Scaling up and evaluating surface reconstruction from point clouds of open scenes

Yanis Marchand

Université Gustave Eiffel

Scaling up and evaluating surface reconstruction from point clouds of open scenes

Thèse de doctorat de l'Université Gustave Eiffel

École doctorale n° 532, Mathématiques et Sciences et Technologies de l'Information et de la Communication (MSTIC)
Spécialité de doctorat: Informatique
Unité de recherche : LASTIG (IGN)

Thèse présentée et soutenue à l'Université Gustave Eiffel, le 28/11/2022, par :

Yanis MARCHAND

Composition du Jury

Pierre GRUSSENMEYER Professeur des universités, INSA Strasbourg	Président du jury
Désiré SIDIBÉ Professeur des universités, Université d'Evry-Paris Saclay	Rapporteur
Anne VERROUST-BLONDET Chargée de recherche, INRIA Paris	Examinatrice
Pooran MEMARI Chargée de recherche CNRS, École Polytechnique	Examinatrice
Cédric DEMONCEAUX Professeur des universités, Université Bourgogne Franche-Comté	Examinateur

Encadrement de la thèse

Bruno VALLET Chargé de recherche HdR, IGN	Directeur de thèse
Laurent CARAFFA Chargé de recherche, IGN	Co-encadrant

Acknowledgements

Three years have passed since I arrived in Paris and stepped into the IGN building for the first time. I would like to express my gratitude to all the people who gave me support and enriched my experience.

This list has to start with my supervisors **Bruno Vallet** and **Laurent Caraffa**. I felt (and I still do!) very lucky to have them as my guides throughout these three years. Laurent helped me settle in at the lab, suggesting activities and giving me tips in order to make things more enjoyable. He was also my cornerman and only supporter at a university boxing competition. I won't forget about that. Bruno helped a lot with establishing project goals, and gave tip top instructions throughout. Our interactions were insightful and I always felt very respected. He is a very kind person who knew when to take more time than he'd planned to provide me with knowledge or support. It was very inspiring to see how he managed my PhD. I learned a lot about human relationships from him and in particular how to be a good supervisor. I feel particularly lucky about having been allowed to give classes on a subject that had nothing to do with my PhD but on which I was keen to gain experience. He also allowed (and encouraged, actually!) me to go to Southampton to work exclusively on a side project for a whole month. Lastly, he's the one who managed to find the right words to cheer me up when I felt under the weather about work or my motivation. Thank you so much Bruno.

A PhD is also about doing some admin work from time to time and as French people, we are quite the specialists in terms of making it hard...!

That's why I have to thank **Sylvie Cach** and **Mariam Sidibé**, who were very supportive and always provided me with quick and clear answers. Thank you both for dealing and having dealt with this many people who, for some reason, find computer programming easier than filling in a form and sending it on time...!
I cannot forget the tremendous help I received from **Alain Sombris**, probably one of the three most important employees at the IGN. This guy can solve any kind of problem to the point of making French administration seem easy to follow!! Thanks for your kindness and patience, Alain!

Mathieu Brédif contributed to one of the papers that we wrote during my PhD and his pieces of advice were very useful. In addition, I greatly appreciated the discussion we had during my IGN annual interview.
Jean-Pierre Papelard helped me use the stationary LiDAR station at the ENSG and Chapter 4 would not be part of this thesis without him. Thank you very much for the time you spent with me, answering my questions and guiding me throughout both the acquisition and the post-processing work.
I had very few interactions with **Emmanuel Clédat** (maybe two or three) but wow, what a dynamic guy! Man, you're so passionate and spread so much positive energy that talking to you always gave me a big boost! Thanks for the work you did with Agisoft Metashape.

None of this would have been possible without two teachers I will never forget. **Olivier Pichon** is a heck of a physics teacher! After he dropped a glue stick and told us that physics was about wondering whether the reason why this object hit the floor was because it was yellow, I was sold! Physics was something I wanted to know about! Hope to see you again soon at "Les poteaux carrés".
Pascale Vérant was my physics teacher in second year of "prépa" and she is the most supportive teacher I've ever had. She always kept an eye on me and kept me on my toes when I was being lazy, dreaming about

anything but physics. That second year was really tough for me but she didn't let me drown. Thank you so much.

I met **Monika Csikos** in September 2022 when I signed up to teach "Algorithm Design" at the ESIEE. She was the PhD student in charge of the module and she provided me with all the support I needed to take on this new challenge!

I want to give a special thanks to **Cédric Demonceaux** who introduced me to the world of computer vision. I was initially seeking an internship in image processing "just to get that skill I lacked to be a real robotics engineer". Cédric hired me and there I was! In Le Creusot! What an experience! Talking about 3D convolution in the morning and going for a run at lunchtime was just the best thing! I swear I miss those hills. Cédric also encouraged Bruno to hire me for the PhD although he wanted us to carry on working together... great man!

In March 2022, I went to Southampton (UK) for a secondment at the Ordnance Survey, the Bri'ish national mapping agency. I had the best time there and this is in large part thanks to the very warm welcome from **David Holland**. David introduced me to everyone and made me feel like a real OS employee, leading me to occasionally forget about being French! I enjoyed the discussions at lunchtimes, our walks around the building, the PieFull Tower at Pie Caramba and our bird watching session on a sunny Saturday mornin'! Top bloke!

I spent most of my time at the IGN sharing an office with **Damien Robert** and **Romain Loiseau**. During covid-related lockdowns and curfews, Damien was one of the very few people I got to see and I am very grateful for those moments we spent together. We were not that many to come to the office every day and I felt very privileged to be able to talk with such a nice guy! Romain started to come more regularly later in the year

and I enjoyed insightful discussions about very different topics including religion, living abroad and international conflicts. I then had the privilege to be invited to his wedding! Damien and Romain offered me a coffee pack for my birthday and what I can tell you is that the cafetiere has been used at least as many times as there are days since I turned 25!

I can strongly recommend the IGN as a great place to work and this is partly thanks to the research group's team of directors. **Sidonie Christophe**, **Ana-Maria Raimond** and **Clément Mallet** are very inclusive and their status does not prevent them from being modest and funny! Clément did not even show his happiness when AJ Auxerre won their promotion match over Saint-Étienne, sealing our relegation to Ligue 2... very thoughtful. I remember Sidonie was one of the first ones to welcome me and make me feel home at the IGN. She has been (and is still) dealing with an unfair situation, being partially excluded from her working environment for being the victim of assault by a colleague. She showed me what it means to be brave and keep battling. Never back down.

Over the last three years I gave quite a few classes at the ENSG, mainly in image processing, and I want to address a big thanks to **Marc Poupée** for all the support he gave me, and for trusting me. He is a very dedicated teacher and students are lucky to have him. He gave his best during the covid pandemic dark times, always seeking better ways of delivering classes. Universities need more people like you!
Arnaud Le Bris also helped me with a class we gave together, providing me with everything I needed to feel confident in front of the students.

Thanks to **Helena** for some nice discussions about Brazil and French culture and for keeping the office tidy. Working in a clean environment definitely helps staying motivated!

Since working for three years in a research group is also about sharing

informal moments with the other people, I'd like to say how much I enjoyed spending time with some of them.

Ameur Zaïbi spent three months at the IGN in autumn 2020 and we started sharing very nice coffee breaks together. We became friends pretty quickly and we were soon able to have deep and meaningful conversations, which I enjoyed a lot. Thanks for everything my friend and good luck.

Katya joined the team (which was pretty small at that time) in January 2021 and we started having coffee together everyday after lunch. That was the beginning of post-lunch coffee breaks! It's become such a tradition now! We had a lot of fun and I really needed that social interaction fix back in the days when I remember being sad on Friday evenings and happy on Monday mornings... Thank you for every minute you spent with me, for taking me to the music festival in June 2021, that meant a lot to me. Cheers La Boss!

Anatol Garioud was the first pal from work I started considering as a friend. For six months I enjoyed our Monday lunchtime debriefs of football matches, and going to that Champion's League match (Paris-SG vs. Dortmund) at your bar was so cool!!! My friends still make fun of me though, saying that I turned into some fan of Neymar or something...

I feel honoured to have met **Paul Chapron** (probably at the coffee room) and man, I feel like I could listen to you for years. You have so much to teach me, and it is so cool to talk with you! Once we realised we both adored the Senseï Hugo TSR, I knew we had some good shit to share with each other! I'm not sure we'll ever get to record that lyrics-based interview but I enjoyed the time we spent writing the script, which we ending up losing because the shared file expired! Stay the same, m8!

Florent Geniet is definitely the one who made me laugh the most! Imagine a fella who always manages to sneak in unexpected jokes... Well it's impossible, forget about it! But still, imagine!! Always positive, I loved spending time with you, Dora!

From our very first interaction, I knew that I would enjoy spending time with **Lâmân Lelégard**. He's a very likeable person who does not follow

the crowd. He is very discrete and modest about what he does best so do not expect him to brag about how well he can draw (I learned that from someone else) or about the last paper he got accepted. Be ready, however, for long explanations about jazz music from the 60's or about how amazing coffee tastes when it is prepared properly. But don't get me wrong, this dude has a lot to teach you, a LOT!
There are so many other people in the lab who I've enjoyed spending time with over the last three years. Thanks to **Mounia Ahmedi**, **Luc Beaudoux**, **Emile Blettery**, **Charles Villard**, **Teng Wu**, **Lulin Zhang**, **Melvin Hersent**, **Grégoire Grzeczkowicz**, **Samuel Mermet**, **Solenn Tual**, **Sami Guembour** and **Mohamed Ali Chebbi** to name but a few.

I want to whack **Helen Mair Rawsthorne** in the spotlight for a million reasons. First, as she was my Bri'ish counterpart at the IGN during my secondment at the Ordnance Survey, she proofread my VOLTA project report, correcting a shit-ton of mistakes and explaining them to me. I feel very grateful that she also proofread this thesis. Not only did she help make the manuscript clearer and easier to read but she also taught me a fuchin' lo'! Well, I'm sure you learned some stuff yourself, you Frenchie! Didn't you no'?? The fact that you pulled out all the stops to help me with my English is only the tip of the iceberg. From the moment we met, I felt really excited about talking to you and getting to know you! Having you around has just made my life so much more enjoyable, you have literally brought light into my life. You have made me feel at home. I feel so lucky that our paths crossed, every day with you is a treat. Wow, what a lucky duck I am. Thank you for making every negative aspect of life so much more manageable. Thank you for making me so happy, for being so supportive, for being my number 1 fan, you are the most wonderful person in the whole wide world!!! Caru ti Helen.

We're getting to the end of the list, so it's time for me to speak a bit about those who have been there for me since way before this all started,

the ones I can rely on no matter what, the ones I want to be able to spend time with for as long as possible.
Massive thanks to my friends with whom I grew up and who are still there today. I have met a lot of people in my life but I have made very few true friends. Thanks to **Jérémy Da Silva**, **Florent Robert**, **Maxime Malmezac**, you are my brothers. Thanks to **Théo Bejannin**, with whom I had a lot of fun in Montreuil. I was sad when you left, buddy!
I didn't make a lot of friends during my studies but **Nabil Mahnane** and **Guillaume Geoffroy** are definitely two who are important to me.
Thanks to **Alexandre Legrand**, you are a truly fantastic and lovable person. Thanks to **Benoît Jourjon** for all these years (22, 23?!) of friendship.

I know my grandfather **Marcel Gaillard** would've loved to attend my PhD defence. I know it would've meant a lot to you. Thank you for all the inspiration, you are forever showing me what tenacity and adaptation could mean!
When I moved to Paris to start the PhD, I felt very lucky that my brother **Théo** was living nearby in Sarcelles. It was the first time were living in the same place as adults and I have some great memories of the weekends we spent together: breakfasts, runs, and watching Rocky ("See ya!")! I miss that stuff. Thank you for all your encouragement and good vibes, brother!
My parents **Anne Gaillard** and **Pierre Marchand** have given me everything, starting with love. They put way more effort into my education than into any personal aspiration of theirs. They did not worry about their own success, but they cared about mine. I'm not a self-made person, I owe them a lot.

Abstract

This PhD tackles two aspects of surface reconstruction from point clouds. Firstly, it addresses the case of large-scale point clouds that are too massive to fit into the memory of one single machine. We present an end-to-end distributed algorithm that can scale up to arbitrarily large point clouds whilst guaranteeing the watertightness of the resulting surface. Secondly, we deal with the assessment of surface reconstruction by proposing two evaluation protocols. The first one requires synthetic data whereas the second one can be set up with data directly obtained from a sensor. These protocols and their respective newly-defined metrics allow the quantification of the quality of surface reconstructions with less bias than previous approaches.

Résumé

Cette thèse de doctorat traite de deux aspects de la reconstruction de surface à partir de nuage de points. Premièrement, elle aborde le cas large échelle où un nuage de points est trop volumineux pour être stocké dans la mémoire d'un seul ordinateur. Nous présentons un algorithme distribué de bout en bout permettant de traiter des nuages de points arbitrairement grands tout en garantissant l'étanchéité de la surface produite. Deuxièmement, cette thèse contribue à l'évaluation de la reconstruction de surface de par la définition de deux protocoles. Le premier nécessite des données synthétiques alors que le deuxième peut être mis en place en ayant uniquement recours à des données provenant de capteurs. Ces protocoles et les nouvelles métriques qui leur sont associées permettent de quantifier la qualité des reconstructions avec un biais moins important que les approches utilisées jusqu'alors.

Contents

List of Figures

List of Tables

Publications

International Conferences

1. Yanis Marchand, Bruno Vallet, Laurent Caraffa. "Evaluating Surface Mesh Reconstruction of Open Scenes." The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences 43 (2021): 369-376.

2. Laurent Caraffa, Yanis Marchand, Mathieu Brédif, Bruno Vallet. "Efficiently distributed watertight surface reconstruction." 2021 International Conference on 3D Vision (3DV). IEEE, 2021.

Journal Paper

Submission: Yanis Marchand, Laurent Caraffa, Raphael Sulzer, Emmanuel Clédat, Bruno Vallet. "Evaluating surface mesh reconstruction using real data", Photogrammetric Engineering & Remote Sensing (PE&RS).

Introduction

Motivation

Surface reconstruction is the task of producing a continuous digital representation of a real surface about which discrete information has been acquired. This information may already be in the form of point clouds produced by a laser scanner. This includes Time-of-flight [2] and Structured-light [3] devices as well as terrestrial and airborne LiDAR [4] that can scan large environments. Point clouds can also be produced from images using Multi-view stereo [5] or Structure from motion [6].

This task has been extensively studied and a tremendous amount of approaches have been proposed as shown in Chapter 1. Nonetheless, two major aspects are still facing a lack of contributions.

Firstly, very few approaches efficiently handle large point clouds as shown in Chapter 1, Section 1.2. With the improvement in sensor capabilities, it has become much easier to acquire massive amounts of visual data in the last few years. In particular, cameras are able to capture more pixels, LiDAR scanners produce more points, and the platforms onto which these device can be embedded have diversified: static, terrestrial and aerial mobile mapping or drones.

As pointed out in [7], *random-access memory (RAM) size* is often a bottleneck when it comes to dealing with large amounts of data. Most surface reconstruction algorithms build the surface all at once by loading the whole point cloud and then creating additional data structures (Delaunay triangulations, voxel grids, octrees, etc.) to represent the output surface. This implies a strong constraint on the maximum input data size for a given amount of RAM. *Processing time* inevitably increases with the number of points to process, at best in a linearly fashion. Even without considering real-time applications, it is important to keep the processing time within reasonable bounds when the data size becomes very large.

The second major aspect of surface reconstruction that has not yet been deeply explored is its evaluation in the case of open scenes. While many methods are available to

build a 3-dimensional mesh from point clouds, there is a lack of relevant metrics to assess the accuracy of these reconstructions. As surface reconstruction is an ill-posed problem, several outputs are possible for the same input point cloud, and they might not even be equivalent in terms of topology (e.g. number of components). How can we know which model is the best? Can we quantify their accuracy? Although it is possible to assess the quality of the models visually, this approach raises several issues. First, it is a subjective comparison and one might be tempted to favour their own or preferred method over others. Secondly, everyone has a different perception of visual quality making it difficult to come to an agreement, even in the absence of a conflict of interest. Thirdly, while it might sound reasonable to visually evaluate a few different models of a relatively small scene, it is very unlikely that one would be able to carry out a large-scale evaluation involving dozens of models representing large areas. Current metrics are not entirely satisfactory as they often depend heavily on the priors used by the algorithms to reconstruct a surface (closed or not, for example) or the ground truth density. This becomes particularly troublesome in the case of open scenes when the ground truth might not be dense and homogeneous, and when the priors have a massive influence on the output surface. It is thus essential to find relevant metrics.

In the context of remote sensing, this topic was first neglected in favour of 2.5D approaches, where the surface reconstruction problem is merely a question of interpolation of possibly sparse height data sampled on a regular grid. This led to the popular Digital Elevation Models (DEMs) and Digital Surface Models (DSMs) used to represent the geometry of the visible surface of a scene seen from above. This representation is, however, becoming more and more limited: an increasing number of applications require terrestrial data (Mobile Mapping Systems, fixed stations and hand-held cameras) for which the 2.5D setting is completely inappropriate. Moreover, the spread of aerial oblique imagery aiming at acquiring more data on vertical surfaces, and the fact that some aerial LiDAR can scan up to 40° away from the vertical, call for more generic 3D models in order to have a continuous geometric representation of the underlying scene. Finally, an increasing number of commercial products such as *Sure* by NFrames or *ContextCapture* by Bentley already offer full 3D processing pipelines for surface mesh reconstruction from remote sensing data. For all these reasons, 3D surface reconstruction, once a topic mainly studied in the geometry processing community, is becoming more and more widespread in remote sensing.

Why has so little work been carried out on the assessment of surface reconstruction of open scenes? One major reason is the complexity of defining an appropriate ground truth. In real-case scenarios when the goal is to produce a digital model of a real object or scene, there is no ground truth other than the real surface itself. It is thus impossible to directly compute the "distance" or the "difference" between a digital model and the ideal real surface. One option is to use synthetic data as in Chapter 3, where a given mesh

is chosen to be the ground truth and is then virtually scanned in a way that simulates the defects of a real acquisition, and then the surface reconstruction algorithms to be evaluated are run on this virtual scan. This makes it more straightforward to compute metrics that assess the difference between the ground truth model and the reconstructed one. Another possibility for working with data representing real scenes is to scan the given scene or object chosen for the evaluation in two different ways such that we can be certain that one scan is significantly "*better*" than the other one. In particular, this high-quality point cloud has to be denser, more accurate and/or contain fewer occlusions. After running the algorithms to be assessed on the low-quality point cloud, the resulting meshes can be evaluated using the additional information contained in the high-quality one.

Definitions and properties of surfaces

A surface is not defined the same way in all branches of mathematics. In topology [8], a **surface** is a two-dimensional manifold (2-manifold).

Definition 0.0.1. An *n-manifold* is a topological space such that each point has a neighbourhood that is homeomorphic to an n-dimensional open disc.

Definition 0.0.2. An *n-manifold with boundary* is a topological space such that each point has a neighbourhood that is homeomorphic to either an n-dimensional open disc or the half-disc.

We can thus define the *boundary* of a manifold as the set of points with half-disc neighbourhoods.

In computer graphics, a triangle mesh is the most common way of representing surfaces, so let us bind the two domains by introducing the concepts that lead to the definition of a triangulated surface.

Definition 0.0.3. An *n-cell* is an n-manifold with boundary with the additional property that its boundary must be divided into a finite number of lower-dimensional cells, called the faces of the n-cell.

In particular:

- A 0-dimensional cell is a point
- A 1-dimensional cell is a segment
- A 2-dimensional cell is a polygon (we will consider it to be a triangle)
- A 3-dimensional cell is a solid polyhedron (we will consider it to be a tetrahedron)

Cells can be bound to each other to form complexes:

Definition 0.0.4. A *complex* K is a finite set of cells $K = \bigcup \{\sigma : \sigma \text{ is a cell}\}$ such that:

1. if σ is a cell in K, then all faces of σ are elements of K.
2. if σ and τ are cells in K, then $Int(\sigma) \cap Int(\tau) = \emptyset$.

The dimension of K is the dimension of its highest-dimensional cell.

Definition 0.0.5. A *simplicial complex* (or *triangulation*) is a 2-complex structure with only triangular cells, and with the additional condition that any two triangles are either connected along a single edge or at a single vertex, or are disjoint.

In all following chapters, we will always refer to triangle meshes as surfaces. Before extending the meaning of this term, let us properly define what we mean by a triangulated surface.

Definition 0.0.6. A *triangulated surface (without boundary)* is a simplicial 2-complex such that:

1. each edge belongs to exactly 2 triangles.
2. the triangles that meet at a vertex can be labelled $T_1, T_2, \ldots, T_n$, with adjacent triangles in this sequence connected along an edge and T_n connected to T_1 along an edge.

A *triangulated surface with boundary* can, however, have edges that belong to only one triangle. The *boundary* is formed by such edges.

In the domain of computer graphics, one may encounter triangle soups (a set of triangles in $\mathbb{R}^3$) that are not triangulated surfaces. It is possible to define a list of vertices with connections between them without the resulting triangle soup respecting the criteria for being a surface. We will thus define some of the properties that triangle soups can have.

Edge-manifold if, for each edge, the set of triangles that share this edge form a topological disk.
Vertex-manifold if, for each vertex, the set of triangles that share this vertex form a topological disk.
Manifold if any point on the mesh has a neighbourhood that is homeomorphic to the Euclidean plane $\mathbb{R}^2$. This is desirable as many mesh-processing algorithms will fail on non-manifold meshes.
Orientable if one can define a consistent continuous orientation of each triangle, meaning that each edge shared by two triangles appears in the opposite order in the index order of each triangle. Once again, mesh processing methods often require orientability,

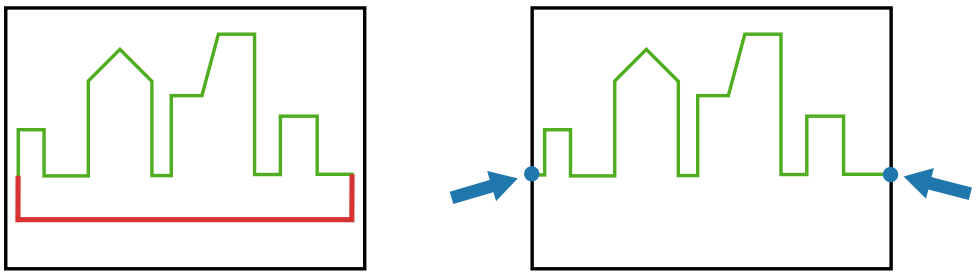


Figure 1: 2D comparison between *hard* watertightness (left) and *soft* watertightness (right). The soft-watertight surface has a border (materialised by the blue dots) but only on the boundary of the domain. When trying to model urban environments, the red piece of surface from the hard-watertight surface does not have any significant meaning so soft-watertightness is often better suited to open scenes.

in particular because the widely-used halfedge data structure has this prerequisite.

Intersection-free if all pairs of triangles that do not share an edge or vertex do not intersect. Self-intersection is always seen as a bad property as real surfaces cannot self intersect.

Watertightness: A surface is watertight if it has no border. In the case of a triangle mesh, this means each edge needs to have exactly two incident faces. We will call this property *hard watertightness*. When trying to reconstruct an open scene, it is often more realistic to authorise the reconstructed surface to intersect the boundary of a domain of interest $\mathcal{B}$ (a bounding box, for instance) as illustrated in Figure 1. We thus define *soft watertightness* as the property that a mesh has when it has no border except on the boundary of the domain $\partial\mathcal{B}$. Only triangle edges lying on the domain boundary can have only one incident face.

Objectives

This thesis aims to address surface reconstruction through the following objectives:

1. Scaling up surface reconstruction by proposing an algorithm that can process arbitrarily-large point clouds.

2. Defining new metrics to evaluate surface reconstruction.

Main Contributions

The main contributions of this thesis are as follows:

1. An end-to-end distributed out-of-core algorithm guaranteeing watertightness and suited to any kind of large-scale point cloud with an arbitrarily-complex 3D geometry.

2. An evaluation protocol to assess surface reconstruction in the case where a ground truth mesh is available. For open scenes, this is only possible by synthetically scanning a realistic mesh, in our case publicly available 3D meshes of open scenes. We thus developed an aerial LiDAR simulator with different scanning patterns and realistic noise, the code of which is open source and available online [1].

3. An evaluation protocol to assess surface reconstruction using only real data, which we applied to public datasets but also to a dataset specifically designed and acquired for this study. This dataset and the code to evaluate a reconstruction has been publicly released[1].

Outline

The dissertation is composed of 4 chapters.

Chapter 1 presents a state of the art of the relevant contributions. In particular, we present surface reconstruction algorithms but also general tools to process massive input data. Moreover, we investigate previous work on the evaluation of surface reconstruction.

In chapter 2, we present a distributed and out-of-core algorithm that is able to process large-scale point clouds. We explain how we managed to parallelise the last step of a reference algorithm, which until now was its main bottleneck.

Chapter 3 presents an evaluation protocol using synthetic data as well as some tools we had to develop to carry out such an assessment, in particular an aerial LiDAR simulator and a high-quality mesh dataset.

Another approach to assess surface reconstruction without the need for synthetic data is proposed in Chapter 4.

Finally, we give the conclusion and perspectives of our work.

[1]Source code is available at: GitHub/SurfaceReconEval

Chapter 1

State of the art

1.1 Surface Reconstruction

Here we review existing methods to reconstruct a triangle mesh from point cloud and classify them by the paradigm they use. See [9] and [10] for an even deeper analysis of them (even though the most recent methods do not appear). Methods evaluated in Chapters 3 and 4 are typesetted in bold.

