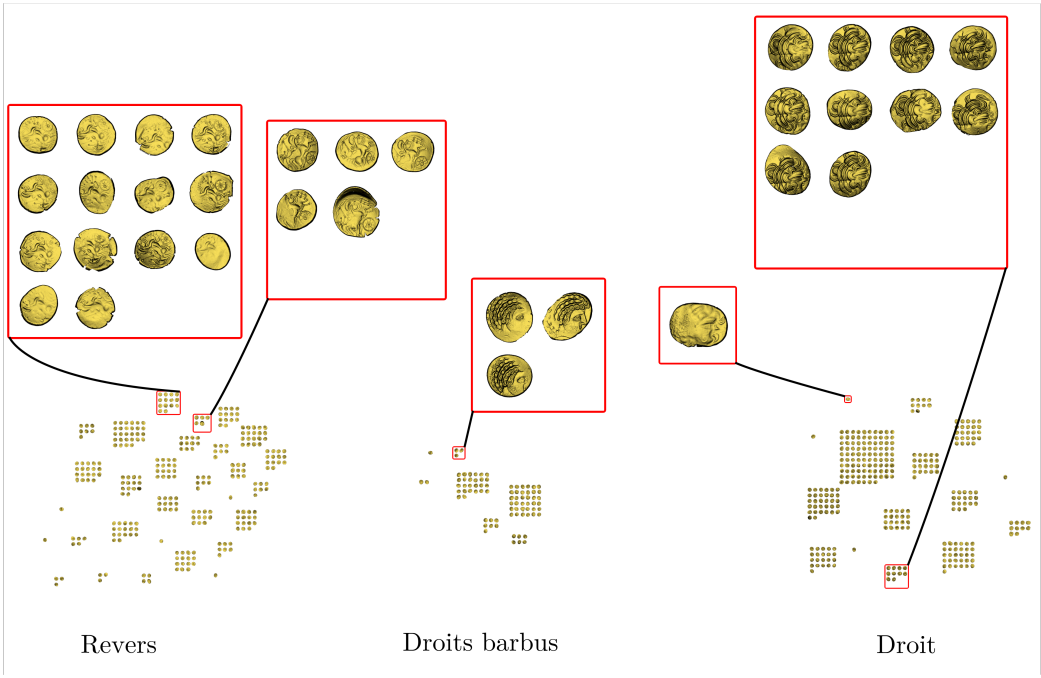


FIGURE 4.5 – Vue synthétique du jeu de données test de Riedones3D pour le regroupement de monnaies par coins (30 coins pour les revers, 7 pour les barbus et 14 pour les droits).

dones3D, nous constatons un grand nombre de monnaies frappées avec le même coin. Donc, ces monnaies sont probablement des prélèvements à la source.

Nous proposons une méthode pour le regroupement des monnaies, basée sur la méthode décrite précédemment (voir figure 4.1). Pour le recalage, nous utilisons FCGF [Choy et al., 2019] entraîné uniquement sur les droits. Après l'alignement, nous calculons la similarité entre les monnaies à partir de l'histogramme des distances entre les paires de monnaies (la régression logistique n'est entraînée que sur les droits). Ensuite, nous construisons le graphe des similarités pour extraire les groupes de monnaies classés selon leur coin. Le seuil α (expliqué plus haut) est fixé avec le jeu de données de validation des droits. Nous avons trouvé $\alpha = 0.95$. Pour évaluer le regroupement par rapport à une vérité terrain construite à la main, nous utilisons l'indice Fowlkes-Mallows (Fowlkes-Mallows Index ou FMI) [Fowlkes and Mallows, 1983]. Le FMI est défini comme :

$$\text{FMI} = \frac{\text{TP}}{\sqrt{(\text{TP}+\text{FP})(\text{TP}+\text{FN})}} \tag{4.4}$$

TP est le nombre de vrais positifs (nombre de paires de monnaies qui sont classées du même coin par l'algorithme proposé et qui sont effectivement du même coin). TN est le nombre de vrais négatifs (le nombre de paires de monnaies qui sont classées de coins différents et qui sont effectivement de coins différents). FP est le nombre de faux positifs (le nombre de paires de monnaies qui sont classées du même coin, mais qui sont de coins différents). FN est le nombre de faux négatifs (le nombre de paires de monnaies qui sont classées de coins différents, mais qui sont de même coin).

Nous rapportons également l'indice de Rand ajusté (Adjusted Rand Index ou ARI) [Hubert and Arabie, 1985] :

$$\text{RI} = \frac{\text{TP}+\text{TN}}{\text{TP}+\text{FP}+\text{TN}+\text{FN}} \tag{4.5}$$

$$\text{ARI} = \frac{\text{RI} - \text{E}[\text{RI}]}{\max(\text{RI}) - \text{E}[\text{RI}]} \tag{4.6}$$

RI est l'indice de Rand (TP, TN, FP, et FN sont définis ci-dessus). E[RI] est l'espérance de l'indice de Rand pour des labels fixés aléatoirement, et max(RI) est l'indice maximum que l'on peut avoir.

Ces deux métriques sont classiques pour l'évaluation du *clustering*. Le tableau 4.2 montre que le ARI et le FMI donnent des résultats quasiment similaires sur le jeu de données test de Riedones3D. Les résultats sont satisfaisants pour les droits, mais pour les revers, il y a encore des erreurs de regroupement ARI = 0.86. Il y a plusieurs explications possibles à cette différence de résultats. Les résultats du recalage sur les revers sont un peu moins bons que sur les droits. Une autre explication est que les revers sont en moins bon état que les droits. En effet, les coins de revers sont plus fragiles que les droits. C'est pour cela qu'il y a plus de coins de revers que de droits. En comparaison, l'approche de Heinecke *et al.* [Heinecke et al., 2021] obtient un score ARI = 0.73 pour les droits sur des monnaies romaines (en supprimant les images dupliquées).

	Revers	Barbus	Droits
FMI	0.87	0.99	0.98
ARI	0.86	0.99	0.97

TABLEAU 4.2 – Évaluation du regroupement par coin en utilisant les métriques FMI et ARI sur le jeu de données test de Riedones3D (l'entraînement et la validation sont faits que sur les droits).

Pour le regroupement des droits sans barbe, nous avons obtenu 16 groupes avec la méthode proposée, alors qu'il y a 14 groupes vérité terrain. Pour le regroupement des barbus, il y a seulement une erreur : une pièce n'est pas associée au bon coin. Pour le regroupement des revers, nous obtenons 35 groupes (il y en a 30 dans la vérité terrain). On peut conclure que la méthode peut identifier les groupes principaux, mais certaines monnaies ne vont pas être correctement classifiées.

4.3 Exploitation des résultats

Dans cette partie, nous allons passer en revue les différentes manières d'exploiter les résultats sur les monnaies. Nous allons analyser les échecs de notre système et proposer des pistes de solution pour les différents échecs. Nous allons aborder l'application de la reconstruction de coins. Et enfin, nous allons voir comment grâce à nos résultats nous pouvons déterminer comment estimer le nombre de monnaies qui ont circulé à l'époque.

4.3.1 Discussions des cas limites de l'outil proposé

Nous avons vu que pour les revers notamment, certains groupes de coins n'étaient pas correctement identifiés. Notre système comporte plusieurs briques et chaque brique peut produire une erreur. Le recalage peut échouer ce qui peut provoquer des erreurs dans le regroupement. De plus, même si le recalage réussit, la similarité peut également être mal estimée pour plusieurs raisons. Les cas que nous allons voir sont plus l'exception que la règle, mais il est important de prendre en compte ces exceptions pour avoir le système le plus fiable possible.

4.3.1.1 Que faire lorsque les monnaies présentent des déformations importantes ?

Les monnaies peuvent être légèrement concaves ou convexes, dû à l'emboutissage lors de la frappe au marteau. cette déformation peut poser problème pour la comparaison, l'hypothèse du recalage rigide n'est plus valide. Par conséquent, la distance point à point va être biaisée (voir figure 4.6). Les motifs ne pourront donc jamais être alignés. Cette erreur vient de l'hypothèse qu'on a faite depuis le début : il n'y a pas de déformation de motifs entre deux monnaies du même coin.

Une solution est de «déformer virtuellement» les modèles 3D afin d'avoir des structures réellement plates. Supposons que nous voulons «déformer» $\mathbf{X} = \{x_1 = (u_1, v_1, w_1) \in \mathbb{R}^3, \ldots, x_n = (u_n, v_n, w_n) \in \mathbb{R}^3\}$. Comme les objets que nous voulons déformer sont planaires, l'objectif est de modifier la coordonnée z en considérant la monnaie alignée sur le plan xy.

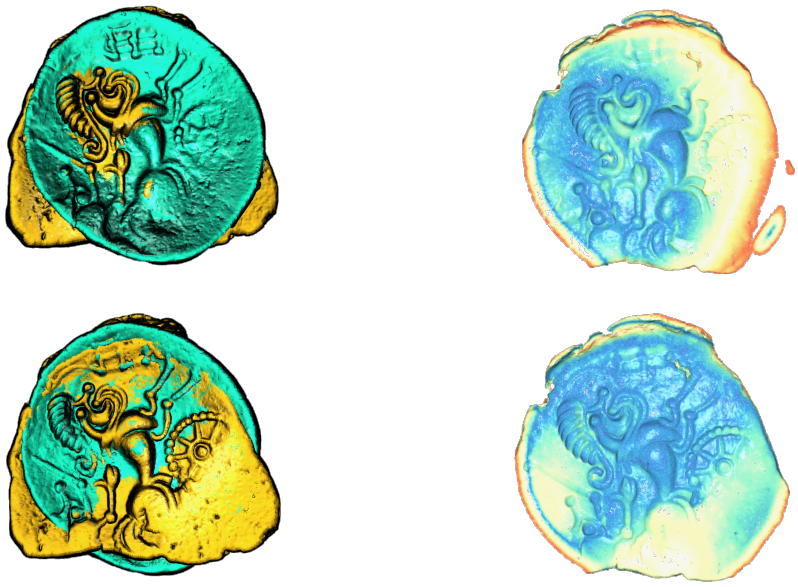


FIGURE 4.6 – Recalage sans tordre la monnaie (première ligne) et recalage en tordant la monnaie (deuxième ligne). La première colonne est le résultat du recalage, la deuxième colonne est la distance point à point. L'estimation s'est faite avec une régression polynomiale ($d = 2$)

.

Nous cherchons donc à régresser z avec $h_\theta : \mathbb{R} \to \mathbb{R}$. h_θ prend en entrée les coordonnées x et y et en sortie, estime la coordonnée z. Si la torsion est suffisamment faible, nous pouvons déformer la monnaie de cette manière.

$$\mathbf{X}' = \{x_1 = (u_1, v_1, w_1 - h_\theta(u_1, v_1)) \in \mathbb{R}^3, \ldots, x_n = (u_n, v_n, w_n - h_\theta(u_n, v_n)) \in \mathbb{R}^3\}. \tag{4.7}$$

$\mathbf{X}'$ est la monnaie déformée. Dit autrement, le principe consiste à estimer une forme globale de la monnaie sans les motifs. Réajuster la hauteur va aplatir la monnaie. Il existe beaucoup de choix pour h_θ, une solution consiste à prendre un modèle polynomial de degré d. Le principe est de trouver les paramètres θ qui minimisent cet objectif :

$$\arg\min_\theta \sum_{i=1}^{n} (w_i - \sum_{k=0}^{d} \sum_{l=0}^{d} \theta_{kl} u_i^k v_i^l) \tag{4.8}$$

Le problème peut s'exprimer comme un problème linéaire et il est possible de régulariser la fonction objective (Ridge, Lasso). Nous pouvons observer en figure 4.6, deux monnaies du même coin (L0027R et L0066R). En regardant le résultat du recalage, nous observons que les motifs sont alignés. Pourtant, nous pouvons voir qu'une des pièces est concave et donc l'algorithme ne les classifie pas du même coin. En appliquant la correction proposée pour «déformer» les monnaies, les résultats sont déjà plus satisfaisants.

Nous pouvons visualiser la fonction polynomiale estimée. C'est une idée très prometteuse, mais plus d'expériences sont à mener. Pour des travaux futurs, il faudrait tester sur plus d'exemples, tester d'autres modèles de régressions comme les Processus gaussiens ou les réseaux de neurones.

4.3.1.2 Que faire lorsque les monnaies sont trop usées ?

Si nous regardons le graphe des similarités dans un grand groupe, nous pouvons voir que beaucoup de monnaies vont être classées comme des coins différents, mais sont du même coin. Prenons par exemple le graphe montré à la figure 4.7. Sur ce graphe, les monnaies de même coin

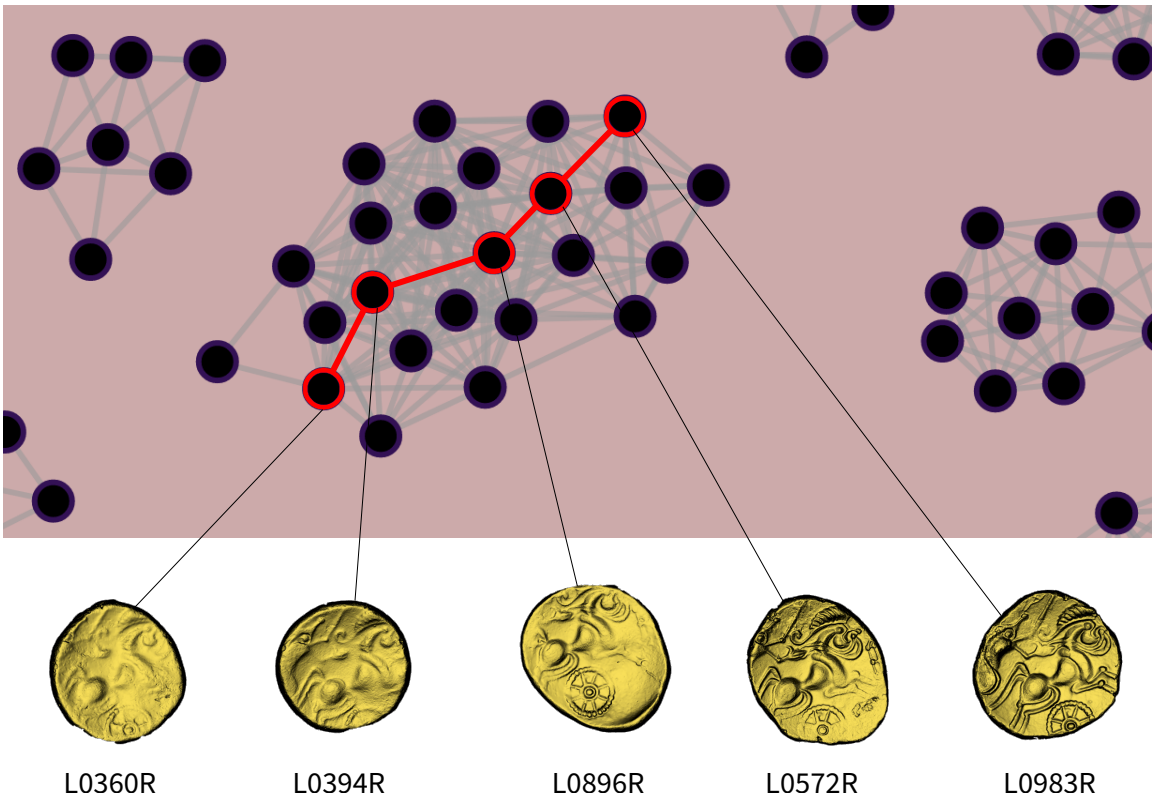


FIGURE 4.7 – Évolution de l'usure des monnaies visible sur le graphe de correspondance des monnaies. Les liens en gris indique si deux monnaies sont classées comme appartenant au même coin. En rouge, nous avons tracé un chemin du graphe qui montre l'usure successif des monnaies

sont liés par un lien en gris. Deux monnaies non liées sont dites de coins différentes par la méthode proposée. La monnaie L0394R et la monnaie L0983R sont classées comme de coin différent, car la monnaie L0983R est très usée (voir figure 4.7). Ces deux monnaies ont finalement été classées dans le même groupe parce qu'il y a des monnaies intermédiaires (la L0896R et la L0572R) qui font le lien entre la L0394R et la L0983R (ce lien est tracé en rouge dans la figure 4.7). Le fait que des monnaies usées et des monnaies non usées ne soient pas en correspondance à 100% offre un avantage, car, pour les experts, il est intéressant de pouvoir identifier cet usure. Ainsi, l'expert peut établir une chronologie relative entre les monnaies du même coin.

Cependant, le revers de la médaille est que si un coin est peu représenté, la méthode risque de ne pas pouvoir regrouper les monnaies ensemble, surtout si une monnaie est très usée par rapport à d'autres. Cependant, l'outil de visualisation permet de limiter le problème. En baissant le seuil de probabilité, nous augmentons le nombre de faux positifs (monnaies de coins différents mises dans le même groupe).

4.3.1.3 Comment gérer les erreurs de recalage?

Nous avons vu que les résultats de recalage ne sont pas 100% fiables comme nous l'avons vu dans le chapitre 3. Ce problème va avoir tendance à créer des faux négatifs (monnaies de mêmes coins classés comme des coins différents). Ce problème est très rare pour les groupes contenant beaucoup de monnaies. Pour les groupes contenant peu de monnaies, ce problème est moins rare (voir la sous-section 4.2.5) et est plus difficile à détecter.

Il peut arriver que ce problème de recalage crée de faux positifs (monnaies de coins différents classés comme du même coin). En effet, il se peut que seule une toute petite partie de la monnaie soit recalée avec une autre petite partie. Lorsque le recouvrement est très faible, l'histogramme contient surtout de faibles valeurs et donc les monnaies vont être classées comme faisant partie du même coin... Le problème vient du fait que lors du calcul de l'histogramme, les grandes distances sont exclues. Heureusement, ces paires sont faciles à exclure également et le cas est rare. De plus, ces cas sont très faciles à détecter grâce à l'outil de visualisation de graphe proposé.

4.3.1.4 Coins différents avec une partie du motif du même coin

Et si des coins différents avaient certains motifs de même poinçon? Prenons par exemple la classe des revers. Pour certaines classes, on peut voir que les corps des chevaux sont très similaires. Une question qu'un numismate peut se poser est donc : est-ce que lors de la fabrication des coins, le même poinçon a été utilisé pour fabriquer le corps du cheval? Cette hypothèse n'est pas absurde, car comme nous l'avons vu précédemment, il y a beaucoup plus de coins de revers que de droits. Il faut donc fabriquer plus de coins de revers parce que les coins de revers s'usent plus vite que les coins de droits. Mais si un même outil a été utilisé pour fabriquer les coins de revers, cela risque de perturber notre algorithme. Certains seront classés comme faux positifs. Nous pouvons voir en figure 4.8 que des monnaies de coins différents peuvent avoir des parties qui sont similaires. Dans cet exemple, les monnaies sont bien conservées donc la méthode proposée différencie bien les monnaies de coins différents, mais dans le cas où les monnaies ne sont pas bien conservées, la méthode proposée va générer un faux positif. La solution adoptée est de choisir un seuil élevé pour la construction du graphe puis de baisser graduellement le seuil. Avec le seuil élevé, nous pouvons supprimer ces faux positifs. Mais, un seuil élevé rajoute beaucoup de faux négatifs. En baissant donc graduellement le seuil, on baisse le nombre de faux négatifs et on crée quelques faux positifs. Nous pouvons les supprimer facilement car ces faux positifs sont rares et peuvent facilement se voir sur le graphe. Cette solution demande toujours l'intervention d'un humain. Cependant,

— Vérifier est beaucoup plus rapide que classer

— Un non-expert peut rapidement conclure de la similarité entre coins grâce au recalage

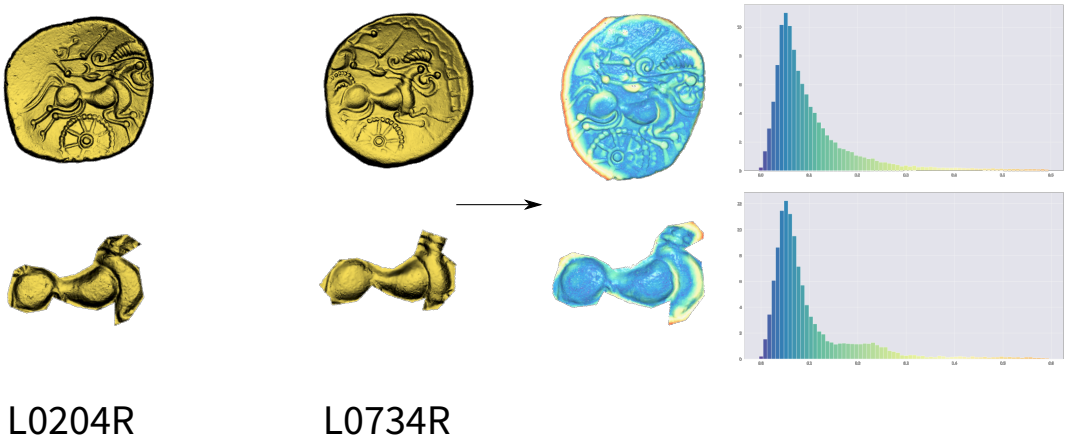


FIGURE 4.8 – Comparaison d'une paire de monnaie de coins différents et comparaison d'une partie de la monnaie (le corps du cheval). La comparaison se fait avec la distance point à point et l'histogramme des distances. La méthode proposée classifie les deux monnaies comme étant de coins différents (probabilité de même coin 3%), mais les corps des chevaux sont similaires (probabilité de 72%).

4.3.1.5 Conclusion

La chaîne de traitement proposée a des limites, nous en avons vu certaines ainsi que des pistes d'améliorations. De plus, dans une approche semi-automatique avec une étape de vérification grâce à l'outil de visualisation, ces erreurs sont limitées. L'analyse des cas extrêmes montre que le système proposé permet de faire des découvertes sur le coin comme le mode de gravure, le recours à des poinçons partiels, des accidents de coins, des regravures, etc.

4.3.2 Reconstruction des coins

Sur une monnaie, le motif va toujours être partiel, le coin est toujours plus grand que la monnaie. Avec les modèles 3D, les numismates souhaitent reconstruire l'image complète du coin. Cette reconstruction peut avoir trois applications :

— C'est un document de travail pour les numismates pour être capable d'analyser plus en détail le style du graveur.

— C'est une ressource exploitable par les musées pour initier le grand public à la numismatique.

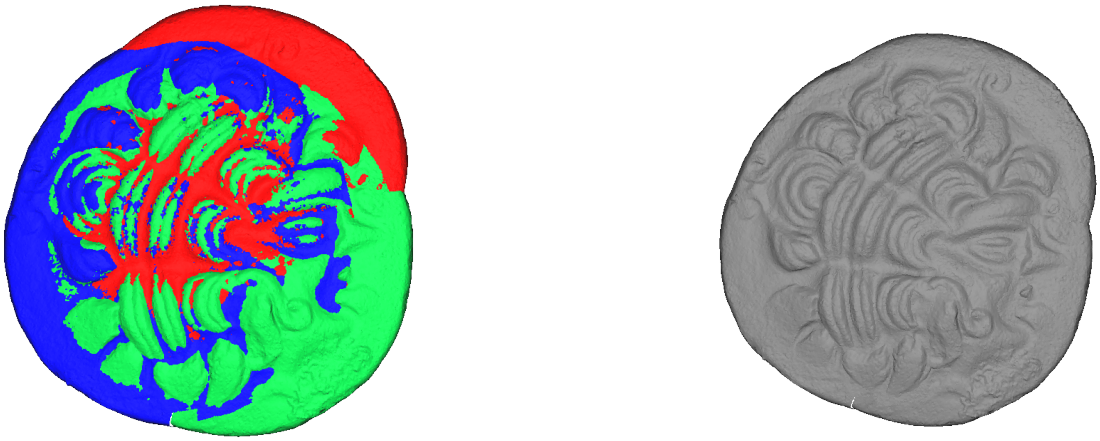
— Pour la reconnaissance automatique, au lieu de comparer deux monnaies, il est possible de comparer une monnaie et un coin, voir deux coins entre eux.

Le dernier point est crucial si nous souhaitons traiter des trésors plus volumineux. En effet, la complexité du regroupement de monnaies est quadratique. Si au lieu d'un groupe de monnaie, nous avons un coin, il y a moins de comparaison à faire. Nous réduisons donc la complexité.