1.1.1 Indicator function

Often used to achieve watertight reconstructions, this class of algorithms proceed by computing a space segmentation. The object itself is defined as the region of space where the labelling equals a certain value. The surface is then computed by finding the changes in the segmentation. A popular approach in this field is **Poisson** reconstruction [11]. Their indicator function χ is defined as 1 inside the object and 0 outside. They show that χ convolved with a smoothing filter has to respect Poisson equation (1.1) where $\overrightarrow{V}$ is a vector field depending on point locations and the associated normals.

$$\Delta\widetilde{\chi} = \nabla.\overrightarrow{V} \tag{1.1}$$

This differential equation is solved numerically and an adaptation of the *marching cube* algorithm [12] is used to extract a triangle mesh approximating the $\widetilde{\chi} = \gamma$ isosurface, where γ is the average of $\widetilde{\chi}$ at the sample positions. This approach is *screened* in [13] to incorporate additional constraints on sample locations, which significantly improves the resulting quality. This implementation also supports two boundary conditions:

- *Dirichlet* specifies the values that χ needs to take along the boundary of the domain ∂M. Hard watertightness is then enforced by imposing $\chi = 0$ along ∂M.
- *Neumann* specifies the values that $\nabla\chi$ needs to take along ∂M. While this boundary condition also allows hard watertightness, it is less restrictive because it enables the

surface to cross ∂M orthogonally, thereby only imposing soft watertightness.

[14] uses the same input data but their algorithm computes the Fourier coefficients of the indicator function χ, and extracts the surface approximating its isovalue.
A differentiable Poisson solver has been introduced in [15] to further increase the robustness to outliers and to serve as a basis for a trainable model to reconstruct surfaces. They predict the effect that shifting the oriented point samples has on the resulting surface in order to find the surface that best approximates the input point cloud. To do so, they use a bi-directional L2 Chamfer Distance $\mathcal{L}_{CD}$ between the mesh and the point cloud and iteratively shift points to optimise $\mathcal{L}_{CD}$. They also regularly update the input point cloud by sampling the largest mesh component with the objective of discarding outliers.

Recently, lots of *learning-based methods* have been proposed. While they often outperform non-learning-based ones on simple geometries, especially closed objects, they have not been proved as able to deal with the complexity of large and open scenes. *IM-NET* [16] is a learning framework which predicts whether any point (x, y, z) is inside or outside the given shape needing to be reconstructed. The input of their network is the 3 coordinates of a point as well as a feature vector that can be computed using *PointNET* [17].
Occupancy Networks [18] presents a similar way of computing the so-called *occupancy function* of the 3D object. However, instead of concatenating a feature vector to the coordinates of points, they use a batch-sampling strategy. An important advantage of the two latter methods is the arbitrary resolution at which the surface can be extracted.
Convolutional Occupancy Networks [19] introduced a learning-based framework to compute implicit surfaces. Taking as input a point cloud x, their encoder computes features which can be projected in three ways, resulting in a 2D or 3D representation. Given a location $p \in \mathbb{R}^3$, linear interpolation is applied to compute its feature vector $\psi(x, p)$. This vector is known at point cloud locations, allowing for interpolation. Occupancy probability at p is then predicted thanks to a fully-connected network $f_\theta\left(p, \psi(x, p)\right) \in [0, 1]$. A mesh can be extracted by the application of *Multiresolution IsoSurface Extraction (MISE)* [18]. A sliding-window implementation enables the algorithm to process large-scale inputs.
Recently, [20] proposed a general learning framework dubbed *AtlasNet* to take as input a 3D point cloud or an RGB image. It proceeds by concatenating this data with a sampling of a *patch*, namely the unit square, before passing it to multilayer perceptrons (MLPs) with rectified linear unit (ReLU) nonlinearities, producing as output a point cloud of arbitrary resolution. A mesh can be generated either by transferring the connections between vertices of a mesh defined on the patch to their 3D image points or by using Poisson Surface Reconstruction [11] on a sufficiently dense point cloud. A third solution is to sample a 3D sphere instead of patches.

1.1.2 Volumetric Segmentation

This is a sub-discipline of indicator functions as it consists in giving information about whether a region of space is filled by the object or is empty. The data structure can be:

- *the Delaunay Triangulation* of input samples as in [21, 22, 23, 24].
- *voxels*: [25] labels them as *free space*, *occupied* or *unknown*. To achieve this, point locations combined with sensor positions allow computing the ray corresponding to a beam of free space. An interesting feature is that undesirable moving objects such as humans can be erased in the final surface thanks to scans of the same area from different sensor positions.

Robust and Efficient Surface Reconstruction (RESR) [21] label as *inside* or *outside* each tetrahedron of the Delaunay triangulation of the point samples. The triangles separating an *empty*-labelled tetrahedron from an *occupied*-labelled one are extracted thanks to a graph-cut optimisation of an energy function defined thanks to the lines of sight (emanating from the vertex and pointing at the laser scanner) and the shape of the triangles.
Similarly, **Watertight Mesh generation With Uncertainties (WMWU)** [23] label as *occupied* or *empty* each tetrahedron t of the Delaunay triangulation T of the point samples. First of all, a set of *mass functions* m_t are computed. Each m_t represents the likelihood of tetrahedron $t \in T$ being *empty* or *occupied*, or the case in which its occupancy is mostly *unknown*. These are computed based on visibility priors that make use of sensor positions. Binary labels l_t are attributed to each tetrahedron $t \in T$ minimising an energy function (equation 1.2). Denoting $l_T = (l_t)_{t \in T}$ as the labelling of each tetrahedron in the triangulation T and $L = \{0, 1\}$ the set of possible labels, the problem is formulated as:

$$l_T = \underset{l_T \in L^T}{\arg\min} \Big(E_{data}(l_T) \ + \ \lambda \ E_{prior}(l_T) \Big) \tag{1.2}$$

$$E_{data}(l_T) = \sum_{t \in T} \|l_t - m_t\|^2 \tag{1.3}$$

$$E_{prior}(l_T) = \sum_{(t_1, t_2) \in T^2} |t_1 \cap t_2| . \|l_{t_1} - l_{t_2}\|^2 \tag{1.4}$$

The *data* term (equation 1.3) enforces a solution close to the overall mass function and the *prior* term (equation 1.4) minimises the total area of the interfaces. The value of the parameter λ balances the two of them.
Delaunay-Graph Neural Networks (DGNN) [26] also use this paradigm (equation 1.2) to label Delaunay cells of the input point cloud tetrahedralisation but they claim that the problem with previous methods like **RESR** [21] and **WMWU** [23] is that the occupancy of the cells is often estimated using visibility models that are not necessarily

robust to noise. In this context, the main asset of their method is the estimation of the occupancy of the tetrahedra thanks to a graph neural network. They can thus adapt to different complexities by learning on different point cloud configurations. A graph-cut-based optimisation then reinforces global consistency.

1.1.3 Signed-distance function

Another way of generating a watertight surface is to compute the signed-distance function f to the surface and to extract its zero-level set. This is the approach chosen in Multi-level Partition of Unity (MPU) [27] and **Smooth Signed Distance (SSD)** [28].

MPU [27] presents an implicit surface reconstruction algorithm which takes as input a normal-equipped point cloud and produces an approximation $f : \mathbb{R}^3 \longrightarrow \mathbb{R}$ of the signed-distance field from the surface. The surface is then approximated by the zero-level set of this function. The space delimiting the region of interest is subdivided using an adaptive octree: if the accuracy of the shape function within a certain cell is not acceptable, the region is subdivided and a new computation is run. This quality-based behaviour offers the possibility of a trade-off between quality preservation (including sharp features) and computation performance (computation time and memory consumption).
Local shape functions: We denote by Ω the bounded domain of interest included in $\mathbb{R}^3$ which contains the point cloud $\mathcal{P} = \{p_1, \ldots, p_N\}$ equipped with the set of normals $\mathcal{N} = \{n_1, \ldots, n_N\}$. As the computation mostly happens within the octree cells, let us analyse the various calculations that will be done for a given cell of centre c and length of the main diagonal d. A support radius (equation 1.5, α being a smoothing parameter typically equal to 0.75) is associated with the cell and only the set of points within the ball $\mathcal{B}(c, R)$, which we will denote by $\mathcal{P}'$, is considered for further operations.

$$R = \alpha d \tag{1.5}$$

For the sake of clarity, we do not detail the conditions under which the cell is supposed to be subdivided or not, let us just say the number of points within $\mathcal{B}$ has to be sufficient and as mentioned above, the quality of the fitting procedure needs to respect a user-specified threshold.
A local shape function $Q(x)$ is computed based on $\mathcal{P}'$. In order to account for the various conditions of density and distribution of the points, three different types of function are considered:

- a general 3D quadric
- a bivariate quadratic polynomial in local coordinates
- a piecewise quadric surface

Approximation of the global solution: Now that each of these local shape functions Q_i have been estimated, the global signed-distance function f is computed by blending all the contributions, weighted by non-negative compactly-supported functions $\{w_i(x)\}$ such that the union of their supports span the whole domain Ω (equation 1.6).

$$f(x) \approx \frac{\sum_i \omega_i(x)\, Q_i(x)}{\sum_{j=1}^{n} \omega_j(x)} \tag{1.6}$$

SSD [28] consists in using a least-square minimisation of an average of several energy functions $\mathcal{E}_{D_i}(f)$, weighted by coefficients λ_i (equation 1.7).

$$\mathcal{E}_D(f) = \sum_i \lambda_i \mathcal{E}_{D_i}(f) \tag{1.7}$$

The two main energy functions used are those of equations 1.8 and 1.9.

$$\mathcal{E}_{D_0}(f) = \frac{1}{N} \sum_{k=1}^{N} f\left(p_k\right)^2 \tag{1.8}$$

$$\mathcal{E}_{D_1}(f) = \frac{1}{N} \sum_{k=1}^{N} ||\nabla f\left(p_k\right) - n_k||^2 \tag{1.9}$$

Equation 1.8 is precisely the one enforcing the condition $f(x) = 0$ at point locations (near the surface). Equation 1.9 comes from the fact that the gradient of the function f represents the normal field where $f(x) = 0$. Consequently, near the surface, the condition $\nabla f(p_k) = n_k$ should be satisfied. A third energy function involving the Hessian matrix of f enforces that the gradient of f should remain almost constant away from the surface. Following this idea, the surface produced is watertight as long as the function f is continuous.
Recently, *DeepSDF* [29] has shown how to learn the surface distance field. They do so by using the 3D coordinates of a point as well as a *latent vector* that accounts for the type of shape to which the point belongs.

1.1.4 Unsigned-distance function

[30] presents a method for reconstructing large-scale watertight and manifold surfaces using only point locations. They proceed by estimating a confidence map over a predefined volumetric grid V. This function $\phi : v \longrightarrow c \in [0, 1]$ associates to any voxel v its confidence c which can be seen as the pseudo-distance to the nearest point location p. The aim is then to minimise the sum of pseudo-distances over a certain set of voxels

(equation 1.10). An algorithm for extracting the corresponding mesh is also proposed.

$$S_{opt} = \arg\min_{S \subset V} \sum_{v \in S} \phi(v) \tag{1.10}$$

1.1.5 Primitive-based

In this field, *PolyFit* [31] uses RANSAC [32] to detect planar segments and refine them. The surface is extracted by combining the optimisation of an objective function which favours data fitting, point coverage and model complexity and the enforcement of water-tightness and manifoldness.

Point Set Structuring (PSS) [22] relies on the Delaunay triangulation of input points and the labelling of its tetrahedrons as *empty* or *occupied*, but their specificity resides in the extraction of primitives as a pre-processing step, a resampling of the resulting structures and the combination of points from planar regions and unstructured ones in the reconstruction step.

1.1.6 MLS-based

Moving least squares (MLS) was first introduced by Lancaster in [33], based on the work conducted by, amongst others, Shepard in [34]. Since then, a tremendous amount of extensions have been added as pointed out by a survey conducted in 2008 in [35]. For instance, [36], [37] and [38] significantly contributed to the advances in MLS-based algorithms. As explained in [35], MLS-based algorithms can be roughly classified into two main categories:

Implicit MLS surfaces require the computation of a level set function of which the zero isosurface can be extracted.

Projection MLS surfaces consist of first computing a projection operator that maps any point of the space to a point on the surface. The surface is then made of the set of stationary points. [39] defined a projection operator on points equipped with normals (that they call *surfels*). That enabled [40] to propose an out-of-core implementation of an MLS-based explicit surface reconstruction method. Their projector, applied to any point p of the space, produces another point $p*$ and a normal $n*$. $n*$ is defined as the vector that best mimics the normal of the surface patch closest to p. This *vector field* is computed as a weighted average on a neighbourhood of p as follows:

$$n_{PSS}(p) = \sum_i \overrightarrow{n_i}\ \vartheta_N(p, x_i) \tag{1.11}$$

where the weighting function is:

$$\vartheta_N(p, x_i) = \frac{e^{-\frac{\|p-x_i\|^2}{h^2}}}{\sum_j e^{-\frac{\left\|p-x_j\right\|^2}{h^2}}} \tag{1.12}$$

$p*$ is defined as the point that minimises the *energy function* defined by equation (1.13). It is then the nearest point to p that belongs to the surface.

$$e_{PSS}(p) = \sum_i dist(p, x_i, n_i)\ \vartheta_N(p, x_i) \tag{1.13}$$

The pair $(p*, n*)$ enables the definition of a plane $\mathcal{P}$ that is, by construction, the best approximation of the nearest surface to p. We are now able to define the distance between the point p and the surface: it is approximated by the distance between p and its projection onto the plane $\mathcal{P}$.
All this being well-defined, the distance between a point and the surface defines the so-called *scalar field*. An octree is used to implement the out-of-core computing of the surface. This structure is also leveraged at this point: the scalar field is evaluated at the corner of the deepest leaves and if a sign change is detected between two corners of the same leaf, the corresponding voxel is sent to a polygonalisation algorithm to extract the local mesh.

1.1.7 Refinement

These methods are particularly useful when data is too massive or when it needs to be processed on the fly. For example, [41] presents an out-of-core algorithm that interactively processes point clouds that do not fit into memory. Their approach consists in sub-sampling the initial input point cloud P to produce a new, smaller one: P_{rep}. The Delaunay triangulation (DT) of P_{rep} is computed and the *geometric convection* algorithm from [42] allows to reconstruct a simplified version of the surface implied by P. After dividing P into n regions of equal size such that: $P_1 \cup P_2 \cup \cdots \cup P_n = P$, points of each $P_{i,\, i=1,\ldots,n}$ are inserted in the triangulation and a surface refinement algorithm processes them in order to update the reconstructed surface.

1.2 Large-scale reconstruction

As explained in the Introduction, massive point clouds are now available thanks to the rise in sensor capabilities but their processing is challenging mainly because of the increased processing time and the RAM size that is necessary in order to store this data. In this section, we present the work of authors who proposed innovative approaches to limit memory footprint or processing time or both. Work addressing surface reconstruction

specifically are summarised and classified in Table 1.1.

1.2.1 Pioneers

Some authors have pioneered Big Data by **introducing new paradigms** bypassing the typical constraints we discussed. As an example, [43] is an algorithm by Hoppe et al. dating back from 1996 that could be performed in streaming given the locality of the information that is necessary to build the surface. The *ball-pivoting* algorithm [44] was proposed in 1999 and has come as a forerunner in terms of out-of-core surface reconstruction. It is an incremental interpolating algorithm which basic idea is the following: a ball pivots around an edge of an initial triangle until it touches another point. If no other point is in the ball, the three points form a triangle. The process is repeated for another edge and then from another seed triangle until the whole point cloud has been explored. The interesting property is that only a small neighbourhood is considered. We can therefore process the data incrementally. To achieve this, two axis-aligned planes π_1 and π_2 define the so-called active region of work for pivoting. When all edges within this space have been tested, the two planes are shifted, data that is not going to be used again is dumped and new points are loaded according to the new active region of work.

In 2005, [45] introduced a *stream-processing* model that enables to process point clouds *out-of-core*. The idea of the so called *sweep-plane* model is that some operators only need a subset of the whole input data at a given time. Thus, this small number of points is loaded in memory and the function can be performed while the rest of the point cloud stays on disk. This has been the theoretical basis for several algorithms including [46] which is still considered as a standard in this domain.
More precisely, [45] describes the concept of *streaming* as a "sliding window" that defines the fraction of the input data that is *living* ie stored in *in-core* main memory and being involved in some calculation. This concept of a *sweep plane* moving forward along a certain axis and finding points on the way implies that the points stored on disk have to be *sorted* along that axis. This pre-processing step is crucial but obviously cannot be carried out in internal memory. The problem has to be re-thought since classical methods for sorting cannot be used as pointed out by Knuth in [47]. Knuth describes in his book several several solutions to *external sorting*. [48] reviews it as well as a couple improvements.

Assuming that $p_1, \dots, p_n \in \mathbb{R}^3$ is the ordered set of points that we want to process, $\mathbb{A} = \{p_{j-m}, \dots, p_j\}$ is called the *active set*: these points are the only ones to reside in main memory. The condition that the active set is orders of magnitude smaller than the full data set, $|\mathbb{A}| = m \ll n$, ensures that we can process arbitrarily large data sets if it can fit on disk.

Two fundamental definitions are given in [45]. The neighbourhood N_i of a point p_i is

often defined as its k-nearest neighbours or points lying within the sphere centred on p_i and of a given radius.

Definition 1.2.1. A *local operator* $\Phi(p_i)$ performs a function on a point p_i that computes or updates a subset of attributes A_i associated with p_i. As function parameters, $\Phi(p_i)$ only accepts p_i, A_i and a set of points $p_j \in N_i$ within close spatial proximity to p_i (and all their associated attributes A_j).

Definition 1.2.2. A local operator $\Phi_k(p_i)$ is *streamable* if it is computed in one single invocation on p_i and not called recursively on points $p_j \in N_i$. Additionally, the FIFO semantic of its queue Q_k ensures no interference between consecutive operators $\Phi_{k\pm1}$.

Indeed, the framework enables to concatenate several operators Φ_k each of these acting as a sequential FIFO queue buffer Q_k on the point stream. In order to fulfil these requirements, operators need to be organised in a compatible order and semantic rules need to be implemented for each operator buffer based on its class.

Three operators need to be highlighted:

1. The *read operator* $\Phi_R(p_i)$ which reads and buffers one new point. It must be the first operator of the stream.

2. The *deffered-write operator* $\Phi_W(p_i)$ which removes and writes point p_{i-m} to the output stream. It is the final operator.

3. The *neighbourhood operator* $\Phi_X(p_i)$ which computes the set N_i. This one can be implemented in a number of ways depending on the definition of the neighbourhood of a point. The k-nearest neighbours is probably the most famous one.

1.2.2 Standardised frameworks

Message Passing Interface (MPI) [49, 50] is the standardisation of a set of tools enabling the communication between several processes that own only local memory. The goal is then to share the workload amongst several nodes to save time. In the surface reconstruction field, it has been used by [51].

MapReduce [52] is a programming model specifically dedicated to processing massive datasets. Data is represented as *key/value* pairs onto which two types of operation can be carried out: *Map* and *Reduce*. The *Map* operator accepts one pair and produces a set of intermediate pairs that are grouped by key before being sent to the *Reduce* operator which processes the intermediate values.
This paradigm has a *master/slave architecture*: one particular program, the *master*, organises the workload across all nodes of the cluster. To achieve this, it first defines the

data partitions. It also attributes operations to the nodes. The *workers*, processes called by the master, then read the necessary data, compute the task they have been assigned and return the result to the master. In case of failure it is the master's responsibility to organise its re-run. As a consequence, a failure of the master is not tolerated.

Spark [53] is a cluster computing framework implemented in Scala programming language that uses the same master/slave architecture as *MapReduce* [52] but it is better-suited than for two particular types of task:

- **iterative** applications which necessitates to loop through the same data and apply certain operators in parallel.
- **interactive** works where data needs to be accessed on demand.

In both these cases, Spark enables to choose a storage strategy that quickens its access and saves an important amount of time. In addition, two advantageous features of *MapReduce*, scalability and fault-tolerance, are preserved.

Resilient Distributed Datasets A *resilient distributed dataset* (RDD) [54] is an abstraction of a parallel data structure that is fault-tolerant and user-manageable. Essentially, the data is virtually *partitioned* across several machines (the *nodes*) and one can create an RDD by defining an operation on data residing in some stable storage device. The result is not necessarily computed but instead, other operations can be defined on this first RDD, resulting in the definition of new RDDs. Such operations are then called *transformations* and include *flatMap* which is the equivalent of the *Map* of the *MapReduce* paradigm.
As opposed to those "lazy" operations, *actions* actually trigger the computation of necessary pre-defined transformations and either return a result or write data to external memory.

Fault-tolerancy is inherently related to this notion of lazy-computing pattern. Indeed, more than the data itself, its *lineage* is crucial because it enables any partition that has been lost due to a node failure to be retrieved. The set of transformations guarantees that any RDD can be recomputed based on the original data and the log of all the transformations it went through. Moreover, only the lost partition necessitates such a process.

Operations are run based on data locality: by default, the scheduler favours the node that already contains the necessary input data to execute the task. But on top of that, the user can influence the organisation of the work in two ways:

- *persistence* can be enforced that is to say one can choose to replicate an RDD. It is particularly useful if it is meant to be transformed several times because in case

of failure, the recovery can start from the latest persisted RDD instead of going all the way from the beginning.

- one can choose where to store some data by using *partitioning*. For instance, an insightful strategy is to maintain in memory on the same nodes partitions that are going to be combined in an operation.

1.2.3 Methods addressing large-scale reconstruction

The refinement-based method [41] already introduced in Section 1.1.7 enables to incrementally refine the surface by treating each of the sub point clouds separately.

The methods presented in *Parallel Poisson Surface Reconstruction* [55] and *Streaming MLS* [59] are both out-of-core and parallel. The external memory feature of Parallel Poisson [55] comes from the fact that it is based on a previous algorithm [46] while being able to run several computations at the same time is the real contribution of their paper. Their approach consists of formulating the surface reconstruction problem as a Poisson equation as introduced in [11]. However, to enable scalability, they get rid of the necessity to consider the whole point cloud at once and rather partition the space in order to solve the problem locally.
Streaming MLS [59] is based on *Moving Least Squares* [33] and has a very straightforward parallel implementation that enables it to be run on a cluster of PCs. Besides, [40] (introduced in Section 1.1.7) also presented an MLS-based out-of-core algorithm.

[57] is also based on the computation of an implicit function and the extraction of a mesh approximating its isovalue. They compute the function by using *wavelets*. An important result is that their streaming implementation is an order of magnitude faster than Poisson [11] and MPU [27] still using far less memory.

[51] use Non-Uniform Rational B-Spline (NURBS) approximation to compute a surface. The optimisation problem is formulated such that the knot vector T is part of the unknowns. The solution is parallelised using MPI [49] on the *Chemnitzer Linux Cluster* supercomputer [60] and the associated speedup is almost linear.
Recently, [58] proposed an extension of a segmentation based method to large scale data set based on local Delaunay triangulation and local graph-cut distribution. It improves the tiling by optimising tile borders for them to cross regions of low complexity (i.e. flat areas). However, it does not guarantee watertightness along borders which is an important property for many applications. Moreover, [58] only deals with scenes in which a dimension is highly dominated by the two others (e.g. a city scene enables to "partition the scene on the ground plane" as said in their paper).

Method	WT	OOC	SG	PL	ALS	LG	AF	OS
Indicator Function (1 in / 0 out) $\forall x \in \mathbb{R}^3$								
Fourier [14]	✓				voxels		n_i	
Poisson [11]	✓				octree		n_i	✓
Screened Poisson [13]	✓				octree		n_i	✓
Streaming Poisson [46]	✓	✓	✓		octree		n_i	
Parallel Poisson [55]	✓	✓	✓	✓	octree		n_i	
Zhou *et al.* [56]	✓			✓(GPU)	octree		n_i	
IM-NET [16]	✓				voxels	✓		✓
Occupancy Networks [18]	✓			✓	octree	✓		✓
Wavelets [57]	✓	✓	✓		octree		n_i	✓
Volumetric segmentation (1 in / 0 out) of a volumetric discretisation of the object								
Kolluri *et al.* [24]	✓				tetrahedra			
Labatut *et al.* [21]	✓				tetrahedra		l.o.s	
Holenstein *et al.* [25]	✓	✓	✓	✓	voxels + tiles		l.o.s	
Caraffa *et al.* [23]	✓				tetrahedra		l.o.s	
Han *et al.* [58]	✓			✓	tetrahedra + tiles		l.o.s	
Signed-distance function								
MPU [27]	✓				octree		n_i	✓
SSD [28]	✓				octree		n_i	✓
DeepSDF [29]	✓					✓		✓
Unsigned-distance function								
Hornung *et al.* [30]	✓				octree			
Randrianarivony *et al.* [51]				✓				
Primitive-based								
PolyFit [31]	✓				primitives			✓
MLS-based								
Out-of-core MLS [40]		✓			octree		n_i	
Streaming MLS [59]		✓	✓	✓	octree		n_i	
Refinement (iterative)								
Ball-Pivoting [44]		✓	✓					✓
Allegre *et al.* [41]	✓	✓			kd-tree		$\lvert n_i \rvert$	

Table 1.1: Categorisation of surface reconstruction in terms of the properties that are guaranteed as regards watertightness (WT), scaling issue i.e. out-of-core (OOC), streaming (SG) or parallel (PL) implementation as well as the use of an additional localisation structure (ALS) ; learning (LG) capabilities, the necessity of additional features (AF) as normals: "n_i" or lines of sight: "l.o.s" in the computation ; and the open-source (OS) availability of the code.

A particular and interesting way of parallelising algorithms is to make them suitable for a Graphics Processing Unit (GPU) implementation to leverage this specifically-designed hardware which contains a massive set of cores. They are yet often limited to a small portion of algorithms namely Deep-Learning-oriented ones, several of them being introduced in Section 1.1. It is therefore a totally different kind of parallel computation which does not necessarily enable to process arbitrarily large point clouds since they still have to be loaded into the memory of one single machine. As an example, [56] and more recently [61] both offer a GPU implementation of their algorithm.

The methods and the properties they offer are summarised in Table 1.1.

1.3 Evaluation of surface reconstruction

In order to assess the quality of a reconstruction, there is need for a ground truth, an input point cloud and a means of calculating the difference between a given output surface and the so-called ground truth. Let us present the various possibilities that have so far been considered for these three aspects.

1.3.1 Ground Truth

Ground truth could potentially take any surface form, i.e. implicit field, triangle mesh, volumetric segmentation, point set, deformed model, skeleton curve, primitives. However, only two have so far been considered: **triangle mesh** [62, 14] and **implicit field** [63].

1.3.2 Input Point Cloud

Producing point samples from a surface can be carried out in several ways:

- **Real scanning**: Based on a physical object (or scene), laser-based scanning generates a point cloud directly. Such technologies include Time-of-Flight [2] and Structured-light [3] devices. In addition, terrestrial or airborne LiDAR [4] offer the possibility to deal with large areas.
- **Image-based**: Multi-view stereo [5] and Structure from motion [6] allow creating a 3D model from images, which can be the starting point for surface reconstruction.
- **Model sampling**: Based on a continuous digital input model, synthetic sampling has the advantage of making it possible to fully control the data. In particular, one can generate more realistic data by adding noise, outliers, misalignment and occlusions, and by setting the density. In this field, several procedures have been considered: **random** or **uniform sampling** [14, 57, 64], **synthetic raytracing** [43, 63] or **z-buffering** [62]. Of particular interest is the recent work presented in [65] and

[66]. *Helios++* [65] is an open-source tool for the simulation of airborne, UAV-based, and terrestrial static and mobile laser scanning implemented in C++ . [66] developed *LiDARsim*: a virtual terrestrial LiDAR platform generating realistic point clouds based on a high-quality mesh, free of moving objects.

1.3.3 Comparison

With regards to comparing an output reconstruction, six main possibilities have been explored:

- **Visually**: Most of the time, surface reconstruction aims at producing a digital representation as visually similar as possible to a real object. Hence, Poisson [11], MPU [27] and SSD [28] have simply compared models on a visual basis. This obviously raises the issues of being sensitive to the observer's perception, conflict of interest and the lack of quantitative information.
  The following five methods come with the advantage of providing a quantitative quality assessment that is independent of any human bias.

- **Point-to-mesh distance computation**: When the only ground truth that is available comes in the form of a point cloud, it is relatively straightforward to compute the distance from each of those points to the reconstructed model. [13] have evaluated their method by randomly partitioning their point cloud into two equal-sized subsets: one of points serving as input for the reconstruction algorithms and one of validation points from which distances to the output meshes are computed.

- **Point-to-point distance computation**: In the case where the ground truth comes in the form of a point cloud that has been acquired by scanning a real surface using a sensor, or if points have been sampled from a ground-truth surface, it is possible to sample the reconstructed surface and measure the distance between the two resulting point sets. In the computer vision community, the Chamfer distance [67] and the Hausdorff distance [68] are common ways of measuring the distance between two point sets. Given two point sets $\mathcal{P}_1$ and $\mathcal{P}_2$, and defining the distance from a point p to a point set $\mathcal{P}$ as:

$$\forall p \in \mathbb{R}^3,\ d\left(p, \mathcal{P}\right) = \min_{q \in \mathcal{P}} ||p - q||_2 \tag{1.14}$$

  the Hausdorff distance is defined [68] as:

$$d_{Hausdorff}(\mathcal{P}_1, \mathcal{P}_2) = \max\left\{\max_{p \in \mathcal{P}_1} d\left(p, \mathcal{P}_2\right),\ \max_{q \in \mathcal{P}_2} d\left(q, \mathcal{P}_1\right)\right\} \tag{1.15}$$

the Chamfer distance is defined [67] as:

$$d_{Chamfer}(\mathcal{P}_1, \mathcal{P}_2) = \frac{1}{|\mathcal{P}_1|} \sum_{p \in \mathcal{P}_1} d\left(p, \mathcal{P}_2\right)^2 + \frac{1}{|\mathcal{P}_2|} \sum_{q \in \mathcal{P}_2} d\left(q, \mathcal{P}_1\right)^2 \tag{1.16}$$

We believe that these metrics are not suited to the specific case of open scene reconstruction because they can be impacted by occlusions. Complex open scenes such as urban environments typically contain many vertical surfaces (building facades, for example) and overhangs (which can be found on bridges). Scanning this type of environment therefore often leads to a tremendous amount of occlusions. Points associated to correct pieces of surface might be unjustifiably penalised by these metrics if lying in occluded areas, for which no ground truth is available.

- **Mesh-to-mesh distance computation**: [62] and [13] use the *Metro* tool [1] to compute the distance between two meshes. It works as follows: given two meshes (a sampled one $\mathcal{M}_s$ and a target one $\mathcal{M}_t$), *Metro* samples $\mathcal{M}_s$ and measures the shortest distance from each sample to $\mathcal{M}_t$. *Metro* then computes the mean distance, the maximum distance and the Root Mean Square (RMS) distance over all samples. It is also possible to inverse the roles of $\mathcal{M}_s$ and $\mathcal{M}_t$ and perform the same calculations. Instead of evaluating how close the reconstruction is to the ground truth, this will assess how *complete* the reconstruction is.
  The Hausdorff distance is defined as the greater value between the maximum distance from $\mathcal{M}_s$ to $\mathcal{M}_t$ and the maximum distance from $\mathcal{M}_t$ to $\mathcal{M}_s$. Again, in the case of open scenes, the input point cloud that is fed to the reconstruction method is likely to contain many occlusions. Therefore, care must be taken when sampling the two meshes. Uniformly sampling the meshes is pertinent in the case of a closed object that can be scanned from different viewpoints. However, when dealing with open scenes, sampling the meshes in unobserved regions will lead the metrics to reflect the capacity of algorithms to fill holes rather than their reconstruction accuracy near the input data. This will be illustrated in Chapter 3.

- **Mesh-to-implicit distance computation**: [63] chose to use an implicit field that we will call Ω as the ground truth, and consequently they adapted the *Metro* methodology in order to compute the distance from a nearly uniform sampling of Ω to the evaluated mesh and vice-versa. The evaluation process answers the question: how well does the reconstructed mesh fit to the implicit surface computed by the MPU [27] algorithm? To address this issue, several measures are proposed by the benchmark [63]: Hausdorff distance (equation 1.19), mean distance (equation 1.20), max (equation 1.21) and mean angle deviation (equation 1.22). The former two allow us to know how *close* the two surfaces are to each other while the latter two give an insight into how similar the *local orientation* is.
  In order to compute these, defining point correspondences between the two surfaces

are needed. Let us denote by M the implicit surface and by $\overline{M}$ the output triangle mesh. As defined in [69], the mapping $\Phi : M \longrightarrow \overline{M}$ attributes to one point $p \in M$ the intersection of the normal line through p and the mesh $\overline{M}$. The inverse mapping $\Phi^{-1} : \overline{M} \longrightarrow M$ attributes to $\Phi(p)$ its closest neighbour on the implicit surface M. This definition, associated with a sampling P_M of M produces a set of nearest neighbour correspondences that we call C_M (equation 1.17).

$$C_M = \{(x, p) | p \in P_M,\ x = \Phi(p)\} \tag{1.17}$$

By defining the corresponding operator $\Psi : \overline{M} \longrightarrow M$ and a sampling $P_{\overline{M}}$ of the reconstructed mesh $\overline{M}$, we get $C_{\overline{M}}$ (equation 1.18).

$$C_{\overline{M}} = \{(p, x) | x \in P_{\overline{M}},\ p = \Psi(x)\} \tag{1.18}$$

Denoting $|S| = |C_M| + |C_{\overline{M}}|$ and with $\gamma(p, x)$ the angle between the normals $N_M(p)$ and $N_{\overline{M}}(x)$, error measures are the following:

$$H(M, \overline{M}) = \max \left\{ \max_{(x,p) \in C_M} |x - p|,\ \max_{(p,x) \in C_{\overline{M}}} |p - x| \right\} \tag{1.19}$$

$$\mu(M, \overline{M}) = \frac{1}{|S|} \left(\sum_{(x,p) \in C_M} |x - p| + \sum_{(p,x) \in C_{\overline{M}}} |p - x| \right) \tag{1.20}$$

$$H_N(M, \overline{M}) = \max \left\{ \max_{(x,p) \in C_M} \gamma(p, x),\ \max_{(p,x) \in C_{\overline{M}}} \gamma(p, x) \right\} \tag{1.21}$$

$$\mu(M, \overline{M}) = \frac{1}{|S|} \left(\sum_{(x,p) \in C_M} \gamma(p, x) + \sum_{(p,x) \in C_{\overline{M}}} \gamma(p, x) \right) \tag{1.22}$$

- **Neural Feature Similarity**: Recently, [70] proposed a benchmark targeting deep-learning-based surface reconstruction algorithms. They compute the similarity between two surfaces in the deep feature space.

1.4 Conclusion

To sum up, whilst surface reconstruction has been extensively studied, few approaches [55, 58] tackle watertight reconstruction of large open scenes. In addition, while metrics to assess surface reconstruction do exist, they are not adapted to deal with the many occlusions that open scenes typically contain. This calls for more contributions to scaling-up and evaluating surface reconstruction from point clouds of open scenes. This PhD

thesis proposes new approaches to tackle both these challenges.

Chapter 2

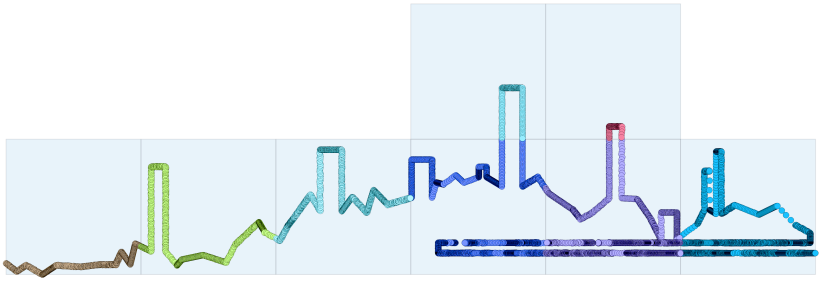
Large scale surface reconstruction

In this chapter, we tackle the task of scaling up surface reconstruction. As highlighted in the Introduction, acquiring massive point clouds has been made possible with the improvement in sensor capabilities but processing them remains a challenge. In Chapter 1, Section 1.2, we surveyed methods contributing to this specific task but very few guarantee that the resulting mesh is *watertight* and matches the *reference* one. The major contribution of the work presented in this chapter is the first Delaunay-based watertight surface reconstruction algorithm that can handle arbitrarily large point clouds. Poisson surface reconstruction [55] is the only other known approach that scales while ensuring watertightness but it is not suited to urban scenes as it fails to reconstruct sharp edges. Recently, [58] proposed an extension of a distributed workflow specific to urban scene reconstruction but it does not guarantee watertightness. Moreover, as their objective is to process urban environment only, the tiling strategy presented in their paper is purely 2-dimensional which restricts their algorithm to scenes in which a dimension is highly dominated by the two others. Conversely, the following end-to-end distributed out-of-core algorithm guarantees watertightness and it is suited to any kind of large-scale point cloud with arbitrarily complex 3-dimensional geometry (overhangs...).

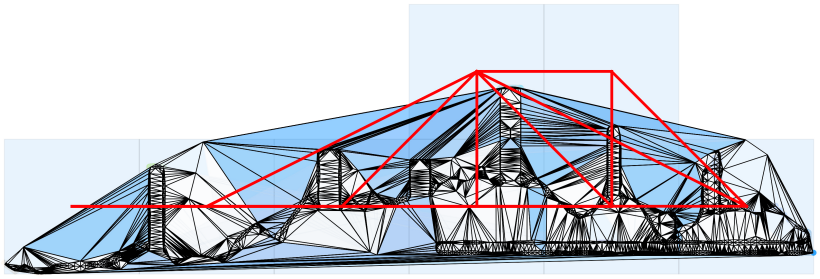
We proceed by first tiling the point cloud with a distributed octree approach which consists in choosing the maximum depth of an octree decomposition and then merging cells for which the total number of points that they contain does not exceed a given value (Figure 2.1a). Then, we compute the exact 3D Delaunay triangulation to discretise space using the algorithm introduced in [71]. Based on a distributed triangulation structure with shared cells between tiles (blue triangles on Figure 2.1b), associated with the adjacency graph of the tiles (red lines on Figure 2.1b), we propose an extension of a distributed graph-cut algorithm [72] (initially introduced for images) to carry out the *in/out* space labelisation that minimises our energy function (equation 2.1). The watertight surface can be extracted on the whole scene (blue line on Figure 2.1c) without any post-processing thanks to the consistent topology ensured by the global Delaunay triangulation.

2.1 Surface reconstruction model

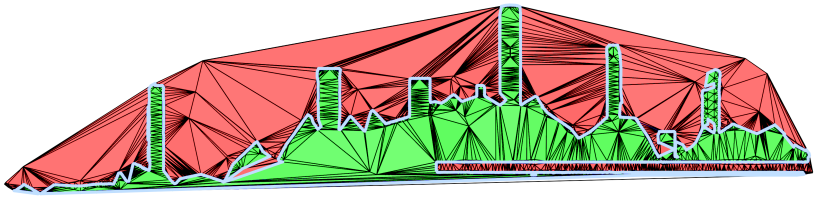
In this section, we present the *reference* (undistributed) surface reconstruction algorithm [23] of which we present a distributed version in Section 2.2.



(a) Tiled input point cloud with multiple acquisitions.



(b) Tiled Delaunay triangulation. The *local* cells are shown in white and the *mixed* ones in blue. The adjacency graph is drawn in red.



(c) Surface extraction. The cells in red and green are labelled as *empty* and *occupied* respectively, the light blue line denotes the extracted surface.

Figure 2.1: 2-dimensional example of the proposed approach.

2.1.1 Local PCA

Several steps of the pipeline rely on a local PCA (Principal Components Analysis) of the inertia matrix of a carefully chosen neighbourhood of each point $\mathbf{p} \in \mathcal{P}$. The resulting normalised eigenvectors define a local frame $f = (\mathbf{p}, \vec{v_1}, \vec{v_2}, \vec{v_3})$ and the corresponding eigenvalues (e_1, e_2, e_3) indicate how much the point cloud locally spreads in each direction. The eigenvector associated to the smallest eigenvalue provides a local (unoriented) surface normal direction. The position of the optical centre can then be used to consistently orient the normals from occupied to empty space.

2.1.2 Sub-sampling

As planar areas can be represented with large triangles, the input point cloud is sub-sampled to reduce its size, but without impairing the geometric precision. To this end, we use an adaptive algorithm based on PCA. Points $\mathbf{p}$ from the original point cloud $\mathcal{P}$ are randomly sampled and added to the set of sub-sampled points $\mathcal{P}^S$ only if $\forall\, j \in \{1,2,3\},\ p_j = |(\mathbf{p}-\mathbf{q}).\vec{v_j}| > \sigma e_j$ where p_j are the coordinates of $\mathbf{p}$ in the local frame of its closest point $\mathbf{q} \in \mathcal{P}^S$. The parameter σ controls the density of the resulting point cloud: as it increases, the distance threshold becomes larger so less points are added to $\mathcal{P}^S$ which density decreases.

2.1.3 Delaunay Triangulation (DT)

Space is discretised with the DT [73] of $\mathcal{P}^S$. In 3D, cells of this DT are tetrahedra which circumspheres do not contain any point of $\mathcal{P}^S$ other than the 4 points lying on them. Additionally, their union exactly covers the convex hull of $\mathcal{P}^S$.

2.1.4 Mass computation

Following [23], the *empty*/*occupied*/*unknown* state of space at any point $\mathbf{y} \in \mathbb{R}^3$ is defined for each input line of sight associated with point $\mathbf{p}$ using Dempster Shafer Theory (DST) masses $\mathbf{m_p}(\mathbf{y}) = (e,o,u) \in \left[0,1\right]^3$ such that $e+o+u=1$. For example, for a given position $\mathbf{y} \in \mathbb{R}^3$:

- $e = 1$ means certainty that space is *empty*
- $o = 1$ means certainty that space is *occupied*
- $u = 1$ means total *uncertainty*

These masses $m_{\mathbf{p}}(\mathbf{y})$ are then combined into an overall mass $m(\mathbf{y}) = (e,o,u)$ and integrated over each tetrahedron t of the DT with a Monte Carlo sampling, yielding a single occupancy value $m_t = o/(1-u)$ that gives the probability of t to be *occupied*.

2.1.5 Volumetric segmentation via graph-cut optimisation

The reconstruction method relies on assigning to each tetrahedron t of the DT the label 0 if t is *empty* or 1 if t is *occupied*. For simplicity we call $\mathcal{T}$ both the DT of $\mathcal{P}^S$ and the set of tetrahedra of this DT. This labelling problem is formulated as a Boolean energy minimisation (2.1). The energy function is composed of a data term (2.2) encouraging labels to be in accordance with overall masses and a prior term (2.3) smoothing the surface, balanced by a parameter α:

$$\widehat{\mathbf{x}} = \underset{\mathbf{x}\in\{0,1\}^{|\mathcal{T}|}}{\arg\min} \Big(E_{data}(\mathbf{x}) \; + \; \alpha \; E_{prior}(\mathbf{x}) \Big) \tag{2.1}$$

where $\mathbf{x} = (x_t)_{t\in\mathcal{T}} \in \{0,1\}^{|\mathcal{T}|}$ is the set of tetrahedra labels and $|\mathcal{T}|$ the number of tetrahedra.

$$E_{data}(\mathbf{x}) = \sum_{t\in\mathcal{T}} V_t.|x_t - m_t| \tag{2.2}$$

$$E_{prior}(\mathbf{x}) = \sum_{(t,t')\in\mathcal{T}^2} A_{t\cap t'}.|x_t - x_{t'}| \tag{2.3}$$

where V_t is the volume of tetrahedron t and $A_{t\cap t'}$ is the area of the interface between tetrahedra t and t'.

This energy (2.1) can be optimised via a graph-cut algorithm. To do so, let us denote by $G = (V, E)$ the dual graph of which the vertices $V = \{s, t\} \cup \mathcal{T}$ are the source s, the sink t and the tetrahedra of $\mathcal{T}$. The edges E are the triangles making the interface between all pairs of adjacent tetrahedra but also those connecting every tetrahedron to both s and t. Each edge is weighted so that the cost of an s-t cut in the dual graph has the same value as the energy function (2.1) after attributing the label values $\mathbf{x}$. To minimise the energy, we therefore need to minimise the sum of cut edges weights:

$$\sum_{(i,j)\in E} c_{i,j} x_{i,j} \tag{2.4}$$

where $c_{i,j}$ is the weight of edge (i, j) and $x_{i,j}$ is a Boolean indicating if the edge (i, j) is cut (in which case $x_{i,j} = 1$) or not ($x_{i,j} = 0$). Moreover, x_i (the label of tetrahedron i) indicates whether vertex i is linked to s ($x_i = 0$) or to t ($x_i = 1$) after a given cut. We then have:

- $x_s = 0$
- $x_t = 1$
- $\forall (i, j) \in E$, $x_{i,j} = |x_i - x_j|$

2.1.6 Surface extraction

The final surface is defined as the set of triangles separating occupied and empty tetrahedra.

2.2 Distributed surface reconstruction

In this section, we explain the distribution strategy. Algorithm 1 sums up in detail the different steps and Figure 2.2 presents the complete distributed workflow.

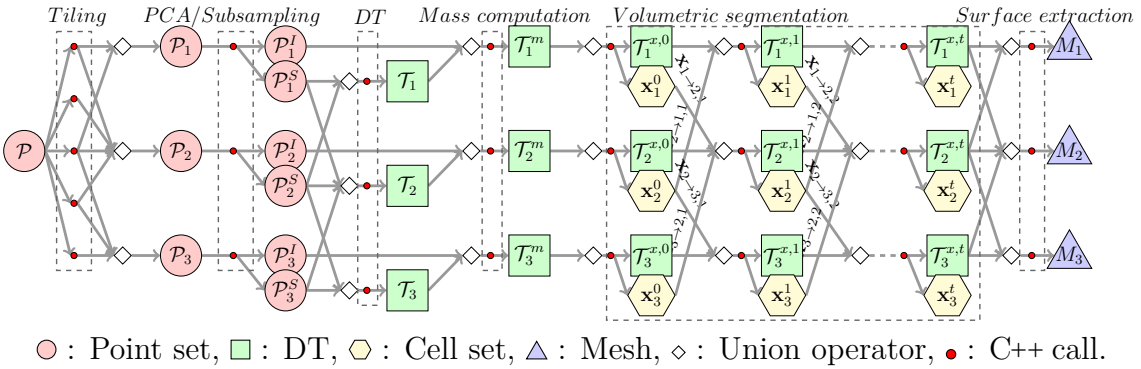


Figure 2.2: The proposed distributed surface reconstruction workflow. $\mathcal{P}$ denotes the input point set, $\mathcal{P}_k$ the tiled point set, $\mathcal{P}_k^I$ and $\mathcal{P}_k^S$ are the tiled point set with PCA **i**nformation and the **s**implified one, $\mathcal{T}_k$ the tile-triangulations, $\mathcal{T}_k^m$ the tile-triangulation with the mass score, $\mathcal{T}_k^{x,t}$ the labelled tile-triangulation at iteration t, $\mathbf{x}_k$ the updated **mixed** cells during the optimisation and M_k the final mesh.

2.2.1 Algorithm distribution

Tiling: Given a point set $\mathcal{P}$, we note the partition $\mathcal{P}_K = (\mathcal{P}_k)_{k\in K}$ its decomposition into $|K|$ disjoint subsets $\mathcal{P}_k$, where K denotes a discrete set of tile indices (Figure 2.1a, Algorithm 1 line 2). We can define the **primary** tile of a point $\mathbf{p} \in \mathcal{P}$ as the unique tile $\mathcal{P}_k$ such that $\mathbf{p} \in \mathcal{P}_k$.

Local PCA: The PCA is distributed by computing it separately on each tile $\mathcal{P}_k$. The tile along with the PCA **I**nformation is denoted $\mathcal{P}_k^I$.

Sub-sampling: Each tile $\mathcal{P}_k$ is sub-sampled separately yielding **S**ub-sampled tiles $\mathcal{P}_k^S$ from which the surface will be extracted (Algorithm 1 line 6).

Distributed Delaunay triangulation: We use the algorithm proposed in [71] to compute a distributed DT of the union of the sub-sampled point clouds $\mathcal{P}_K^S = \bigcup_{k\in K} \mathcal{P}_k^S$ (Algorithm 1 line 8). The **distributed DT** of a tiled point set $\mathcal{P} = \bigcup_{k\in K} \mathcal{P}_k$ is defined in [71] as a set of DTs: $\mathcal{T}_K = (\mathcal{T}_k)_{k\in K}$ such that $\mathcal{P}_k$ is a subset of the vertices of $\mathcal{T}_k$. The vertices of $\mathcal{T}_k$ are said to be **local** in $\mathcal{T}_k$ if they belong to $\mathcal{P}_k$ and **foreign** in $\mathcal{T}_k$ if they belong to another tile $\mathcal{P}_l$ with $l \neq k$. By extension, cells and facets of $\mathcal{T}_k$ are **local** or **foreign** in $\mathcal{T}_k$ if all their vertices are local in $\mathcal{T}_k$ or foreign in $\mathcal{T}_k$ respectively, and **mixed** otherwise. For a mixed cell t, we call K_t the set of indices of tiles containing t. We call a mixed cell *main* in $\mathcal{T}_k$ if $\min(K_t) = k$. The algorithm of [71] guarantees that the union of all local and main mixed cells is exactly the DT of $\mathcal{P}$ and that these cells are disjoint. We define the adjacency graph of the triangulation $G^A = (\mathcal{T}^A, \mathcal{E}^A)$ where $\mathcal{T}^A$ are the tile-triangulations and $\mathcal{E}^A$ the edges between them. Two triangulations are connected by an edge if and only if these triangulations share mixed points (see Figure 2.1b).

Algorithm 1: Distributed surface reconstruction

Input: Point set $\mathcal{P}$

`// Tiling`

1 **for** $\mathbf{p} \in \mathcal{P}$ **do** in parallel

2 $\quad k \leftarrow TileId(\mathbf{p})$

3 $\quad \mathcal{P}_k \leftarrow \mathcal{P}_k \bigcup \{\mathbf{p}\}$

4 **end**

`// PCA/Sub-sampling`

5 **for** $k \in K$ **do** in parallel

6 $\quad {\mathcal{P}^I}_k, \mathcal{P}^S_k \leftarrow PCA(\mathcal{P}_k)$

7 **end**

`// Delaunay Triangulation (DT)`

8 $\mathcal{T}^A, \mathcal{E}^A = DDT(\mathcal{P}^S)$

`// Mass computation`

9 **for** $k \in K$ **do** in parallel

10 $\quad \mathcal{T}^m_k \leftarrow mass\_computation(\mathcal{T}^A_k \cup \mathcal{P}^I_k)$

11 **end**

`// Volumetric segmentation`

12 $t \leftarrow 0$

13 **for** $k \in K$ **do** in parallel

14 $\quad \mathcal{T}^x_{k,t}, \mathbf{x}_{k \rightarrow l \neq k,t} \leftarrow graph\_cut(\mathcal{T}^m_k)$

15 **end**

16 $t \leftarrow 1$

17 **while** $t < max\_iterations$ **do**

18 $\quad$ **for** $k \in K$ **do** in parallel

19 $\quad\quad \boldsymbol{\lambda}^k_t \leftarrow update\_lagrange\,(\bigcup_l \mathbf{x}_{l \rightarrow k,t-1})$

20 $\quad\quad \mathcal{T}^x_{k,t}, \mathbf{x}_{k \rightarrow l \neq k,t} \leftarrow graph\_cut(\mathcal{T}^x_{k,t-1}, \boldsymbol{\lambda}^k_t)$

21 $\quad$ **end**

22 $\quad t \leftarrow t + 1$

23 **end**

`// Surface extraction`

24 **for** $k \in K$ **do** in parallel

25 $\quad \mathcal{T}^{'}_k \leftarrow aggregate\_neigbors(\mathcal{T}_{k,t}, \mathcal{E}^A)$

26 $\quad M_k = extract\_surface(\mathcal{T}^{'}_k)$

27 **end**

28 **return** M

Mass computation: Given that the effect of an input point $\mathbf{p}$ on a tetrahedron t decreases with the distance, the mass function is only computed locally on each tile-triangulation $\mathcal{T}_k$ using $\mathcal{P}^I_k$ (Algorithm 1 line 10).