Avec ce regroupement par coin ainsi que le recalage entre les monnaies, l'intérêt est de pouvoir reconstruire les coins. Avoir les modèles 3D déjà alignés aide à la reconstruction du coin en entier.

4.3.2.1 Peut-on reconstruire le coin automatiquement ?

La reconstruction automatique a des enjeux importants pour la mise à l'échelle. En effet, pour un trésor de 5000 monnaies, notre méthode prendrait beaucoup de temps (environ 175 jours). Pour 1000 monnaies, il faut 7 jours environ. Pour accélérer les calculs, il est possible de diviser le trésor en petit groupe et de créer des coins intermédiaires. Il y aurait moins de comparaisons à faire. Mais il faut être capable de reconstituer des coins automatiquement. La reconstruction 3D de coin est finalement aussi un problème de recalage (recalage multiple). Cependant, il y a une



(a) Trois monnaies de couleur différente pour la reconstitution

(b) Reconstitution du coin

FIGURE 4.9 – Reconstitution d'un coin de droit avec trois monnaies. Crédit : Caroline Plumel (AOROC-ENS), outil : CloudCompare [Cloud Compare, 2013]

contrainte supplémentaire. L'expert souhaite garder la partie du motif la mieux conservée. Une question ouverte serait donc de pouvoir détecter les défauts et les parties de la monnaie intactes. Mais pour résoudre ce problème, si nous utilisons des méthodes basées sur l'apprentissage, nous aurons besoin d'annoter les monnaies. Il y a également le problème des monnaies déformées. C'est le même problème que nous avons vu précédemment et les mêmes solutions peuvent être envisagées.

4.3.2.2 Reconstruction manuelle

La reconstruction manuelle pour le moment est nécessaire. La méthode proposée nous donne un groupe de monnaies alignées et de même coin. Cependant, les experts veulent uniquement garder les parties des monnaies les mieux conservées. L'utilisateur va donc sélectionner les monnaies les moins abîmées et enlever les parties abîmées manuellement. Nous pouvons voir un exemple de reconstruction de coin en figure 4.9 Cette reconstruction va permettre de créer une base de données d'annotation de défaut de monnaies. En effet, en comparant un coin et une monnaie abîmée, il sera sans doute possible de détecter les défauts (usure, cassure, etc.). S'il est possible de détecter les défauts automatiquement, il est possible de reconstruire le coin sans les défauts. Cependant, la détection de défauts est un autre sujet, qui nécessite de constituer une base de données d'évaluation et aussi d'entraînement. Cela pourra être envisagé dans des travaux futurs. Après constitution du jeu de données d'entraînement, il sera possible d'entraîner un algorithme d'apprentissage automatique, et donc de pouvoir reconstruire automatiquement et virtuellement des coins.

4.3.3 Nombre de coins ayant circulé

Une application de la méthode proposée est de pouvoir estimer le nombre de coins et par conséquent, estimer le nombre de monnaies qui ont circulé. Pour cela, il faut faire appel à des méthodes statistiques. Contrairement aux civilisations grecques ou romaines, les Celtes nous ont laissé très peu de traces écrites. Il est donc beaucoup plus difficile d'étudier l'histoire économique de ces derniers. Ces estimations du nombre de monnaies ayant circulé sont donc des données précieuses pour les spécialistes des celtes. Nous allons nous baser sur les travaux d'Esty [Esty, 1986] pour estimer le nombre de coins. Particulièrement, nous allons nous baser sur les méthodes qui ne font pas l'hypothèse qu'un coin produit le même nombre de monnaies (nous observons que le nombre de monnaies par coin peut être très variable). Le problème d'estimation du nombre de

coins ressemble beaucoup aux problèmes d'estimation du nombre d'espèces inconnus en biologie. Si nous connaissons le nombre de coins ayant circulé et si nous savons combien de monnaies un coin fabrique, nous pouvons estimer le nombre de coins qui ont circulé. Tout d'abord, définissons quelques variables. Soit n le nombre de monnaies ayant circulé. Soit n_{obs}, le nombre de monnaies observées et n', le nombre de monnaies frappé par tous les coins observés. On appelle d' le nombre total de coin et d_{obs} le nombre de coins observés. Soit D_i le nombre de coins vus par i monnaies. Dans le cas de Riedones3D [1], nous avons $n_{obs}^{(droits)} = 1106$, $n_{obs}^{(revers)} = 1107$ $d_{obs}^{(droits)} = 83$, $d_{obs}^{(revers)} = 199, \mathrm{D}_1^{(droits)} = 22, \mathrm{D}_2^{(droits)} = 10, \mathrm{D}_1^{(revers)} = 82, \mathrm{D}_2^{(revers)} = 22$. Notre objectif est d'estimer d'.

Un concept intéressant pour les numismates est celui de couverture C, car la couverture peut nous permettre d'estimer d' [Esty, 1986]. Il s'agit de la part de la production totale réalisée à partir des coins attestés dans l'échantillon observé. Dit autrement :

$$\mathrm{C} = \frac{n'}{n} \tag{4.9}$$

Estimer C est compliqué, car on ne connaît ni n' ni n. Heureusement, C peut être estimé avec la méthode de Good [Good, 1953].

Méthode de Good La méthode de Good [Good, 1953] a l'avantage d'être simple pour estimer la couverture de coins. L'hypothèse est que nous avons un échantillon aléatoire du nombre de monnaie émise. Le principe est d'estimer la couverture à partir des monnaies représentées par un seul coin. L'estimation de C appelé $\hat{\mathrm{C}}$ est définie (avec un intervalle à 95%) :

$$\hat{\mathrm{C}} = 1 - \frac{\mathrm{D}_1}{n_{obs}} \pm \frac{2}{n_{obs}} \sqrt{\mathrm{D}_1 + 2\mathrm{D}_2 - \frac{\mathrm{D}_1^2}{n_{obs}}} \tag{4.10}$$

Avec la couverture, une estimation du nombre de coins est possible avec cette formule

$$d_{good} = \frac{d_{obs}}{\hat{\mathrm{C}}} \tag{4.11}$$

Avant d'être appliqué sur les monnaies, la méthode de Good était appliquée pour estimer la taille d'une population d'espèce. D'après De Callatay [De Callatay, 1993], la méthode de Good est la plus utilisée par les numismates.

Méthode de Carter Cette méthode n'utilise pas la couverture pour estimer le nombre de coins. La méthode de Carter fait l'hypothèse que la durée de vie d'un coin suit une loi gamma [Esty, 1986]. Si nous avons $n_{obs} > 3d_{obs}$, nous pouvons estimer d_{carter} de la manière suivante :

$$d_{carter} = \frac{n_{obs} d_{obs}}{1.069 n_{obs} - 0.843 d_{obs}} \tag{4.12}$$

Il existe d'autres méthodes, dont beaucoup font l'hypothèse que le nombre de monnaies par coin est égal. Pour le jeu de données Riedones3D, il est très probable que le nombre de monnaies par coin soit inégal. Par conséquent, cet hypothèse ne semble pas valide pour Riedones3D.

Applications et discussions Le tableau 4.3 montre les résultats de ces méthodes statistiques pour estimer le nombre total de coins. Comme nous avons un trésor volumineux, le taux de couverture est élevé pour les droits et les revers. Par conséquent, la plupart des coins de ce monnayage sont représentés dans le jeu de données Riedones3D. Nous pouvons voir que l'estimation du nombre de coins dans l'échantillon est importante pour la méthode de Carter. Et dans la méthode de Good, il est encore plus important de trouver le bon nombre de monnaies représenté par un seul coin.

1. Ici, nous prenons les chiffres totaux et non ceux du jeu de données d'évaluation pour le recalage et le regroupement de coins.

	Droits	Revers
C_{good}	98,0%	92,9%
d_{good}	84,7	214,9
d_{carter}	82,5	216,1
d_{obs}	83	199

TABLEAU 4.3 – Estimation du nombre total de coins avec la méthode de Good et de Carter. Le nombre de coins estimé est proche du nombre de coins du trésor (taux de couverture très élevé).

Par conséquent, si nous souhaitons estimer le nombre de coins du monnayage, il faut limiter les faux positifs et les faux négatifs sur le nombre de coins de l'échantillon. Particulièrement, trop de faux négatifs peuvent biaiser les résultats, car ils créent beaucoup de coins isolés. Cela montre que la fiabilité de l'algorithme est très important. Particulièrement, il est important de pouvoir estimer correctement le bon nombre de coin dans l'échantillon, quitte à faire une correction manuelle ensuite. Dans le cas de Riedones3D, on peut noter que même si les méthodes ont des formules très différentes, les nombres de coins estimés restent assez proches.

4.4 Applications à d'autres objets

Les travaux réalisés sur les monnaies peuvent-ils s'appliquer à l'analyse de motifs estampés sur d'autres objets, comme le casque d'Agris (Charente) ou le fourreau d'épée de Moscano di Fabriano (Italie, Marches) ? Pour ces deux objets, nous ne savons pas si les motifs sont du même poinçon. Par conséquent, nous ne pouvons pas faire d'évaluation quantitative du recalage comme pour les monnaies. Malgré tout, les résultats produits sur ces objets peuvent être analysés et validés par un expert. Pour les prochaines expériences de recalage, nous utilisons FCGF [Choy et al., 2019] entraîné uniquement sur les droits de Riedones3D, comme pour le recalage des revers et des barbus. De même, la régression logistique pour le calcul de similarité est uniquement entraînée sur les droits.

4.4.1 Fourreau de Moscano di Fabriano

Le fourreau de Moscano di Fabriano a été trouvé en 1955 en Italie à Fabriano (de nombreux indices matériels témoignent d'une présence celtique dans ce territoire médio-adriatique où les témoignages littéraires antiques localisent les Sénons, derniers peuple gaulois installé dans la péninsule). Ce fourreau est une référence pour l'étude de l'art celtique du IVe siècle avant notre ère. Le fourreau en fer est recouvert sur l'avant d'une tôle en bronze ornée d'une double ligne longitudinale de rinceaux à base d'esses et de triscèles (voir figure 4.10) caractéristiques du Style Végétal continu du IVe siècle avant notre ère (également dénommé Style de Waldalgesheim d'après le nom d'une célèbre sépulture d'Allemagne). Les motifs, identiques, sont imprimés de manière à simuler une frise continue. L'intérêt est de comparer les méthodes de fabrication avec d'autres objets similaires. D'un point de vue de la vision par ordinateur, la problématique est finalement semblable à celle que pose l'étude des monnaies. On veut pouvoir extraire des informations sur l'outil qui a permis d'imprimer les motifs. Si les motifs sont similaires, cela veut dire que très probablement, ils ont été frappés avec le même poinçon. La figure 4.10 montre une vue d'ensemble du modèle 3D.

Comme les monnaies de Riedones3D, le décor du fourreau a été numérisé avec le Smartscan d'Hexagon. Les motifs, au nombre de 22, sont découpés et extraits manuellement pour procéder à la comparaison des formes paire par paire. Les motifs isolés, constitués de rinceaux, sont plus simples et petits que les images monétaires.

La figure 4.11 permet de visualiser le résultat de la méthode proposée appliquées à deux motifs. Bien que très différents des images monétaires, il apparaît clairement que la méthode basée apprentissage profond peut aussi généraliser pour ce type de motif.

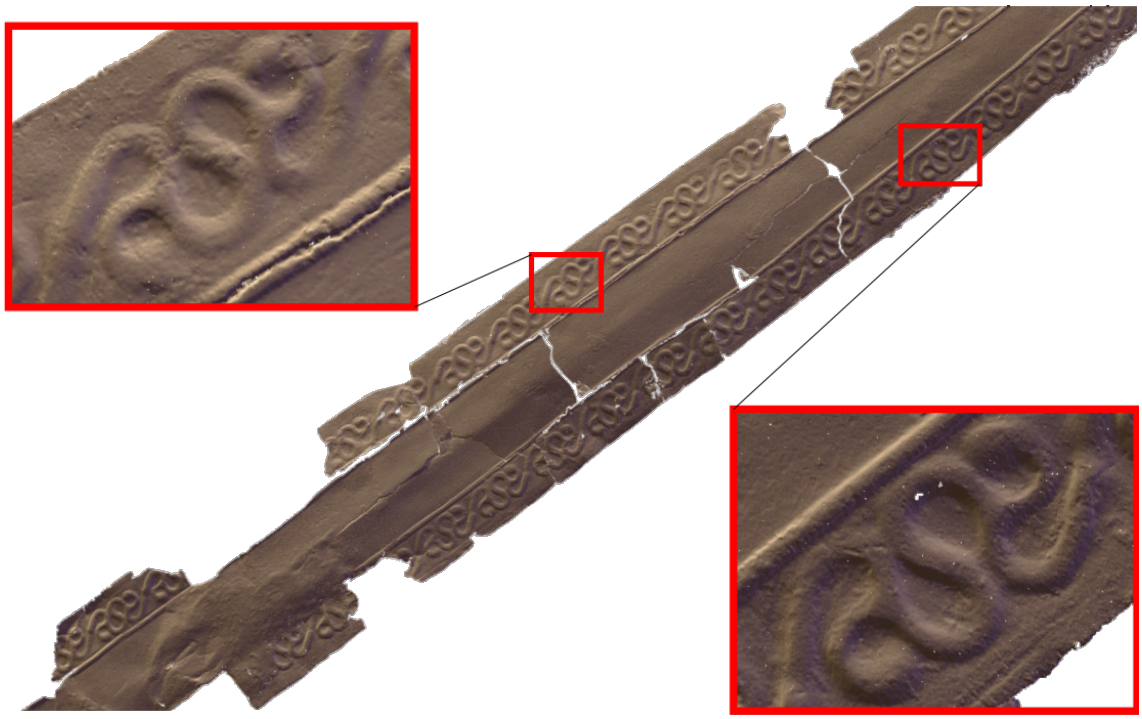


FIGURE 4.10 – Relevé 3D du fourreau d'épée de Moscano di Fabriano avec extraction d'une paire de motifs à rinceaux pour comparaison. L'objectif est de savoir si les différents motifs ont été imprimés avec un même poinçon.

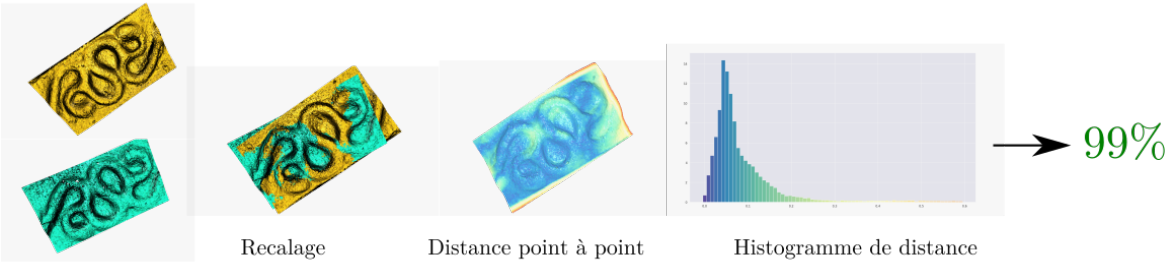


FIGURE 4.11 – Évaluation de la méthode proposée sur une paire de rinceaux

Contrairement au jeu de données Riedones3D, nous ne disposons pas de la vérité terrain (ni des transformations, ni des labels). Par conséquent, sur ce jeu de données, nous ne pouvons pas faire d'évaluation quantitative précise comme pour les jeux de données précédents. Mais qualitativement, pour la majorité des paires, les motifs sont visiblement bien alignés. De plus, le calcul de similarité indique une probabilité forte que les motifs viennent du même outil. Par conséquent, nous pouvons conclure que probablement, les rinceaux du fourreau de Moscano di Fabriano viennent du même poinçon. Cet exemple montre que nous pouvons facilement élargir la problématique à d'autres types d'objets et que la méthode proposée peut s'appliquer à d'autres disciplines que la numismatique.

4.4.2 Casque d'Agris

La deuxième application est l'analyse de motifs du casque d'Agris. Le casque d'Agris est un casque celtique trouvé dans la grotte des Perrats, à Agris, au nord-est d'Angoulême (Charente). Ce casque est daté du IVe siècle avant notre ère et est exposé au musée d'Angoulême. Chef-d'œuvre

FIGURE 4.12 – Casque d'Agris, Musée d'Angoulême (Charente). Source : Wikipédia

de l'art celtique, ce casque constitué de fer, bronze, or, argent et corail appartient à une série de couvre-chefs prestigieux connue par de rares exemplaires disséminés à travers le continent européen (voir figure 4.12). Le choix des matériaux et la richesse des décors déclinés en frises horizontales montrent qu'il ne s'agit pas d'un casque ordinaire mais plus probablement d'un attribut divin ou plus généralement d'un objet lié à la sphère religieuse. Le décor en léger relief a été réalisé sur la plaque en bronze recouvert par une feuille d'or. En l'état il est encore difficile d'évaluer l'incidence de la dorure sur le décor lui-même. Comme pour le fourreau de Moscano di Fabriano, comme pour les monnaies, on cherche à savoir si les différents motifs répétés à l'identique pour chaque registre ont été imprimés à partir d'une même matrice ou réalisés individuellement à la main. Pour cela, on dispose d'un modèle 3D réalisé comme pour les exemples précédents avec le Smartscan d'Hexagon (voir figure 4.13). Comme pour le fourreau d'épée, les motifs doivent être préalablement découpés et extraits. Dans le cas présent, les difficultés sont plus importantes en raison de l'ajout de blocs de corail, conservés en partie seulement, qui viennent perturber l'ordonnancement des décors imprimés.

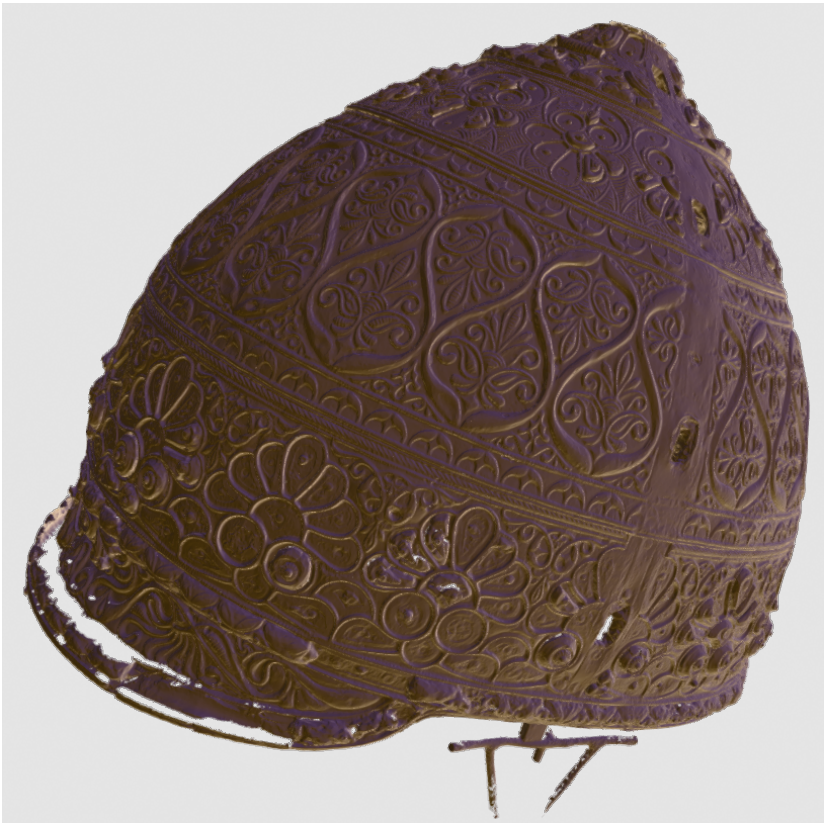


FIGURE 4.13 – Modèle 3D du casque d'Agris

En figure 4.14, nous pouvons voir une version du casque d'Agris déplié réalisée à partir du modèle numérique avec indications d'une série de motifs que l'on souhaite comparer. P1, P2, P3, PA et PB représentent des types de motifs différents. P1 et P3 sont difficilement comparables, car ce sont les motifs avec des pierres de corail qui rendent plus difficile le recalage et le calcul de probabilité. Si nous regardons les résultats du recalage, pour P1 et P2, l'algorithme de recalage proposé arrive approximativement à aligner les motifs (même si le recalage n'est pas systématiquement une réussite). Pour les motifs P3, le taux de succès du recalage est en revanche très bas. En figure 4.15, des exemples de paires de motifs recalées.

Comme pour le fourreau de Moscano di Fabriano, nous n'avons pas accès à la vérité terrain. Pour des motifs assez simples comme P2, la méthode proposée aligne approximativement les motifs. Cependant, dans le cas de P2, nous pouvons voir que les motifs ne s'alignent pas exactement (voir figure 4.15b). Pour les motifs P2 de la figure 4.15b, la méthode proposée estime que les motifs ne viennent pas du même poinçon (probabilité de 4%). Cela signifie donc que probablement, les motifs du casque d'Agris n'ont pas été imprimés avec un poinçon comme les rinceaux du fourreau de Moscano di Fabriano. La généralisation de la méthode de recalage sur le casque d'Agris est difficile à obtenir sur les motifs P1 et P3 (voir figure 4.15a et figure 4.15c), car les poinçons sont différents et certains motifs ont des pierres de corail incrustées alors que d'autres n'en ont pas. Le recalage et la comparaison deviennent donc beaucoup plus difficiles.

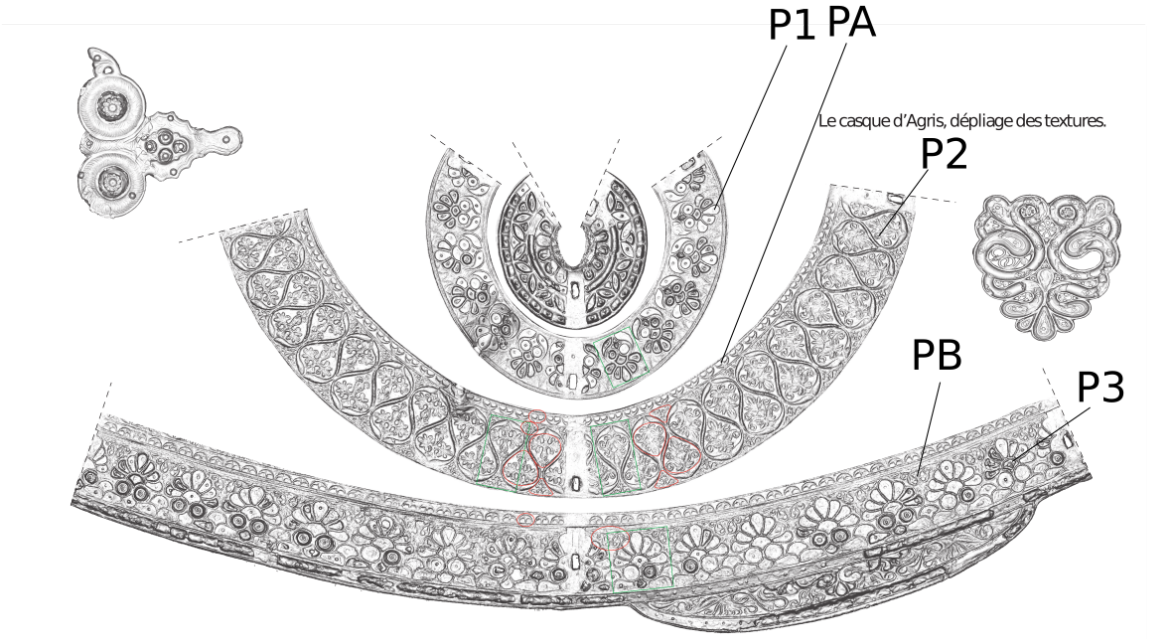
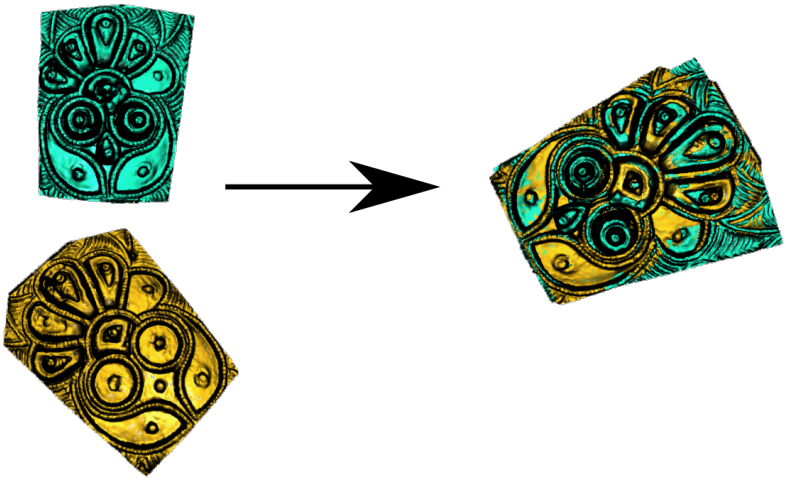
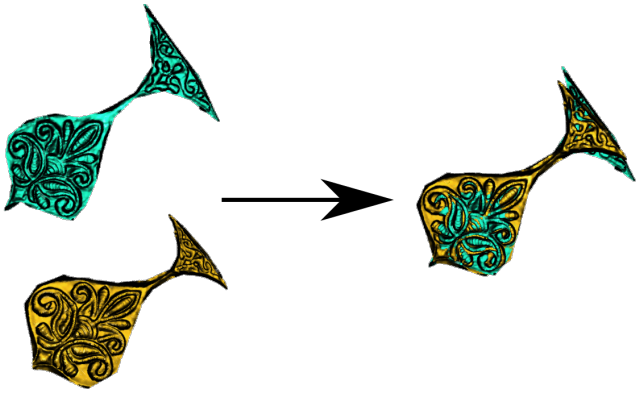


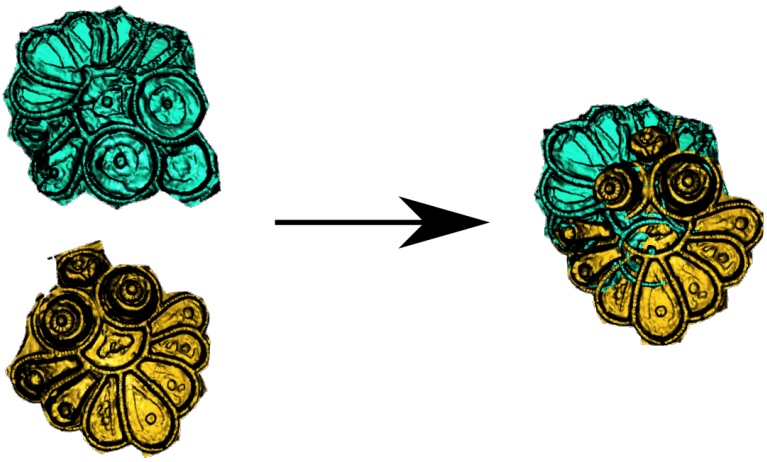
FIGURE 4.14 – Agris déplié à partir du modèle 3D. En vert et en rouge sont entourés une série de motifs intéressants à comparer, les types de motifs sont nommés P1, P2, P3, PA, PB. Crédit : Benjamin Houal (AOROC-ENS)



(a) recalage de motifs P1



(b) recalage de motifs P2



(c) recalage de motifs P3

FIGURE 4.15 – Recalage des motifs P1, P2, P3 du casque d'Agris (P3 est un échec du recalage). Pour les trois exemples, les motifs ne sont pas exactement les mêmes, les motifs ne peuvent pas s'aligner exactement. Par conséquent, nous pouvons en conclure que probablement, les motifs n'ont pas été imprimés avec le même poinçon.