Volumetric segmentation: The energy function 2.1 is optimised in parallel with independent graph-cuts on each tile (Algorithm 1 line 12) where the labels of mixed cells are encouraged to agree with an iterative scheme as explained in Section 2.2.2.

Surface extraction: To extract the full surface, each tile is loaded with its respective neighbours regarding the adjacency graph's edges E (Algorithm 1 line 24). We denote by $aggregate\_neigbors(\mathcal{T}_k^A, \mathcal{E}^A)$ the function that aggregates the triangulation and its respective neighbours. Because the iterative scheme does not provide the guarantee to converge on the same label on **mixed** cells (for which copies exist in multiple tile-triangulations), the label from the main mixed cell is systematically chosen for the surface extraction. Note that the surface is watertight even if the labels do not agree since the optimisation ends up with a unique set of labels which inevitably leads to a watertight mesh.

2.2.2 Distributed graph-cut

The main challenges to distribute a DT-based surface reconstruction method are distributing the DT itself, for which we use the implementation of [71], and distributing the graph-cut optimisation. Graph-cut distribution approaches are highly dependent on the structure of the graph. While many approaches exist to distribute graph cuts as surveyed very recently in [74], we chose the *dual decomposition* paradigm because it offers the best distribution potential. Very little information has to be shared between the different tiles. While dual decomposition has been used for a graph representing the regular 2-dimensional structure of images [72], we did not find any work tackling the specific structure of the adjacency graph G of the cells of a DT (see Section 2.1) and of its tiling-induced split into $|K|$ sub-graphs $G_k = (V_k, E_k)$. Thus, one of the main contributions of this work is the generalisation of the method proposed in [72] to our tiled graph G which allows to distribute the graph-cut optimisation across multiple tiles. We use the structure of the tiled DT to split G into $|K|$ subgraphs G_k, one for each tiled-triangulation T_k. In practice, each G_k is defined as the adjacency graph of the set of local and mixed cells of T_k (Figure 2.4). The main issue of the distributed problem is to ensure that the mixed cells have the same label relatively to the tiles that share them. For any $k \in |K|$, we denote by $x_{i,j}^k$ and x_i^k the values of $x_{i,j}$ in sub-graph k and x_i in sub-graph k respectively. We also define $c_{i,j}^k = \frac{c_{i,j}}{|\{k:(i,j)\in E_k\}|}$ as the weight of edge (i,j) normalised by the number of tiles it appears in. Using this decomposition, the optimised function becomes:

$$f(\mathbf{x}) = \sum_{(i,j)\in E} c_{i,j}x_{i,j} = \sum_{k\in K}\sum_{(i,j)\in E_k} c_{i,j}^k x_{i,j}^k \tag{2.5}$$

under the condition that labels x_i^k should have the same value amongst all subgraphs they appear into:

$$\forall\, k,l \in K^2, \quad \forall\, i \in V_k \cap V_l, \quad x_i^k - x_i^l = 0 \tag{2.6}$$

To split this problem in sub-problems using our tiled graph structure, we solve the graph-cut on each sub-graph G_k with variables $x_{i,j}^k$. To guarantee the condition expressed in

equation 2.6, we add a penalty term to the energy using Lagrangian duality:

$$L(\mathbf{x}, \boldsymbol{\lambda}) = \sum_{k \in K} \sum_{(i,j) \in E_k} c_{i,j}^k x_{i,j}^k + \sum_{k \in K} \sum_{l > k} \sum_{i \in V_k \cap V_l} \lambda_i^{k,l} (x_i^k - x_i^l) \tag{2.7}$$

with $\boldsymbol{\lambda} = \{\lambda_i^{k,l} : (k,l) \in K^2,\ i \in V_k \cap V_l\}$. By defining, for any tile index $k \in K$:

$$L_k(\mathbf{x}^k, \boldsymbol{\lambda}) = \sum_{(i,j) \in E_k} c_{i,j}^k x_{i,j}^k + \sum_{l \neq k} \sum_{i \in V_k \cap V_l} \begin{Bmatrix} 1 & \text{if } l > k \\ -1 & \text{otherwise} \end{Bmatrix} \lambda_i^{k,l} x_i^k \tag{2.8}$$

We then have:

$$L(\mathbf{x}, \boldsymbol{\lambda}) = \sum_{k \in K} L_k(\mathbf{x}^k, \boldsymbol{\lambda}) \tag{2.9}$$

Figure 2.3 allows to visualise the dual graph weights update in the case of a single split resulting in two tiles. We define the Lagrange dual function:

$$g(\boldsymbol{\lambda}) = \min_{\mathbf{x}} L(\mathbf{x}, \boldsymbol{\lambda}) \tag{2.10}$$

Since the Lagrange dual function $g(\boldsymbol{\lambda})$ (equation 2.10) is concave [75], we can find an optimal distributed solution to our problem (2.5+2.6) by maximising $g(\boldsymbol{\lambda})$ via an ascent method. We denote by t the iteration index and by $\boldsymbol{\tau} = \{\tau_i^{k,l} : (k,l) \in K^2,\ i \in V_k \cap V_l\}$ the vector of step amplitudes by which we increment $\boldsymbol{\lambda}$. We initialise the Lagrange multiplier vector $\boldsymbol{\lambda}(t=0) = \mathbf{0}$ and the vector of steps $\boldsymbol{\tau}(t=0)$ by $\tau_i^{k,l} = \tau_0$. As demonstrated in [72], $(\mathbf{x}^k - \mathbf{x}^l)_{k,l \in K^2}$ is a supergradient to g at $\boldsymbol{\lambda}$. Consequently, we can solve the distributed graph cut by iterating:

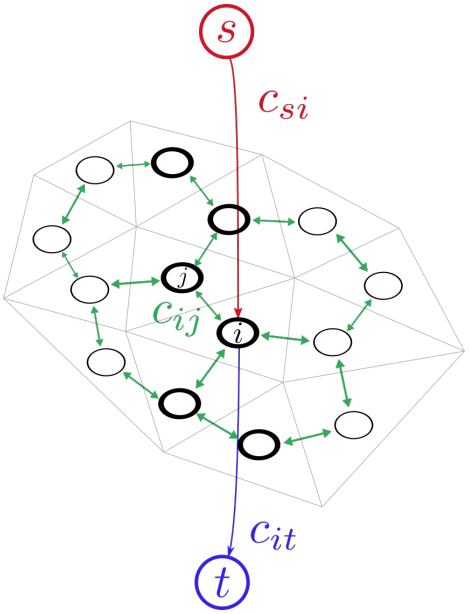
- Solve the graph-cut problems $L_k(\mathbf{x}^k, \boldsymbol{\lambda})$ for $\mathbf{x}^k$
- Update $\boldsymbol{\lambda}$ and $\boldsymbol{\tau}$:

$$\text{for } k \in K \text{ and } l > k:$$

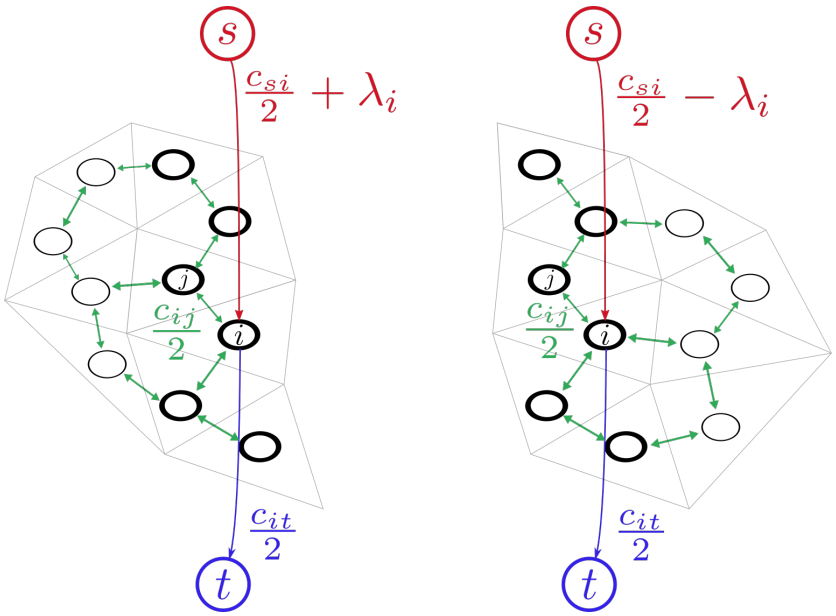
$$\lambda_{i,t+1}^{k,l} = \lambda_{i,t}^{k,l} + \tau_{i,t}^{k,l} (x_{i,t}^k - x_{i,t}^l) \tag{2.11}$$

$$\tau_{i,t+1}^{k,l} = \frac{\tau_t^{k,l}}{2} \text{ if } x_{i,t}^k - x_{i,t}^l \neq x_{i,t-1}^k - x_{i,t-1}^l \tag{2.12}$$

To perform this update at iteration step t, each tile G_l sends to its neighbour tiles G_k the labels of their shared mixed cells (according to G_l) in a vector $\mathbf{x}_{l \to k,t}$.



(a) Dual graph before split



(b) Dual sub-graphs after split

Figure 2.3: Weight update after a single split in the dual graph. We focus on node i. Nodes in bold are the ones that have been duplicated after the split. s and t represent the *source* and the *sink*.

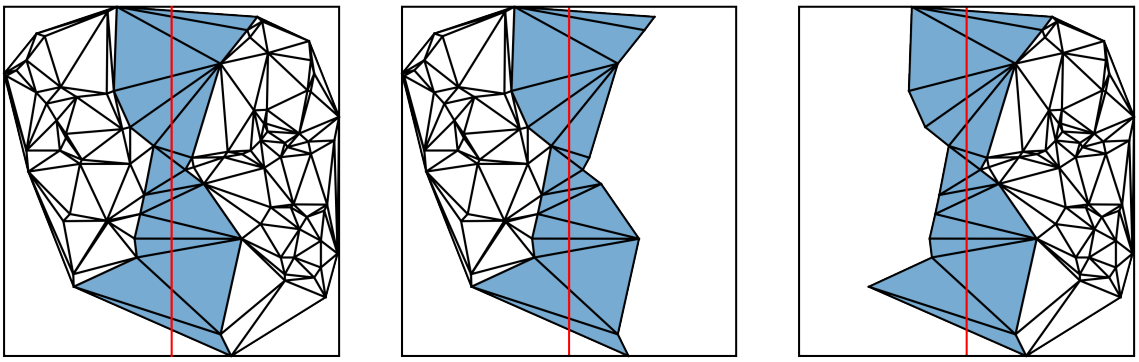


Figure 2.4: 2D visualisation of the full triangulation (left) being split into 2 tiles (middle and right) with local cells in white and mixed cells in blue. Mixed cells are duplicated in both sub-graphs.

2.2.3 Implementation details

The workflow is implemented with the Apache Spark framework [53]. The input of the algorithm is a point cloud separated in several files stored on the Hadoop distributed filesystem (HDFS) [76]. To ensure a reasonable memory footprint in the tiling, we split large files to ensure that each input file contains less than 1 million points. Each such file is serialised into an RDD [54] with a *Base64* encoding in a *String*. The key/value formalism is used: each element of the *RDD* is represented by a key and a value which is a list of sets (list of point sets, list of tile-triangulations, etc.). A transformation on an *RDD* (Local graph-cut, mass computation, etc.) is performed with a C++ call in parallel on each *RDD* chunk by using the *pipe* operator. The union operator ($\cup$ in Algorithm 1 and $\diamond$ in Figure 2.2) is a union followed by a *ReduceByKey* in Spark.

2.3 Experimental results

While memory and computational efficiency are the main goals when trying to distribute an algorithm, the most important aspect of evaluation is to verify that the *distributed* algorithm reproduces as closely as possible the output of the *reference* algorithm. For that reason, we will first evaluate the *distributed* algorithm in comparison with the *reference*, and then focus on the scalability. Moreover, as PCA and sub-sampling are expected to suffer minor border effects, and the distributed Delaunay triangulation is proved to produce the exact Delaunay Triangulation [71], we mainly focus our evaluation on our major contribution: the distributed graph cut. In all the following experiments, unless otherwise stated, we used the values: $\alpha = 0.005$, $\tau_0 = 5$, and 30 iterations.

2.3.1 Distributed graph-cut convergence

Graph cut implementations provably find the global minimum cut of the graph [77] which equivalently achieves the global minimum of the associated energy (eq 2.1). We will therefore assess our distributed graph cut implementation by comparing the energy obtained with our distributed implementation with this global minimum energy. In order to evaluate our capacity to handle a large proportion of mixed cells (which makes the distributed problem harder), the algorithm is run with different octree depths for a fixed number of points. A larger depth induces more and smaller tiles, resulting in a higher proportion of mixed cells. As our iterative scheme is mainly influenced by the value of τ_0, we also evaluate its impact on convergence. Figure 2.5 shows the evolution along iterations t of the ratio between the energy (eq 2.1) of the labelling computed with the ***d**istributed* graph-cut E_d and the labelling computed with the ***r**eference* graph-cut E_r. It is important to note that even if two surfaces with the same energy are not necessarily the same, they are considered equally good. For all values of τ_0 and octree depths, the energy ratio increases during the first iterations until it reaches a maximum around iteration 5, then decreases until converging significantly under the initialisation value,which shows the robustness of our distributed algorithm to these two factors.

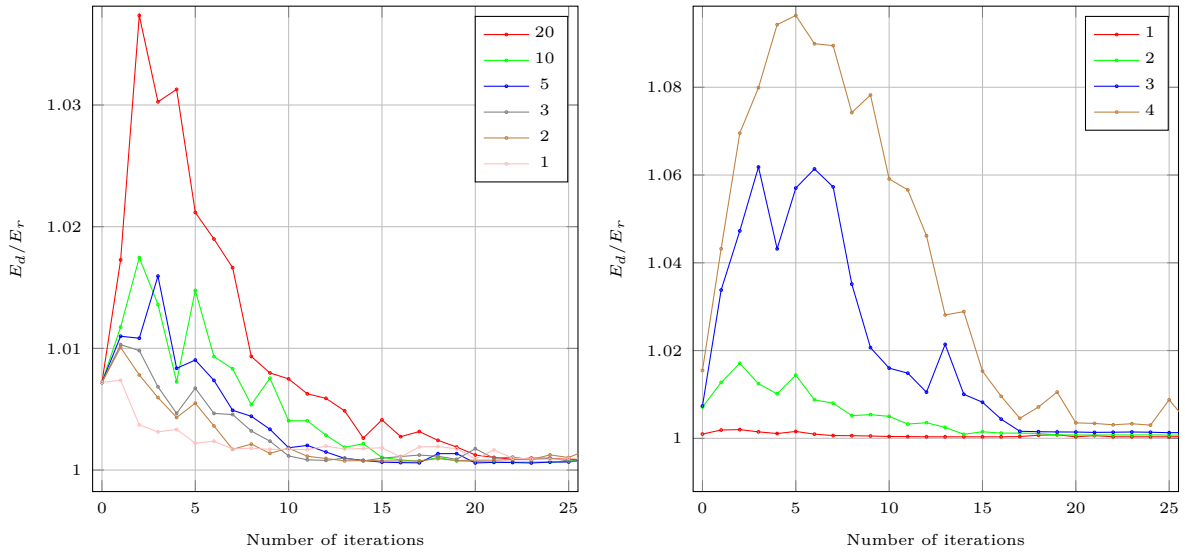


Figure 2.5: Ratio between the energy of the labelling given by the distributed graph-cut E_d and the reference graph-cut with one thread E_r for different initialisations of the **Lagrangian step** $\tau_0 = 1, 2, 3, 5, 10, 20$ (left) and different **octree depths** $1, 2, 3, 4$ (right).

Figure 2.7 shows the resulting mesh along iterations on a LiDAR data set. Shared cells are visually the most impacted after the first graph-cut. In some areas, tiles are not or badly connected, only triangles between local cells at the centre of a tile are well reconstructed. After 3 iterations, the mesh is better regularised but the quality remains bad at the boundary. After 15 iterations, the main errors are removed and the tiles are

globally well connected. At iteration 30, small shared areas like the car hood in the example are well reconstructed.

2.3.2 Speedup

The efficiency of the proposed approach is evaluated on a Spark cluster with 28 cores and 100GB of RAM. This configuration is close to the $m5$[1] setup of the Amazon EMR service dedicated to general usage that provides 4GB per core. We evaluate the **strong scaling** speedup which quantifies how the algorithm scales with an increasing number of cores for a fixed number of points. We compute the surface of a 10 million points data set with different cluster configurations: a varying number of executors (7,4,2 and 1) and, for each executor, a varying number of cores (4,3,2,1) for a fixed 12GB of RAM per executor in every case. The total number of cores is the number of executors times the number of cores per executor. The result is shown on Figure 2.6 where the strong scaling factor is defined as $t_{1\,\text{core}}/(n.t_{n\,\text{cores}})$, where $t_{x\,\text{cores}}$ is the time it takes to compute the solution to the problem using x cores. The strong scaling factor is expressed as a function of the number of cores. As our application is fully distributed, we compare it to the perfect scale factor of 1. Results show that every configuration has a scale factor > 0.6. In other words, the execution time with a single thread is at least divided by 0.6 times the number of cores with the distributed implementation.

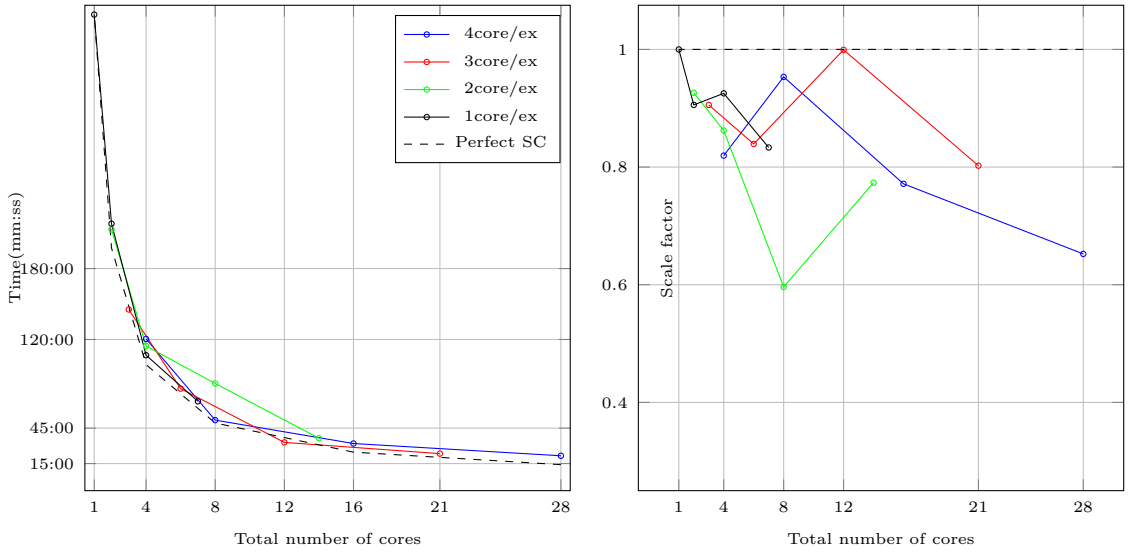


Figure 2.6: **Execution time** (left) and **strong scaling factor** $\frac{t_{1\,\text{core}}}{n.t_{n\,\text{cores}}}$ (right) as a function of the number of cores with 4, 3, 2 and 1 core(s)/executor (12Go RAM/executor) on a 10 million points data set.

[1] https://aws.amazon.com/ec2/instance-types/m5/

2.3.3 Large data set result

We tested our algorithm on a very large data set that combines more than 350 million points acquired with both aerial and terrestrial LiDAR (Figure 2.8), on which our available hardware could not run the reference implementation by lack of RAM. The point cloud was split into 9520 tiles and it took 17 hours to process using 28 cores. The proposed approach produces a watertight mesh of the whole area with highly detailed and complex 3D structures, composed of around 80M vertices. We empirically set the sub-sampling parameter $\sigma = 0.3$ to have a good trade-off between accuracy and computing time (a higher σ would reduce computing time at the cost of accuracy).

2.4 Conclusion

In this chapter, we presented a fully distributed and out-of-core surface reconstruction method. The extension of the distributed graph-cut method of [72] to the structure of the tiled 3D Delaunay Triangulation of [71] leads to a fully distributed method that ensures both watertightness of the resulting mesh and preservation of sharp edges. The evaluation confirms that the proposed method produces meshes of similar quality to the reference (non-distributed) implementation. The fully distributed framework allows an efficient speedup while maintaining a reasonable memory footprint even for very large inputs. The implementation on the state-of-the-art Apache-Spark framework shows very good results on managing very large data sets.

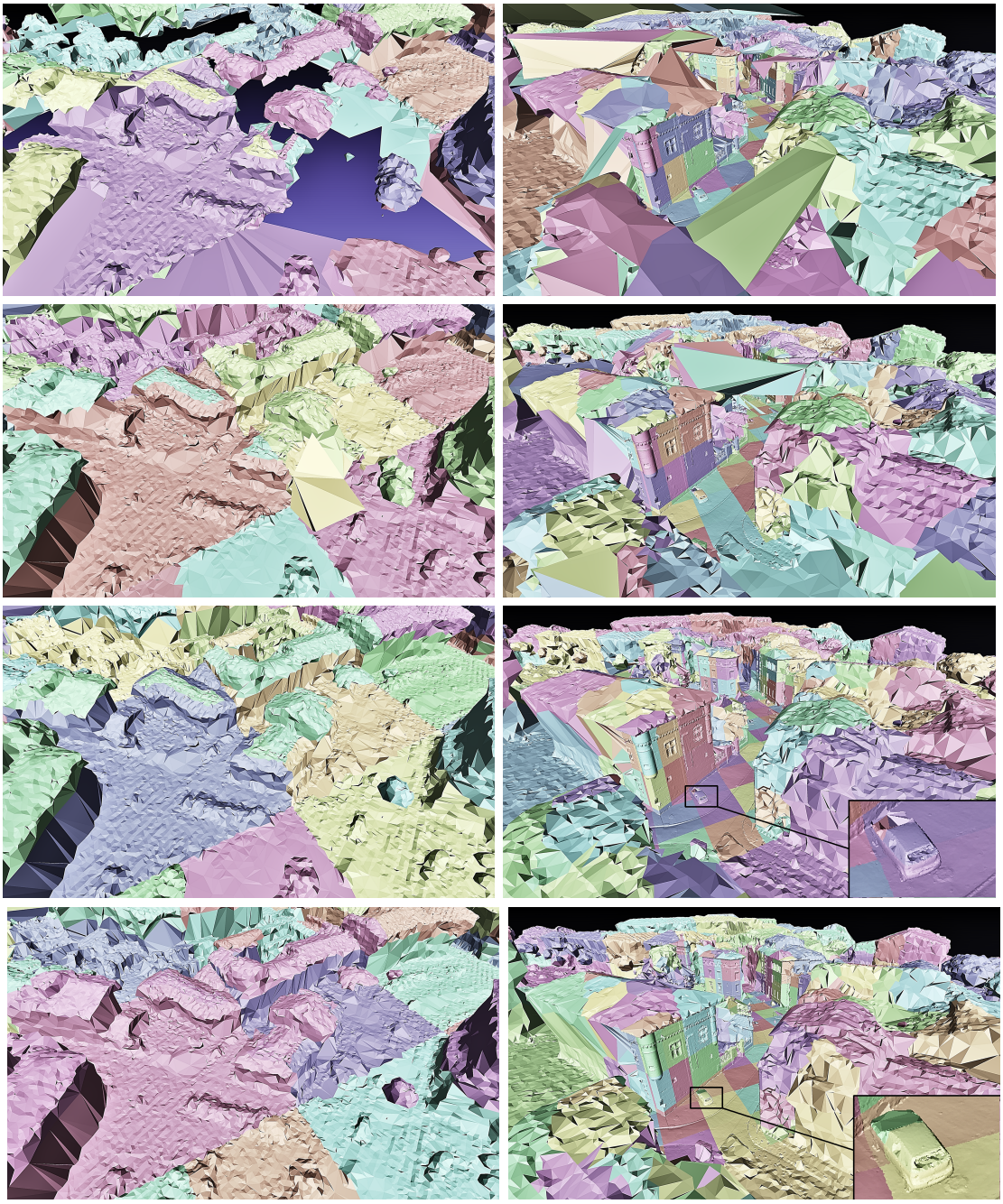


Figure 2.7: Evolution of the resulting mesh along iterations on an urban scene. Each line shows two different parts of the mesh. The first line shows the result with the local graph-cut without iterating, the second line corresponds to iteration n°3, the third line to iteration n°15 and the fourth line to iteration n°30.

Figure 2.8: Reconstruction result on a 350 million points data set. Tiling is visualised through a random colour per tile.