4.5 Implémentation et mise en production

Un travail plus important est la mise en production de cette recherche à destination des archéologues. Si les principaux intéressés ne peuvent pas profiter de cette recherche, tous les efforts faits seraient presque vains. Pour ce travail de recherche, la reproductibilité et la mise en production sont d'autant plus importantes. L'enjeu principal de cette étude pluri-disciplinaire est de pouvoir transférer les résultats de cette contribution scientifique à des chercheurs en sciences humaines peu familiers à la manipulation de lignes brutes de code et d'aboutir à une application conviviale pouvant être mise en pratique sur d'autres corpus par des archéologues, des numismates, des historiens d'art.

4.5.1 Code actuel

Le code est ouvert sur ce site : https://github.com/humanpose1/riedones3d. Un tutoriel est disponible pour l'installation et l'utilisation. Le langage utilisé est Python. Pour les méthodes basées sur l'apprentissage profond, nous nous sommes basées sur la bibliothèque torch-points3d [Chaton et al., 2020]. Mais pour exécuter le code, il faut une machine avec une carte graphique pour la partie sur l'apprentissage profond. Le code a été réalisé sous l'OS Ubuntu 18.04, il n'a pas été conçu pour tourner sur Windows ou Mac. De plus, il faut installer beaucoup de bibliothèques avec des versions spécifiques (pytorch, numpy, pandas, MinkowskiEngine, etc.).

Mais juste, mettre le code source disponible n'est pas suffisant. Cela demande des compétences en informatique pour exécuter le code. Il faut être familier avec python pour l'installation. Il faut aussi pouvoir installer les différentes bibliothèques avec les bonnes versions. Pour une personne qui n'est pas formée ou qui n'a pas l'habitude, ce n'est pas facile. De plus, le code est garanti de tourner sur une machine suffisamment puissante sur un certain OS. Pour résumer, les contraintes sont :

— La méthode doit être facile à installer

— La méthode doit également être facile d'utilisation

Nous pouvons envisager deux possibilités. Une première est de faire un logiciel. Une autre possibilité est que le code tourne sur une seule machine à distance. L'utilisateur interagit avec le code grâce à une interface web.

4.5.2 Logiciel

Il faut rendre l'outil proposé plus facile d'accès. Et pour cela, il faut créer une interface utilisateur graphique. Il faut aussi que n'importe qui puisse l'installer. Dans ce cas, il faut s'adapter à la machine de l'utilisateur. Il faut s'adapter à différents OS, il faut s'adapter au matériel informatique de l'utilisateur (est-ce que l'utilisateur a une carte graphique ?). Que faire si l'utilisateur a uniquement un processeur ? Un autre inconvénient de cette approche est qu'il faudra ré-implémenter beaucoup de briques. Faire un logiciel qui tourne sur plusieurs plates-formes demandera de revoir la structure du code. Pour l'interface utilisateur, nous pouvons partir de zéro. Il est aussi possible d'offrir un plug-in sur un autre logiciel comme Cloud Compare [Cloud Compare, 2013]. Si nous partons de zéro, nous pouvons nous inspirer des interfaces utilisateurs de logiciels 3D open source déjà existants comme Instance Mesh [Jakob et al., 2015] ou CloudCompare [Cloud Compare, 2013] (voir figure 4.17 et figure 4.16). Quelle que soit la solution choisie, cela demandera une réécriture du code.

4.5.3 Application web

Une autre possibilité est que le code tourne sur une seule machine à distance puissante. L'archéologue lance les calculs à partir d'une interface web.

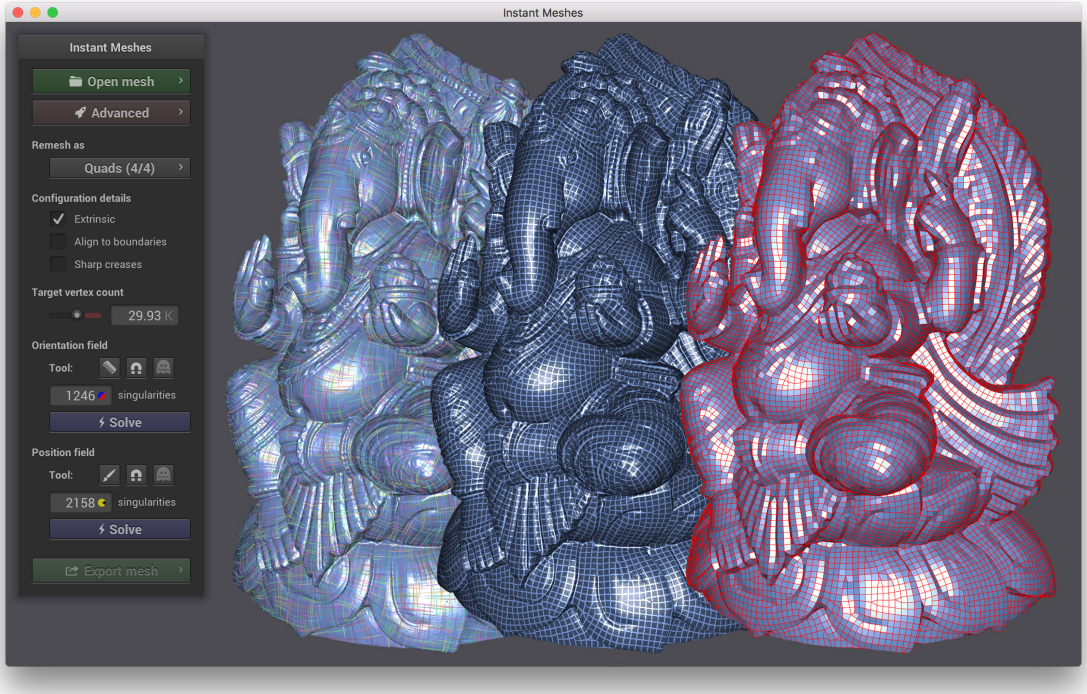


FIGURE 4.16 – Interface graphique de Instance Mesh. Credit : Wenzel Jakob

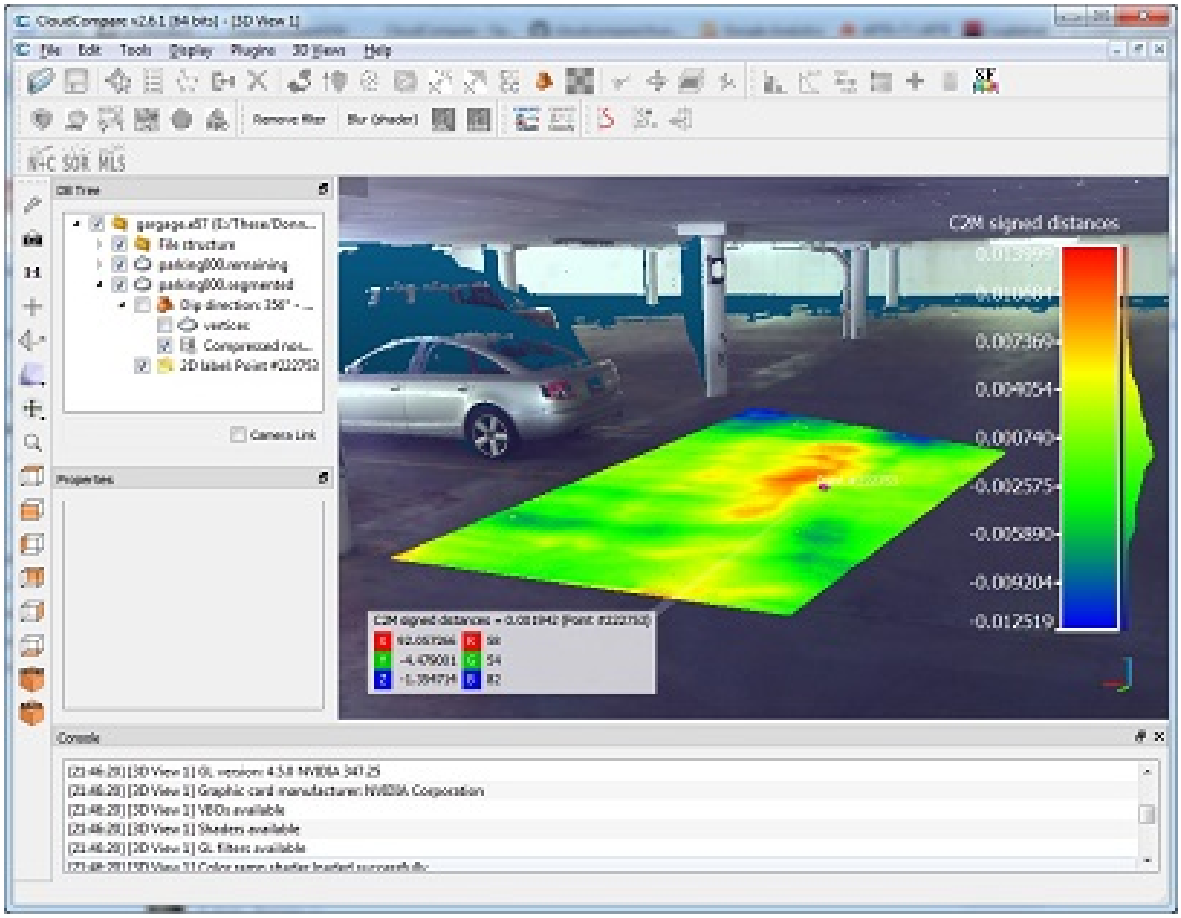


FIGURE 4.17 – Interface graphique de Cloud Compare. Credit : Daniel Girardeau-Montaut

Pour l'application web, nous pouvons nous inspirer du projet Filigrane [Bounou et al., 2020]. Cependant, contrairement à Filigrane, nous n'avons pas la contrainte d'avoir une application mobile.

Il faut pour cela créer une interface web et il est possible d'utiliser les langages du web (HTML, CSS et JavaScript). Pour les interfaces utilisateurs, il existe plusieurs bibliothèques comme React [react, 2013] ou Angular [Jain et al., 2014]. Pour visualisation des données, il est possible d'utiliser D3.js [D3, 2011] et Three.js [three, 2010] pour les modèles 3D. En matière de *backend*, il est possible de rester sur python avec des bibliothèques comme Django [Django Software Foundation, 2005] ou Flask [Grinberg, 2018]. L'avantage pour une application web est qu'il y aura moins de ré-écriture de code pour la partie recalage et comparaison, contrairement à un logiciel. Il est possible d'utiliser un environnement Docker [Merkel, 2014] par exemple pour que le code tourne sur un environnement bien défini et isolé.

Si nous choisissons de développer une application web, plusieurs contraintes vont se poser à nous comme la question du transfert de données, la gestion de différents comptes ou bien la question de la sécurité.

4.6 Conclusion

Nous avons présenté notre contribution d'identification de coins grâce aux algorithmes de recalage vu dans le précédent chapitre. Après le recalage, nous pouvons estimer de la probabilité que deux monnaies soient du même coin grâce au calcul de l'histogramme des distances. Nous avons également proposé un outil qui permet la visualisation et une vérification rapide des regroupements de monnaies. Cet outil également permet aux experts d'identifier les monnaies avec des coins différents qui ont des parties similaires comme les monnaies de revers avec un coin différent, mais le corps du cheval identique. Nous avons vu les nombreuses applications que la méthode proposée de regroupements de monnaies peut avoir, comme la reconstitution du coin originel ou l'estimation du nombre de coins ayant été utilisés pour ce monnayage.

Nous avons vu que la méthode proposée de recalage basée sur FCGF [Choy et al., 2019] fonctionne sur Riedones3D. Sur des données inconnues comme les Revers de Riedones3D ou le fourreau de Moscano, les résultats du recalage restent satisfaisants même s' il y a des erreurs pour certaines paires. Ces nouvelles expériences confirment la capacité de généralisation de FCGF [Choy et al., 2019] sur des jeux de données différents. Pour le casque d'Agris, le recalage fonctionne moins bien parce que les motifs n'ont probablement pas été imprimés par le même poinçon. Le recalage rigide ne peux pas les aligner parfaitement.

Nous avons également vu que pour les travaux futurs, il y a plusieurs perspectives de recherche afin d'améliorer la méthode de reconnaissance de coin proposée sur d'autres monnaies ou autres types d'objets. Avec la base de données Riedones3D, d'autres sujets de recherches complémentaires peuvent être envisagés comme la reconstruction de coins. Mais nous pouvons conclure qu'un sujet important est la mise en production de la méthode proposée, afin de rendre cette recherche accessible aux spécialistes. Il y a plusieurs choix possibles pour mettre en production le code de recherche développé. Il est possible de partir sur un logiciel ou un site web. Les deux approches ont des avantages et des inconvénients. Il faut prendre en compte que la création du site web ou la création du logiciel demande des compétences spécifiques et pointues qui sont différentes.

Le casque d'Agris et le fourreau de Moscano ont été acquis avec le même système d'acquisition que les monnaies de Riedones3D. On peut donc se demander : que faire lorsque le système d'acquisition est différent? Une solution serait de labéliser les nouvelles données acquises et de ré-entraîner un nouvel algorithme. Cependant, labéliser des données prend beaucoup de temps et ne peut être réalisé que par un expert comme nous avons pu le voir avec Riedones3D. De plus, que faire si le nouveau jeu de données est trop petit? La méthode actuelle peut dans une certaine mesure s'adapter à d'autres motifs mais ne peux pas s'adapter à d'autres capteurs. Le prochain chapitre propose une méthode pour s'adapter à n'importe quel type de capteur.

Chapitre 5

Recalage sur différents capteurs par transfert non supervisé

Sommaire

5.1 Introduction

Nous avons principalement travaillé sur le jeu de données Riedones3D pour développer un algorithme automatique de regroupement de monnaies selon leur coin. Nous avons vu que la méthode de recalage proposée entraînée sur les monnaies de droits réussit à recaler les monnaies de revers alors que le motif du revers est complètement différent du droit. Dans une certaines mesures, ils peuvent même s'appliquer à d'autres types de motifs. À partir des données expérimentales que nous avons, nous pouvons raisonnablement penser que la méthode de recalage proposée fonctionnera probablement sur d'autres monnaies seulement si les monnaies ont été acquises avec le même scanner. Cependant, nous n'avons pour l'instant pas beaucoup de garanties que les méthodes testées jusqu'à présent fonctionnent sur d'autres acquisitions.

Nous avons même de bonnes raisons de penser que FCGF, l'algorithme de recalage utilisé sur Riedones3D ne fonctionnera pas sur d'autres types d'acquisition, surtout si la densité de points n'est pas la même ou est irrégulière [Poiesi and Poiesi, 2021]. Cependant, l'acquisition 3D est très longue. Pour réaliser des études à plus grande échelle, il faudra probablement un système d'acquisition plus rapide. La question de la généralisation, c'est à dire la question de savoir à quel point une méthode entraînée sur un jeu de données d'entraînement peut fonctionner sur un autre jeu de données se pose. Si la méthode proposée ne généralise pas, il faut ré-entraîner le modèle sur les nouvelles données (ce qui va prendre beaucoup de temps). Nous devons donc nous assurer que la méthode de recalage proposée puisse généraliser à d'autres types d'acquisitions. Ce chapitre tente donc de répondre à cette question : la méthode proposée peut-elle être adaptée sur d'autres types d'acquisitions ? Ce chapitre aborde une fois de plus la généralisation des méthodes basées sur l'apprentissage profond. Cependant, cette fois-ci nous nous intéressons à la généralisation à d'autres types de capteur et pas à d'autres type de motifs. La question est de savoir si la méthode de recalage basée sur l'apprentissage profond va fonctionner sur des scènes différentes venant de capteurs différents, sachant que la méthode a été entraînée avec un autre système d'acquisition.

Nous avons déjà évoqué des éléments de réponse dans le chapitre 3. Nous avons vu les limites des méthodes d'apprentissage profond état de l'art. C'est pourquoi dans ce chapitre, nous allons proposer une nouvelle méthode de recalage pour généraliser à d'autres types d'acquisitions. La méthode proposée repose sur deux composantes : **MS-SVConv**, une nouvelle architecture basée sur FCGF [Choy et al., 2019] et une méthode d'apprentissage par transfert non supervisé appelé **UDGE**. Nos contributions sont :

— Nous développons ici une méthode basée sur l'apprentissage profond capable de généraliser sur un jeu de données totalement différent en utilisant l'apprentissage par transfert. Elle s'appuie sur deux modules complémentaires, un premier, MS-SVConv (une nouvelle architecture basée FCGF qui permet de mieux généraliser sur d'autres types de capteurs), qui utilise un U-Net sur plusieurs échelles avec des poids partagés et un second, UDGE (*Unsupervised transfer learning using Data GEneration*) qui permet de faire de l'apprentissage non supervisé par transfert.

Ces contributions ont donné lieu à un article publié à la conférence *International Conference on 3D Vision* [Horache et al., 2021a]. Nous n'avons pas de modèles 3D de monnaies acquises avec d'autres capteurs. Par conséquent, nous utilisons d'autres jeux de données issues du monde réel comme 3DMatch [Zeng et al., 2017], ETH [Pomerleau et al., 2012] ou TUM [Sturm et al., 2012]. L'utilisation de ces jeux de données va nous permettre de nous comparer plus facilement avec d'autres méthodes. Nous montrons que nous pouvons entraîner MS-SVConv sur un jeu de données synthétiques puis transférer de manière non supervisé sur un jeu de données réelles. Le code, ouvert, est disponible à ce lien : `https://github.com/humanpose1/MS-SVConv`.

5.2 Transfert non supervisé sur d'autres types de capteurs

Dans les méthodes à la pointe de l'état de l'art, nous avons vu que les méthodes basées U-Net se caractérisent par leurs capacités de généralisation faibles sur des jeux de données test issues de

capteurs différents par rapport au jeu de données d'entraînement [Ao et al., 2020, Poiesi and Poiesi, 2021]. Nous avons vu que les méthodes basées *patch* généralisent mieux sur des jeux de données tests issus de capteurs différents. En revanche, les expériences sur Riedones3D montrent que DIP a des capacités de généralisations assez limitées pour des motifs différents. De plus, ces méthodes sont très lentes et donc difficilement exploitables dans des conditions réelles. Par conséquent, il n'existe pas de méthode rapide avec une bonne capacité de généralisation à d'autres capteurs et à d'autres motifs (voir tableau 5.1).

Méthodes	Rapide	Généralisation à d'autres motifs	Généralisation à d'autres capteurs
FCGF [Choy et al., 2019]	Oui	Oui	Non
DIP [Poiesi and Poiesi, 2021]	Non	Non	Oui
Méthode proposée [Horache et al., 2021a]	Oui	Oui	Oui

TABLEAU 5.1 – Comparaison de différentes méthode. Les méthodes U-Net comme FCGF sont rapides et généralisent aux même motifs mais généralisent mal lorsqu'on change de capteurs [Poiesi and Poiesi, 2021]. Les méthodes par *patch* comme DIP sont lentes et ne généralisent pas sur des motifs similaires (dans le cas de Riedones3D). La méthode de recalage proposée répond à tous les critères.

Pour avoir le meilleur des deux approches, il nous faudrait une méthode basée U-Net qui offre de bonnes capacités de généralisation.

Dans cette section, nous présentons la méthode proposée basée sur MS-SVConv et UDGE. Nous montrons qu'il y a une très grande synergie entre les deux composantes qui permet de pré-entraîner sur un jeu de données synthétique puis de transférer de manière non supervisée sur un jeu de données différent et inconnu. Pour comparer correctement notre méthode aux autres, nous avons décidé d'utiliser des jeux de données réelles et difficiles comme le jeu de données 3DMatch [Zeng et al., 2017] ou ETH [Pomerleau et al., 2012].

Nous montrons que la méthode proposée a de meilleurs résultats que l'état de l'art sur 3DMatch ou ETH même si nous nous entraînons sur un jeu de données synthétique comme ModelNet [Wu et al., 2015]. Tout d'abord, nous expliquons comment calculer des descripteurs grâce aux méthodes U-Net. Nous décrivons ensuite la contribution proposée c'est-à-dire MS-SVConv et UDGE.

5.2.1 Méthodes U-Net

Une architecture U-Net peut être divisée en deux modules : un encodeur et un décodeur, le premier module sous-échantillonne le nuage de points voxélisé et calcule des *features* intermédiaires, grâce aux convolutions. Le deuxième module sur-échantillone les voxels en fusionnant les *features* calculés par l'encodeur dans chaque couche. Ainsi, un U-Net calcule des descripteurs pour tous les points du nuage de points. Nous appelons ψ_θ le U-Net avec comme paramètre θ. En entrée, ψ_θ prend des nuages de points $\mathbf{X}$ et un ensemble de *features* $\mathbf{F}_\mathbf{X}^{(in)} = \{f_{x_i}^{(in)} \in \mathbb{R}^{d^{(in)}} | i = 1 \dots |\mathbf{X}|\}$ (respectivement, les *features* d'entrées de $\mathbf{Y}$ sont $\mathbf{F}_\mathbf{Y}^{(in)}$). $d^{(in)}$ est la dimension des *features* d'entrée. Nous utilisons ψ_θ pour calculer des *features* de sorties : $\mathbf{F}_\mathbf{X} = \{f_{x_i} \in \mathbb{R}^d | i = 1 \dots |\mathbf{X}|\}$, et $\mathbf{F}_\mathbf{Y} = \{f_{y_i} \in \mathbb{R}^d | i = 1 \dots |\mathbf{Y}|\}$:

$$\mathbf{F}_\mathbf{X} = \psi_\theta(\mathbf{X}, \mathbf{F}_\mathbf{X}^{(in)}) \tag{5.1}$$

$$\mathbf{F}_\mathbf{Y} = \psi_\theta(\mathbf{Y}, \mathbf{F}_\mathbf{Y}^{(in)}) \tag{5.2}$$

d est la dimension des *features* de sortie.