Chapter 3

Evaluating surface reconstruction using synthetic data

This chapter addresses the evaluation of algorithms reconstructing a surface from a point cloud, using synthetic data. The objective is to set a rigorous protocol measuring the quality of the reconstruction. This work is of fundamental interest for two reasons. First, it enables the assessment of different algorithms tackling the same problem to determine the best approach. Secondly, as most algorithms can be tuned via a set of parameters, it makes it possible to quantitatively study the influence of the different settings and adjust them in order to achieve the best performance. This is a new approach, that introduces less bias than previous contributions. We use ground truth surfaces and we proceed by filtering out parts of the meshes that are not relevant to assess, at different thresholds. We compute distances between samplings of these filtered meshes and samplings from the filtered ground truth surfaces.

3.1 Evaluation protocol

3.1.1 Input data

In order to assess surface reconstruction, we first assume the availability of a ground truth model taking the form of a triangle mesh $\mathcal{M}_{GT}$. Secondly, a point cloud $\mathcal{P}$ representing a realistic scan of $\mathcal{M}_{GT}$ must have been generated using a LiDAR simulator like the one we present in section 3.2. Finally, we will denote by $\mathcal{M}_E$ a reconstructed triangle mesh produced with one of the methods to evaluate.

3.1.2 Intuition

As explained in Chapter 1, one way of assessing the quality of a given reconstruction is to uniformly sample the corresponding mesh $\mathcal{M}_E$ and to compute the distance from each of these samples to the ground-truth mesh $\mathcal{M}_{GT}$, and then doing the same but inverting the

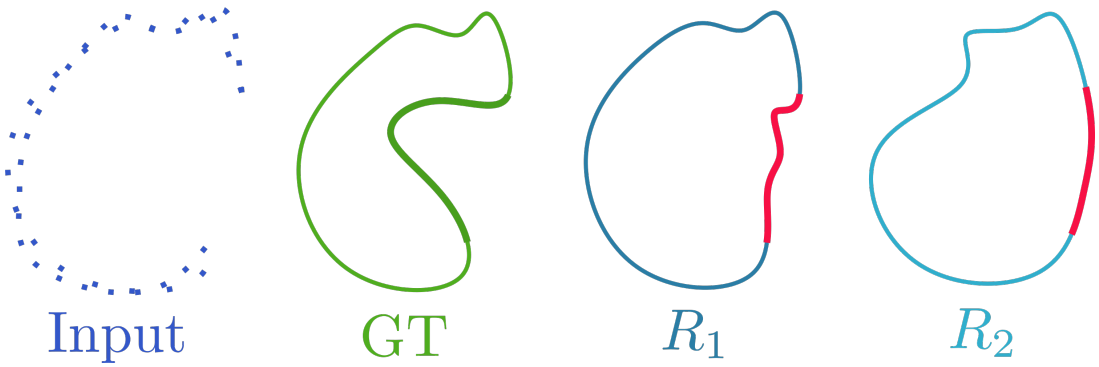


Figure 3.1: In this situation, the real surface has been partially scanned, resulting in a serious occlusion in the input point cloud. If we uniformly sampled the entirety of both reconstructed meshes R_1 and R_2 and used the *Metro* tool [1] to compute the distances from them to the ground-truth, the metrics would be dominated by occlusion-based errors (highlighted in bold).

roles of $\mathcal{M}_E$ and $\mathcal{M}_{GT}$. Scanning open scenes such as urban environments often results in severe occlusions because of the presence of many vertical surfaces and overhangs. Sampling points from the reconstructed surface $\mathcal{M}_E$ in unobserved regions and taking into account the corresponding distances to $\mathcal{M}_{GT}$ is very likely to dramatically affect the global performance. This is because the main errors made by the algorithm are very likely to correspond to those occluded areas. This approach would thus give an indication of the capacity of the algorithm to fill holes rather than its reconstruction accuracy near the input data. This is illustrated in Figure 3.1.

3.1.3 Best expected surface

In the general case, as we explained it in Section 3.1.2, only a limited portion of the ground truth surface $\mathcal{M}_{GT}$ is covered by the acquisition. If we can expect an algorithm to fill some holes in the data in a reasonable manner, we cannot expect it to recover the shape of the scene far from this covered area. Thus, we need to define the "*reconstructible*" part $\mathcal{M}_{GT}^{\alpha}$ of $\mathcal{M}_{GT}$, which can be non connected and have holes, but is still orientable and manifold. To this end, we propose to define an explicit maximum interpolation (hole filling, super resolution) and extrapolation (uncropping) distance α at which we evaluate the algorithm. We can then compute $\mathcal{M}_{GT}^{\alpha}$ by removing all triangles of $\mathcal{M}_{GT}$ for which none of its vertices lie closer than α to a point of the sampling $\mathcal{P}$. See algorithm 2 for a description of this mesh processing step. $\mathcal{M}_{GT}^{\alpha}$ is then the best surface an algorithm can be expected to reconstruct. Consequently, this is the surface we will compare every reconstructed mesh with. However, watertight reconstruction pipelines interpolate surface even where point samples are absent. While watertightness is desirable for the reconstruction of closed objects, it is less for open scenes that naturally have a border corresponding to the limit of the coverage of the scene by the acquisition. In order to be impartial with every algorithm prior, we will not evaluate those interpolated mesh parts further away than α

from the input data. As a consequence, we also need to compute the part of $\mathcal{M}_E$ that is relevant to assess i.e. the sub-mesh $\mathcal{M}_E^\alpha$ of $\mathcal{M}_E$ (applying algorithm 2) containing only triangles closer to $\mathcal{P}$ than a distance α. This is not limiting as in practice, it is an easy post-processing step that the user can choose to perform if he does not require a fully watertight mesh. For all these reasons, we believe this approach introduces less bias than previous work. Figure 3.2 allows to visualise the computation of $\mathcal{M}_{GT}^\alpha$ and $\mathcal{M}_E^\alpha$ for a given α.

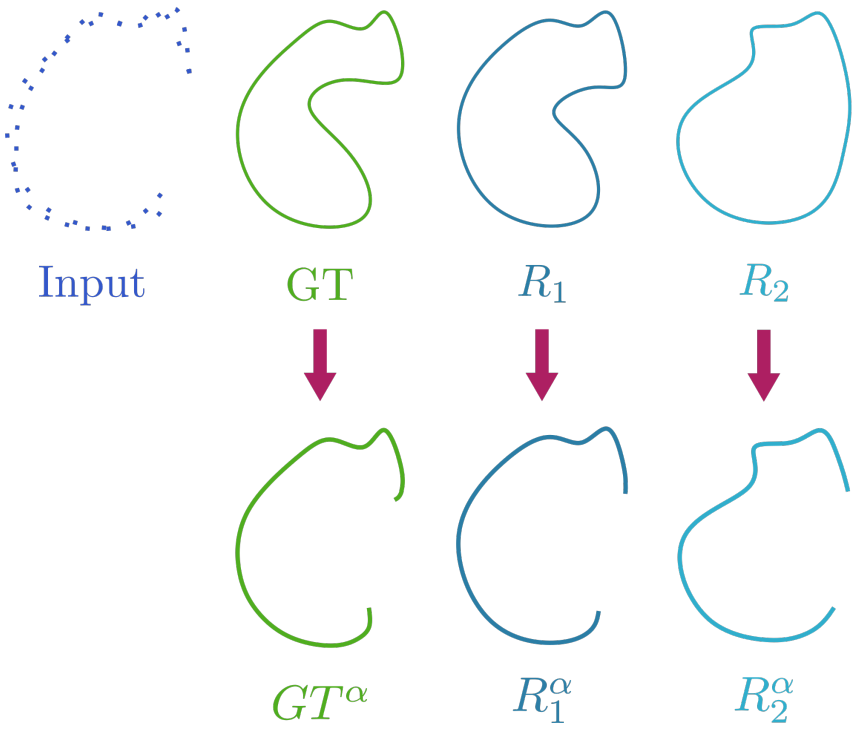


Figure 3.2: Visualisation of the computation of the best expected surface for a small value of α. Starting from the input point cloud, the ground-truth mesh and two reconstructed meshes R_1 and R_2, we compute the best expected surface and the relevant parts to assess R_1^α and R_2^α by removing all triangles lying further than α from a point of the input point cloud.

3.1.4 Sampling

Once $\mathcal{M}_{GT}^\alpha$ and $\mathcal{M}_E^\alpha$ have been computed, we need a way to measure the difference between them. We consider that computing error metrics based on distances between vertices of one mesh and triangles of the other is biased, as different algorithm can produce triangles of very different size. Thus we propose to perform a Poisson-disk sampling of the triangles, which guarantees an even distribution of samples so that we can consider that each sample represents the same amount of surface area. We choose a Poisson-disk radius R significantly smaller than α. We denote by $\mathcal{P}^R_{\mathcal{M}_{GT}^\alpha}$ and $\mathcal{P}^R_{\mathcal{M}_E^\alpha}$ the Poisson-disk samplings of radius R of $\mathcal{M}_{GT}^\alpha$ and $\mathcal{M}_E^\alpha$ respectively.

Algorithm 2: Reconstructible part of mesh $\mathcal{M}$ computation

```
Data: M, P, α
// Browse vertices of M:
for each vertex v ∈ vertices(M) do
    compute d = d(v, P) = min_{P_i} d(d, P_i)
    if d < α then
        // Browse all triangles incident to v:
        for each triangle t ∈ {T : v ∈ T} do
            if t ∉ M^α then
                // Browse vertices adjacent to t:
                compute (v_0, v_1, v_2) = vertices(t)
                for each vertex v_t ∈ {v_0, v_1, v_2} ∈ do
                    if v_t ∉ M^α then
                        add v_t to M^α
                end
                add t to M^α
        end
end
```

3.1.5 Quality measure

Thanks to notations introduced in Section 3.1.4 and using the point-to-mesh distance $d(p, \mathcal{M})$ defined in equation 3.1, bi-directional distances can be computed (equations 3.2, 3.3).

$$\forall p \in \mathbb{R}^3,\ d(p, \mathcal{M}) = \min_{q \in \mathcal{M}} d(p, q) \tag{3.1}$$

$$\text{Mean Precision: } \frac{1}{|\mathcal{P}^R_{\mathcal{M}^\alpha_E}|} \sum_{p \in \mathcal{P}^R_{\mathcal{M}^\alpha_E}} d(p, \mathcal{M}^\alpha_{GT}) \tag{3.2}$$

$$\text{Mean Recall: } \frac{1}{|\mathcal{P}^R_{\mathcal{M}^\alpha}|} \sum_{p \in \mathcal{P}^R_{\mathcal{M}^\alpha_{GT}}} d(p, \mathcal{M}^\alpha_E) \tag{3.3}$$

Precision measures how close are points from the reconstructed mesh to the ground truth and *recall* indicates which distance separates points from the ground truth to the reconstruction. While these terms are named in analogy with the machine learning community metrics, it is important to note that they do not measure a ratio of relevant information but a distance and thus the lower, the better.

Instead of choosing an arbitrary α, we propose to compute these distances for a range of α values and draw the charts corresponding to each of these. Note that the case where $\alpha \longrightarrow \infty$ corresponds to computing the distances on samplings of the raw meshes $\mathcal{M}_{GT}$ and $\mathcal{M}_E$, without applying algorithm 2. The resulting curves will both indicate the precision of the algorithm for small α values and the quality of the interpolation/extrapolation/hole filling for larger α values.

3.2 Aerial LiDAR simulator

Similar to *LiDARsim* [66] for terrestrial scan, we developed our own airborne LiDAR simulation platform in order to generate realistic scans of a given environment.

3.2.1 Virtual environment

We used the open dataset from [78] which was financed by the European Union as part of a FEDER (*Fonds Européen de Developpement Régional*). It consists of a 3D mesh of a large area covering the metropole of Strasbourg. It was produced by photogrammetry using high resolution (between 4 and 7cm GSD) oblique imagery acquired by helicopter platform such that it presents details at a higher resolution than typical aerial LiDAR acquisitions. Figure 3.3 shows a 250m by 250m tile of this mesh.

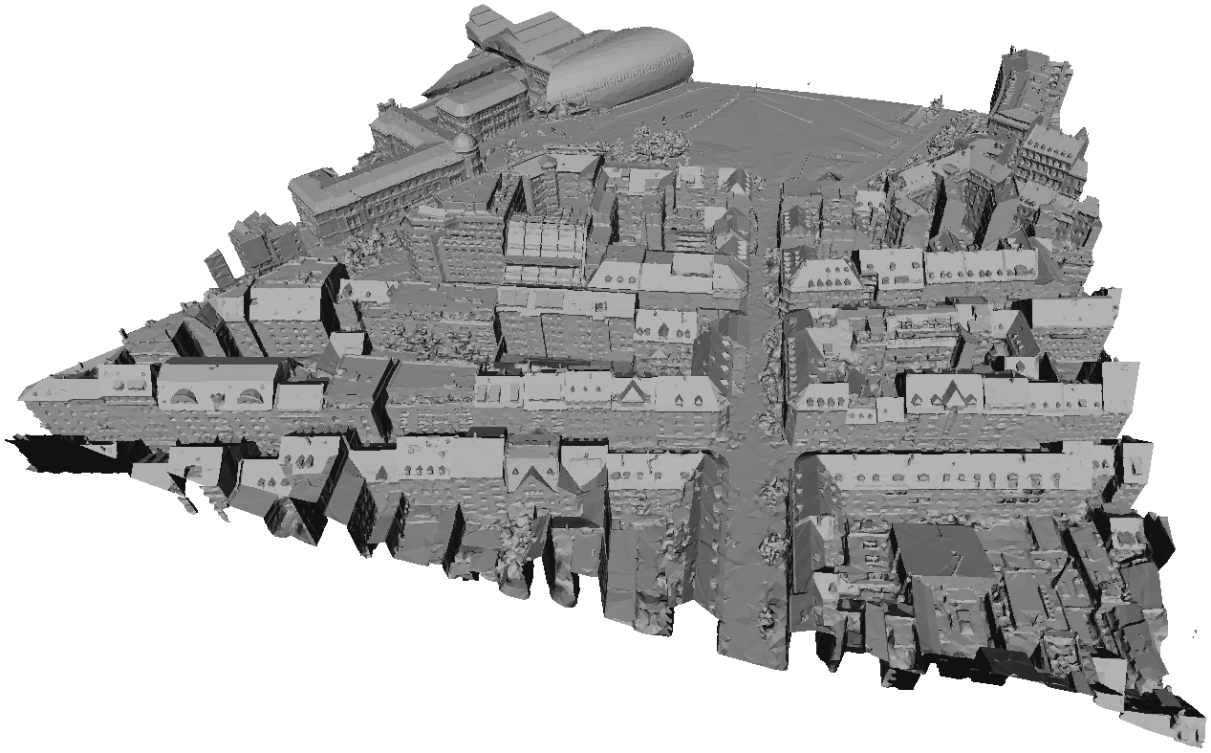


Figure 3.3: Virtual environment from Strasbourg open dataset

3.2.2 Scanning process

In this section, we formalise our aerial LiDAR simulator. We assume the plane trajectory can be modelled as a sequence of straight lines and we implemented two scanning patterns. Among the many possibilities available on the market (zig-zag, elliptical, circular...), we chose to replicate the **parallel line** and the **elliptical** scanning pattern produced by a rotating mirror mechanism.

3.2.2.1 Plane trajectory

We use $(O, \overrightarrow{e_x}, \overrightarrow{e_y}, \overrightarrow{e_z})$ as the global coordinate frame, the one in which mesh vertices coordinates are expressed as detailed in Figure 3.4. We model the acquisition by a linear trajectory of the LiDAR optical center M moving from $A\,(x_A, y_A, z_A)$ to $B(x_B, y_B, z_B)$ at constant speed v_0. We also use a local coordinate frame $\left(M, \overrightarrow{i}, \overrightarrow{j}, \overrightarrow{k}\right)$ associated to M defined as:

$$\overrightarrow{k} = \frac{\overrightarrow{AB}}{\left\|\overrightarrow{AB}\right\|} \; ; \quad \overrightarrow{j} = \frac{\overrightarrow{e_z} \wedge \overrightarrow{k}}{\left\|\overrightarrow{e_z} \wedge \overrightarrow{k}\right\|} \; ; \quad \overrightarrow{i} = \overrightarrow{j} \wedge \overrightarrow{k} \tag{3.4}$$

which ensures that $\overrightarrow{j}$ is orthogonal to $\overrightarrow{AB}$ and to the vertical direction $\overrightarrow{e_z}$. Note that $\overrightarrow{k}$ is undefined only when $\overrightarrow{AB}$ and $\overrightarrow{e_z}$ are colinear which never happens in practice (a plane does not fly vertically). $\overrightarrow{i}$ completes the frame so that $\left(\overrightarrow{i}, \overrightarrow{j}, \overrightarrow{k}\right)$ is right-handed. $M(t)$ moves in a straight line and thus can be defined as:

$$\forall\, t \in [0, t_B]\,, \; \overrightarrow{OM}(t) = \overrightarrow{OA} + v_0\, t\, \overrightarrow{k} \tag{3.5}$$

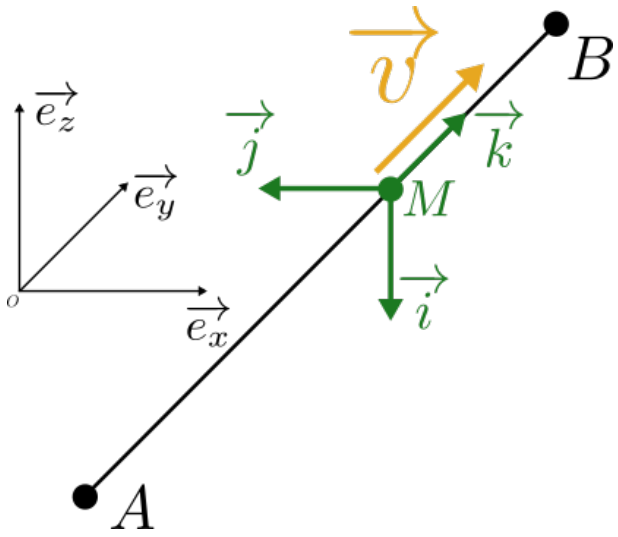


Figure 3.4: Global frame and flight trajectory.

3.2.2.2 Parallel-line scanning pattern

We denote as $\overrightarrow{r}$ the direction of the laser ray. $\overrightarrow{r}$ is rotating around $\overrightarrow{k}$ at constant angular speed $\omega = \dot{\theta}$ (see Figure 3.5). Thus:

$$\overrightarrow{r} = \cos(\theta)\, \overrightarrow{i} + \sin(\theta)\, \overrightarrow{j} \tag{3.6}$$

$$\forall\, t \in [0, t_B]\,, \; \theta(t) = \omega\, t + \theta_0 \tag{3.7}$$

Such a system is also defined by its *field of view* i.e. the angle range $[\theta_{min}, \theta_{max}]$ to which θ must belong for laser pulses being actually emitted, and by the *pulse rate* (frequency at

which pulses are emitted) f_p.

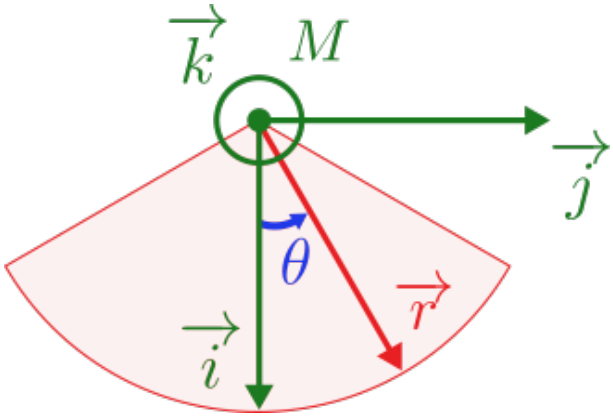


Figure 3.5: Parallel-line scanning pattern: zoom in on the local frame and laser pulse direction. The red-blured area represents the *field of view*.

3.2.2.3 Elliptical scanning pattern

Denoting as $\overrightarrow{r}$ the direction of the laser ray, and using $(M, \overrightarrow{u}, \overrightarrow{v}, \overrightarrow{w})$ as the canonical spherical coordinate frame (see Figure 3.6), $\overrightarrow{r}$ is rotating around $\overrightarrow{w}$ at constant angular speed $\omega = \dot{\phi}$ with ϕ being the azimuthal angle of $\overrightarrow{r}$. The polar angle θ is constant.

The laser ray's direction $\overrightarrow{r}$ is:

$$\begin{aligned}
\overrightarrow{r} &= \sin(\theta)\cos(\phi)\overrightarrow{u} + \sin(\theta)\sin(\phi)\overrightarrow{v} + \cos(\theta)\overrightarrow{w} \\
\overrightarrow{r} &= \sin(\theta)\cos(\phi)\overrightarrow{-j} + \sin(\theta)\sin(\phi)\overrightarrow{k} + \cos(\theta)\overrightarrow{-i} \\
\overrightarrow{r} &= -\cos(\theta)\overrightarrow{i} - \sin(\theta)\cos(\phi)\overrightarrow{j} + \sin(\theta)\sin(\phi)\overrightarrow{k}
\end{aligned}$$

with :

- $\forall t,\ \theta(t) = \theta_0$ (e.g. $\pi - \frac{\pi}{9}$)
- $\forall t,\ \phi(t) = \omega t + \phi_0$

3.2.2.4 Noise model

The quality of LiDAR data has recently been surveyed by [79]. We are especially interested in the accuracy of the point coordinates that a real LiDAR can achieve. Most of studies in this area agree to state that it can be split up in an **altimetric** and a **planimetric** component. Values are obviously influenced by the modernity of the system but they also vary depending on the type of terrain that is considered (bare soil, low grass, forestry) and the flight parameters (altitude, speed). See [80], [81] for studies on altimetric error and [82], [83] for planimetric error analysis. As suggested by these contributions, we

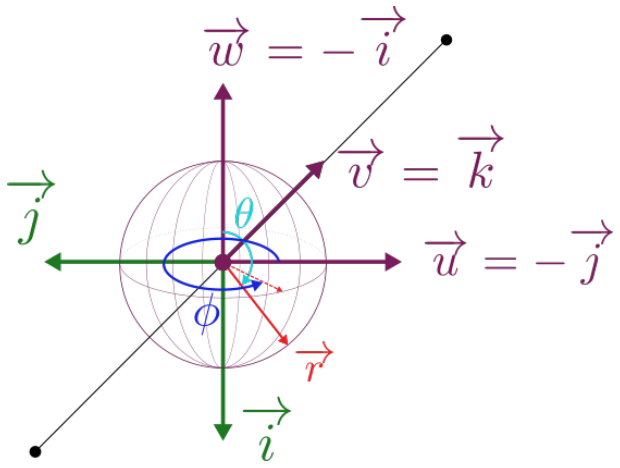


Figure 3.6: Elliptical scanning pattern: zoom in on the local frame and laser pulse direction.

suppose **a normal distribution of errors**, differentiating them into planimetric $\Delta x, \Delta y$ and altimetric Δz components:

$$\Delta x, \Delta y \sim \mathcal{N}\left(\mu_{xy}, \sigma_{xy}^2\right) \;\; ; \quad \Delta z \sim \mathcal{N}\left(\mu_z, \sigma_z^2\right) \tag{3.8}$$

3.2.2.5 Implementation

We implemented the method described above in C++ using CGAL library [84]. Rays are traced from the virtual aerial station in direction $\overrightarrow{r}$ and their exact intersections with the virtual environment are computed. Gaussian noise following 3.8 is added to these perfect intersections. As *oriented normals* or *sensor positions* are required for some reconstruction algorithms, point clouds containing this additional information are generated after any simulation.

3.3 Evaluation

The algorithms we assessed have been run on a point cloud we generated using the LiDAR simulator described in section 3.2 and the virtual environment displayed in Figure 3.3. We selected the following ones:

- **Poisson N**: *Poisson* [13] using Neumann boundary condition
- **Poisson D**: *Poisson* [13] using Dirichlet boundary condition
- **SSD**: *Smooth Signed Distance* [28]
- **PSS**: *Point Set Structuring* [22]
- **RESR**: *Robust and Efficient Surface Reconstruction* [21]

- **WMWU**: *Watertight Mesh generation With Uncertainties* [23]

Resulting surfaces are shown on Figure 3.7.

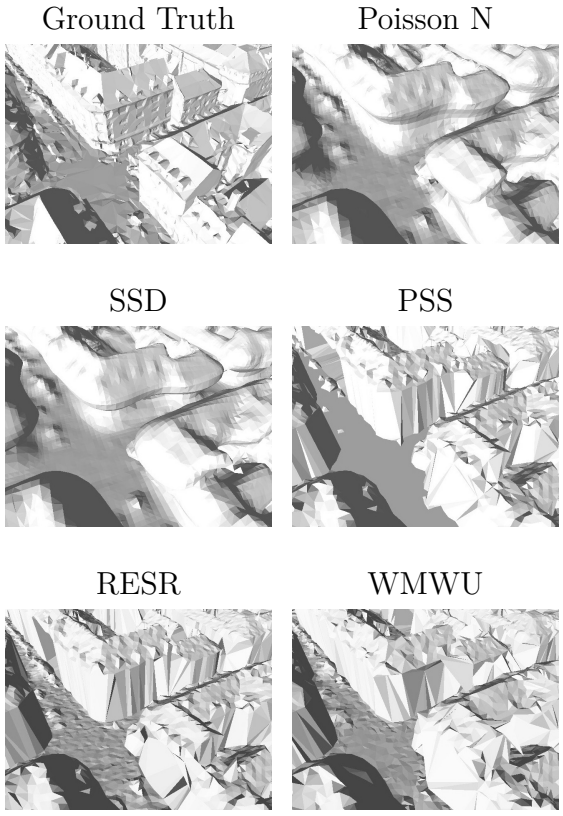


Figure 3.7: Result of the different algorithms evaluated on a crossroad of the scene

3.3.1 Experimental parameters and methodology

To generate the point cloud on which we evaluated the various algorithms, we used the *parallel-line* scanning pattern of the aerial simulator presented in Section 3.2. Values of the experimental parameters we used are to be found in Table 3.1. As regards to the evaluation part, we set to $R = 0.3\,m$ the Poisson-disk sampling radius. This parameter is chosen by considering the trade-off between sampling density and computational time.
As every algorithm we assessed can be tuned, we first studied the performance of each of them for a small number of α's, changing some parameter values. We then selected the best version of each algorithm and carried out the full evaluation for which the results are presented in section 3.3.2.