Avec cette formule, nous pouvons exprimer des architectures pour le recalage comme FCGF [Choy et al., 2019] ou D3Feat [Bai et al., 2020]. Ainsi, trouver le bon descripteur est un problème d'apprentissage de métrique. Nous voulons que les motifs similaires aient des descripteurs qui soient proches dans le sens de la distance euclidienne et que des motifs différents aient des descripteurs éloignés. Dans le cas supervisé, nous avons les correspondances vérité terrain entre $\mathbf{X}$ et $\mathbf{Y}$. Ces correspondances sont les correspondances positives et nous pouvons tirer aléatoirement des correspondances négatives (correspondance de points où les motifs sont différents localement). Pour chaque paire de nuages de points $\mathbf{X}$ et $\mathbf{Y}$, nous minimisons la *hard negative contrastive loss* :

$$L(\theta) = \sum_{(i,j)\in\mathbf{M}^+} \{[\|f_{x_i} - f_{y_j}\| - m_+]_+^2 \tag{5.3}$$

$$+ \frac{1}{2}[m_- - \min_{k|(i,k)\in\mathbf{M}^-} \|f_{x_i} - f_{y_k}\|]_+^2 \tag{5.4}$$

$$+ \frac{1}{2}[m_- - \min_{k|(k,j)\in\mathbf{M}^-} \|f_{x_k} - f_{y_j}\|]_+^2\}, \tag{5.5}$$

Où $[.]_+ = \max(.,0)$, $\mathbf{M}^+$ est l'ensemble des correspondances positives. $\mathbf{M}^-$ est l'ensemble des correspondances négatives. $m+$ est un hyper-paramètre appelé la marge positive, et $m-$ est appelé la marge négative. Pour toutes les expériences et les jeux de données, $m+ = 0,1$ et $m- = 1,4$ comme dans FCGF [Choy et al., 2019].

Bien qu'il soit possible d'utiliser la *triplet loss* [Hadsell et al., 2006], la *contrastive loss* [Hoffer and Ailon, 2015] a montré de meilleurs résultats. En apprentissage supervisé, nous utilisons la transformation vérité terrain entre les paires de nuages de points pour obtenir ces ensembles de correspondances positives et négatives.

5.2.2 Influence du type de la convolution

Utilisé avec l'estimateur RANSAC [Fischler and Bolles, 1981], FCGF [Choy et al., 2019] a montré des résultats à la pointe de l'état de l'art sur le jeu de donnée 3DMatch [Zeng et al., 2017] et Riedones3D. Cette méthode peut gérer des nuages de points volumineux et est plus rapide que la plupart des autres méthodes basées sur l'apprentissage profond. Néanmoins, quand le nuage de points vient d'un autre environnement ou d'un autre capteur, FCGF ne peut pas généraliser (comme noté par Bai *et al.* [Bai et al., 2020] et Poiesi *et al.* [Poiesi and Poiesi, 2021]). FCGF [Choy et al., 2019] utilise les convolutions parcimonieuses, et a donc besoin de voxéliser le nuage de points. Mais il semble que la convolution peut sur-apprendre sur une taille de voxel fixe (contrairement à D3Feat, qui utilise *KPConv*). À cause de ce phénomène, FCGF ne peut pas généraliser quand l'échantillonnage est différent. Une solution est de sous-échantillonner le nuage de points afin d'éliminer l'influence de la densité. Cependant, le sous-échantillonnage mène à une perte de détails et à une représentation de la scène avec moins de points. Le problème est que la perte de détails nuit à la mise en correspondance de parties similaires et donc à l'estimation de la transformation. De plus, il est difficile de fixer la taille de voxel. Ce problème est présent pour des acquisitions de scènes d'intérieur ou d'extérieur. Mais dans le cas de Riedones3D, nous avons des acquisitions 3D de monnaies avec une densité très forte constante. De plus, les motifs sont fins et complexes. Le problème se pose donc moins sur le jeu de données Riedones3D.

5.2.3 Réseau de neurones multi-echelles, MS-SVConv

Les nuages de points sous-échantillonnés ont une densité homogène, mais peu de détails, alors que les nuages de points denses contiennent beaucoup de détails, mais n'ont pas forcément la même densité. L'idée est donc de sous-échantillonner le nuage de points à différentes échelles et ensuite d'appliquer U-Net pour chaque échelle. Ensuite, nous fusionnons les *features* calculées par chaque U-Net en utilisant un perceptron multi-couche (MLP). Une illustration de notre architecture est disponible en figure 5.1.

Soit $\mathbf{X}^{(s)}$ le nuage de points sous-échantillonné à l'échelle s ($\mathbf{F}^{(in)}_{\mathbf{X}^{(s)}}$). Pour MS-SVConv, les *features* de sorties sont calculés ainsi :

$$\mathbf{F}_{\mathbf{X}} = \text{MLP}(\bigoplus_{s=1}^{S} \psi_\theta(\mathbf{X}^{(s)}, \mathbf{F}^{(in)}_{\mathbf{X}^{(s)}})) \tag{5.6}$$

Où $\bigoplus$ signifie concaténation. Nous appliquons le même MLP pour chaque point du nuage de points.

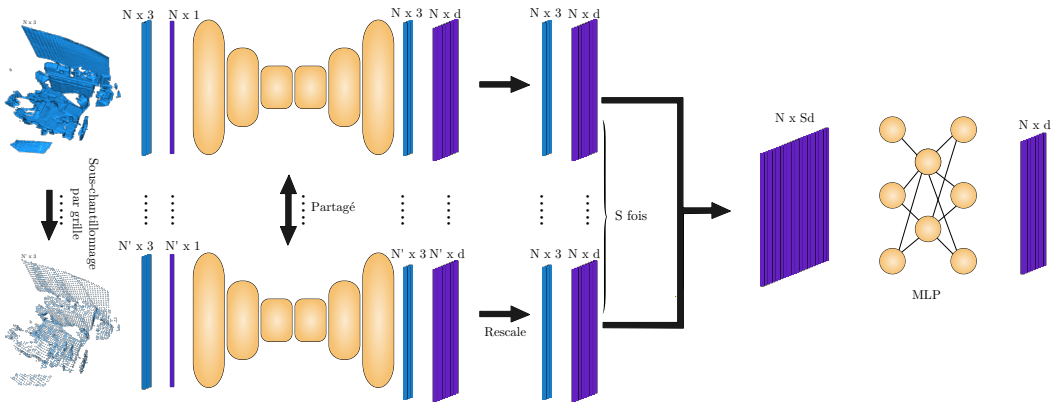


FIGURE 5.1 – Architecture de MS-SVConv pour le recalage. À chaque échelle, le nuage de points est sous-échantillonné avec une taille de voxel qui est multiplié par deux à chaque fois.

Le MLP va apprendre à sélectionner et filtrer les sorties à chaque échelle. En entrée, nous voxélisons le nuage de points en multipliant la taille du voxel par deux à chaque échelle (en effectuant un sous-échantillonnage par grille). Le nombre de voxels non vides va être différent pour chaque U-Net. Nous assignons donc le même descripteur de sortie pour tous les points qui étaient à l'origine dans le même voxel à une échelle spécifique (nous appelons cette opération *Rescale* en figure 5.1). Dans le U-Net, il y a déjà des opérations pour sous-échantillonner le nuage de point pour agrandir son champ récepteur. Cependant, un U-Net classique pour un nuage de points 3D avec trois échelles où nous sous-échantillonnons le nuage de points voxélisé offre un champ récepteur multiplié par huit fois la taille du voxel. En utilisant trois échelles de nuage de points, nous sommes capables de multiplier le champ récepteur par 64, ce qui apporte plus de contexte lors du calcul des descripteurs.

Nous utilisons le même réseau pour chaque échelle avec des poids partagés : cela veut dire que nous gardons le même nombre de paramètres que pour une échelle et nous ajoutons seulement les paramètres du MLP final.

Des travaux précédents ont étudié les architectures multi-échelles en 3D, mais d'une manière différente et dans un contexte totalement distinct. Mu-Net par exemple [Lee et al., 2020], est un U-Net en série avec des poids non partagés, sur une grille de voxel dense pour du débruitage. Cette architecture est très différente de MS-SVConv avec plusieurs U-Net en parallèle, à des échelles multiples et des poids partagés. MS-DeepVoxNet [Roynard et al., 2018] prend plusieurs voisinages (échelles) en entrée pour classifier un point et est utilisé en segmentation sémantique de nuages de points. Dans notre cas, nous prenons tout le nuage de points en entrée, sous-échantillonné à des échelles différentes. L'approche multi-échelle, améliore les résultats dans le cas supervisé, et la capacité de l'architecture U-Net à généraliser sur d'autres jeux de données.

5.2.4 Apprentissage non supervisé avec UDGE

L'apprentissage par transfert ou *transfer learning* est une branche de l'apprentissage automatique qui s'intéresse à apprendre sur un jeu de données source et pouvoir ré-appliquer dans un jeu de données cible ce qui a été appris précédemment. L'objectif est de concevoir un algorithme d'apprentissage par transfert non supervisé qui permet de généraliser les descripteurs appris pour le recalage sur des données issues de capteurs différents.

Dans le cadre supervisé, nous avons des paires de nuages de points ainsi que la pose relative entre les nuages de points. Pour une scène, les nuages de points sont différents, car ils viennent de différents points de vue. Le principe de *Unsupervised transfer learning using Data GEneraration* (UDGE) est de générer une paire de nuages de points à partir d'un seul. Pour y parvenir, nous appliquons du rognage aléatoire (*random cropping*) et un sous-échantillonnage périodique. Le but

de cette génération de données est de simuler deux points de vue différents d'une scène. Avec la paire obtenue, nous connaissons parfaitement les correspondances positives et négatives, et nous pouvons ainsi entraîner notre réseau dans le même cadre que l'apprentissage supervisé. C'est cependant de l'apprentissage non supervisé car il n' y a pas besoin de connaître la transformation entre une paire de nuage de points pour entraîner le modèle. Un seul nuage suffit. Pour notre méthode, le principe est d'entraîner le réseau, de manière supervisée sur un jeu de données source. Et nous utilisons UDGE pour entraîner de manière non supervisée sur un autre jeu de donnée T (cible). UDGE montre une très bonne efficacité, même si le jeu de données cible T est petit.

Le travail le plus proche de UDGE vient de [Yuan et al., 2021]. Mais ils ne font pas de pré-entraînement. Nous montrons que sans pré-entraînement, l'apprentissage auto-supervisé ne peut marcher que sur de gros jeux de données. De plus, nous utilisons des fonctions spécifiques comme le rognage aléatoire et le sous-échantillonnage périodique, qui ne sont pas utilisés dans [Yuan et al., 2021].

5.2.4.1 Génération de nuages de points

La figure 5.2 montre comment les données sont générées à partir d'un seul nuage de points. Nous faisons la différence entre la génération de données et l'augmentation de données. La génération de données consiste à faire deux nuages de points différents à partir d'un seul. L'objectif de la génération de données est de générer sans supervision des paires de nuages de points pour faire comme si nous étions dans un cas supervisé. La génération de données dépend du jeu de données cibles. L'augmentation de données est utilisée pour augmenter artificiellement la taille du jeu de données. Dans notre cas, nous faisons également de l'augmentation de données, particulièrement des rotations aléatoires, des changements d'échelles aléatoires et de l'ajout de bruit. La génération de données est utilisée par certains auteurs sur ModelNet, mais pour tester si leur méthode de recalage marche même sur des nuages de points partiels [Yew and Lee, 2020]. La génération de données n'est pas utilisée pour l'entraînement. Dans notre cas, nous l'utilisons aussi sur ModelNet pour de l'entraînement, mais aussi sur des données RGBD ou bien sur des acquisitions LiDAR. Bien qu'un travail similaire soit présenté dans [Zhang et al., 2021], ils réalisent de l'apprentissage auto-supervisé seulement comme pré-entraînement pour de la segmentation sémantique. Dans le cas de UDGE, nous utilisons la génération de données pour de l'apprentissage par transfert non supervisé pour du recalage de nuages de points.

5.2.4.1.1 Rognage Comme pour les images, il est possible de sélectionner une zone locale et d'exclure le reste. Cela permet de générer de nouvelles données d'entraînement. Pour cela, nous sélectionnons un point aléatoire ainsi qu'une taille de rognage aléatoire (un cube ou une sphère), les points à l'intérieur de la forme sont gardés et les points à l'extérieur sont rejetés. La sélection d'une zone locale aléatoire est utile pour simuler des nuages de points qui se recouvrent partiellement.

5.2.4.1.2 Sous échantillonnage aléatoire Les acquisitions 3D de scènes ont souvent un échantillonnage irrégulier, qui dépend du point de vue. Nous proposons une technique de génération de données qui permet de simuler cet échantillonnage irrégulier, appelé l'échantillonnage périodique (voir la figure 5.2). Le principe est de supprimer des points périodiquement selon un centre choisi aléatoirement. Soit $c \in \mathbb{R}^3$ un centre aléatoire. Nous calculons le masque $M \in \{0,1\}^N$ du nuage de ponts $\mathbf{X}$. m_i est la valeur binaire qui nous indique si nous gardons x_i ou pas. Le masque est défini ainsi :

$$m_i = \mathbf{1}(|\cos(\frac{2\pi}{T}\|x_i - c\|)| > \cos(\alpha\pi)), \tag{5.7}$$

$\alpha \in [0,1]$ est un seuil qui indique la proportion de points qu'on garde.

Si $\alpha = 0$, tous les points sont supprimés. Si $\alpha = 1$, nous gardons tous les points. T est la période de l'échantillonnage aléatoire. En changeant la période T et le seuil α, nous pouvons simuler

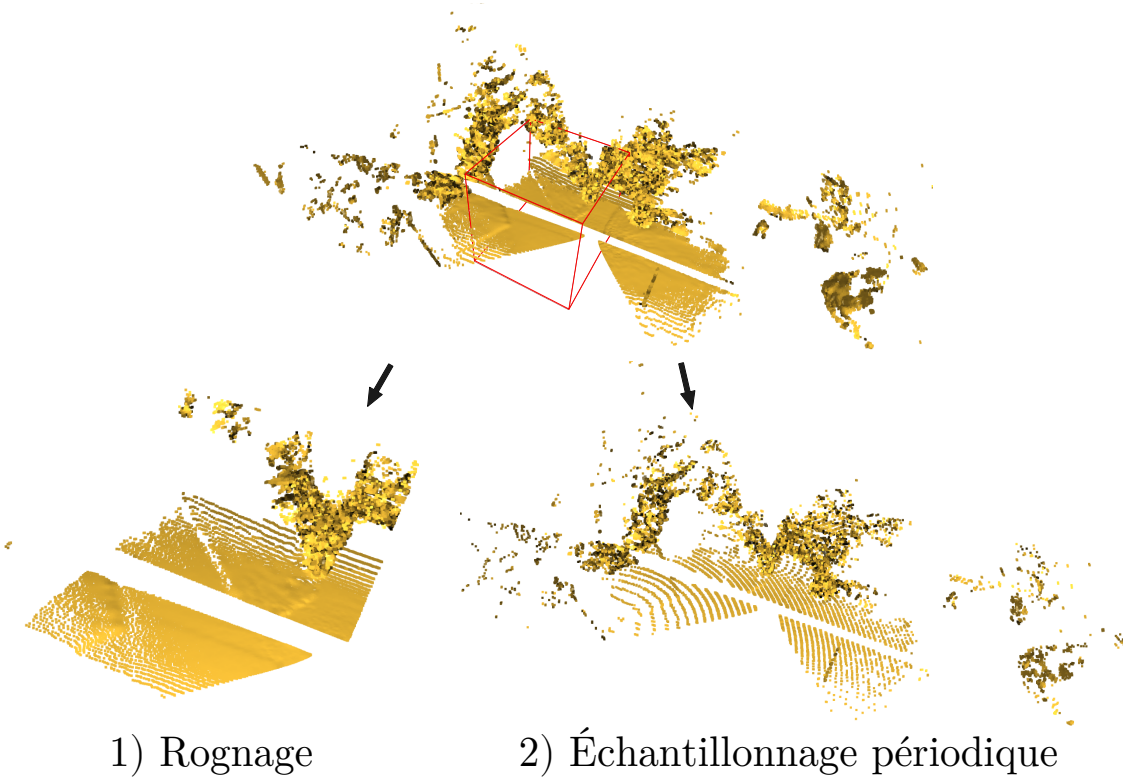


FIGURE 5.2 – Les deux techniques de générations de nuages de points que nous utilisons pour créer une paire de nuages de points à partir d'un seul nuage de points

une multitude de types d'échantillonnage. L'échantillonnage aléatoire est particulièrement utile surtout lorsque nous rencontrons des scènes qui ont une densité irrégulière.

5.2.5 Expérimentations sur des acquisitions d'intérieur et d'extérieur

Pour montrer que MS-SVConv et UDGE permettent la généralisation à de nouveaux jeux de données, nous évaluons la méthode proposée sur les jeux de données 3DMatch, ETH et TUM. Tout d'abord, nous décrivons les jeux de données utilisés pour les expérimentations, ainsi que les détails d'implémentation. Puis, nous expliquons notre protocole expérimental. Ensuite, nous montrons l'impact de l'architecture multi échelle proposée dans un cadre supervisé, et nous montrons les résultats de la généralisation de MS-SVConv. Finalement, nous montrons des résultats qualitatifs.

5.2.5.1 Présentation des jeux de données.

Nous allons présenter les jeux de données utilisées pour valider notre contribution. «Jeu de données source» veut dire un jeu de données où nous entraînons en premier pour avoir un pré-entraînement. «Jeu de données cible» veut dire que nous testons notre méthode d'apprentissage par transfert non supervisé après un pré-entraînement sur un jeu de données source.

5.2.5.1.1 ModelNet40 (source) : ModelNet40 [Wu et al., 2015] est un jeu de données CAO qui contient environ 10 000 objets avec 40 catégories. Ce jeu de données est utilisé pour de la classification. Mais beaucoup de méthodes basées sur l'apprentissage profond utilisent ModelNet pour le recalage. Dans notre cas, nous utilisons ModelNet [Wu et al., 2015] comme un jeu d'entraînement source. Notre objectif est de montrer que même si le jeu de données est synthétique, la méthode proposée peut quand même généraliser sur un jeu de données issu du monde réel comme 3DMatch [Zeng et al., 2017] ou ETH [Pomerleau et al., 2012].

5.2.5.1.2 3DMatch (source et cible) : 3DMatch [Zeng et al., 2017] est un jeu de données qui est composé d'image RGBD qui viennent d'autres jeux de données d'intérieur utilisé pour la reconstruction 3D, comme BundleFusion [Dai et al., 2017], SUN3D [Halber and Funkhouser, 2017], ou 7-Scene [Shotton et al., 2013]. Le jeu de données est constitué de 62 scènes. Comme Choy *et al.* [Choy et al., 2019], nous séparons le jeu de données en un jeu de donnée d'entraînement composé de 48 scènes, un jeu de données de validation composé de 6 scènes et un jeu de données test composé de 8 scènes. Pour créer des nuages de points, il faut traiter les images de profondeurs : nous fusionnons 50 images de profondeurs avec une TSDF comme [Deng et al., 2018, Choy et al., 2019, Bai et al., 2020, Gojcic et al., 2019]. Le jeu de données test est fourni par le site de 3DMatch [Zeng et al., 2017]. 3DMatch est utilisé pour de l'entraînement supervisé (paires de nuages de points avec un recouvrement de 30% minimum comme dans [Gojcic et al., 2019]). Nous utilisons 3DMatch [Zeng et al., 2017] comme un jeu de données cible pour tester UDGE avec comme pré-entraînement ModelNet [Wu et al., 2015].

5.2.5.1.3 ETH (cible) : Le jeu de données ETH [Pomerleau et al., 2012] est un jeu de données de nuages de points 3D acquis en intérieur et extérieur avec un LiDAR 2D. Il est composé de 8 scènes : 6 scènes d'extérieur et 2 scènes d'intérieur. Le jeu de données est considéré comme difficile, car c'est un jeu de données bruité avec des densités irrégulières. Nous allons faire une différence entre ETH-8-scenes, qui suit le protocole rigoureux décrit par Fontana *et al.* [Fontana et al., 2021] et ETH-4-scenes qui suit le protocole de Gojcic *et al.* [Gojcic et al., 2019] et qui est suivi par la majorité des travaux publiés précédemment comme [Gojcic et al., 2019, Bai et al., 2020, Ao et al., 2020, Poiesi and Poiesi, 2021, Poiesi and Boscaini, 2021]. Ce jeu de données est utilisé pour évaluer les capacités de généralisation de la méthode proposée.

5.2.5.1.4 TUM (cible) : TUM [Sturm et al., 2012] est un jeu de données RGBD qui est utilisé pour évaluer l'odométrie ou le SLAM RGBD. Nous utiliserons seulement une seule image de profondeur comme nuage de points, en suivant le protocole de Fontana *et al.* [Fontana et al., 2021]. Ce jeu de données est aussi utilisé pour évaluer la méthode proposée de transfert non supervisé.

5.2.5.2 Implémentation

5.2.5.2.1 Détails d'implémentation Pour gérer les différentes expériences, nous utilisons Pytorch Points 3D [Chaton et al., 2020]. Cette bibliothèque utilise massivement la bibliothèque hydra [Yadan, 2019] pour gérer les hyper-paramètres, l'architecture des réseaux et les augmentations de données. Pour la convolution parcimonieuse et les tenseurs parcimonieux, nous utilisons l'implémentation de Pytorch Points 3D qui utilise comme *backend* torchsparse (implémenté par Tang *et al.* [Tang et al., 2020]). Pytorch Points 3D [Chaton et al., 2020] aussi supporte comme *backend* MinkowskiEngine [Choy et al., 2019], mais la convolution parcimonieuse dans torchsparse [Tang et al., 2020] est plus rapide que celle de MinkowskiEngine [Choy et al., 2019]. Pour tous les entraînements, nous utilisons la descente de gradient stochastique (SGD) avec un *momentum* de 0,8 est une taille de *batch* de 4. Pour le pré-entraînement, le nombre d'époques est fixé à 400 pour ModelNet [Wu et al., 2015] (environ 48h), et 300 pour le jeu de données 3DMatch [Zeng et al., 2017] (environ 3 semaines), et le *learning rate* est de 0,1. Pour l'apprentissage par transfert non supervisé, le nombre d'époques est de 200 pour tous les jeux de données (environ 2 h pour TUM [Sturm et al., 2012], 24 h pour 3DMatch [Zeng et al., 2017], 18 h pour ETH [Pomerleau et al., 2012]), et le *learning rate* est de 0,001.

Sur 3DMatch, ModelNet et TUM, la taille du voxel est fixée à 2 cm et est doublée à chaque échelle. Pour le jeu de données ETH (seulement quand on utilise UDGE), la taille de voxel est fixée à 4 cm et est aussi doublée (ETH est plus parcimonieux que 3DMatch). Le choix de la taille de voxel va dépendre de la densité du nuage de points. La dimension du descripteur de sortie est de 32 pour tous les jeux de données (comme la plupart des papiers publiés récemment [Choy et al., 2019, Gojcic et al., 2019, Bai et al., 2020, Ao et al., 2020, Poiesi and Poiesi, 2021, Poiesi and Boscaini, 2021]). Pour la génération de données dans UDGE, le rognage a une forme cubique, le centre est

un point aléatoire et la taille du rognage est de 2 m pour ModelNet [Wu et al., 2015], 3 m pour 3DMatch [Zeng et al., 2017] et TUM [Sturm et al., 2012], et 10 m pour ETH [Pomerleau et al., 2012]. Comme pour les paramètres du rognage, les paramètres de l'échantillonnage périodique changent selon le jeu de données. Pour ETH, nous tirons aléatoirement α entre 15 et 30%. La période T est choisie aléatoirement entre 4 cm et 16 cm. Pour le jeu de données TUM, nous tirons aléatoirement α entre 10% et 40% et la période T est choisie aléatoirement entre 2 cm et 8 cm. Pour 3DMatch, nous n'utilisons pas d'échantillonnage périodique, car les nuages de points ont une densité qui varie peu. Finalement, nous réalisons une augmentation de données (pour tous les jeux de données) avec des rotations aléatoires selon tous les axes, un changement d'échelle aléatoire entre 0,9 et 1,2 et un bruit gaussien avec un écart-type de 0,7 cm pour TUM et 3DMatch et 1 cm pour ETH et ModelNet.

Pour calculer la translation et la rotation, nous utilisons l'algorithme TEASER [Yang et al., 2020] (avec comme paramètre principal une borne de bruit fixée à $\tau_1 = 0,1$). Nous choisissons TEASER par rapport à RANSAC [Fischler and Bolles, 1981] parce que TEASER est plus rapide [Yang et al., 2020]. Pour toutes les expériences, nous échantillonnons aléatoirement 5 000 points, pour évaluer la mise en correspondance entre les nuages de points. Les expériences ont principalement été faites avec un ordinateur équipé d'une Carte Graphique Nvidia GTX 1080Ti et d'un processeur Intel(R) Core(TM) i7-4790K.

5.2.5.2.2 Architecture du U-Net Dans la figure 5.3, nous reportons l'architecture d'un réseau U-Net de MS-SVConv. Parce que les poids sont partagés quand on utilise MS-SVConv avec S échelles, le modèle a toujours quasiment le même nombre de paramètres (dans nos expériences, le modèle a environ 9 millions de paramètres). La seule différence vient du MLP à la fin qui agrège les *features* calculées à différentes échelles, ce qui donne dans notre cas 3104 paramètres supplémentaires pour trois échelles. L'architecture de MS-SVConv (à une échelle) et FCGF [Choy et al., 2019] sont presque similaires, mais MS-SVConv (à une échelle) a un bloc résiduel supplémentaire. De plus, MS-SVConv est plus rapide que FCGF, car FCGF utilise la bibliothèque MinkowskiEngine et MS-SVConv la bibliothèque torchsparse. Pour l'évaluation, contrairement à [Gojcic et al., 2019, Bai et al., 2020, Ao et al., 2020], FCGF ne fait pas de test symétrique quand il évalue la méthode [Choy et al., 2019]. C'est pourquoi les résultats de FCGF sont plus bas en matière de FMR (le FMR est défini ci-dessous dans l'équation 5.9).

5.2.5.3 Métriques

Pour évaluer la méthode proposée, nous utilisons deux métriques : le *Feature Match Recall* (FMR) [Gojcic et al., 2019] et le *Scaled Registration Error* (SRE) [Fontana et al., 2021]. Soit $(\mathbf{X}_i, \mathbf{Y}_i)$ une paire de nuages de points du jeu de données test que nous souhaitons évaluer. Nous appelons $\mathbf{M}(\mathbf{X}_i, \mathbf{Y}_i)$ les correspondances entre $\mathbf{X}_i$ et $\mathbf{Y}_i$, $|\mathbf{M}|$ est le nombre de correspondances, et $(\mathrm{R}^{(gt)}, t^{(gt)})$ les rotations et translations vérités terrain. Le hit ratio est défini ainsi :

$$\mathrm{H}(\mathbf{X}_i, \mathbf{Y}_i) = \frac{1}{|\mathbf{M}|} \sum_{k,l \in \mathbf{M}(\mathbf{X}_i, \mathbf{Y}_i)} \mathbf{1}(\|\mathrm{R}^{(gt)} x_k + t^{(gt)} - y_l\| \leq \tau_1). \tag{5.8}$$

Le *Feature Match Recall* (FMR) est défini comme

$$\mathrm{FMR} = \frac{1}{\mathrm{N}} \sum_{i=1}^{\mathrm{N}} \mathbf{1}(\mathrm{H}(\mathbf{X}_i, \mathbf{Y}_i) \geq \tau_2), \tag{5.9}$$

Où N est dans ce cas le nombre de paires de scènes.

Comme dans les travaux précédents [Gojcic et al., 2019, Choy et al., 2019, Bai et al., 2020, Ao et al., 2020, Poiesi and Poiesi, 2021], nous assignons $\tau_1 = 0,1$ m et $\tau_2 = 0,05$ par défaut et nous réalisons un test symétrique pour filtrer les correspondances avant l'évaluation. La métrique permet d'évaluer la mise en correspondance des descripteurs. Bien que ce soit une métrique très utile pour comparer les différentes méthodes de mise en correspondance de descripteurs, nous ne pouvons utiliser cette métrique que pour des approches basées descripteurs (et par conséquent, nous

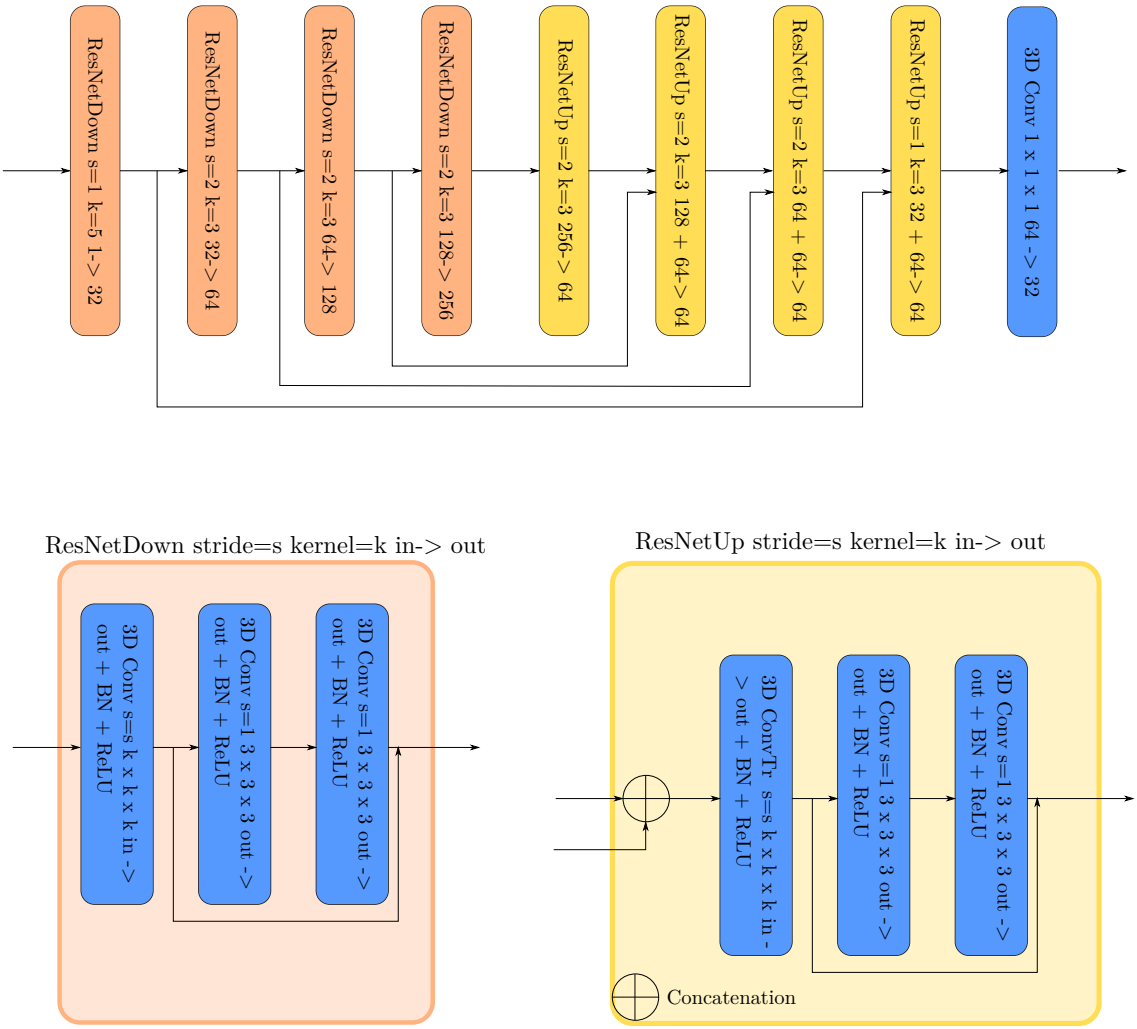


FIGURE 5.3 – Architecture du réseau U-Net de MS-SVConv (BN veut dire «Batch Normalization»).

ne pouvons pas l'utiliser pour évaluer ICP par exemple). Nous utilisons également le SRE défini à l'équation 3.33.

5.2.5.4 Protocole sur ETH [Pomerleau et al., 2012]

Pour nos tests sur le jeu de données ETH [Pomerleau et al., 2012], nous suivons le protocole de Fontana *et al.* [Fontana et al., 2021] et de Gojcic *et al.* [Gojcic et al., 2019]. Le premier est plus rigoureux et utilise huit scènes au lieu de quatre. Fontana *et al.*ont introduit le SRE [Fontana et al., 2021], qui est plus rigoureux que l'erreur de translation et rotation. Pour calculer les résultats des méthodes précédentes sur ETH-8-scenes, nous avons dû calculer nous-mêmes avec le code des auteurs disponible en ligne. Nous présentons les détails expérimentaux des méthodes que nous avons testées sur ETH 8-scenes en suivant le protocole de Fontana *et al.* [Fontana et al., 2021]. Nous avons varié les paramètres des différentes méthodes pour garder les meilleurs résultats.

5.2.5.4.1 FPFH [Rusu et al., 2009] Nous utilisons la version Open3D [Zhou et al., 2018]. Nous sous-échantillonnons le nuage de points avec une taille de voxel de 6 cm et nous choisissons 5000 points aléatoirement. Pour calculer les normales nécessaires au calcul des descripteurs, nous utilisons un rayon de 50 cm (maximum 30 voisins).

5.2.5.4.2 DIP [Poiesi and Poiesi, 2021] Pour DIP, nous sous-échantillonnons le nuage de points avec une taille de voxel de 6 cm. Nous utilisons également 5000 points aléatoirement. La zone locale a un rayon de $60\sqrt{3} = 104$ cm. Nous fixons le paramètre $p_\rho = 0$. Le modèle a été fourni par les auteurs.

5.2.5.4.3 D3Feat [Bai et al., 2020] Pour D3Feat [Bai et al., 2020], nous sous-échantillonnons le nuage de points avec une taille de voxel de 6 cm (comme dans la démo fournie par les auteurs). Nous utilisons l'implémentation Tensorflow. Nous doublons l'échelle des *kernel points* pour augmenter le champ récepteur, comme Bai *et al.* [Bai et al., 2020]. Nous utilisons le modèle entraîné avec la *contrastive loss* donnée par les auteurs (nous avons remarqué que le modèle entraîné avec la *circle loss* avait de moins bons résultats). Nous choisissons 5000 points avec le meilleur score pour calculer le FMR et le SRE.

5.2.5.4.4 MultiView [Li et al., 2020] Pour MultiView [Li et al., 2020], nous utilisons les poids pré-entraînés par les auteurs. Nous sous-échantillonnons le nuage de points avec une taille de voxel de 6 cm et nous prenons 5000 points aléatoires.

5.2.5.5 MS-SVConv dans un cadre supervisé

5.2.5.5.1 Évaluation sur 3DMatch Nous évaluons l'impact de MS-SVConv sur l'apprentissage supervisé pour voir l'influence de l'architecture multi échelle. Le tableau 5.2 montre les résultats sur 3DMatch [Zeng et al., 2017]. Bien que le *benchmark* 3DMatch soit très compétitif, MS-SVConv surpasse toutes les autres méthodes publiées. Si nous comparons MS-SVConv avec la meilleure méthode publiée (MultiView [Li et al., 2020]), nous avons une augmentation de +3% sur le FMR avec un $\tau_2 = 0,2$. Le tableau 5.2 montre que MS-SVConv avec trois échelles est bien meilleur qu'avec une seule échelle, ce qui démontre l'intérêt de la méthode multi-échelle pour l'apprentissage supervisé. MS-SVConv avec une échelle et FCGF sont similaires conceptuellement, mais MS-SVConv a une implémentation différente. De plus, contrairement à la plupart des méthodes publiées, FCGF ne filtre pas les correspondances avec un test symétrique. Si nous filtrons les correspondances avec un test symétrique, les résultats de FCGF et MS-SVConv avec une échelle sont quasiment similaires (voir le tableau 5.2)

Apprentissage supervisé sur 3DMatch		
Méthodes	FMR (%) $\tau_2 = 0,05$	FMR (%) $\tau_2 = 0,2$
SHOT [Salti et al., 2014]	23,8	-
FPFH [Rusu et al., 2009]	35,9	-
3DMatch [Zeng et al., 2017]	59,6	-
PPFNet [Deng et al., 2018]	62,3	-
3DSmoothNet [Gojcic et al., 2019]	94,7	72,9
DIP [Poiesi and Poiesi, 2021]	94,8	-
FCGF [Choy et al., 2019]	95,2	67,4
FCGF* [Choy et al., 2019]	97,5	87,3
D3Feat [Bai et al., 2020]	95,8	75,8
MultiView [Li et al., 2020]	97,5	86,9
SpinNet [Ao et al., 2020]	97,6	85,7
Predator [Huang et al., 2021]	96,6	-
GeDI [Poiesi and Boscaini, 2021]	97,9	-
MS-SVConv (1 échelle)	97,6	87,2
MS-SVConv (3 échelles)	**98,4**	**89,9**

TABLEAU 5.2 – FMR sur 3DMatch [Zeng et al., 2017] avec $\tau_2 = 0,05$ et $\tau_2 = 0,2$. Les résultats des méthodes publiés viennent des papiers d'origine. FCGF* signifie que nous évaluons nous-mêmes FCGF avec le code original et un test symétrique avant de calculer le FMR.

Cible / **Architecture**	ETH	TUM	3DMatch
MS-SVConv(1)	56,4	99,0	97,6
MS-SVConv(3)	**76,8**	**100**	**98,4**

TABLEAU 5.3 – FMR ($\tau_2 = 0,05$) de MS-SVConv entraîné sur le jeu de données 3DMatch avec une taille de voxel de 2 cm et évalué sur ETH-8-scenes, TUM et 3DMatch sans UDGE

5.2.5.5.2 Capacité de généralisation de MS-SVConv sans UDGE Le tableau 5.3 montre le FMR sans UDGE sur les jeux de données ETH-8-scenes [Pomerleau et al., 2012], TUM [Sturm et al., 2012] et 3DMatch [Zeng et al., 2017] après avoir entraîné nos modèles sur 3DMatch [Zeng et al., 2017] (Tableau 5.4 quand nous entraînons sur ModelNet [Wu et al., 2015]). Ces tables montrent que l'architecture multi-échelle améliore les capacités de généralisation. De plus, quand nous entraînons sur le jeu de données ModelNet [Wu et al., 2015], le tableau 5.4 montre que l'approche multi-échelle apporte une amélioration de +71%. MS-SVConv avec trois échelles est bien meilleur que MS-SVConv avec une seule échelle. Ces résultats nous indiquent que grâce à l'architecture multi-échelle, ModelNet [Wu et al., 2015] en pré-entraînement est suffisant pour calculer des descripteurs pertinents.

5.2.5.5.3 Robustesse aux rotations aléatoires Les méthodes basées U-Net ne sont pas désignées pour être robustes aux rotations aléatoires. Effectivement en général, ces méthodes ne sont pas invariantes par rotation, mais peuvent devenir robustes grâce à l'augmentation de données. Pendant nos entraînements, nous rajoutons des rotations aléatoires dans toutes les directions. Pour tester la robustesse, nous testons également MS-SVConv sur le jeu de données test de 3DMatch. Le tableau 5.5 montre que les résultats sont presque équivalents sur le jeu de données 3DMatch avec ou sans rotation.

5.2.5.5.4 Influence du nombre de points Pour toutes les expériences précédentes sur le jeu de données 3DMatch [Zeng et al., 2017] et ETH [Pomerleau et al., 2012], nous choisissons 5000 points aléatoirement avant la mise en correspondance. Pour montrer que MS-SVConv calcule toujours

Cible / **Architecture**	ETH	TUM	3DMatch
MS-SVConv(1)	37,4	28,3	29,2
MS-SVConv(3)	**74,1**	**99,3**	**85,0**

TABLEAU 5.4 – FMR ($\tau_2 = 0,05$) de MS-SVConv entraîné sur le jeu de données ModelNet avec une taille de voxel de 2 cm et évalué sur ETH-8-scenes, TUM et 3DMatch sans UDGE

Architecture	Sans rotation	Avec rotation
MS-SVConv(1)	87,2	87,0
MS-SVConv(3)	89,9	89,7

TABLEAU 5.5 – FMR sur 3DMatch ($\tau_2 = 0,2$) dans un cadre supervisé avec et sans rotation aléatoire dans le jeu de données test

des descripteurs pertinents, nous avons également varié le nombre de points choisi. Le tableau 5.6 montre que même si nous choisissons 250 points, nous avons un FMR de 96,3 soit une perte de 2,1%. À l'exception de la méthode basée *patch* GeDI [Poiesi and Boscaini, 2021], toutes les autres méthodes ont une plus grosse perte de performance lorsque le nombre de points pour faire les correspondances diminue. Cette expérience montre que les descripteurs calculés par MS-SVConv sont pertinents en chaque point, grâce à l'architecture multi-échelle qui vient rajouter plus de contextes.

Nombre de points	5000	2500	1000	500	250	Av.
Perfect Match [Gojcic et al., 2019]	94,7	94,2	92,6	90,1	82,9	90,9
FCGF [Choy et al., 2019]	95,2	95,5	94,6	93,0	89,9	93,6
D3Feat(rand) [Bai et al., 2020]	95,3	95,1	94,2	93,6	90,8	93,8
D3Feat(pred) [Bai et al., 2020]	95,8	95,6	94,6	94,3	93,3	94,7
SpinNet [Ao et al., 2020]	97,6	97,5	97,3	96,3	94,3	96,6
GeDI [Poiesi and Boscaini, 2021]	97,9	97,7	**97,6**	97,2	**97,3**	**97,5**
MS-SVConv(3)	**98,4**	**97,9**	97,0	**97,9**	96,3	**97,5**

TABLEAU 5.6 – Influence du nombre de points sur le FMR en apprentissage supervisé sur 3DMatch (FMR avec $\tau_2 = 0,05$).

5.2.5.5.5 Résultats sur 3DLoMatch 3DLoMatch [Huang et al., 2021] est un jeu de données qui contient les mêmes nuages de points que 3DMatch sauf que les paires de nuages de points pour l'évaluation sont différentes. En effet, 3DLoMatch se concentre sur les paires de nuages de points qui ont un taux de recouvrement assez faible (10-30%). Nous comparons MS-SVConv avec Predator [Huang et al., 2021] (une méthode très récente) sur 3DMatch et 3DLoMatch. Predator est une méthode sophistiquée qui utilise des réseaux convolutionnels sur des graphes et un module d'attention après l'encodeur pour calculer des descripteurs pertinents même lorsque le recouvrement est faible. Au contraire, MS-SVConv utilise une simple architecture multi-échelle. Nous comparons MS-SVConv(3) sur 3DMatch et 3DLoMatch. Le tableau 5.7 montre que MS-SVConv est meilleur que 3DMatch et 3DLoMatch (on peut noter que contrairement à Predator, MS-SVConv n'utilise pas de détecteur pour trouver des points d'intérêt, mais seulement des points aléatoires). Nous pouvons conclure de ces expériences que MS-SVConv a des bonne performances de recalage, même si le recouvrement entre les deux nuages de points est faible.

5.2.5.6 Impact de MS-SVConv et UDGE

	3DMatch	3DLoMatch
Predator [Huang et al., 2021]	96,6	73,0
MS-SVConv(3)	**98,4**	**77,2**

TABLEAU 5.7 – Comparaison du FMR entre Predator [Huang et al., 2021] et MS-SVConv(3) sur le jeu de données 3DMatch et 3DLoMatch dans un cadre supervisé.

Apprentissage non supervisé sur ETH				
	ETH 4-scenes	**ETH 8-scenes**		
Méthodes	FMR (%)	FMR (%)	SRE	Temps (s)
Méthodes sans Apprentissage				
FPFH [Rusu et al., 2009]	22,1	0,75	85,1	0,71
Méthodes basées patch				
3DSmoothNet [Gojcic et al., 2019]	79,0	-	-	-
Multiview [Li et al., 2020]	92,3	42,6	44,0	56,0
DIP [Poiesi and Poiesi, 2021]	92,8	**93,9**	**6,9**	8,27
GeDI [Poiesi and Boscaini, 2021]	98,2	-	-	-
Méthodes basées U-Net sans UDGE				
D3Feat [Bai et al., 2020]	61,6	64,5	95,0	0,43
MS-SVConv(1)	34,9	56,4	151,0	0,24
MS-SVConv(3)	71,8	76,8	82,2	0,52
Méthodes basées U-Net avec UDGE				
MS-SVConv(1)	88,0	87,5	44,0	**0,16**
MS-SVConv(3)	**98,9**	93,6	**6,9**	0,40

TABLEAU 5.8 – FMR et SRE médian x1000 sur le jeu de données ETH avec des méthodes entraînées sur 3DMatch [Zeng et al., 2017]. ETH 4-scenes montre le protocole de Gojcic [Gojcic et al., 2019], et les résultats des autres méthodes sont des résultats publiés dans les articles correspondants aux méthodes (seul le FMR est disponible). ETH 8-scenes suit le protocole de Fontana [Fontana et al., 2021], et les autres méthodes sont évaluées avec le code disponible en ligne. « MS-SVConv(1) » signifie MS-SVConv sur une échelle et « MS-SVConv(3) », MS-SVConv sur trois échelles. Nous reportons uniquement le temps d'extraction des descripteurs.

5.2.5.6.1 Résultats sur le jeu de données ETH Le tableau 5.8 nous permet de voir les résultats sur le jeu de données ETH [Pomerleau et al., 2012] avec ou sans UDGE après un pré-entraînement sur 3DMatch. ETH est un jeu de données difficile parce que la densité de points varie dans l'espace et il n'y a qu'une vue partielle de la scène. Sans UDGE, MS-SVConv avec une échelle a un FMR de 34,9% sur ETH 4-scenes et 56,4% FMR sur ETH 8-scenes. MS-SVConv sur trois échelles a +36,9% FMR sur ETH 4-scenes et +20,4% FMR d'améliorations sur ETH 8-scenes sans UDGE, seulement grâce à l'architecture multi-échelle. Cette expérience montre que l'impact de l'architecture multi-échelle sur la généralisation est déjà grand, ce qui est très utile pour le recalage dans un cas où nous recevons les paires à recaler à la volée.

Avec UDGE, MS-SVConv(3) est l'état de l'art avec 98,9% FMR sur ETH-4-scenes et 93,6% FMR sur ETH 8-scenes comme la meilleure méthode basée *patch*, DIP [Poiesi and Poiesi, 2021], tout en étant 20 fois plus rapide. Nous pouvons remarquer qu'il y a une synergie entre l'architecture multi-échelle et UDGE, avec +18,9% et +6,1% d'améliorations du FMR entre une et trois échelles. Ajouter des échelles améliore les capacités de généralisation, mais le temps d'entraînement et le temps d'inférence est multiplié par 2,5 (de 0,16 s à 0,40 s comme temps moyen d'extraction de descripteurs). Cependant, cela reste toujours plus rapide que les autres méthodes testées.

Apprentissage non supervisé sur ETH		
Source	FMR (%) (sans UDGE)	FMR (%) (avec UDGE)
ModelNet	74,1	93,4
3DMatch	76,8	93,6

TABLEAU 5.9 – Résultats de MS-SVConv(3) selon le jeu de données source S. ETH 8-scenes est le jeu de cible.

5.2.5.6.2 Influence du jeu de données source De manière inattendue, grâce à l'approche proposée multi-échelle et la génération de données pour de l'apprentissage par transfert, un pré-entraînement sur le jeu de données synthétique ModelNet [Wu et al., 2015] est suffisant pour obtenir de bons résultats sur le jeu de données réel ETH [Pomerleau et al., 2012] (voir le tableau 5.9). Ce résultat montre que MS-SVConv et UDGE peuvent s'adapter à des jeux de données complètement différents même si le jeu de données source n'a rien à voir avec le jeu de données cible.

Source \ **Cible**	ETH	TUM	3DMatch
∅	0,0	16,0	60,3
ModelNet	93,4	100	96,5
3DMatch	93,6	99,7	97,8

TABLEAU 5.10 – UDGE sur les jeux de données ETH, TUM, et 3DMatch sans pré-entraînement (∅) ou avec pré-entraînement sur ModelNet ou 3DMatch. Les résultats sont le FMR en % avec MS-SVConv(3).