3.3.2 Quantitative results

We plotted the mean precision on Figure 3.8 and mean recall on Figure 3.9 (following equations 3.2 and 3.3) as a function of α.

Symbol	Value	Unit	Description
h	1 000	m	Flying altitude
v_0	60	$m.s^{-1}$	Flying speed
ω	150	Hz	Angular speed
$\Delta\theta$	40	$(^\circ)$	Field of view
f_p	400 000	Hz	Pulse frequency
σ_{xy}	0.13	m	Planimetric error
σ_z	0.05	m	Altimetric error

Table 3.1: Values of experimental parameters we used (parallel-line pattern).

RESR [21] achieves the best performance both in terms of precision and recall and for any value of α. Additional information provided by sensor positions is certainly helping but we can assume they are making the best use of it, in comparison to **WMWU** [23].

Results of the evaluation of **Poisson** [13] confirm that *Neumann* (**Poisson N**) leads to a better reconstruction of open scenes as it enables the surface to extend out to the boundary of the domain. In contrast, *Dirichlet* (**Poisson D**) enforces hard watertightness. Consequently, an undesirable closure of the surface from the bottom leads to a poor precision, but only for high α's. We now see how important it is to differentiate between global and local evaluation.
WMWU [23] evaluation shows a comparable result as those provided in the original article. This method provides good results regarding the recall metrics while precision quickly worsens when α increases. The main explanation is the same as for Poisson with Dirichlet boundary condition: the surface is artificially closed to ensure watertightness, leading a great amount of surface to lie away from the ground truth.
SSD [28] curve logically follows the one of Poisson N, as both methods use smooth basis functions to solve their equation.
PSS [22] could have been expected to perform better since it is based on primitive detection and an urban scene is full of planar regions. However, it is quite sensitive to missing data and facades of the buildings are mostly out of reach for a parallel line scanning pattern LiDAR.

3.4 Conclusion

Surface mesh reconstruction from remote sensing point clouds is a challenging task that becomes more and more important as data acquisition starts from a purely vertical perspective. Our evaluation shows important differences between the various state of the art approaches. In this chapter, we evaluate the quality of the interpolation that surface reconstruction algorithms perform between the input points. We argue that this should be measured as a function of the distance at which we expect the algorithm to interpolate

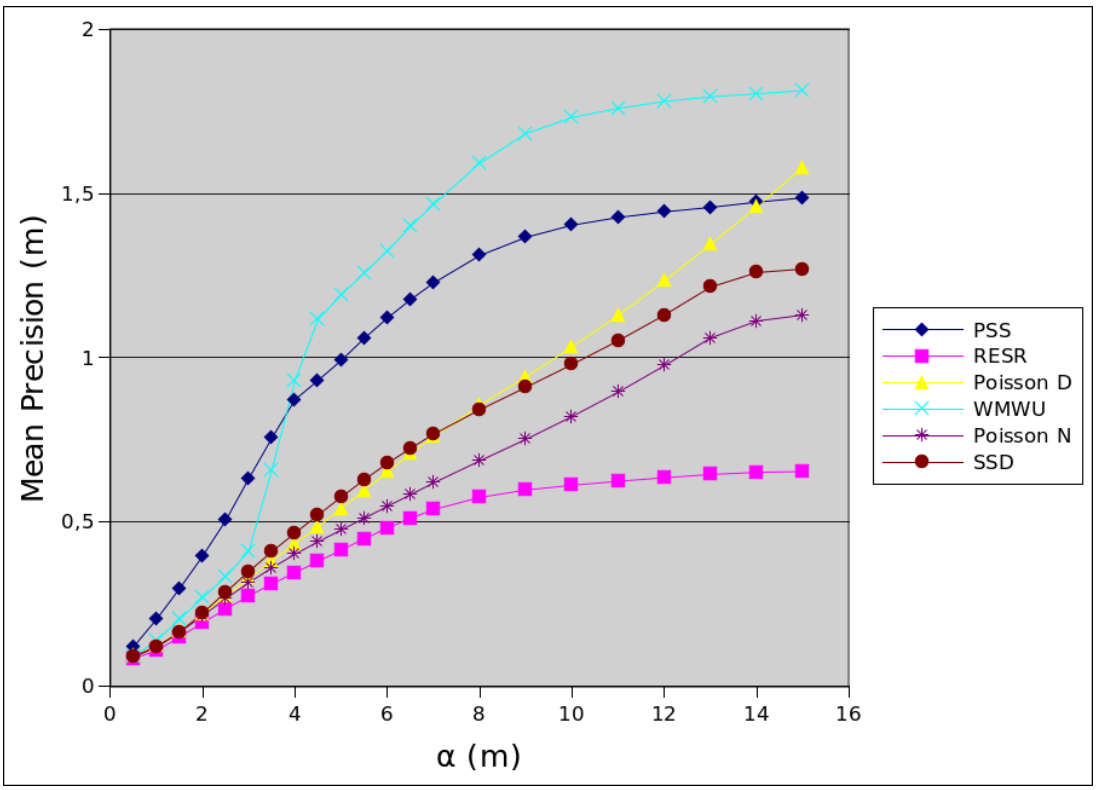


Figure 3.8: Mean Precision

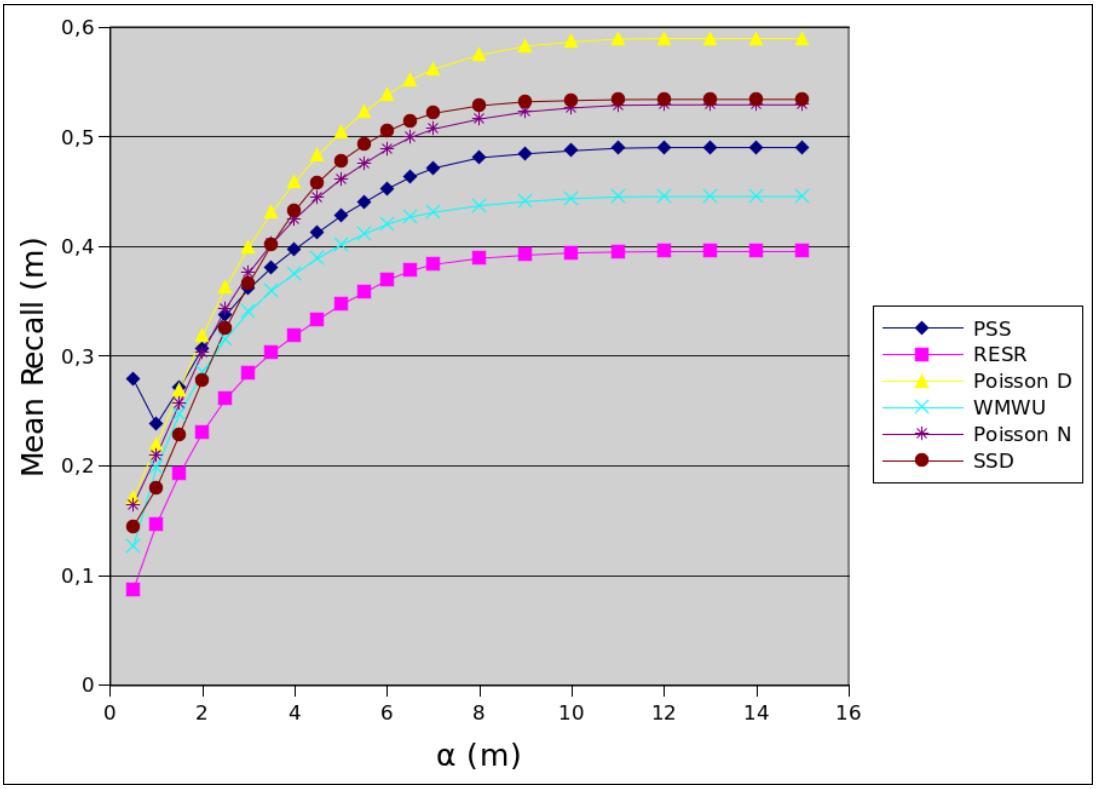


Figure 3.9: Mean Recall

the surface. The resulting metric take the form of curves indicating two quality criteria as a function of the maximum interpolation distance. Our study shows that [85], while quite old, remains a very good choice. The very popular **Poisson** method [11] is efficient and scales well but does not preserve sharp features that are very present in our urban evaluation scene.

Chapter 4

Evaluating surface reconstruction using real data

4.1 Introduction

The evaluation of surface reconstruction from synthetic data, as presented in Chapter 3, offers several advantages including the possibility to directly compute the difference between the reconstructed surface and the ideal one. However, high-quality synthetic meshes representing large scenes are expensive to obtain thus reducing the range of possibilities. This Chapter tackles the evaluation of surface reconstruction in the real-world case where we do not have access to such a synthetic ground truth. We call "**real data**" the data acquired in the physical world with real sensors. This includes LiDAR scans, images, RGB-D images, etc. As is usually done to address this issue, we assess the reconstruction of real scenes from real data only based on other real data of significantly higher quality. Even though this idea is quite typical, the main contribution of the work presented in this chapter lies in the way that we assess the difference between the reconstructed surface and the high-quality real data as inconsistencies, inspired by recent work on change detection [86]. This endeavour is hard because of several aspects.

Limits in the quality of the ground truth: the specificity of working with real data is the presence of *noise* in what we consider as the ground truth. In addition, real data is always sparse and incomplete, which means that we do not know the state of space (*occupied* by the object or *empty*) everywhere. This raises the question of how to assess pieces of reconstructed surface in *unseen, unobserved* regions.

Our contributions are twofold:

1. We propose a setting where the high-quality data used to compute metrics is significantly better than the low-quality data on which surface reconstruction is performed in three separate ways:

- Coverage: we use multiple data sources acquired from multiple points of view to ensure that the high-quality data has a significantly better coverage of the surface to reconstruct than the low-quality data.

- Density: we ensure that the density of the points in the high-quality data is significantly better than that of the low-quality data.

- Noise: we ensure that the noise level is lower in the high-quality data than in the low-quality data.

2. We propose metrics that penalise inconsistencies between the surface to be evaluated and the high-quality data: a piece of surface reconstructed within a volume unseen by the high-quality data will simply not be evaluated as we have no information on it. This does not mean that we do not evaluate the hole-filling capacity of the evaluated methods. As the high-quality data has more coverage, we evaluate hole filling exactly where we have the data to do so.

Assumptions and priors: algorithms make different assumptions about the type of shape that needs to be reconstructed and this leads to very different properties. Consequently, depending on the metrics' definitions, these assumptions can dramatically influence the assessment and sometimes in an unjustified manner. An example of such a situation is shown in Figure 1 where the left model would be attributed a bad mark because of the red piece of surface even though the rest of the model is correct. Does this red piece of surface need to be taken into account when assessing the model? We tackle this issue by defining metrics that only assess hole filling where relevant *high-quality* data is available.

Our proposed metrics are based on the visibility information contained in the high-quality data. We assess the reconstructed surface only where the real one has been observed. For that purpose, we make an extensive use of *sensor positions* (positions from which points have been acquired). This information is easy to access and it provides us with a full ray along which we know that space is free, instead of just a single position where we know that the real surface lies. We used these newly-defined metrics to assess several open-source surface reconstruction algorithms and a licensed one on different types of scenes. In Section 4.2, we define our metrics and discuss their nature. In Section 4.3, we present the three datasets on which we tested our evaluation protocol and present the experimental setup we used to generate the high-quality data. Results are detailed and analysed in Section 4.4, before we present the conclusions of this work in Section 4.5.

4.2 Evaluation protocol

4.2.1 Intuition

Let us give some examples of situations where current metrics are not adequate to evaluate surface reconstruction, and what we suggest would be an improvement. This is going to help understand our metrics' definitions in Section 4.2.3.

First, as presented in Section 1.3, comparing a reconstructed mesh with a ground truth point cloud can be done by computing the distances from those points to the mesh model. While this seems like a good starting point to assess how well holes have been filled around those points, it is inadequate to evaluate the overall accuracy of the surface. Figure 4.1 shows an example of such a situation where a surface would be evaluated as almost perfect even though large portions are clearly incompatible with the ground truth if we take into account the positions from which points have been acquired.

Secondly, it is possible to measure accuracy solely with ground truth points by sampling the reconstructed surface and measuring the distance from these samples to the ground truth points. Nevertheless, large pieces of the reconstructed surface might be judged as being of poor quality (if lying far from the nearest ground truth point) despite being correct, just because of a low ground truth density. We want to assess both accuracy and completeness only in regions where ground truth information is available, and this is possible using sensor positions as shown in Figure 4.1.

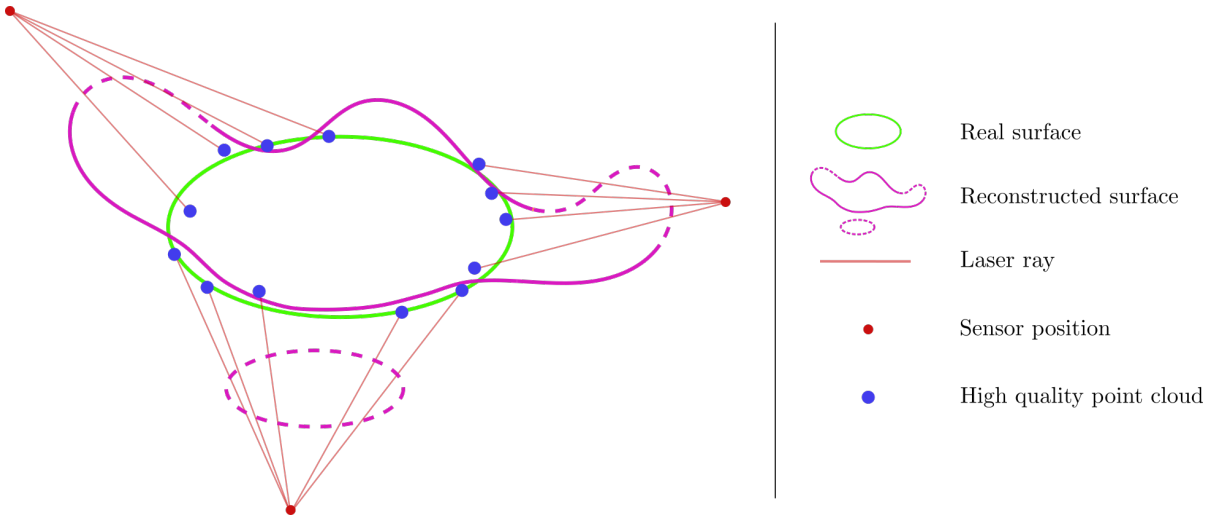


Figure 4.1: The importance of sensor positions: the dashed part of the reconstructed surface can be identified as wrong by making use of sensor positions when the high-quality point cloud does not provide enough information.

Thirdly, as surface reconstruction has often been evaluated visually, we wanted to find metrics that would imitate this human intuition-based assessment. We believe that

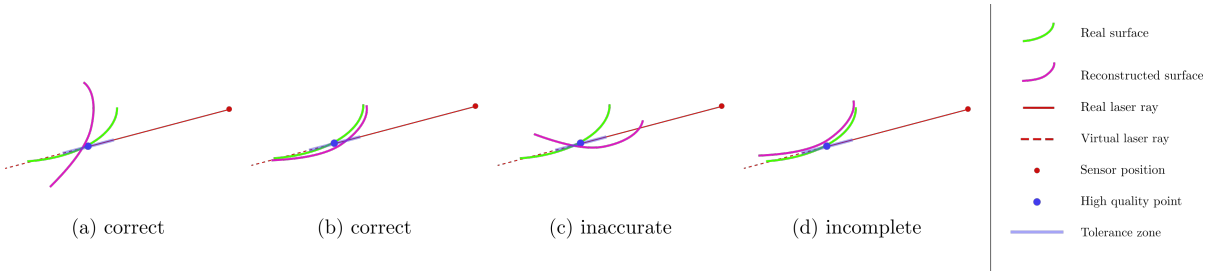


Figure 4.2: These four cases would be evaluated in the same way by a basic point-to-mesh distance. However, they are very different in terms of what a human being would be able to see from the sensor position.

visibility-based metrics are more appropriate for assessing how the reconstructed surface matches the real one everywhere where we can *see* and compare them. Figure 4.2 shows four situations for which the piece of reconstructed surface would be marked similarly by a point-to-mesh distance. The distance from the high-quality point to the nearest piece of reconstructed surface is indeed the same in all four situations. However, we are certain that they should correspond to three completely different outcomes and we want our metrics to be able to differentiate between them. In particular:

- in (a), the reconstructed surface lies slightly *behind* the high-quality point. We consider it as correct at the threshold defined by our tolerance zone.
- in (b), the reconstructed surface lies slightly *in front of* the high-quality point. As in (a), we consider it as correct even though we can measure a slight error.
- in (c), whilst the piece of real surface corresponding the high-quality point has been recovered just like in (a), the reconstructed surface hides the nearest intersection by crossing the laser ray, resulting in an obvious inaccuracy. What one would see by looking in the direction of the laser ray is not the right piece of reconstructed surface but rather something else far in front of it.
- situation (d) might look similar to (b) but there is actually no intersection between the reconstructed surface and the laser ray. Consequently, what one would see by looking in the direction of the laser ray is not the right piece of reconstructed surface but something else in the background. We thus consider that the surface has not been recovered at all here.

ETH3D benchmark [87] proposed metrics to evaluate how well a two- or multi-view stereo point cloud matches a LiDAR-based one, using sensor positions. While surface reconstruction is a different task to MVS (in particular in terms of the expected output properties), [87] still evaluates the matching between two 3-dimensional structures and we were inspired by their use of sensor positions. However, we chose different paradigms that are more adapted to surfaces, so our evaluation protocol is considerably different.

The *ETH3D* benchmark [87] defines *completeness* as the proportion of ground truth points for which the distance to its closest reconstructed point is below a given threshold. We could keep this definition and find the point from the reconstructed surface that minimises the distance to each point of the high-quality point cloud. Nonetheless, given that we know in what direction this point is supposed to be encountered thanks to the sensor position, we find it more relevant to compute the distance between the high-quality point and the nearest intersection between the corresponding laser ray and the mesh model. In other words, while the most natural adaptation of this point-to-point distance would be a point-to-mesh distance, we believe that for each ray that hit the the real surface there should be a piece of reconstructed surface close by and *along the ray*.

However, we also leverage the information given by each laser ray: the space between each sensor position and its associated high-quality point should be *empty*. We soften this property by defining a tolerance zone: a piece of reconstructed surface will be considered as correct if its distance along the ray from the high-quality point is smaller than a given threshold d_{max}. Every piece of surface further than d_{max} and situated *in front of* the high-quality point will affect the accuracy of the model.

Contrary to *ETH3D* benchmark [87], we do not model the shape of a laser beam as a truncated cone. The first reason for this is that we do not *need* to, since the model we are trying to evaluate (a surface) is continuous instead of discrete (a point cloud). Hence, we are not at risk of missing any part of it. In addition, it makes it simpler to get a point as the intersection between a ray and a piece of surface. This way, every couple (ray - piece of surface) gives the same amount of information.
[87] uses voxels to prevent a "cheating" strategy from achieving both high accuracy and completeness despite raising other issues. For example, regions of low ground truth density contribute as much as high density ones while encapsulating less information. A cheating strategy for surface reconstruction would be to add several parallel layers of surface in regions of high confidence. We do not need to discretise space as in [87] since for each ray we propose to keep only the closest intersection as a potential correct one and penalise all the ones situated in front of it. Note that even layers situated behind the closest intersection might be obstacles to rays pointing at another object in the background.

If the nearest intersection is found *behind* the high-quality point at a greater distance than d_{max}, or if no intersection is found at all, then we consider that this piece of surface has not been recovered. This therefore affects the completeness of the model.

4.2.2 Definitions and notations

In this chapter, we want to evaluate the quality of surface reconstructions from low-quality data $\mathcal{P}_{If}$, with only having access to high-quality data $\mathcal{P}_{GT}$ of the same scene and without having access to the perfect ground truth surface that the algorithms are supposed to produce.

- $\mathcal{P}_{If}$: the *low-quality* point cloud that will be fed to the evaluated surface reconstruction methods to produce the output surface meshes to be evaluated.

- $\mathcal{P}_{GT}$: the *high-quality* (ground-truth) point cloud with better coverage, higher density and less noise than $\mathcal{P}_{If}$ and for which we know the sensor positions, defining one ray per point.

- $\mathcal{M}_E$: the reconstructed mesh to evaluate, produced by an algorithm from $\mathcal{P}_{If}$.

- d_{max}: the maximum distance at which we evaluate the reconstruction. It is a parameter which influences the different metrics as we use it to separate noise (distance $< d_{max}$) from outliers (distance $> d_{max}$).

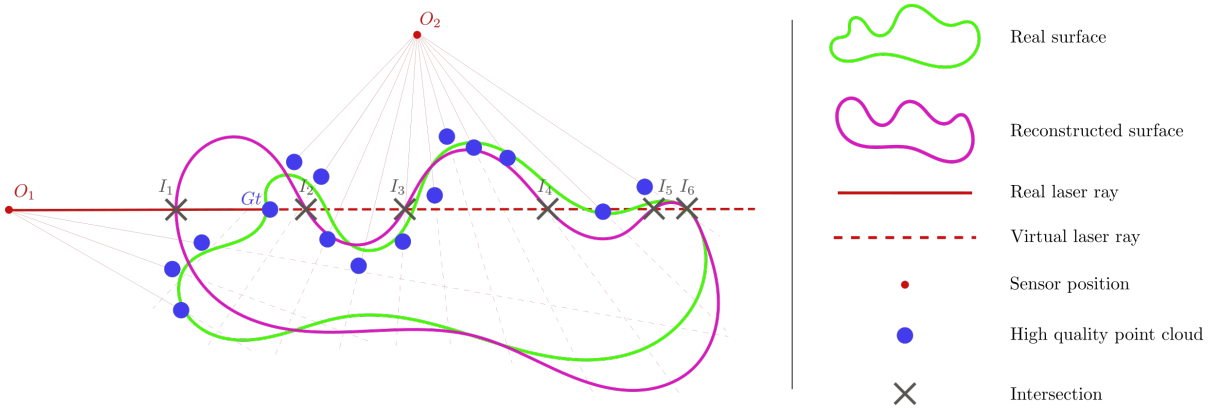


Figure 4.3: Toy example to visualise the definitions of the metrics. The real surface has been scanned from two positions: O_1 and O_2. A given real laser ray (the thick one) was cast from O_1 and it hit the real surface at Gt. Note that the position of the intersection might be noisy, hence the shift between the real surface and the high-quality point cloud. We compute all intersections between the associated virtual ray (i.e. the extension of the real laser ray) and the reconstructed surface. In this case, it results in six intersections $I_{1,\dots,6}$. The closest intersection to Gt happens to be I_2 so the "**ray distance**" metric for that particular ray is the distance (Gt, I_2). If $(Gt, I_2) < d_{max}$, Gt is to be counted as a **TP**, otherwise it will not be taken into account in the evaluation since it is situated *after* Gt. Note that this piece of surface might still be evaluated thanks to another ray (as one emanating from O_2, for example). Besides, we found one intersection I_1 *on the way to* the closest intersection I_2 which is counted as a **FP**.

4.2.3 Metrics definitions

Similar to [88], we would like to assess both the precision of each part of the reconstructed surface (each part of the surface should lie near some part of the real surface) and the completeness of the model (there should be as few missing parts of the real surface as possible). As we do not have access to a digital model of the real surface, we cannot compute the actual *precision* and *recall* as in [88]. Our knowledge of the ground truth is limited to the high-quality data $\mathcal{P}_{GT}$. However, we also know where the surface is **not** supposed to lie since we know the sensor position from which every point of $\mathcal{P}_{GT}$ has been acquired, thus defining a ray of free space. Therefore, we define a *precision* metric that penalises inconsistencies between the reconstructed surface and the visibility information contained in $\mathcal{P}_{GT}$. We propose an equivalent of the *recall* metric: for each $[OP]$ ray from $\mathcal{P}_{GT}$, we compute the distance from the P to the closest intersection between the $[OP)$ half line and the reconstructed surface. Here is the formalisation of these metrics in more detail:

- "Ray distance": for each point/ray $(p, r) \in \mathcal{P}_{GT}$, we compute the distance from p to the closest intersection (which we will denote as c) between r and $\mathcal{M}_E$ (among all potential intersections, we choose the one with the smallest distance with respect to p). If this distance is $< d_{max}$ then the piece of reconstructed surface is considered as being *correct* and we add this distance to an array of distances.

- "Precision": for each point/ray $(p, r) \in \mathcal{P}_{GT}$, if the "ray distance" (between p and c) is $< d_{max}$ then we count c as a *True Positive (**TP**)* (since there is a piece of surface and it is correct). Otherwise, if the intersected point c lies at a distance greater than d_{max} and it is *before* the corresponding high-quality point p, we consider it as being false and we count it as a *False Positive (**FP**)* (since there is a piece of surface and it is false). We consider that we cannot say anything about the closest point c if it lies at a greater distance than d_{max} and it is situated *after* the corresponding high-quality point p (neither can we for all intersected points lying after it), so we just ignore them. All intersected points lying *before* this closest intersection c are also counted as ***FP*** as they are inconsistent with the corresponding ray of free space (p, r). This is enough to define the precision ratio (equation 4.1).