Le tableau 5.10 nous permet de conclure que le pré-entraînement joue un rôle très important. UDGE a de mauvais résultats sur les petits jeux de données (0,0% sur ETH et 16,0% sur TUM). Cependant, le pré-entraînement sur ModelNet est suffisant pour avoir des résultats proche de l'état de l'art, grâce à UDGE sans aucune vérité terrain pour les jeux de données cible (ETH, TUM, 3DMatch). Cela montre que UDGE apporte une amélioration des résultats même quand le jeu de données cible est petit.

Apprentissage non supervisé sur le jeu de données ETH									
	Scènes avec UDGE				Scènes sans UDGE				
Méthodes	Haupt.	Stairs	Plain	Apart.	Gaz. Sum.	Gaz. Wint.	Wood Aut.	Wood Sum.	Moy.
MS-SVConv(1)	53	93	76	100	85	99	84	89	84,9
MS-SVConv(3)	72	99	88	100	96	100	99	99	94,1

TABLEAU 5.11 – FMR avec $\tau_2 = 0,05$ par scène sur le jeu de données ETH 8-scenes [Fontana et al., 2021]. MS-SVConv a été pré-entraîné sur ModelNet, UDGE a uniquement été appliqué sur les scènes Hauptegebaude, Stairs, Plain et Appartment.

5.2.5.6.3 Est-ce que UDGE permet la généralisation sur des scènes complètement inconnues ? Dans toutes les expériences, UDGE consiste à prendre le jeu de données test et à créer artificiellement des paires de nuages de points pour que le réseau de neurones s'adapte à un nouveau jeu de données. Même si nous utilisons le jeu de données test pour l'apprentissage par transfert, le protocole proposé est valide, car UDGE n'utilise pas la vérité terrain du jeu de données test. Mais à quel point les descripteurs transférés généralisent-ils à des scènes inconnues ?

Pour voir si les résultats restent similaires, nous coupons en deux le jeu de données test ETH : après un pré-entraînement sur ModelNet, nous appliquons UDGE sur seulement quatre scènes du jeu de données ETH (Plain, Stairs, Hauptgebaude, Apartment) et nous évaluons la méthode proposée sur les quatre autres (Gazebo Summer et Winter, Wood Autumn et Summer). Nous utilisons

les mêmes hyper-paramètres que l'entraînement précédent (avec un nombre d'époques de 400 au lieu de 200). Le tableau 5.11 montre que même si MS-SVConv n'a jamais vu Gazebo ou Wood, il peut quand même généraliser à des scènes inconnues.

Nous avons réalisé une autre expérience sur 3DMatch pour montrer que UDGE apprend des descripteurs pertinents. Premièrement, nous entraînons sur le jeu de données synthétique ModelNet. Ensuite, nous appliquons notre stratégie d'apprentissage par transfert non supervisé sur UDGE sur le jeu de données d'entraînement de 3DMatch. Nous évaluons MS-SVConv sur le jeu de données test de 3DMatch. Le tableau 5.12 nous permet de conclure que MS-SVConv entraîné de manière non supervisée est assez similaire à MS-SVConv entraîné de manière supervisé. Cette deuxième expérience est une preuve supplémentaire que UDGE n'est pas juste en train de sur-apprendre sur le jeu de données cible.

Méthode	FMR ($\tau_2 = 0,05$)	FMR ($\tau_2 = 0,2$)
Supervisé	98,4	89,9
UDGE	96,7	85,0

TABLEAU 5.12 – Comparaison entre l'apprentissage supervisé et l'apprentissage non supervisé sur 3DMatch. Pour le cas non supervisé : le modèle est pré-entraîné sur ModelNet et entraîné sur le jeu de données d'entraînement de 3DMatch avec UDGE (sans utiliser les poses vérité terrain de 3DMatch). Le modèle est MS-SVConv(3).

rognage	échantillonnage périodique	FMR (%)
		88,4
✓		91,4
✓	✓	**93,4**

TABLEAU 5.13 – Influence de la génération de données proposée dans UDGE : MS-SVConv est pré-entraîné sur ModelNet et UDGE est appliqué sur le jeu de données ETH 8-scenes.

5.2.5.6.4 Influence de la génération de données proposée Nous avons réalisé des expériences pour voir l'influence de la génération de données proposée de UDGE. Après un pré-entraînement sur ModelNet, nous appliquons UDGE sur le jeu de données ETH sans rognage et sans échantillonnage périodique (on utilise le même nuage de points comme paires de la même manière que [Yuan et al., 2021]). Le tableau 5.13 montre que le rognage est très important dans la génération de données. Si nous ne réalisons pas cette opération, les performances chutent de 91,4% à 88,4% sur le FMR. L'échantillonnage aléatoire aussi apporte une amélioration. Cette expérience montre que la génération de données contribue à l'amélioration des résultats.

5.2.5.6.5 To share or not to share? Nous comparons les résultats du multi-échelle avec des poids partagés et sans poids partagés entre les U-Net. Le tableau 5.14 montre que quand nous ne partageons pas les poids (trois U-Net différents pour les trois échelles), les capacités de généralisation sont plus faibles qu'avec des poids partagés. En effet, le gain n'est que de +11% en FMR avec les poids non partagés, contre +20,4% avec les poids partagés quand on teste sur le jeu de données ETH [Pomerleau et al., 2012] sans UDGE. Une raison qui peut expliquer que les poids partagés sont meilleurs est que sans poids partagé, chaque U-Net va se spécialiser sur une seule échelle, alors que lorsqu'ils sont partagés, un U-Net apprend sur plusieurs échelles, il apprend à être robuste à l'échelle.

Nous pouvons remarquer qu'avec UDGE, les résultats sont similaires entre les poids partagés et les poids non partagés. Mais MS-SVConv avec des poids partagés est quand même meilleur. Cette expérience explique pourquoi nous avons choisi de partager les poids entre les U-Net.

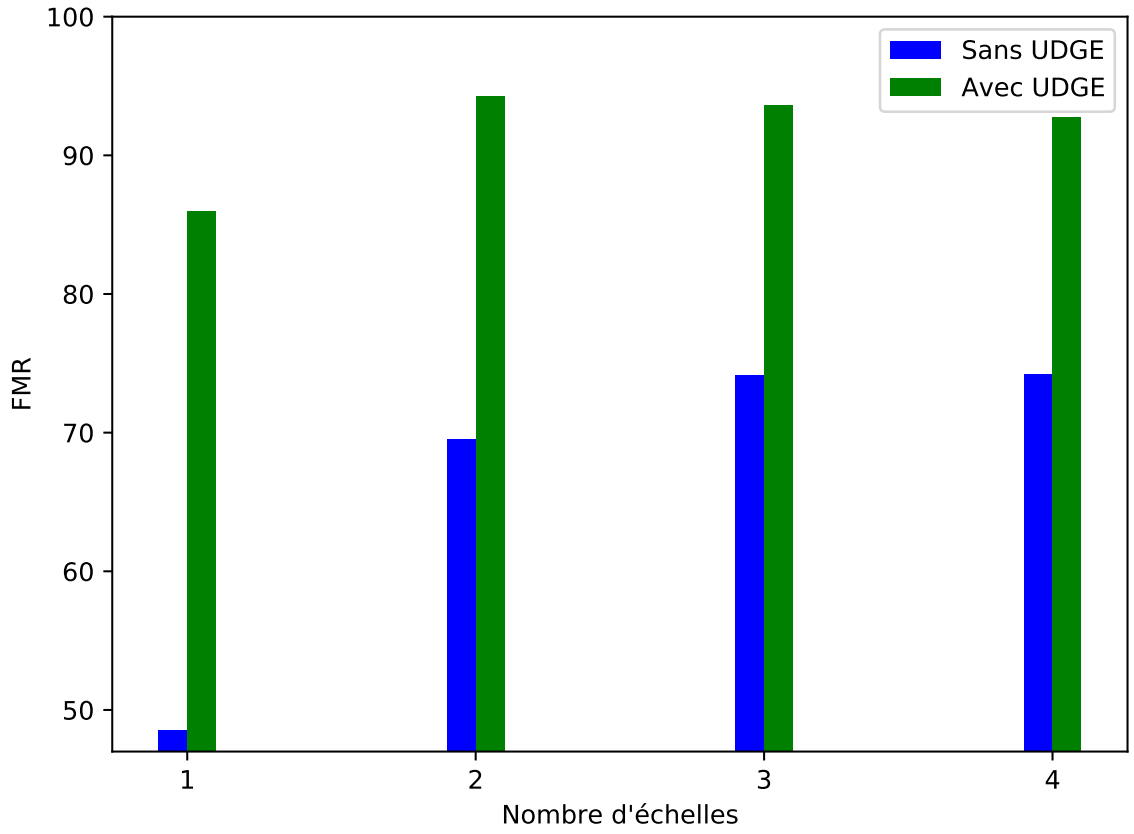


FIGURE 5.4 – FMR sur ETH 8-scenes en fonction du nombre d'échelles avec UDGE et sans UDGE (modèle pre-entraîné sur le jeu de données ModelNet).

ETH 8-scenes Dataset			
Méthodes	FMR (%)	SRE	Temps (s)
sans UDGE			
MS-SVConv(1)	56,4	151,0	0,24
MS-SVConv(3) Unshared	67,4	129,6	0,50
MS-SVConv(3)	76,8	82,2	0,52
avec UDGE			
MS-SVConv(1)	87,5	44,0	**0,16**
MS-SVConv(3) Unshared	93,5	33,1	0,335
MS-SVConv(3)	**93,6**	**6,9**	0,40

TABLEAU 5.14 – FMR et SRE médian x1000 sur le jeu de données ETH 8-scenes avec un réseau pré-entraîné sur 3DMatch. « MS-SVConv(1) » signifie MS-SVConv avec 1 échelle et « MS-SVConv(3) », MS-SVConv avec 3 échelles. Unshared veut dire que les U-Net ont des poids différents pour chaque échelle. Nous affichons également le temps moyen d'extraction des descripteurs.

5.2.5.6.6 Influence du nombre d'échelles pour MS-SVConv Dans les expériences précédentes sur les jeux de données 3DMatch et ETH, nous avons montré qu'un réseau avec un U-Net sur trois échelles différentes est meilleur qu'un réseau avec un U-Net sur une seule échelle dans un cadre supervisé et non supervisé. Mais quelle est l'influence du nombre d'échelles ? Pour répondre à cette question, nous avons entraîné plusieurs versions de MS-SVConv avec un nombre variable d'échelles. Nous avons calculé le FMR sur ETH-8-scenes avec ou sans UDGE. Comme dans les expériences précédentes, la taille de voxel est fixée à 2 cm sans UDGE et à 4 cm avec UDGE (la taille du voxel est toujours multipliée par 2 par rapport à l'échelle précédente). La figure 5.4 montre le FMR en fonction du nombre d'échelles. Ces résultats montrent que plus le nombre de têtes est grand, plus nous sommes capables de généraliser (sans UDGE) entre ModelNet et le jeu de données ETH (les résultats diminuent à partir de 4 échelles). Avec UDGE, seulement deux échelles sont suffisantes pour avoir des descripteurs pertinents pour ETH avec UDGE. Trois têtes sont un bon compromis pour avoir une bonne capacité de généralisation sur de nouveaux jeux de données, tout en ayant de bonnes performances avec UDGE et tout en gardant un temps d'inférence correct (trois échelles sont 2,5 plus lents qu'une échelle).

Taille de voxel (cm)	2	4	6
MS-SVConv(1)	67,3	87,5	91
Taille de voxel (cm)	2, 4, 8	4, 8, 16	6, 12, 24
MS-SVConv(3)	90,9	93,6	93,8

TABLEAU 5.15 – Influence de la taille du voxel quand on applique notre méthode non supervisée UDGE sur ETH-8-scenes. Les résultats sont exprimés avec les FMR en % avec des réseaux pré-entraînés sur 3DMatch avec une taille de voxel de 2 cm pour MS-SVConv(1) et des tailles de voxel de 2, 4 et 8 cm pour MS-SVConv(3).

5.2.5.6.7 Influence de la taille de voxel sur UDGE Les réseaux de neurones basés sur les convolutions parcimonieuses sont en général sensibles au choix de la taille de voxel. MS-SVConv ne fait pas exception. C'est pourquoi, lorsque nous appliquons MS-SVConv sur un nouveau jeu de données, nous ne pouvons pas changer la taille de voxel. Cependant, grâce à l'apprentissage par transfert non supervisé, nous pouvons changer la taille de voxel pour le jeu de données cible. Par exemple par rapport à 3DMatch, ETH a des scènes plus grandes avec une densité qui peut être moins grande ce qui nous pousse à augmenter la taille de voxel pour améliorer la qualité des descripteurs. Nous observons effectivement ce phénomène avec le tableau 5.15 où nous avons pré-entraîné MS-SVConv sur 3DMatch et transféré de manière non supervisée sur le jeu de données ETH. Avec une seule échelle, en gardant la même taille de voxel que durant le pré-entraînement,

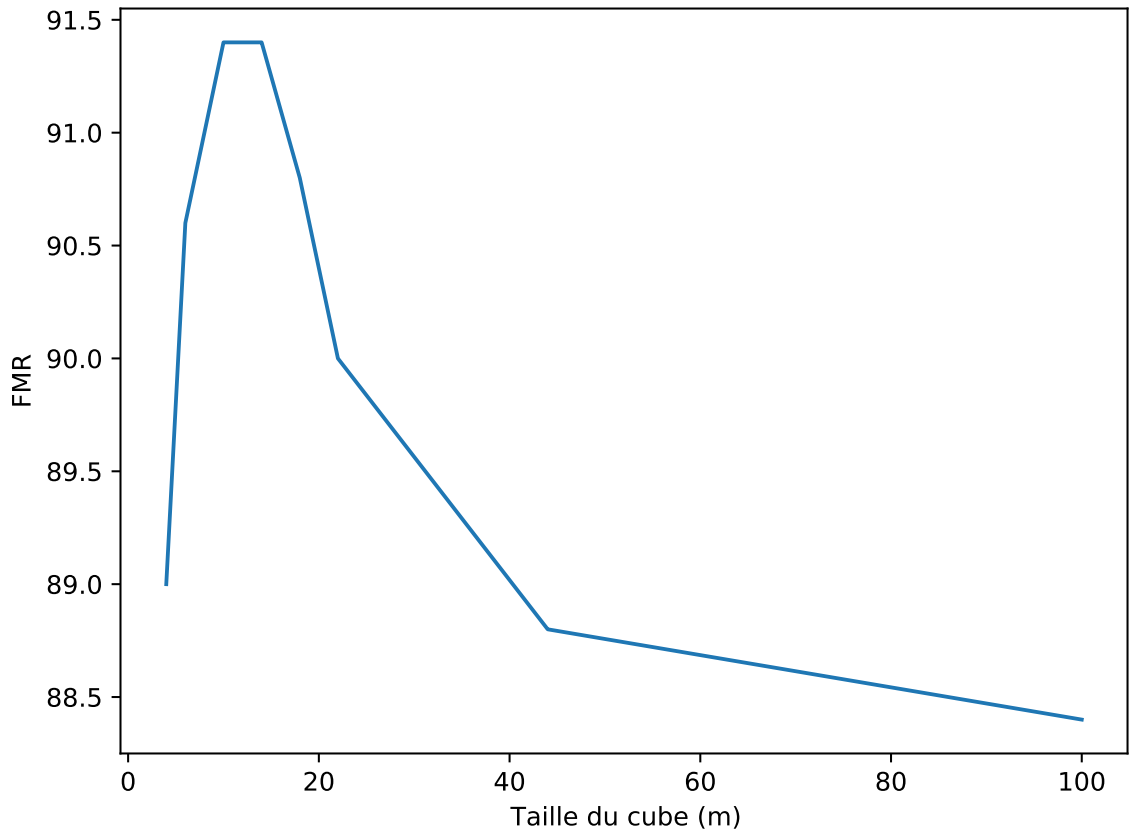


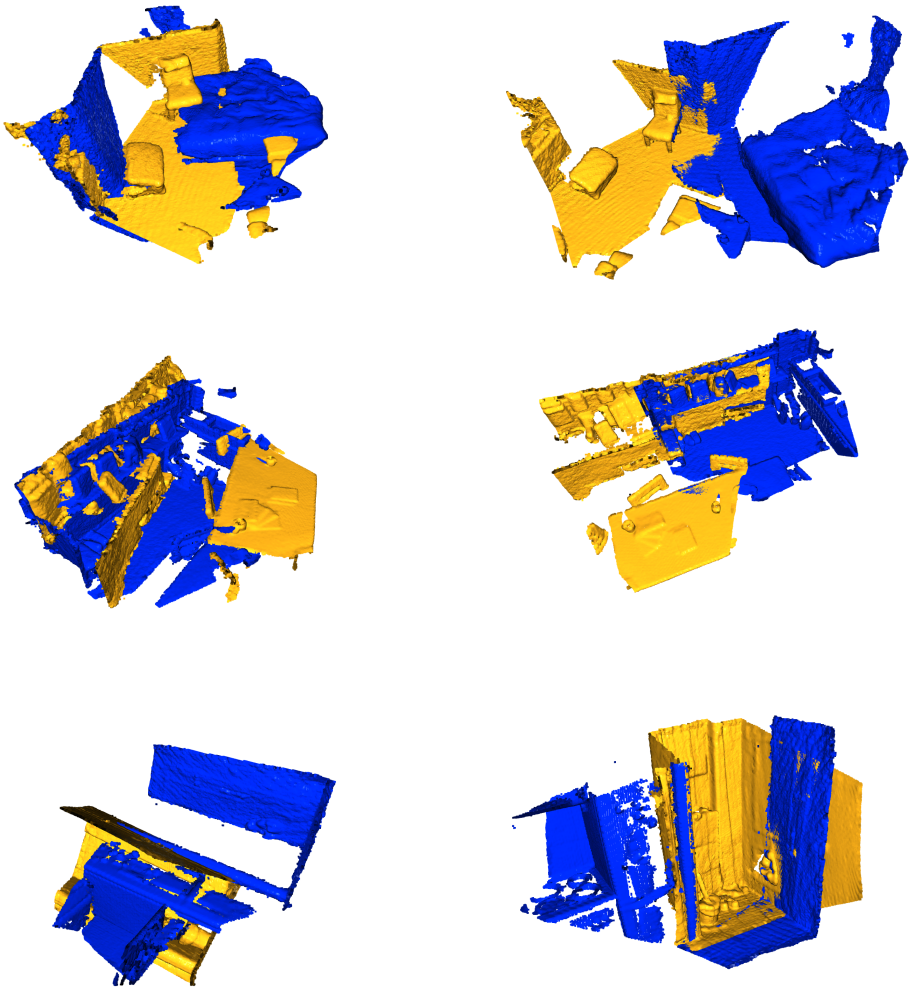
FIGURE 5.5 – FMR de MS-SVConv(3) sur ETH 8-scenes selon la taille du rognage de la génération de données proposée pour UDGE. Les modèles sont pré-entraînés sur ModelNet. Ces expériences ont été réalisées sans échantillonnage périodique.

le réseau ne génère pas des descripteurs pertinents. En revanche, grâce à UDGE, en ayant une taille de voxel de 6 cm, MS-SVConv à une échelle obtient de meilleurs résultats. En rajoutant deux échelles supplémentaires, nous remarquons que MS-SVConv devient moins sensible au choix de ce paramètre et obtient de bons scores (90% de FMR) même sans changer la taille de voxel entre 3DMatch et le jeu de données cible ETH.

5.2.5.6.8 Influence de la taille du rognage pour UDGE Nous avons également mesuré l'influence de la taille du rognage sur la génération de données proposées pour UDGE (voir figure 5.5). La figure 5.5 montre le FMR selon la taille du rognage sur ETH-8-scenes (les modèles sont pré-entraînés sur ModelNet). Une grande taille de rognage équivaut à ne pas avoir de rognage du tout. Cette expérience montre que ce paramètre est important pour UDGE et doit être choisi selon le jeu de données.

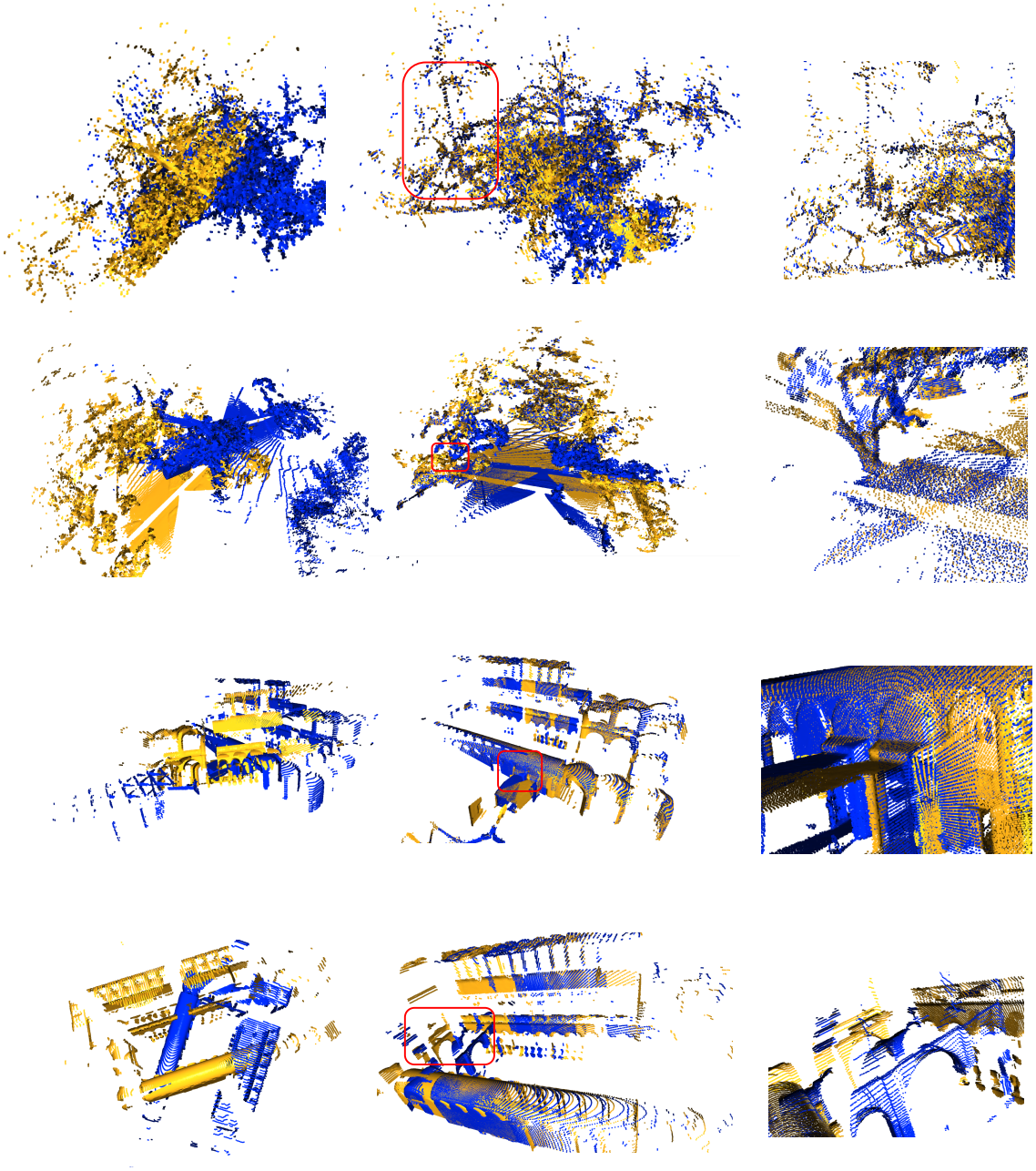
5.2.5.7 Résultats qualitatifs

Nous montrons dans les figures 5.6, 5.7 et 5.8 des résultats qualitatifs du recalage en utilisant MS-SVConv(3) pour les jeux de données 3DMatch, ETH, et TUM.