- "Recall": it is defined as the ratio between the number of ***TP*** and the number of cast rays (equation 4.2. Every high-quality point $p \in \mathcal{P}_{GT}$ is either mapped to its corresponding ***TP*** (in which case the piece of real surface has actually been recovered by the algorithm) or not (meaning a lack of exhaustiveness of the reconstruction and corresponding to a *False Negative* ***FN***). The number of rays thus equals the sum of the ***TP*** and the ***FN***.

- "Cumulative distances": the cumulative histogram of the ray distances where the x-axis corresponds to the distance and on the y-axis we plot the number of points for which the ray distance is below the x-distance, divided by the total number of rays cast. It contains the information of both mean ray distance and recall. The right-most value will be at a given distance from the $y = 1$ line, which corresponds to the number of points for which the ray distance is above d_{max} (if "re-multiplied" by the total number of rays).

$$\text{Precision} = \frac{TP}{TP + FP} \tag{4.1}$$

$$\text{Recall} = \frac{TP}{TP + FN} = \frac{TP}{\text{number of rays}} \tag{4.2}$$

$$\text{F-score} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{4.3}$$

As we define precision and recall as ratios, the harmonic mean (F-score, see equation 4.3) allows ranking the methods by taking into account both metrics.

4.2.4 Tuning / Training

The algorithms we evaluate in this chapter are either *tunable* for the most part or they *learn* parameters in order to reconstruct surfaces. For example, **DGNN** [26] needs a training dataset in order to learn the parameters of its model and **Poisson** [11] can be run at different resolutions by changing the *depth* of the octree that is used. In order to be as fair as possible, we used the same dataset to *tune* or *train* the algorithms. We therefore ran the non-learning-based methods with different values for their parameters and carried out an evaluation. A first interesting result is that for some methods, it can be hard to obtain a good performance regarding the *precision* and the *recall* metrics at the same time. For **PSS**, the higher the value of the *trade-off* parameter λ (therefore the more importance given to the prior term, see equation 1.2), the higher the precision (up to a certain value) but the lower the recall. Maximising the score of one metric (by varying λ) results in minimising the score of the other one (we explain why in Section 4.4). Thus, we selected the two λ values that maximise each of these metrics individually.

The *tuning/training* dataset that we decided to use is composed of three scenes from *STRAS*. The reason behind this choice is the availability of ground-truth meshes which are absolutely necessary for the training phase of **DGNN**. We then also used these three scenes to find the best parameters (in terms of *ray distance*, *precision* and *recall*) for the non-learning-based algorithms.

4.3 Input data

We aim to evaluate the algorithms in very different scenarios as the methods' priors might influence the quality of the reconstruction depending on the type of scene or the type of data involved. We therefore compute the metrics introduced above on three significantly varying datasets.

4.3.1 *STRAS*: Strasbourg dataset and LiDAR simulator

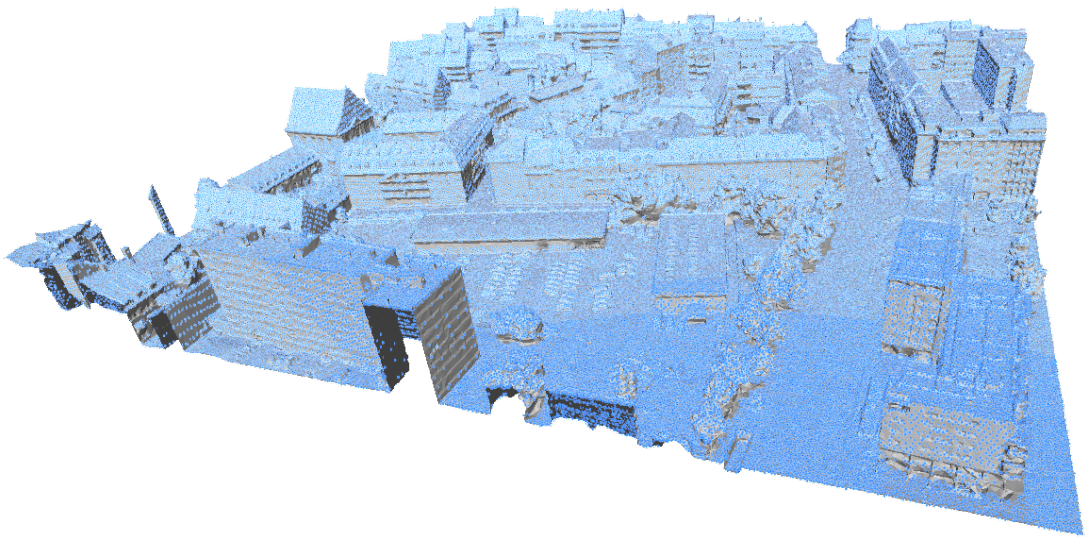


Figure 4.4: Strasbourg scene (mesh in grey, point cloud in blue)

4.3.1.1 Elliptical aerial LiDAR simulator

In order to control the data itself, we started by using the same synthetic dataset [78] as in Chapter 3. Figure 4.4 shows a 250 metre by 250 metre tile of this mesh. In order to generate the ground-truth and the input point clouds $\mathcal{P}_{GT}$ and $\mathcal{P}_{If}$, we used the **aerial LiDAR simulator** that we introduced in Chapter 3 [88]. This is because such a large urban environment is typically scanned using an airborne LiDAR system. The problem with the parallel line pattern is that facades that are perpendicular to the direction of the plane are not reachable by the laser ray and thus are absent from the resulting point cloud. In order to overcome this issue, we used the **elliptical scanning pattern**. It is indeed better suited to urban environments, for the laser ray will be able to point at far more facades than with the parallel line pattern, resulting in a better coverage. The values of all the parameters we used can be found in Table 4.1.

4.3.1.2 Experimental setup

We aim to produce two point clouds $\mathcal{P}_{GT}$ and $\mathcal{P}_{If}$ in such a way that $\mathcal{P}_{GT}$ should have less occlusions and be denser than $\mathcal{P}_{If}$. The scenes are all 250 metre by 250 metre tiles

Symbol	Value	Unit	Description
h	1 000	m	Flying altitude
v_0	60	$m.s^{-1}$	Flying speed
ω	150	Hz	Angular speed
θ_0	160	$(^\circ)$	Polar angle
f_p	400 000	Hz	Pulse frequency
σ_{xy}	0.13	m	Planimetric error
σ_z	0.05	m	Altimetric error

Table 4.1: Values of experimental parameters used.

of an urban environment for which positions are expressed in a global coordinate frame $(O, \overrightarrow{e_x}, \overrightarrow{e_y}, \overrightarrow{e_z})$ such that $\overrightarrow{e_z}$ represents the ascending vertical direction. In our setup, a trajectory of the LiDAR system for each scene is a single straight pass of the plane along axis $\overrightarrow{e_y}$ for $x = \alpha_x \, (x_{max} - x_{min})$, $\alpha_x \in [0, 1]$. We generate $\mathcal{P}_{If}$ thanks to one pass of the plane with $\alpha_x = 0.5$ and $\mathcal{P}_{GT}$ thanks to three passes with $\alpha_x \in \{0.25, 0.5, 0.75\}$. We denote P_{α_x} as the point cloud resulting from the flight $x = \alpha_x \, (x_{max} - x_{min})$. We then have: $\mathcal{P}_{If} = P_{0.5}$ and $\mathcal{P}_{GT} = P_{0.25} \cup P_{0.5} \cup P_{0.75}$. This way, $\mathcal{P}_{If}$ forms part of $\mathcal{P}_{GT}$ and contains a lot more occlusions in particular on facades parallel to the direction of the plane. Figure 4.5 gives an example of such a situation.

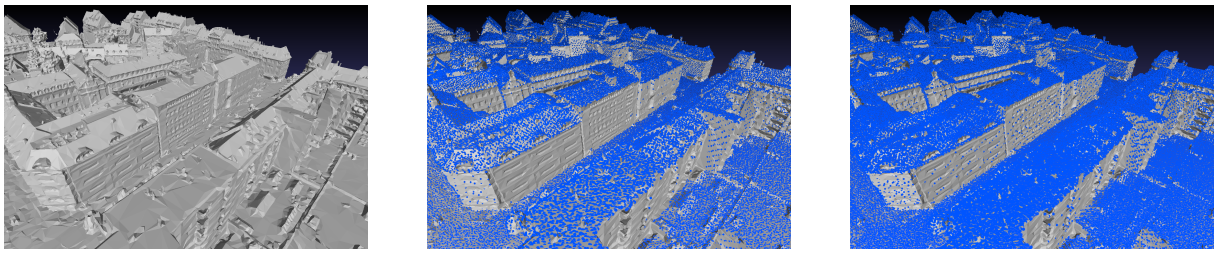


Figure 4.5: Left: Mesh, Centre: $\mathcal{P}_{If}$, Right: $\mathcal{P}_{GT}$. The facades parallel to the direction of the plane are a lot more occluded in $\mathcal{P}_{If}$.

4.3.2 *ENSG* dataset: indoor and outdoor terrestrial LiDAR scan

As the original goal of our study was to propose an evaluation protocol suited for real data, this dataset is only based on real data that we acquired ourselves. The resulting point clouds intensity channel can be visualised on Figure 4.6.

4.3.2.1 Stationary LiDAR station

We used the stationary LiDAR station "*Leica ScanStation P40*" for which we will give a brief introduction. Once the station is settled, rays are cast 360° horizontally around its origin and 290° vertically (the ground area immediately underneath the station remains unobserved during the acquisition). It can acquire up to $1,000,000$ points per second

from 0.4 metre to 270 metres with a 3D position accuracy of 3 millimetres at 50 metres. In order to satisfy the condition of $\mathcal{P}_{GT}$ being of higher quality than $\mathcal{P}_{If}$, we decided to acquire points from more viewpoints to generate $\mathcal{P}_{GT}$, using the same LiDAR station. We thus scanned each environment from five positions: the four vertices of an approximately regular 1.5 metres side length tetrahedron (O_1, O_2, O_2, O_4) and its centre of mass O_5 as shown in Figure 4.7. In order to evaluate surface reconstruction algorithms in several different scenarios, we repeated this procedure for three scenes:

- "*Building*": an outdoor scene made of a building, a sloped road, trees and an opening to another scene called "*Car park*".
- "*Car park*": an indoor scene with pipes, partially occluded cars and open doors, including one communicating with the outdoor scene "*Building*".
- "*Clutter*": a closed indoor scene with a lot of occlusions due to a high density of objects.

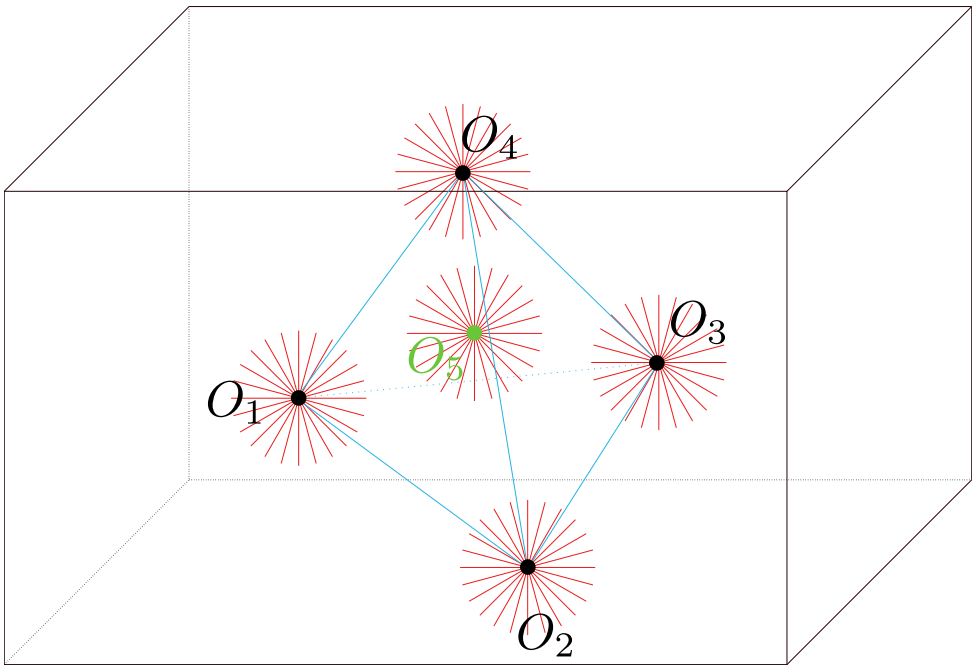


Figure 4.7: Tetrahedron-like viewpoints. (O_1, O_2, O_2, O_4) form an approximately regular 1.5 metres side length tetrahedron and O_5 is positioned at its centre of mass. The black rectangle represents a room in which we installed our stationary LiDAR.

4.3.2.2 Matrix format and sub-sampling

Following the definition of our protocol (see Section 4.2), we need to generate poorer-quality point clouds to run reconstructions. Our LiDAR system has a spherical geometry as illustrated in Figure 4.8. The horizontal resolution and vertical resolution of the scanner define a fixed number w of values for ϕ and a fixed number h of values for θ respectively. Laser rays are thus cast in the directions given by every pair of angles (ϕ, θ). We can

thus represent the points as a matrix of *height* h and *width* w and then index all points by their $(i, j) \in [|1, h|] \times [|1, w|]$ coordinates. Figure 4.6 shows the intensity of the returns in this matrix-like format. Each raw acquired point cloud did not fit into the memory of our machine so we down-sampled them by keeping odd-indexed points as illustrated in Figure 4.8. Starting from the raw point clouds with origins centred on O_1, O_2, O_3 O_4 and O_5, we down-sampled them all (roughly four times) following this matrix-based scheme and $\mathcal{P}_{GT}$ is the union of these five four-time down-sampled point clouds. We repeated this matrix-based down-sampling scheme on the point cloud centred on O_5 to generate $\mathcal{P}_{If}$.

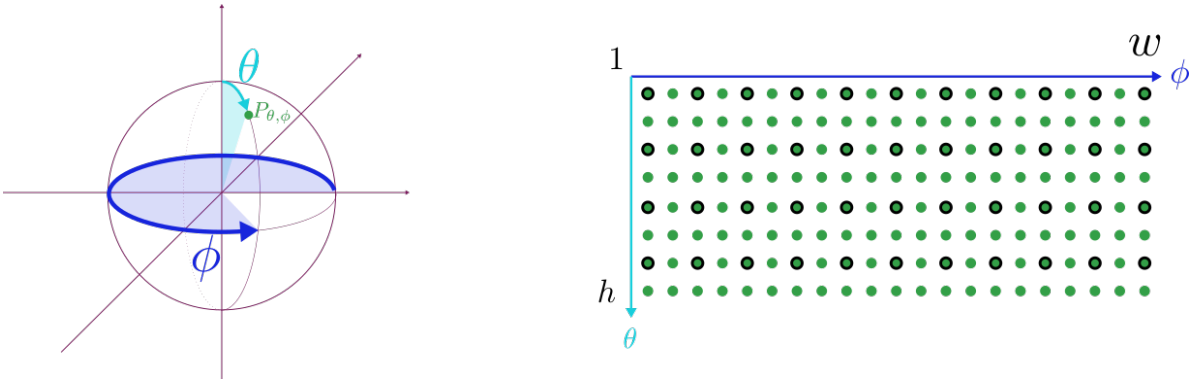


Figure 4.8: Matrix format-based subsampling. Left: spherical representation of a point cloud acquired with the stationary LiDAR station "*Leica ScanStation P40*". Each point acquired is defined by its spherical coordinate angles (ϕ, θ). Right: Matrix format-based sub-sampling: only points represented with a black outskirt are kept in the sub-sampled point cloud.

4.3.3 *ETH3D* dataset

[87] presents a two- and multi-view stereo benchmark. Their dataset contains several scenes with:

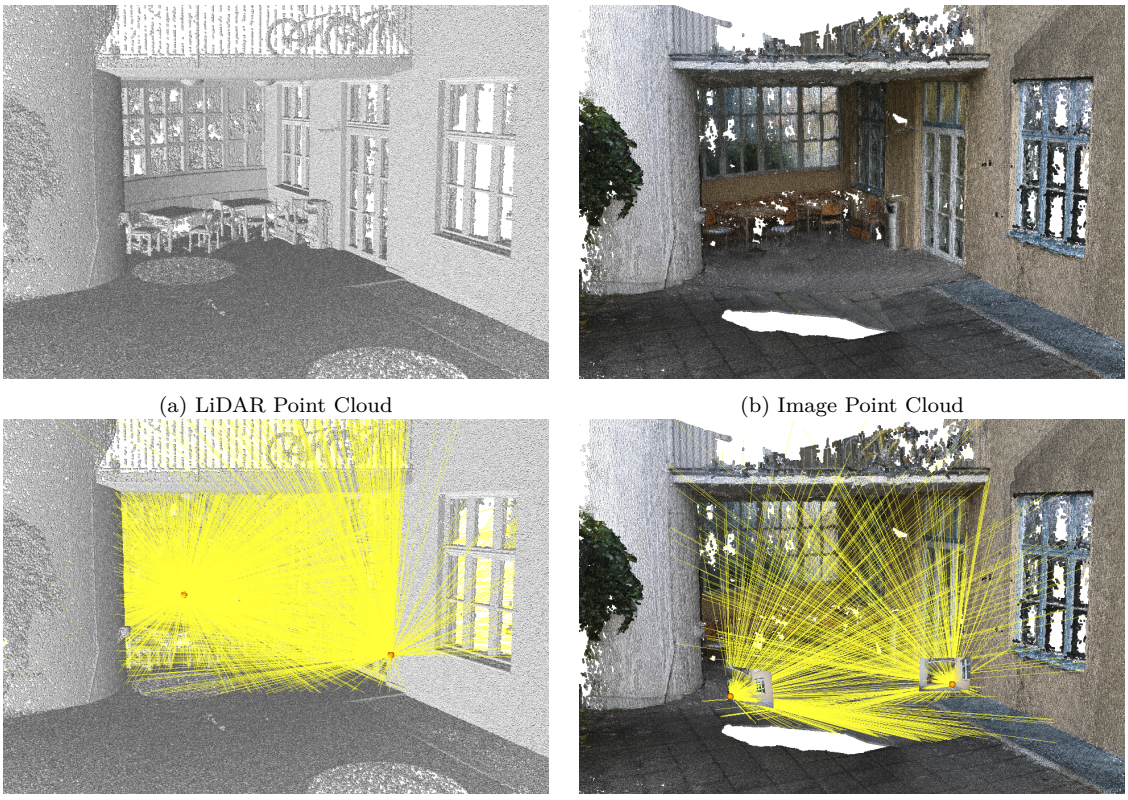
- input images at 24 Megapixel resolution on several scenes
- ground truth 3D laser scan point clouds

4.3.3.1 Generating the point clouds: Multi-View Stereo and real LiDAR

In order to generate the low-quality data, we used the OpenMVS [89] library to generate dense point clouds from images using the provided camera poses of three scenes (*Terrace*, *Courtyard*, *Pipes*) of the ETH3D train dataset. We used the DensifyPointCloud tool of OpenMVS with the standard settings, except for the following parameters: *number-views-fuse* $= 2$, *optimize* $= 0$ and *resolution-level* $= 4$. We used the provided LiDAR point clouds as our high-quality data. A typical MVS pipeline generates a much sparser and more noisy point cloud than what a laser scan provides. It also contains more outliers. For all these reasons, we consider it relevant to carry out an evaluation on a set of

MVS-based point clouds. We thus used three scenes from the *ETH3D* dataset [87], which can be seen in Figure 4.10.

Figure 4.6: Equirectangular projection of the LiDAR points. **Top**: Outdoor scene ("*Building*") from O_3 - **Middle**: indoor scene ("*Car park*") from O_2 - **Bottom**: Indoor Scene ("*Clutter*") from O_1. "*Building*" and "*Car park*" share some space thanks to what can be seen through the open door in the middle of both images.



(a) LiDAR Point Cloud

(b) Image Point Cloud

(c) LiDAR Point Cloud With Visibility

(d) Image Point Cloud With Visibility

Figure 4.9: **Point Clouds with Visibility Information:** Terrestrial point clouds (a, b) from the *Terrace* scene of ETH3D [87]. We visualise some of the sensor positions ● and lines-of-sight — (c, d).

Figure 4.10: Images of the three scenes from the *ETH3D* dataset we used. **Top**: Outdoor scene ("*Courtyard*") - **Middle**: indoor scene ("*Pipes*") - **Bottom**: Outdoor Scene ("*Terrace*").

4.4 Results

In this survey, we assessed:

- **RESR** [21]
- **SSD** [28] with two combinations of the octree *depth* and the B-spline *degree* parameters: $(depth = 8, degree = 2)$ and $(depth = 12, degree = 3)$.
- **Poisson** [13] with two values for the octree *depth* parameter: 8 and 11. The B-spline *degree* will always be 2.
- **DGNN** [26]
- **PSS** [22] with two values for the *trade-off* parameter: 0.1 and 0.6
- **Agisoft Metashape** 1.6.4 (the user manual can be found here) with *Extrapolated mode* and *ultra high* resolution.

For the *ETH3D* dataset only, we assess two other surface reconstruction algorithms as they are part of the OpenMVS [89] pipeline. Mesh reconstruction is initiated using **Exploiting Visibility Information in Surface Reconstruction to Preserve Weakly Supported Surfaces** (**WSS**) [90]. A refinement step is then carried out using **High Accuracy and Visibility-Consistent Dense Multiview Stereo** (**DMS**) [91]. We thus compute the metrics on the resulting meshes from both these methods.

STRAS PC3E44_3 ($d_{max} = 50\,cm$)							
Method	TP	FP	TP + FN	MD (cm)	P (%)	R (%)	F1 (%)
RESR	1278734	57089	1365120	6.58	95.73	93.67	94.69
DGNN	1246228	72899	1365120	6.79	94.47	91.29	92.85
Poisson 11/2	1267301	128888	1365120	8.91	90.77	92.83	91.79
PSS 0.6	1217138	81549	1365120	7.39	93.72	89.16	91.38
SSD 12/3	1271969	172285	1365120	9.56	88.07	93.18	90.55
SSD 8/2	1203150	167204	1365120	13.04	87.80	88.14	87.97
Poisson 8/2	1174069	151134	1365120	12.77	88.60	86.00	87.28
A. Metashape	1033179	195725	1365120	19.41	84.07	75.68	79.66
PSS 0.1	1280230	686825	1365120	7.47	65.08	93.78	76.84

Table 4.2: Raw numerical results for $d_{max} = 50\,cm$ (MD stands for mean distance, P for precision, R for recall and F1 for F-score)

4.4.1 *STRAS* dataset

In accordance with the survey conducted and published in [88] on the same dataset but with different assumptions and metrics, Table 4.3 shows that **RESR** achieves the best performance again on the urban environment of [78] regarding both precision and recall.

STRAS ($d_{max} = 50\,cm$)				
Method	mean distance (cm)	precision (%)	recall(%)	F-score(%)
RESR	**5.98**	**96.68**	94.88	**95.77**
DGNN	6.13	96.08	92.56	94.29
Poisson 11/2	7.99	93.19	94.31	93.75
PSS 0.6	6.67	95.08	91.35	93.17
SSD 12/3	8.53	90.72	94.63	92.63
SSD 8/2	11.57	90.11	90.48	90.30
Poisson 8/2	11.49	91.11	88.39	89.73
A. Metashape	16.21	87.92	80.63	84.11
PSS 0.1	6.76	72.72	**95.00**	82.28
mean methods	9.04	90.40	91.36	90.67

Table 4.3: Average numerical results on the three scenes from *STRAS* dataset sorted by decreasing F-score for $d_{max} = 50\,cm$

As evaluating surface reconstruction from real data only is harder than using synthetic data (we do not have an exhaustive ground truth), it is a very sound validation that the metrics that we defined without access to the ground truth surface show similar tendencies to the metrics that have access to ground truth surfaces.

We also carried out a more detailed evaluation by testing several values for the main parameters of selected methods. We found that the *trade-off* of **PSS** [22] has a big effect on the metrics. More precisely, the lower we set it (the more confidence we give to the data), the lower the precision but the higher the recall (and vice versa). A high confidence in the data results in a lot more interfaces between occupied tetrahedra and empty ones. Conversely, a higher *trade-off* λ gives more power to the regularisation term, resulting in fewer couples of adjacent tetrahedra being labelled differently, and so fewer triangles in the output mesh. When more confidence is given to the data, there are a lot more undesired triangles "floating" in regions of free space, which dramatically affects the precision. However, small structures might be erased from the mesh if less confidence is given to the data term, resulting in a poorer recall.

Poisson [11] and **SSD** [28] are both influenced positively by an increase in the octree depth. This was expected since more points are used to reconstruct the mesh, which results in an increase in the computation time and memory footprint.

DGNN performs a lot better on the *STRAS* dataset than on the two others, which highlights a problem in its capacity to generalise to scenes that differ from the ones in the training set. However, its poorer F-score performance on *ETH3D* and *ENSG* is mostly due to the precision metric. **DGNN** often succeeds in recovering the scene features but adds too many undesired triangles in the scene.