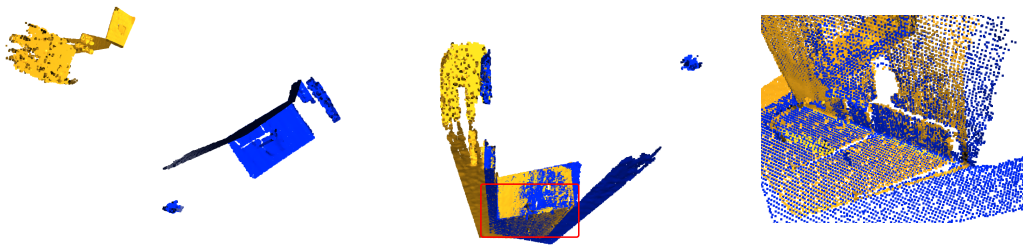
Initialisation MS-SVConv 3 échelles + TEASER

FIGURE 5.6 – Résultats qualitatifs sur 3DMatch (apprentissage supervisé). Même sans structure et avec un recouvrement faible, MS-SVConv(3) contribue à trouver la bonne transformation entre les scènes.



Initialisation MS-SVConv 3 échelles + TEASER

FIGURE 5.7 – Résultats qualitatifs sur le jeu de données ETH (modèles pré-entraînés sur 3DMatch et *fine-tunés* avec UDGE sur ETH). La dernière ligne montre un échec de MS-SVConv(3).



Initialisation MS-SVConv 3 échelles + TEASER

FIGURE 5.8 – Résultats qualitatifs sur le jeu de données TUM (modèles pré-entraînés sur 3DMatch et *fine-tunés* avec UDGE sur TUM).

5.3 Conclusion

Nous proposons MS-SVConv, une architecture multi-échelle, ainsi que UDGE, une méthode d'apprentissage par transfert non supervisé. Nous constatons une synergie entre ces deux modules ce qui permet d'avoir des résultats compétitifs sur des jeux de données réelles issues de capteurs RGBD et LiDAR. Mieux encore, nous pouvons entraîner MS-SVConv sur des données synthétiques et avec UDGE, généraliser sur des données réelles.

Ainsi, cette méthode est un petit pas vers une méthode de recalage qui peut s'adapter à tous types de capteurs. Dans le cas des monnaies ou autres objets archéologique, cette méthode a un potentiel si les archéologues ou numismates décident de changer de système d'acquisitions (un capteur moins cher ou plus rapide).

Chapitre 6

Conclusions et perspectives

Cette thèse avait pour but de proposer des algorithmes pour le classement automatique des monnaies d'un trésor selon leur coin.

Tout d'abord, nous avons décrit et analysé la manière dont les numismates travaillent pour classer les monnaies selon leur coin, de la méthode de Colbert de Beaulieu jusqu'aux analyses manuelles des modèles 3D. Nous en avons conclu que la reconnaissance de coin passe par du recalage. Avant d'établir une méthode, nous avons proposé Riedones3D, un jeu de données d'acquisitions 3D de monnaies annotées par coin pour l'évaluation du recalage et du regroupement de monnaies par coin. Ce jeu de données offre de nombreuses perspectives. Pour les chercheurs en vision par ordinateur, il peut servir à développer et tester de nouveaux algorithmes de reconnaissance de motifs sur des nuages de points 3D. Pour les numismates, ces scans 3D sont très utiles afin de mieux comprendre les techniques de fabrication et le style du graveur. Enfin, Riedones3D a des applications en muséographie. Il sera possible de mieux transmettre au public les prouesses de l'art celtique.

Comme le recalage est un problème fondamental, nous y avons consacré un chapitre en explicitant l'état de l'art. Nous avons vu que la plupart des algorithmes de recalage étaient conçus pour du recalage de forme et non de motifs. C'est pourquoi, nous avons proposé une méthode de recalage basée sur l'ICP fonctionnant sur Riedones3D. Nous avons comparé différentes méthodes, dont des méthodes basées sur l'apprentissage profond. Nous en avons conclu que les méthodes basées sur l'apprentissage profond avec un jeu de données d'entraînement constitué de 200 monnaies de droits ont de meilleures performances que les autres méthodes. Nous avons donc retenu cette méthode qui offrait les meilleurs résultats sur Riedones3D.

Ensuite, nous avons proposé une nouvelle chaîne de traitement pour classer les monnaies selon leurs coins basées sur les méthodes de recalage. Grâce à la distance point à point entre les monnaies, nous pouvons estimer une probabilité que deux monnaies soient du même coin. Nous avons évalué la méthode proposée sur Riedones3D avec des résultats très satisfaisants sur les droits et sur les revers en matière de regroupement. Nous avons également fait une analyse des cas extrêmes qui montrent des pistes d'améliorations de la méthode proposée. Cette analyse montre également les découvertes possibles sur la fabrication des coins. Nous avons également montré d'autres applications concrètes de ce classement qui sont la reconstitution de coins virtuels ou l'estimation du nombre de coins pour ce monnayage. Nous avons utilisé la méthode proposée de classement de monnaies par coin sur d'autres types d'objets comme les motifs du casque d'Agris ou ceux du fourreau de Moscano di Fabriano. Les méthodes de recalage basées sur l'apprentissage profond ont des capacités de généralisation pour les motifs du fourreau de Moscano di Fabriano. Pour le casque d'Agris, les méthodes de recalage ont échoué, car les motifs à recaler se sont révélés être très différents. Comme nous l'avons montré, la méthode proposée fonctionne dans une certaines mesure sur de nombreux motifs même s'ils sont très différents de ceux utilisés pour l'entraînement. La méthode proposée a donc un énorme potentiel pour le regroupement automatique de trésors volumineux selon leur coins ou plus généralement, le regroupement de motifs selon leur poinçon.

La méthode de recalage utilisée semble fonctionner lorsque les motifs testés sont différents des motifs avec lesquels le modèle a été entraîné, à condition que système d'acquisition soit toujours le même. Or, la généralisation à d'autres types de capteurs est encore un domaine actif de recherche. La plupart des méthodes ont des capacités de généralisation limitées sur des données issues de capteurs différents. Nous en avons déduit que la méthode proposée allait être limitée si les données sont issues de capteurs différents. C'est pourquoi nous avons proposé une méthode pour généraliser les méthodes de recalage à des jeux de données différents. La méthode de généralisation proposée est basée sur une architecture multi-échelle **MS-SVConv** pour s'adapter à des densités variables et sur une méthode d'apprentissage par transfert non supervisé appelé **UDGE**. Nous avons montré que la méthode proposée a de meilleurs résultats que les autres méthodes de l'état de l'art tout en restant relativement rapide en inférence. Nous avons également observé que **MS-SVConv** peut être entraîné sur un jeu de données synthétique puis, avoir de bons résultats sur des scans LiDAR grâce à **UDGE**, montrant l'intérêt de la méthode de recalage proposée.

Ces travaux laissent de nombreuses questions ouvertes. Une piste de recherche est de pouvoir réduire le nombre de comparaisons pour améliorer la recherche par coin. Si nous collectons des bases de données plus volumineuses, il serait envisageable d'avoir un réseau de neurones convolutionel qui calcule un descripteur global par motif comme ClareNet [ClaReNet, 2021]. Une autre piste de recherche est l'amélioration de l'identification de coins lors de cas limites. Par exemple, la méthode proposée peut être améliorée dans le cas où les monnaies sont déformées ou trop usées. Il est également envisageable de faire de la recherche sur la reconstruction de coins automatique. Mais une autre piste de recherche et développement très importante est l'amélioration de l'interface utilisateur. Si nous voulons que les algorithmes puissent être utilisés, il est important de réfléchir à une mise en production de ces algorithmes et à la mise en place d'une interface simple facilement exploitable par les principaux intéressés.

Bibliographie

[Aagaard and Märcher, 2015] Aagaard, S. and Märcher, M. (2015). The microscope drawing tube method (mdtm) - an easy and efficient way to make large scale die studies. *The Numismatic Chronicle (1966-)*, 175 :249–262. 19

[Aiger et al., 2008] Aiger, D., Mitra, N. J., and Cohen-Or, D. (2008). 4-points congruent sets for robust pairwise surface registration. *ACM Trans. Graph.*, 27(3) :1–10. 40, 51

[American Numismatic Society, 2020] American Numismatic Society (2020). Coins and computation : New developments in the computer-aided die study. [Online ; accessed 13-July-2021]. 19

[Anwar et al., 2021] Anwar, H., Anwar, S., Zambanini, S., and Porikli, F. (2021). Coinnet : Deep ancient roman republican coin classification via feature fusion and attention. *Pattern Recognit.*, 114 :107871. 24

[Ao et al., 2020] Ao, S., Hu, Q., Yang, B., Markham, A., and Guo, Y. (2020). SpinNet : Learning a general surface descriptor for 3d point cloud registration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11753–11762. 41, 45, 47, 49, 50, 55, 93, 98, 99, 102, 103

[Aoki et al., 2019] Aoki, Y., Goforth, H., Srivatsan, R. A., and Lucey, S. (2019). Pointnetlk : Robust and efficient point cloud registration using pointnet. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7156–7165. 9, 45, 46, 47, 48, 51

[Arandjelović and Zachariou, 2020] Arandjelović, O. and Zachariou, M. (2020). Images of roman imperial denarii : A curated data set for the evaluation of computer vision algorithms applied to ancient numismatics, and an overview of challenges in the field. *Sci*, 2(4). 24, 25

[Babin et al., 2019] Babin, P., Giguère, P., and Pomerleau, F. (2019). Analysis of robust functions for registration algorithms. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 1451–1457. 35, 37, 44, 53

[Bai et al., 2020] Bai, X., Luo, Z., Zhou, L., Fu, H., Quan, L., and Tai, C.-L. (2020). D3feat : Joint learning of dense detection and description of 3d local features. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6358–6366. 41, 45, 46, 47, 48, 50, 93, 94, 98, 99, 101, 102, 103, 104

[Balntas et al., 2017] Balntas, V., Lenc, K., Vedaldi, A., and Mikolajczyk, K. (2017). Hpatches : A benchmark and evaluation of handcrafted and learned local descriptors. In *CVPR*. 48

[Barath and Matas, 2018] Barath, D. and Matas, J. (2018). Graph-cut RANSAC. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6733–6741. ISSN : 2575-7075. 43

[Barath et al., 2019] Barath, D., Matas, J., and Noskova, J. (2019). MAGSAC : Marginalizing sample consensus. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10189–10197. ISSN : 2575-7075. 43

[Belongie et al., 2001] Belongie, S., Malik, J., and Puzicha, J. (2001). Shape context : A new descriptor for shape matching and object recognition. In Leen, T., Dietterich, T., and Tresp, V., editors, *Advances in Neural Information Processing Systems*, volume 13. MIT Press. 42

[Besl and McKay, 1992] Besl, P. J. and McKay, N. D. (1992). A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2) :239–256. 10, 32, 34, 43, 52, 58, 61

[Biber and Strasser, 2003] Biber, P. and Strasser, W. (2003). The normal distributions transform : a new approach to laser scan matching. In *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453)*, volume 3, pages 2743–2748 vol.3. 37

[Black and Rangarajan, 1996] Black, M. J. and Rangarajan, A. (1996). On the unification of line processes, outlier rejection, and robust statistics with applications in early vision. *Int J Comput Vision*, 19(1) :57–91. 44

[Bouaziz et al., 2013] Bouaziz, S., Tagliasacchi, A., and Pauly, M. (2013). Sparse iterative closest point. *Computer Graphics Forum*, 32(5) :113–123. 35, 37

[Bounou et al., 2020] Bounou, O., Monnier, T., Pastrolin, I., SHEN, X., Benevent, C., Limon-Bonnet, M.-F., Bougard, F., Aubry, M., Smith, M. H., Poncet, O., and Raverdy, P.-G. (2020). A Web Application for Watermark Recognition. *Journal of Data Mining & Digital Humanities*, volume 335 title. 90

[Cakir et al., 2019] Cakir, F., He, K., Xia, X., Kulis, B., and Sclaroff, S. (2019). Deep metric learning to rank. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1861–1870. 25

[Campbell and Petersson, 2016] Campbell, D. and Petersson, L. (2016). Gogma : Globally-optimal gaussian mixture alignment. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5685–5694. 40

[Caron et al., 2018] Caron, M., Bojanowski, P., Joulin, A., and Douze, M. (2018). Deep clustering for unsupervised learning of visual features. In *European Conference on Computer Vision*. 25

[Chaton et al., 2020] Chaton, T., Chaulet, N., Horache, S., and Landrieu, L. (2020). Torch-points3d : A modular multi-task frameworkfor reproducible deep learning on 3d point clouds. *arXiv :2010.04642 [cs, stat]*. 88, 98

[Chen and Medioni, 1992] Chen, Y. and Medioni, G. (1992). Object modelling by registration of multiple range images. *Image and Vision Computing*, 10(3) :145–155. Range Image Understanding. 35

[Cherner, 2021] Cherner, S. (2021). Une intelligence artificielle pour reconstituer les fresques de pompéi. *Le Figaro*. [Online ; accessed 01-March-2022]. 13

[Chetverikov et al., 2002] Chetverikov, D., Svirko, D., Stepanov, D., and Krsek, P. (2002). The trimmed iterative closest point algorithm. In *2002 International Conference on Pattern Recognition*, volume 3, pages 545–548 vol.3. 37

[Choi et al., 2015] Choi, S., Zhou, Q.-Y., and Koltun, V. (2015). Robust reconstruction of indoor scenes. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5556–5565. 32

[Choy et al., 2020] Choy, C., Dong, W., and Koltun, V. (2020). Deep global registration. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2511–2520. 26, 45, 48, 50, 51, 53

[Choy et al., 2019] Choy, C., Gwak, J., and Savarese, S. (2019). 4d spatio-temporal ConvNets : Minkowski convolutional neural networks. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3070–3079. IEEE. 45, 46, 98

[Choy et al., 2019] Choy, C., Gwak, J., and Savarese, S. (2019). 4d spatio-temporal convnets : Minkowski convolutional neural networks. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3070–3079. 56

[Choy et al., 2019] Choy, C., Park, J., and Koltun, V. (2019). Fully convolutional geometric features. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 10, 41, 45, 47, 48, 50, 51, 56, 58, 59, 63, 68, 69, 74, 82, 90, 92, 93, 94, 98, 99, 102, 103

[Chum and Matas, 2005] Chum, O. and Matas, J. (2005). Matching with PROSAC - progressive sample consensus. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 220–226 vol. 1. ISSN : 1063-6919. 43

[ClaReNet, 2021] ClaReNet (2021). Eaa 2021 clarenet. [Online ; accessed 03-March-2022]. 25, 116

[Cloud Compare, 2013] Cloud Compare (2013). Cloud compare. 11, 21, 23, 80, 88

[Colbert De Beaulieu, 1949] Colbert De Beaulieu, J.-B. (1949). L'atelier monétaire de la Rennes celtique et la trouvaille de Saint-Jacques-de-la-Lande. *Rennes : Société d'histoire et d'archéologie de Bretagne*, page 24. 19, 20

[Curless and Levoy, 1996] Curless, B. and Levoy, M. (1996). A volumetric method for building complex models from range images. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '96, page 303–312, New York, NY, USA. Association for Computing Machinery. 26, 51, 57

[D3, 2011] D3 (2011). D3.js - data-driven documents. https ://d3js.org. Accessed : 2022-02-13. 90

[Dai et al., 2020] Dai, A., Diller, C., and Niessner, M. (2020). SG-NN : Sparse generative neural networks for self-supervised scene completion of RGB-d scans. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 846–855. ISSN : 2575-7075. 45

[Dai et al., 2017] Dai, A., Nießner, M., Zollöfer, M., Izadi, S., and Theobalt, C. (2017). Bundlefusion : Real-time globally consistent 3d reconstruction using on-the-fly surface re-integration. *ACM Transactions on Graphics 2017 (TOG)*. 98

[Danelljan et al., 2016a] Danelljan, M., Meneghetti, G., Khan, F. S., and Felsberg, M. (2016a). Aligning the dissimilar : A probabilistic method for feature-based point set registration. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 247–252. 38

[Danelljan et al., 2016b] Danelljan, M., Meneghetti, G., Khan, F. S., and Felsberg, M. (2016b). A probabilistic framework for color-based point set registration. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1818–1826. ISSN : 1063-6919. 38

[De Callatay, 1993] De Callatay, F. (1993). Estimation du nombre originel de coins : en augmentant l'échantillon... *Acta numismàtica, 21-23*, pages 31–48. 81

[De Callatay, 2007] De Callatay, F. (2007). L'historique de l'étude des liaisons de coins (xviiie-xxe siècle). *Bulletin de la Société française de numismatique*. 18

[de Jersey, 2018] de Jersey, P. (2018). What next for the coriosolitae? some implications of le châtillon ii. *Monnaies et archéologie en Europe celtique : Mélanges en l'honneur de Katherine Gruel*, pages 145–148. 18, 25

[Dellenbach et al., 2021] Dellenbach, P., Deschaud, J.-E., Jacquet, B., and Goulette, F. (2021). Ct-icp : Real-time elastic lidar odometry with loop closure. 32, 34

[Deng et al., 2018] Deng, H., Birdal, T., and Ilic, S. (2018). Ppf-foldnet : Unsupervised learning of rotation invariant 3d local descriptors. In Ferrari, V., Hebert, M., Sminchisescu, C., and Weiss, Y., editors, *Computer Vision – ECCV 2018*, pages 620–638, Cham. Springer International Publishing. 46, 48, 49

[Deng et al., 2018] Deng, H., Birdal, T., and Ilic, S. (2018). Ppfnet : Global context aware local features for robust 3d point matching. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 195–205. 47, 48, 49, 50, 98, 102

[Django Software Foundation, 2005] Django Software Foundation (2005). Django. 90

[Dong et al., 2017] Dong, Z., Yang, B., Liu, Y., Liang, F., Li, B., and Zang, Y. (2017). A novel binary shape context for 3d local surface description. *ISPRS Journal of Photogrammetry and Remote Sensing*, 130 :431–452. 26

[Eckart et al., 2018] Eckart, B., Kim, K., and Kautz, J. (2018). Fast and accurate point cloud registration using trees of gaussian mixtures. *arXiv :1807.02587 [cs]*. 38

[Esty, 1986] Esty, W. W. (1986). Estimation of the Size of a Coinage : a Survey and Comparison of Methods. *The Numismatic Chronicle (1966-)*, 146 :185–215. Publisher : Royal Numismatic Society. 18, 80, 81

[Evangelidis et al., 2014] Evangelidis, G. D., Kounades-Bastian, D., Horaud, R., and Psarakis, E. Z. (2014). A generative model for the joint registration of multiple point sets. In Fleet, D., Pajdla, T., Schiele, B., and Tuytelaars, T., editors, *Computer Vision – ECCV 2014*, Lecture Notes in Computer Science, pages 109–122. Springer International Publishing. 38

[Fischer-Bossert, 1999] Fischer-Bossert, W. (1999). *Chronologie der Didrachmenpragung von Tarent 510-280 v. Chr.* W. de Gruyter, Berlin. 19

[Fischler and Bolles, 1981] Fischler, M. A. and Bolles, R. C. (1981). Random sample consensus : A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6) :381–395. 43, 44, 48, 50, 94, 99

[Fontana et al., 2021] Fontana, S., Cattaneo, D., Ballardini, A. L., Vaghi, M., and Sorrenti, D. G. (2021). A benchmark for point clouds registration algorithms. *Robotics and Autonomous Systems*, 140 :103734. 8, 58, 98, 99, 101, 104, 105

[Fowlkes and Mallows, 1983] Fowlkes, E. B. and Mallows, C. L. (1983). A method for comparing two hierarchical clusterings. *Journal of the American Statistical Association*, 78(383) :553–569. 74

[Gao and Tedrake, 2019] Gao, W. and Tedrake, R. (2019). FilterReg : Robust and efficient probabilistic point-set registration using gaussian filter and twist parameterization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11095–11104. 38, 53

[Geiger et al., 2012] Geiger, A., Lenz, P., and Urtasun, R. (2012). Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. 26, 32, 47, 48, 51

[Gelfand et al., 2003] Gelfand, N., Ikemoto, L., Rusinkiewicz, S., and Levoy, M. (2003). Geometrically stable sampling for the icp algorithm. In *Fourth International Conference on 3-D Digital Imaging and Modeling (3DIM)*, pages 260–267. 35

[Gelfand et al., 2005] Gelfand, N., Mitra, N. J., Guibas, L. J., and Pottmann, H. (2005). Robust global registration. In *Proceedings of the third Eurographics symposium on Geometry processing*, SGP '05, pages 197–es, Goslar, DEU. Eurographics Association. 42, 51

[Gojcic et al., 2019] Gojcic, Z., Zhou, C., Wegner, J. D., and Wieser, A. (2019). The perfect match : 3d point cloud matching with smoothed densities. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5540–5549. 8, 9, 43, 45, 47, 48, 49, 50, 51, 55, 98, 99, 101, 102, 103, 104

[Good, 1953] Good, I. J. (1953). The population frequencies of species and the estimation of population parameters. *Biometrika*, 40(3-4) :237–264. 81

[Grinberg, 2018] Grinberg, M. (2018). *Flask web development : developing web applications with python.* " O'Reilly Media, Inc.". 90

[Groueix et al., 2018] Groueix, T., Fisher, M., Kim, V. G., Russell, B., and Aubry, M. (2018). AtlasNet : A Papier-Mâché Approach to Learning 3D Surface Generation. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 48

[Gruel, 1981] Gruel, K. (1981). Le trésor de trébry (côtes-du-nord) - ier siècle avant notre ère (Études de numismatique celtique 1). *Collection de l'Institut des Sciences et Techniques de l'Antiquité*. 20, 21

[Gruel et al., 2003] Gruel, K., Matterne, V., and Villard, A. (2003). Contextes archéologiques et restes carpologiques associés à des dépôts monétaires armoricains. *Revue archéologique de l'Ouest*, 10 :37–42. 18

[Gruel et al., 2017] Gruel, K., Nieto-Pelletier, S., Demierre, M., and Hiriart, E. (2017). Évaluation des indices de métallurgie monétaire au second âge du Fer. In Marion, S., Deffressigne, S., and Kaurin, J., editors, *Production et proto-industrialisation aux âges du Fer : perspectives sociales et environnementales. Actes du 39e colloque international de l'Association française pour l'étude de l'âge du Fer (Nancy, 14-17 mai 2015)*, volume Mémoires, pages 497–520. Ausonius Éditions. 18

[Gruel and Tangue, 2020] Gruel, K. and Tangue, M. (2020). *Detur dignissimo Studies in honour of Johan Van Heesch*, chapter Le portrait barbu des Riedones, pages 125–146. Centre d'études numismatiques - European Centre for Numismatic Studies. 20, 22, 23, 33

[Guo et al., 2020] Guo, Y., Wang, H., Hu, Q., Liu, H., Liu, L., and Bennamoun, M. (2020). Deep learning for 3d point clouds : A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1. 46

[Hadsell et al., 2006] Hadsell, R., Chopra, S., and LeCun, Y. (2006). Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742. ISSN : 1063-6919. 94

[Halber and Funkhouser, 2017] Halber, M. and Funkhouser, T. (2017). Fine-to-coarse global registration of rgb-d scans. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6660–6669. 98

[Han et al., 2018] Han, X., Jin, J. S., Xie, J., Wang, M., and Jiang, W. (2018). A comprehensive review of 3d point cloud descriptors. *CoRR*, abs/1802.02297. 41, 42

[He et al., 2016] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778. 56

[Heinecke et al., 2021] Heinecke, A., Mayer, E., Natarajan, A., and Jung, Y. (2021). Unsupervised statistical learning for die analysis in ancient numismatics. *CoRR*, abs/2112.00290. 25, 70, 75

[Hexagon, 2021] Hexagon (2021). 3d reshaper. 23

[Hoffer and Ailon, 2015] Hoffer, E. and Ailon, N. (2015). Deep metric learning using triplet network. In Feragen, A., Pelillo, M., and Loog, M., editors, *Similarity-Based Pattern Recognition*, Lecture Notes in Computer Science, pages 84–92. Springer International Publishing. 94

[Horache et al., 2021a] Horache, S., Deschaud, J.-E., and Goulette, F. (2021a). 3d point cloud registration with multi-scale architecture and unsupervised transfer learning. In *2021 International Conference on 3D Vision (3DV)*, pages 1351–1361. 14, 92, 93

[Horache et al., 2021b] Horache, S., Deschaud, J.-E., Goulette, F., Gruel, K., Lejars, T., and Masson, O. (2021b). Riedones3D : a Celtic Coin Dataset for Registration and Fine-grained Clustering. In Hulusic, V. and Chalmers, A., editors, *Eurographics Workshop on Graphics and Cultural Heritage*, pages 83–92. The Eurographics Association. 14, 20, 29, 32, 53, 68

[Horache et al., 2020] Horache, S., Goulette, F., Deschaud, J.-E., Lejars, T., and Gruel, K. (2020). Automatic clustering of celtic coins based on 3d point cloud pattern analysis. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, V-2-2020 :973–980. 14, 32, 39, 53

[Hu et al., 2020] Hu, Q., Yang, B., Xie, L., Rosa, S., Guo, Y., Wang, Z., Trigoni, N., and Markham, A. (2020). RandLA-net : Efficient semantic segmentation of large-scale point clouds. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11105–11114. IEEE. 46

[Huang et al., 2021] Huang, S., Gojcic, Z., Usvyatsov, M., Wieser, A., and Schindler, K. (2021). Predator : Registration of 3d point clouds with low overlap. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4267–4276. 8, 45, 47, 50, 102, 103, 104

[Huang et al., 2020] Huang, X., Mei, G., and Zhang, J. (2020). Feature-metric registration : A fast semi-supervised approach for robust point cloud registration without correspondences. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11363–11371. 48

[Hubert and Arabie, 1985] Hubert, L. and Arabie, P. (1985). Comparing partitions. *Journal of Classification*, 2(1) :193–218. 74

[Ioffe and Szegedy, 2015] Ioffe, S. and Szegedy, C. (2015). Batch normalization : Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, page 448–456. JMLR.org. 56

[Jain et al., 2014] Jain, N., Bhansali, A., and Mehta, D. (2014). Angularjs : A modern mvc framework in javascript. *Journal of Global Research in Computer Science*, 5(12) :17–23. 90

[Jakob et al., 2015] Jakob, W., Tarini, M., Panozzo, D., and Sorkine-Hornung, O. (2015). Instant field-aligned meshes. *ACM Transactions on Graphics (Proceedings of SIGGRAPH ASIA)*, 34(6). 88

[Jian and Vemuri, 2011] Jian, B. and Vemuri, B. C. (2011). Robust point set registration using gaussian mixture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(8) :1633–1645. 37, 38, 40

[Johnson and Hebert, 1999] Johnson, A. E. and Hebert, M. (1999). Using spin images for efficient object recognition in cluttered 3d scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(5) :433–449. 42, 53

[Kabsch, 1978] Kabsch, W. (1978). A discussion of the solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A*, 34(5) :827–828. 33

[Koppe, 2022] Koppe, M. (2022). L'art et l'archéologie, nouveaux horizons de l'ia. *CNRS Le journal*. [Online ; accessed 01-March-2022]. 13

[Krause et al., 2013] Krause, J., Stark, M., Deng, J., and Fei-Fei, L. (2013). 3d object representations for fine-grained categorization. In *2013 IEEE International Conference on Computer Vision Workshops*, pages 554–561. 25

[Körtgen et al., 2003] Körtgen, M., Park, G.-J., Novotni, M., and Klein, R. (2003). 3d shape matching with 3d shape contexts. In *In The 7th Central European Seminar on Computer Graphics*. 42

[Le et al., 2019] Le, H. M., Do, T.-T., Hoang, T., and Cheung, N.-M. (2019). SDRSAC : Semidefinite-based randomized approach for robust point cloud registration without correspondences. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 124–133. ISSN : 2575-7075. 43

[Lee et al., 2020] Lee, S., Negishi, M., Urakubo, H., Kasai, H., and Ishii, S. (2020). Mu-net : Multi-scale u-net for two-photon microscopy image denoising and restoration. *Neural Networks*, 125 :92–103. 95

[Li et al., 2020] Li, L., Zhu, S., Fu, H., Tan, P., and Tai, C.-L. (2020). End-to-end learning local multi-view descriptors for 3d point clouds. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1916–1925. 49, 50, 101, 102, 104

[Li et al., 2021] Li, X., Pontes, J. K., and Lucey, S. (2021). Pointnetlk revisited. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12763–12772. 48

[Lista, 2019] Lista, L. (2019). A simple numeric algorithm for ancient coin dies identification. 25

[Liu et al., 2017] Liu, W., Wen, Y., Yu, Z., Li, M., Raj, B., and Song, L. (2017). Sphereface : Deep hypersphere embedding for face recognition. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6738–6746. 25

[Liu et al., 2021] Liu, X., Killeen, B. D., Sinha, A., Ishii, M., Hager, G. D., Taylor, R. H., and Unberath, M. (2021). Neighborhood normalization for robust geometric feature learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13049–13058. 50

[Liu, 2007] Liu, Y. (2007). A mean field annealing approach to accurate free form shape matching. *Pattern Recognition*, 40(9) :2418–2436. 38

[Liu et al., 2018] Liu, Y., Wang, C., Song, Z., and Wang, M. (2018). Efficient Global Point Cloud Registration by Matching Rotation Invariant Features Through Translation Search. In Ferrari, V., Hebert, M., Sminchisescu, C., and Weiss, Y., editors, *Computer Vision – ECCV 2018*, Lecture Notes in Computer Science, pages 460–474, Cham. Springer International Publishing. 40

[Lowe, 2004] Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60 :91–110. 48

[Lu et al., 2019] Lu, W., Wan, G., Zhou, Y., Fu, X., Yuan, P., and Song, S. (2019). Deepvcp : An end-to-end deep neural network for point cloud registration. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 12–21. 45, 46

[Marchand, 2014] Marchand, S. (2014). IBISA : Making Image-Based Identification of Ancient Coins Robust to Lighting Conditions. In Klein, R. and Santos, P., editors, *Eurographics Workshop on Graphics and Cultural Heritage - Short Papers / Posters*. The Eurographics Association. 25, 69, 70

[Mellado et al., 2014] Mellado, N., Aiger, D., and Mitra, N. J. (2014). Super 4pcs fast global pointcloud registration via smart indexing. *Computer Graphics Forum*, 33(5) :205–215. 41, 51

[Merkel, 2014] Merkel, D. (2014). Docker : lightweight linux containers for consistent development and deployment. *Linux journal*, 2014(239) :2. 90

[Myronenko and Song, 2010] Myronenko, A. and Song, X. (2010). Point set registration : Coherent point drift. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(12) :2262–2275. 38, 51, 53, 56

[Natarajan et al., 2021] Natarajan, A., Iorio, M. D., Heinecke, A., Mayer, E., and Glenn, S. (2021). Cohesion and repulsion in bayesian distance clustering. 25

[Newcombe et al., 2011] Newcombe, R. A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A. J., Kohi, P., Shotton, J., Hodges, S., and Fitzgibbon, A. (2011). Kinectfusion : Real-time dense surface mapping and tracking. In *2011 10th IEEE International Symposium on Mixed and Augmented Reality*, pages 127–136. 32, 34

[Pavlov et al., 2018] Pavlov, A. L., Ovchinnikov, G. W., Derbyshev, D. Y., Tsetserukou, D., and Oseledets, I. V. (2018). AA-ICP : Iterative Closest Point with Anderson Acceleration. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3407–3412. ISSN : 2577-087X. 35, 37

[Poiesi and Boscaini, 2021] Poiesi, F. and Boscaini, D. (2021). Generalisable and distinctive 3D local deep descriptors for point cloud registration. *arXiv e-prints*, page arXiv :2105.10382. 49, 50, 98, 102, 103, 104

[Poiesi and Poiesi, 2021] Poiesi, F. and Poiesi, D. (2021). Distinctive 3d local deep descriptors. In *IEEE Proc. of Int'l Conference on Pattern Recognition (ICPR)*, Milan, IT. 7, 41, 45, 46, 49, 50, 55, 56, 58, 59, 69, 92, 93, 94, 98, 99, 101, 102, 104

[Pomerleau et al., 2015] Pomerleau, F., Colas, F., and Siegwart, R. (2015). A review of point cloud registration algorithms for mobile robotics. *Foundations and Trends® in Robotics*, 4(1) :1–104. 35, 37

[Pomerleau et al., 2012] Pomerleau, F., Liu, M., Colas, F., and Siegwart, R. (2012). Challenging data sets for point cloud registration algorithms. *The International Journal of Robotics Research*, 31(14) :1705–1711. 26, 49, 50, 59, 92, 93, 97, 98, 99, 101, 102, 104, 105, 106

[Prakhya et al., 2015] Prakhya, S. M., Liu, B., and Lin, W. (2015). B-shot : A binary feature descriptor for fast and efficient keypoint matching on 3d point clouds. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1929–1934. 43

[Qi et al., 2019] Qi, C. R., Litany, O., He, K., and Guibas, L. J. (2019). Deep hough voting for 3d object detection in point clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9277–9286. 45

[Qi et al., 2017] Qi, C. R., Su, H., Mo, K., and Guibas, L. J. (2017). Pointnet : Deep learning on point sets for 3d classification and segmentation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 77–85. 46, 47, 50, 55

[Qi et al., 2017] Qi, C. R., Yi, L., Su, H., and Guibas, L. J. (2017). PointNet++ : Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc. 45, 46, 50

[react, 2013] react (2013). A javascript library for building user interfaces. https ://reactjs.org. Accessed : 2022-02-13. 90

[Roynard et al., 2018] Roynard, X., Deschaud, J.-E., and Goulette, F. (2018). Classification of Point Cloud Scenes with Multiscale Voxel Deep Network. *arXiv e-prints*, page arXiv :1804.03583. 46, 95

[Rusinkiewicz and Levoy, 2001] Rusinkiewicz, S. and Levoy, M. (2001). Efficient variants of the icp algorithm. In *Proceedings Third International Conference on 3-D Digital Imaging and Modeling*, pages 145–152. 35, 37, 52, 53, 56, 57

[Rusu et al., 2009] Rusu, R. B., Blodow, N., and Beetz, M. (2009). Fast point feature histograms (fpfh) for 3d registration. In *2009 IEEE International Conference on Robotics and Automation*, pages 3212–3217. 41, 43, 45, 51, 53, 56, 58, 101, 102, 104

[Rusu et al., 2008] Rusu, R. B., Blodow, N., Marton, Z. C., and Beetz, M. (2008). Aligning point cloud views using persistent feature histograms. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3384–3391. 42

[Salgado, 2016] Salgado, L. (2016). Medieval coin automatic recognition by computer vision. *computer vision*, page 98. 24

[Salti et al., 2014] Salti, S., Tombari, F., and Di Stefano, L. (2014). Shot : Unique signatures of histograms for surface and texture description. *Computer Vision and Image Understanding*, 125 :251–264. 33, 41, 43, 45, 49, 53, 102

[Schlag and Arandjelovic, 2017] Schlag, I. and Arandjelovic, O. (2017). Ancient Roman Coin Recognition in the Wild Using Deep Learning Based Recognition of Artistically Depicted Face Profiles. In *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, pages 2898–2906, Venice. IEEE. 24

[Segal et al., 2009] Segal, A., Haehnel, D., and Thrun, S. (2009). Generalized-icp. In *Robotics : science and systems*, volume 2, page 435. Seattle, WA. 35

[Shen et al., 2019] Shen, X., Efros, A. A., and Aubry, M. (2019). Discovering Visual Patterns in Art Collections With Spatially-Consistent Feature Learning. In *2019 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9278–9287. 13

[Shen et al., 2021] Shen, X., Pastrolin, I., Bounou, O., Gidaris, S., Smith, M., Poncet, O., and Aubry, M. (2021). Large-Scale Historical Watermark Recognition : dataset and a new consistency-based approach. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 6810–6817. ISSN : 1051-4651. 25

[Shotton et al., 2013] Shotton, J., Glocker, B., Zach, C., Izadi, S., Criminisi, A., and Fitzgibbon, A. (2013). Scene coordinate regression forests for camera relocalization in rgb-d images. In *Proc. Computer Vision and Pattern Recognition (CVPR)*. IEEE. 48, 98

[Sipiran and Bustos, 2011] Sipiran, I. and Bustos, B. (2011). Harris 3D : a robust extension of the Harris operator for interest point detection on 3D meshes. *The Visual Computer*, 27(11) :963. 41

[Sipiran et al., 2021] Sipiran, I., Lazo, P., and Lopez, C. (2021). SHREC2021. http ://www.ivan-sipiran.com/shrec2021.html. [Online; accessed 13-July-2021]. 25

[Song et al., 2016] Song, H. O., Xiang, Y., Jegelka, S., and Savarese, S. (2016). Deep metric learning via lifted structured feature embedding. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 25

[Straub et al., 2017] Straub, J., Campbell, T., How, J. P., and Fisher, J. W. (2017). Efficient global point cloud alignment using bayesian nonparametric mixtures. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2403–2412. 40

[Sturm et al., 2012] Sturm, J., Engelhard, N., Endres, F., Burgard, W., and Cremers, D. (2012). A benchmark for the evaluation of rgb-d slam systems. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 573–580. 26, 92, 98, 99, 102

[Su et al., 2015] Su, H., Maji, S., Kalogerakis, E., and Learned-Miller, E. (2015). Multi-view convolutional neural networks for 3d shape recognition. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 945–953. IEEE. 46

[Szeliski, 2010] Szeliski, R. (2010). Computer vision : Algorithms and applications. 13, 43

[Tamaki et al., 2010] Tamaki, T., Abe, M., Raytchev, B., and Kaneda, K. (2010). Softassign and em-icp on gpu. In *2010 First International Conference on Networking and Computing*, pages 179–183. 38, 40

[Tang et al., 2020] Tang, H., Liu, Z., Zhao, S., Lin, Y., Lin, J., Wang, H., and Han, S. (2020). Searching efficient 3d architectures with sparse point-voxel convolution. In Vedaldi, A., Bischof, H., Brox, T., and Frahm, J.-M., editors, *Computer Vision – ECCV 2020*, Lecture Notes in Computer Science, pages 685–702. Springer International Publishing. 45, 46, 98

[Tang et al., 2020] Tang, H., Liu, Z., Zhao, S., Lin, Y., Lin, J., Wang, H., and Han, S. (2020). Searching efficient 3d architectures with sparse point-voxel convolution. In *European Conference on Computer Vision (ECCV)*. 56

[Thomas et al., 2019] Thomas, H., Qi, C. R., Deschaud, J.-E., Marcotegui, B., Goulette, F., and Guibas, L. (2019). KPConv : Flexible and deformable convolution for point clouds. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6410–6419. ISSN : 2380-7504. 45, 46

[three, 2010] three (2010). Three.js - javascript 3d library. https ://threejs.org/. Accessed : 2022-02-13. 90

[Tolksdorf et al., 2017] Tolksdorf, J. F., Elburg, R., and Reuter, T. (2017). Can 3d scanning of countermarks on roman coins help to reconstruct the movement of varus and his legions. *Journal of Archaeological Science : Reports*, 11 :400–410. 18, 24, 25

[Tombari et al., 2010] Tombari, F., Salti, S., and Di Stefano, L. (2010). Unique shape context for 3d data description. In *Proceedings of the ACM Workshop on 3D Object Retrieval*, 3DOR '10, page 57–62, New York, NY, USA. Association for Computing Machinery. 42

[Tsin and Kanade, 2004] Tsin, Y. and Kanade, T. (2004). A Correlation-Based Approach to Robust Point Set Registration. In Pajdla, T. and Matas, J., editors, *Computer Vision - ECCV 2004*, pages 558–569, Berlin, Heidelberg. Springer Berlin Heidelberg. 37

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6000–6010, Red Hook, NY, USA. Curran Associates Inc. 47

[Vinyals et al., 2016] Vinyals, O., Blundell, C., Lillicrap, T., kavukcuoglu, k., and Wierstra, D. (2016). Matching networks for one shot learning. In Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc. 25

[Wah et al., 2011] Wah, C., Branson, S., Welinder, P., Perona, P., and Belongie, S. (2011). The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology. 25

[Wang et al., 2021] Wang, H., Liu, Y., Dong, Z., Wang, W., and Yang, B. (2021). You only hypothesize once : Point cloud registration with rotation-equivariant descriptors. *arXiv :2109.00182 [cs]*. 50

[Wang and Solomon, 2019a] Wang, Y. and Solomon, J. M. (2019a). Deep closest point : Learning representations for point cloud registration. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3522–3531. 9, 45, 46, 47, 48, 51

[Wang and Solomon, 2019b] Wang, Y. and Solomon, J. M. (2019b). Prnet : Self-supervised learning for partial-to-partial registration. In *33rd Conference on Neural Information Processing Systems*. 47, 48

[Wang et al., 2019] Wang, Y., Sun, Y., Liu, Z., Sarma, S. E., Bronstein, M. M., and Solomon, J. M. (2019). Dynamic graph cnn for learning on point clouds. *ACM Trans. Graph.*, 38(5). 47

[Weinmann et al., 2015] Weinmann, M., Jutzi, B., Hinz, S., and Mallet, C. (2015). Semantic point cloud interpretation based on optimal neighborhoods, relevant features and efficient classifiers. *ISPRS Journal of Photogrammetry and Remote Sensing*, 105 :286–304. 45

[Wu et al., 2015] Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., and Xiao, J. (2015). 3d shapenets : A deep representation for volumetric shapes. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1912–1920. 26, 47, 51, 93, 97, 98, 99, 102, 105

[Xie et al., 2020] Xie, S., Gu, J., Guo, D., Qi, C. R., Guibas, L. J., and Litany, O. (2020). Pointcontrast : Unsupervised pre-training for 3d point cloud understanding. In *European Conference on Computer Vision (ECCV)*. 50

[Yadan, 2019] Yadan, O. (2019). Hydra - a framework for elegantly configuring complex applications. Github. 98

[Yang et al., 2020] Yang, H., Antonante, P., Tzoumas, V., and Carlone, L. (2020). Graduated non-convexity for robust spatial perception : From non-minimal solvers to global outlier rejection. *IEEE Robotics and Automation Letters*, 5(2) :1127–1134. 44

[Yang et al., 2020] Yang, H., Shi, J., and Carlone, L. (2020). TEASER : Fast and certifiable point cloud registration. *IEEE Trans. Robot.*, 37(2) :314–333. 44, 53, 69, 99

[Yang et al., 2016] Yang, J., Li, H., Campbell, D., and Jia, Y. (2016). Go-icp : A globally optimal solution to 3d icp point-set registration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(11) :2241–2254. 40, 51, 57

[Yew and Lee, 2020] Yew, Z. J. and Lee, G. H. (2020). Rpm-net : Robust point matching using learned features. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11821–11830. 47, 96

[Yuan et al., 2021] Yuan, Y., Hou, J., Nüchter, A., and Schwertfeger, S. (2021). Self-supervised point set local descriptors for point cloud registration. *Sensors*, 21(2). 96, 106

[Zambanini and Kampel, 2013] Zambanini, S. and Kampel, M. (2013). Coarse-to-Fine Correspondence Search for Classifying Ancient Coins. In Park, J.-I. and Kim, J., editors, *Computer Vision - ACCV 2012 Workshops*, Lecture Notes in Computer Science, pages 25–36. Springer Berlin Heidelberg. 24

[Zambanini et al., 2013] Zambanini, S., Kavelar, A., and Kampel, M. (2013). Improving Ancient Roman Coin Classification by Fusing Exemplar-Based Classification and Legend Recognition. In Petrosino, A., Maddalena, L., and Pala, P., editors, *New Trends in Image Analysis and Processing – ICIAP 2013*, Lecture Notes in Computer Science, pages 149–158. Springer Berlin Heidelberg. 24

[Zambanini et al., 2009] Zambanini, S., Schlapke, M., Kampel, M., and Muller, A. (2009). Historical coins in 3d : Acquisition and numismatic applications. In Perlingieri, C. and Pitzalis, D., editors, *The 10th International Symposium on Virtual Reality, Archaeology and Cultural Heritage VAST*. The Eurographics Association. 24

[Zeng et al., 2017] Zeng, A., Song, S., Nießner, M., Fisher, M., Xiao, J., and Funkhouser, T. (2017). 3dmatch : Learning local geometric descriptors from rgb-d reconstructions. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 199–208. 7, 8, 26, 32, 41, 45, 46, 47, 48, 49, 50, 51, 59, 92, 93, 94, 97, 98, 99, 101, 102, 104

[Zhang et al., 2021] Zhang, Z., Girdhar, R., Joulin, A., and Misra, I. (2021). Self-Supervised Pretraining of 3D Features on any Point-Cloud. *arXiv e-prints*, page arXiv :2101.02691. 96

[Zhao et al., 2019] Zhao, Y., Birdal, T., Deng, H., and Tombari, F. (2019). 3d point capsule networks. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1009–1018. 46, 49

[Zheng et al., 2015] Zheng, L., Shen, L., Tian, L., Wang, S., Wang, J., and Tian, Q. (2015). Scalable person re-identification : A benchmark. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1116–1124. 25

[Zhirong Wu et al., 2015] Zhirong Wu, Song, S., Khosla, A., Fisher Yu, Linguang Zhang, Xiaoou Tang, and Xiao, J. (2015). 3d ShapeNets : A deep representation for volumetric shapes. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1912–1920. IEEE. 46, 48

[Zhong, 2009] Zhong, Y. (2009). Intrinsic shape signatures : A shape descriptor for 3D object recognition. In *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*, pages 689–696. 41

[Zhou et al., 2016] Zhou, Q.-Y., Park, J., and Koltun, V. (2016). Fast global registration. In Leibe, B., Matas, J., Sebe, N., and Welling, M., editors, *Computer Vision – ECCV 2016*, Lecture Notes in Computer Science, pages 766–782. Springer International Publishing. 39, 43, 44, 48, 51, 53

[Zhou et al., 2018] Zhou, Q.-Y., Park, J., and Koltun, V. (2018). Open3D : A Modern Library for 3D Data Processing. *arXiv e-prints*, page arXiv :1801.09847. 53, 101

RÉSUMÉ

Regrouper les monnaies selon leur coin est un problème qui a de nombreuses applications en numismatique. Ce regroupement est crucial pour comprendre l'histoire économique de certains peuples, surtout pour les peuples dont peu de traces écrites existent, comme les peuples celtes. C'est une tâche difficile, qui demande beaucoup de temps et d'expertise. Pourtant, les travaux qui se sont penchés sur l'identification automatique de coins monétaires sont très rares. Cette thèse propose un outil automatique pour savoir si deux motifs ont été imprimés avec le même motif, particulièrement savoir si deux monnaies ont été frappées avec le même coin. Basé sur des algorithmes de recalage basées apprentissage profond, la méthode proposée a permis de classer un trésor d'un millier de monnaies Riedones datant du IIème siècle avant notre ère. Ce trésor nous a permis de constituer un jeu de données annotées d'acquisitions 3D de monnaies appelée Riedones3D. Cette base de données est utile pour les spécialistes en monnaies celtiques, mais également à la communauté vision par ordinateur pour développer de nouveaux algorithmes de reconnaissance de coins monétaires. Des évaluations rigoureuses sur Riedones3D et sur d'autres œuvres celtiques montrent l'intérêt de la méthode proposée. En effet, elle peut s'adapter à des motifs inconnus. Finalement, nous proposons un nouvel algorithme de recalage qui peut s'adapter à n'importe quel type de capteur. Grâce à cet algorithme, il est potentiellement possible pour un spécialiste d'utiliser des capteurs plus rapide ou moins onéreux pour faire l'acquisition des monnaies ou des motifs gravés.

MOTS CLÉS

Nuage de Points 3D, Vision par ordinateur, Reconnaissance de formes, Apprentissage profond, Apprentissage automatique, Art celtique, Archéologie, Numismatique

ABSTRACT

Clustering coins according to their die is a problem that has many applications in numismatics. This clustering is crucial for understanding the economic history of tribes (especially for tribes for whom few written records exist, such as the Celts). It is a difficult task, requiring a lot of times and expertises. However, there is very little work that has been done on coin die identification. This thesis project aims at proposing an automatic tool to know if two patterns have been impressed with the same tool, especially to know if two coins have been struck with the same die. Based on deep learning-based registration algorithms, the proposed method has allowed us to classify a hoard of a thousand Riedone coins dating from the 2nd century BC. This treasure allowed us to build an annotated dataset of 3D acquisitions called Riedones3D. Riedones3D is useful for Celtic coin specialists, but also for the computer vision community to develop new coin die recognition algorithms. Rigorous evaluations on Riedones3D and on other Celtic works show the interest of the proposed method. Indeed, it can be adapted to unknown patterns. Finally, we propose a new registration algorithm that can be adapted to any type of sensor. Thanks to this algorithm, it is potentially possible for a specialist to use faster or less expensive sensors to acquire coins or engraved patterns.

KEYWORDS

3D Point Clouds, Computer vision, Shape recognition, Deep Learning, Machine Learning, Celtic art, Archaeology, Numismatics