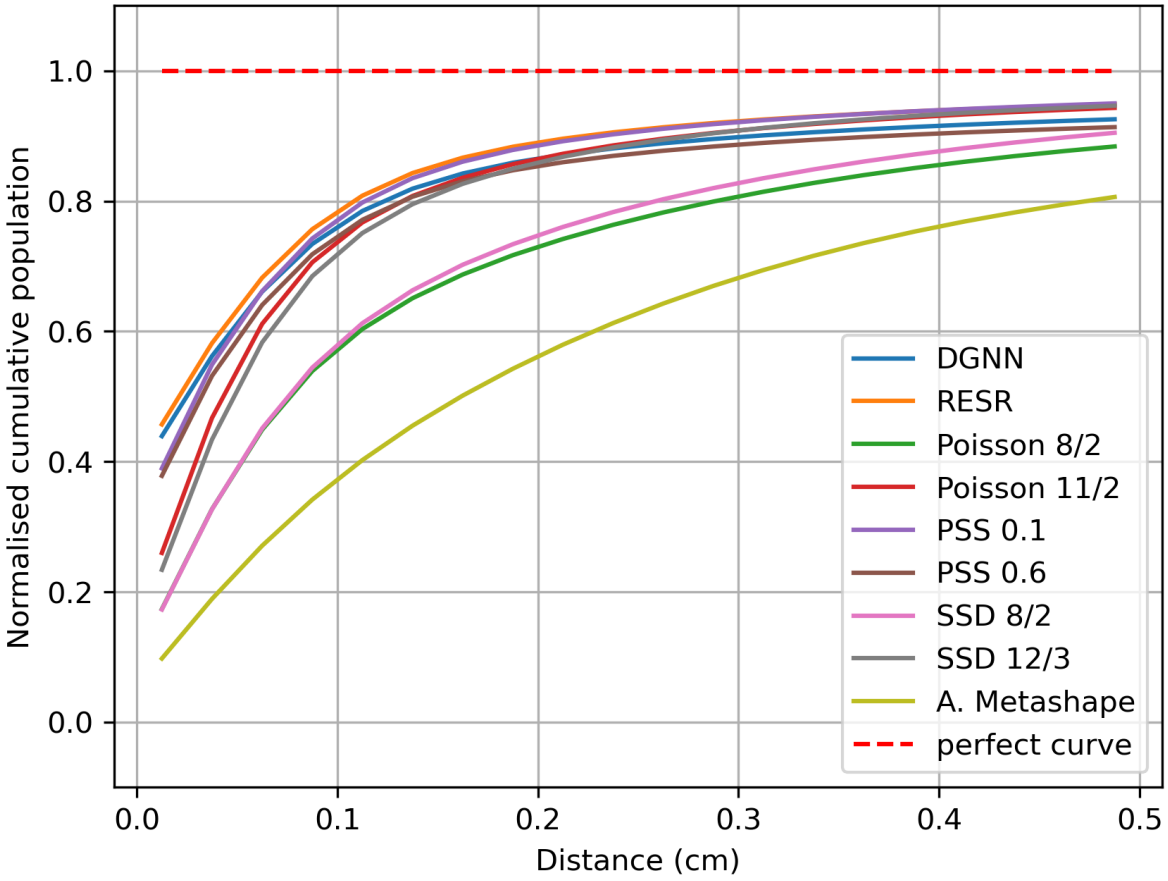


Figure 4.11: Cumulative distances over the three scenes from *STRAS* dataset for $d_{max} = 50\,cm$.

While precision, recall and ray distance provide complete information on the quality of the reconstruction, one might find it more intuitive to start by having a look at the cumulative distances shown in Figure 4.11. The precision at small range can be estimated as the area under the curve. The closer the curve is to the top left-hand corner, the better it is since this means that all the *True Positives* are actually very close to it. Besides, the highest value of each curve is the recall of the corresponding method so the gap between the cumulative population in the last category and the line $y = 1$ should be as small as possible.

4.4.2 *ENSG* dataset

The *ENSG* dataset, having been generated using a stationary LiDAR system, is the one containing the least amount of noise hence the overall good performance of all of the methods. In particular, we can see that the mean distance is generally a lot smaller than with other datasets even though the scenes themselves have much more complicated geometries and more occlusions.

Figure 4.12 shows the meshes reconstructed by every assessed algorithm on the **Park-**

ENSG ($d_{max} = 20\,cm$)				
Method	mean distance (cm)	precision (%)	recall(%)	F-score(%)
RESR	**0.45**	**93.10**	95.99	**94.51**
Poisson 11/2	0.90	78.38	96.96	86.22
Poisson 8/2	2.72	78.66	88.16	83.05
SSD 12/3	1.27	72.41	96.04	82.23
A. Metashape	1.63	79.00	83.95	81.37
SSD 8/2	3.04	71.95	87.77	78.96
DGNN	0.49	52.99	96.22	68.28
PSS 0.6	0.54	28.44	97.92	43.94
PSS 0.1	0.54	22.30	**98.19**	36.18
mean methods	1.29	64.14	93.47	72.75

Table 4.4: Average numerical results on the three scenes from *ENSG* dataset sorted by decreasing F-score for $d_{max} = 20\,cm$

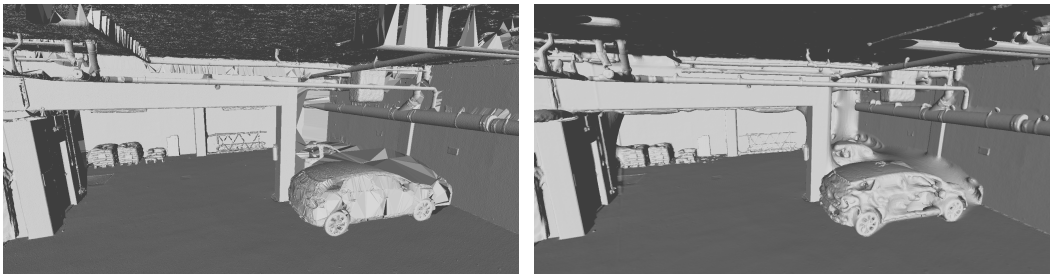
ing Lot scene (part of the *ENSG* dataset). One can fairly easily interpret the performance achieved by these methods by analysing the type of mistake they made on the corresponding scene. **RESR** succeeds at reconstructing most of the visible parts, and very few undesired triangles lie in free space (most of them are connecting the pipes to the wall and the ceiling).

At first glance, **Poisson** 11/2 reconstruction seems to be a lot more accurate than **Poisson** 8/2 so it might not be obvious why they have the same precision. This situation actually shows the interest of the mean distance metric. While the two reconstructed models are structurally the same, the difference between them is visible at close range: under the threshold d_{max}. Consequently, **Poisson** 8/2 has a much higher mean distance than **Poisson** 11/2 but achieves a similar precision. The same kind of argument holds for explaining the relatively poor performance of **SSD** 12/3 and **A. Metashape**: whilst being *locally* more accurate than **Poisson** 8/2, the meshes are *structurally* not in accordance with the visibility information provided by the high-quality point cloud. The precision metric is dramatically affected by large portions of surface lying in free space. **DGNN** and **PSS** have the same problem: whilst having a high recall, denoting their capacity to recover most of the existing pieces of surface (and very accurately, given the very low mean distance metric), they connect too many regions of space with triangles lying in empty space, thus affecting their precision.

The accordance between all of these visual observations and the corresponding quantitative results given by our metrics prove their relevance.

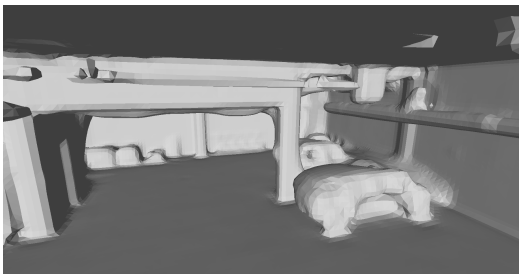
4.4.3 *ETH3D* dataset

MVS-based point clouds are known to be noisy and contain quite a lot of outliers. This seems to have an effect on the performances of the different methods. The ones performing best on *ENSG* and *STRAS* seem to struggle more, and surprisingly, **Poisson** 8 [11]

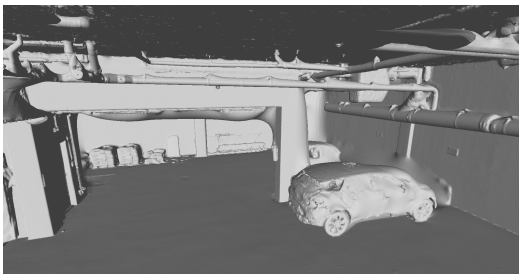


(a) **RESR**

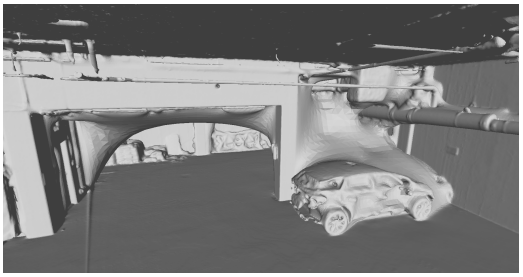
(b) **Poisson** 11/2



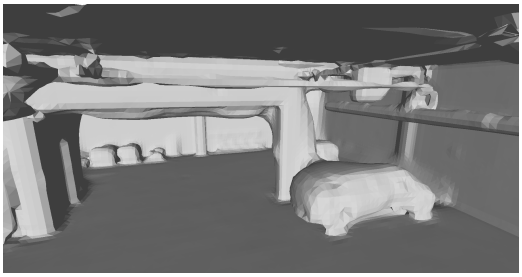
(c) **Poisson** 8/2



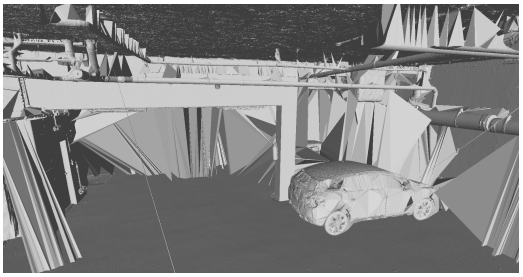
(d) **SSD** 12/3



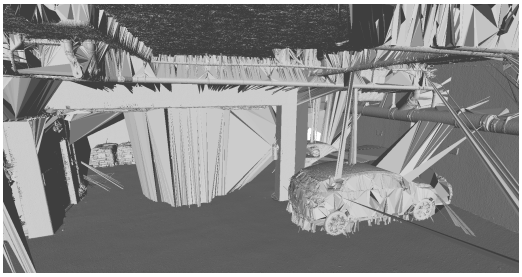
(e) **A. Metashape**



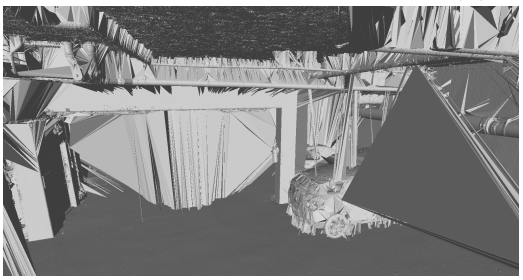
(f) **SSD** 8/2



(g) **DGNN**



(h) **PSS** 0.6



(i) **PSS** 0.1

Figure 4.12: Reconstructed meshes from the **ENSG** Parking Lot scene.

ETH3D ($d_{max} = 20\,cm$)				
Method	mean distance (cm)	precision (%)	recall(%)	F-score(%)
DMS	**2.04**	**95.33**	93.72	**94.47**
Poisson 11/2	2.56	93.91	94.04	93.96
RESR	2.62	94.12	92.65	93.29
A. Metashape	2.57	95.09	91.18	93.00
WSS	2.66	91.31	93.79	92.51
SSD 12/3	2.66	90.65	94.44	92.48
SSD 8/2	4.21	93.38	89.03	91.13
Poisson 8/2	3.99	94.66	86.94	90.59
DGNN	2.61	79.47	93.63	85.95
PSS 0.6	2.52	67.41	94.63	78.52
PSS 0.1	2.56	53.79	**95.16**	66.99
mean methods	2.82	86.28	92.66	88.44

Table 4.5: Average numerical results on the three scenes from *ETH3D* dataset sorted by decreasing F-score for $d_{max} = 20\,cm$

achieves a fairly high precision on this dataset. We believe that this is because it is more capable of filtering out the noise with an 8-depth than with an 11-depth octree. That would explain why **SSD** 8/2 also has a better precision than **SSD** 12/3. However, their poorer recall indicates that more pieces of real surface have not been recovered.

The method that performs best, however, is **DMS**, which is not very surprising considering the fact that it is the best version of a real MVS pipeline, fed with images and not with an image-derived point cloud.

The relatively good performance of **A. Metashape** on *ETH3D* compared to the other datasets might suggest it copes pretty well with outliers. More generally, considering it is a licensed solution, we might have expected a better overall performance on at least one of our datasets.

4.4.4 General remarks

Overall, **RESR** is the method that performs best almost everywhere. **DGNN** shows that learning how to reconstruct large, complex and open scenes is indeed possible, but it faced generalisation problems since the metrics on *ENSG* and *ETH3D* datasets are significantly lower than those on *STRAS* (from which its training set was extracted). However, with **RESR** and **DGNN** being the only methods making use of sensor positions, we believe that this is an important reason behind their good results. Sensor positions give an important piece of information that neither the points themselves nor the associated normals provide.

Poisson generally performs structurally better than **SSD**. Small-scale differences are noticeable when changing the octree depth used by both these algorithms.

PSS often reconstructs meshes very close to the real surface but also connects pieces of surface in regions of space that should remain empty. We can assume that we failed to find the right parameter settings because it was definitely the hardest algorithm to tune, but this is the best performance we managed to get.

4.5 Conclusion

Surface reconstruction is hard to evaluate since it is impossible to directly compute the difference between the real surface and a reconstructed one. It has often been assessed visually because it seems fairly intuitive to know whether a piece of surface has been accurately recovered. However, human perception can be unfair and a purely visual evaluation lacks quantitative information. In this chapter, we proposed new metrics to assess surface reconstruction. We have leveraged the visual information obtained by combining the acquired points and the associated sensor positions in order to define what we believe are more relevant metrics than the ones currently used. They imitate the process of a human being looking at and comparing the two surfaces (the real one and the reconstructed one). This goal has been achieved since our survey validates behaviours that a human can interpret by just looking at the meshes. In Section 4.4 we drew parallels between the specific visual observations and the quantitative evidence provided by our metrics that confirms them.

Our metrics enable the assessment of the completeness of the reconstructions as well as their precision both *locally* and *globally*. One can thus analyse the results from different points of view. As a relevant outcome, our survey also confirms that sensor positions are very relevant when trying to separate occupied from empty space.

As well as all these advantages that make surface reconstruction evaluation more objective, the fact that we only use raw data acquired with basic sensors makes it easy to set up a new experiment. Having access to expensive data is not a requirement. We provide a tool to make surface reconstruction evaluation easier and wish to see it used widely.

General conclusion and perspectives

Conclusion

Two major challenges in surface reconstruction have been tackled in this PhD. We first presented an end-to-end distributed surface reconstruction pipeline in order to make it possible to scale up to arbitrarily-large point clouds whilst guaranteeing the watertightness of the model. Secondly, we proposed two protocols to assess the quality of any reconstruction with the aim to make assessments more rigorous and less biased than previous approaches.

Large-scale surface reconstruction

Improvements in sensor capabilities have made it possible to acquire massive point clouds representing large areas. The *LiDAR HD* project, led by the French Mapping Agency, aims to map all French territory at 10 points per square meter resolution. However, working with such massive data is impossible without re-thinking point cloud processing algorithms and storage. Memory footprint and computation time quickly become prohibitive when working on a standard machine, so we investigated the parallelisation of a surface reconstruction algorithm [23] and in particular the graph-cut optimisation step.

We generalised the work on the parallelisation of an imaged-based graph-cut optimisation [72], which uses the method of Lagrange multipliers [75], to the more complex case of a 3-dimensional structure dual graph. We proceeded by splitting the input point cloud into tiles and computing its distributed Delaunay triangulation using the algorithm and data structure presented in [71]. In order to label each tetrahedron as *empty* or *occupied*, we needed to minimise an energy function. While it has become standard to use graph-cut optimisation to solve this problem, we wanted to make it possible to split the computation on the nodes of a cluster. We split the Delaunay triangulation dual graph after duplicating the frontier nodes so that the energy function can be divided into as many sub-energies as there are tiles. However, in order to end up with a globally consistent surface, we needed frontier nodes to be labelled identically in all the sub-graphs that they belong to. We showed how to incorporate these constraints into the energy function itself and optimise the global energy function by iterating *local* graph-cut optimisations and Lagrange multipliers' updates. This leads to a watertight surface that is close to the

one that would have been produced by the non-distributed method.

Evaluation of surface reconstruction

Evaluating surface reconstruction is difficult because of the nature of the problem: the goal is to produce a digital model that represents as best as possible a real object or environment, of which discrete information has been acquired using a noisy sensor. For complex open scenes, it is nearly impossible to define the ground truth surface as it should be the real surface itself, so we cannot directly compute the difference between this real surface and the reconstructed one.

We have shown that one possible solution is to work with synthetic data that we consider as being as close to the real surface as possible. We thus considered such a mesh to be the perfect surface that an algorithm is supposed to reproduce, which solves the problem of defining a ground-truth that we can directly compare a reconstruction with. We next had to simulate the acquisition of a point cloud from this ground-truth mesh by a sensor. In order to produce a realistic input point cloud, the simulation includes adding noise to the acquired point positions. We could then run the algorithms that we wish to assess on this simulated data. To achieve this, we developed an aerial LiDAR simulator that replicates the production of a realistic point cloud representing an urban environment. We used it on a publicly available high-quality dataset [78]. However, even after solving this problem, measuring how different the reconstruction is from the ground truth is not straightforward. Point clouds are noisy and contain occlusions so all the features of the scene might not be present in the input point cloud, or they might be scarcely sampled. In this context, cannot expect an algorithm to recover such shapes. To deal with this question, we proposed processing both the ground truth and the reconstructed meshes. We filtered out triangles from both meshes that lie further than a given threshold from the input point cloud, thus defining the "best expected surface" and its reconstructed equivalent. We repeated this process for several threshold values and computed the distance between Poisson-Disk samplings of both these filtered meshes, thus resulting in two curves of distances. This protocol offers the possibility to analyse the precision and the recall of the reconstructions *locally* and *globally*. This proved to be insightful since we observed the algorithms we tested had different behaviours depending on the range at which we assessed them.

In order to avoid having to work with synthetic data, we showed that it was possible to set a rigorous protocol to evaluate surface reconstruction exclusively with real data. We only had to make sure that we have access to two different point clouds representing the same scene, with one of them being of significantly better quality than the other: it has to be denser, have fewer occlusions and less noise. In addition, we made extensive

use of sensor positions, which are easy-to-access pieces of information in our controlled setting. The combination of high-quality points and the positions from which these points have been acquired allow us to define rays along which space is supposed to be *free*. Each ray also provides us with a direction along which space is supposed to become *occupied*, close to the corresponding high-quality point. We therefore expect to find a piece of surface in this region. We proposed computing the intersections between these rays and the reconstructed model, and analysing the ones situated in *observed space*. These intersections that denote pieces of reconstructed surface can be classified as *True Positives* or *False Positives* depending on how close they are to the corresponding high-quality point and how compatible they are with the visibility information provided by the laser ray. We defined *False Negatives* as pieces of surface that were expected but were not recovered, and these entities allow us to calculate *precision* and *recall* ratios. We combined these two complementary metrics with the distance from each high-quality point to the nearest intersection, which provides information on the *local* precision of the reconstruction. We showed that these metrics are relevant when trying to assess surface reconstruction since they allow validating and quantifying human visual assessment.

Perspectives

The end-to-end distributed surface reconstruction pipeline that we presented in Chapter 2 is an improvement on previous Delaunay-based algorithms. We have nevertheless identified a number of perspectives for our work. In particular, there is no theoretical guarantee that the algorithm converges to a unique labelling over all tiles. In other words, there might be residual disagreements between the labelling of a single tetrahedron in the different tiles that it belongs to. To guarantee watertightness, we decided that each of these inconsistently labelled tetrahedra would be labelled according to one particular tile, defined by our distributed Delaunay triangulation structure [71]. In the case where a tetrahedron is split across an odd number of tiles, we could probably change that decision making process to a majority vote between the tiles. However, a shared tetrahedron is much more likely to belong to either two or four tiles, making this source of improvement quite limited. Another possibility to reduce the number of disagreements would be to duplicate more tetrahedra than just the ones defined by the distributed Delaunay triangulation structure [71]. This would add more *contextual information* to the decision making process, helping to guess the right label of each tetrahedron. The drawback of this approach is the induced increase in the memory footprint and the extra computations carried out for each tile involved.

An ambitious avenue would be to seek a more efficient way to update the Lagrange multipliers. The algorithm, in its current form, needs quite a few iterations to converge and we might be able to save computation time by finding a more efficient strategy.

The contributions to the evaluation of surface reconstruction that we presented in Chapter 3 and 4 could also be strengthened. Our synthetic data evaluation pipeline has been tested on a mesh that represents a real scene (the city of Strasbourg in France). We can theoretically use it on any mesh representing a fictitious environment. An evident continuation of this work would therefore be to use purely handmade synthetic data in order to diversify the type of environment on which we evaluate algorithms.

We could even turn it into an open benchmark by extending to more complex scanning geometries but also to other scanning devices, in particular terrestrial, drone, or satellite platforms. That would also require incorporating the new metrics that we defined on real data, which would result in a multi-scene, multi-data and multi-metric benchmark.

A final perspective that we have identified concerns *learning-based* algorithms. Such methods are gaining in popularity despite still facing problems when trying to reconstruct large and complex scenes. Our work could provide a way to produce more training data and learn more efficiently thanks to our metrics.

Résumé de la thèse en français

Passage à l'échelle et évaluation de la reconstruction de surface à partir de nuages de points de scènes ouvertes

Introduction

La reconstruction de surface à partir de nuages de points consiste à trouver le modèle continu représentant au mieux une surface réelle dont seule une information ponctuelle est disponible. Ce nuage de points peut avoir été produit à partir d'images de la scène grâce aux techniques de Stéréoscopie multi-vues (*Multi-view stereo*) [5] ou de Structure acquise à partir d'un mouvement (*Structure from motion*) [6]. Mais nous nous intéresserons surtout au cas où les nuages de points ont été acquis grâce à un dispositif laser de type LiDAR aérien ou terrestre permettant de scanner des zones étendues. D'autres appareils de plus courte portée sont basés sur le temps de vol (*time-of-flight*) [2] ou la lumière structurée (*structured-light*) [3].

De nombreuses contributions ont été proposées pour résoudre ce problème. *Poisson* [11] est une méthode qui consiste à calculer une fonction indicatrice (valant 1 à l'intérieur de l'objet et 0 en dehors) en se basant sur les positions des points du nuage ainsi que sur les normales qui leur sont associées. Un maillage triangulé correspondant à une isovalue est ensuite extrait. [23] calcule la triangulation de Delaunay du nuage de points et utilise les positions à partir desquelles chaque point a été acquis afin d'attribuer à chaque tétrahèdre un statut (intérieur ou extérieur). La surface est alors définie comme l'ensemble des triangles à l'interface entre un tétraèdre intérieur et un tétraèdre extérieur. Plus précisément, la segmentation des tétraèdres provient de l'optimisation par coupe de graphe d'une fonction énergie composée de deux termes :

- un **terme de données** qui favorise des statuts (intérieur ou extérieur) en accord avec les hypothèses faites grâce aux données (points et positions du capteur)

- un **terme de régularisation** qui minimise le nombre de couples de tétraèdres adjacents ayant des statuts différents.

Plus récemment, des algorithmes basés sur l'apprentissage machine ont été développés pour estimer cette segmentation intérieur-extérieur : *IM-NET* [16], *Occupancy Networks* [18], *Convolutional Occupancy Networks* [19].

Les performances des appareils d'acquisition de données cités plus haut ont beaucoup augmenté au cours des dernières années, rendant possible la production massive de données. Les ordinateurs sont quant à eux limités par leur mémoire vive et leur puissance de calcul. Il est par conséquent nécessaire de repenser les algorithmes de traitement de données et certains travaux permettent de répondre à ce besoin. De nouveaux outils et paradigmes ont vu le jour notamment le *streaming* [45] mais aussi des procédures standardisées comme le *Message Passing Interface* (MPI) [49, 50] ou encore *Spark* [53], un cadre de programmation en Scala plus adapté que le paradigme *MapReduce* [52] pour les traitements itératifs et interactifs. D'autre part, des algorithmes de reconstruction de surface dédiés au cas particulier des nuages de points large échelle ont vu le jour : l'algorithme *ball-pivoting* [44] datant de 1999 fait figure de pionnier en termes de méthode *out-of-core*. La méthode proposée dans [58], basée sur la triangulation de Delaunay, consiste à découper le nuage de points en tuiles et calculer les bouts de surface correspondant aux différentes tuiles en parallèle. Le problème d'une telle approche est que la surface globale résultante n'est pas forcément étanche, propriété importante des surfaces réelles.

Alors que de nombreux algorithmes utilisant un panel d'approches variées ont été développés pour reconstruire un maillage triangulé à partir d'un nuage de points, peu de méthodes d'évaluation sont aujourd'hui disponibles. L'évaluation permet pourtant de déterminer quelle surface est la plus fidèle par rapport à la réalité et idéalement, de quantifier les écarts entre les différentes méthodes. De nombreux auteurs ont simplement évalué la qualité des modèles produits *visuellement*, ce qui pose évidemment des problèmes d'honnêteté dus aux conflits d'intérêt et de manque de quantification. À condition de disposer d'un maillage que l'on considère idéal, *Metro* [1] permet de calculer la distance entre un échantillonnage de la surface reconstruite et le maillage vérité terrain (et vice versa).

Passage à l'échelle

L'une des contributions de cette thèse est l'adaptation d'un algorithme de reconstruction de surface [23] au cas large échelle. En particulier, nous présentons comment paralléliser l'étape d'optimisation de la fonction énergie (équation 4.4).

Algorithme de référence

$$\hat{\mathbf{x}} = \underset{\mathbf{x}\in\{0,1\}^{|\mathcal{T}|}}{\arg\min} \Big(E_{data}(\mathbf{x}) \; + \; \alpha \; E_{prior}(\mathbf{x}) \Big) \tag{4.4}$$

La méthode consiste à attribuer à chaque tétraèdre t de la triangulation de Delaunay $\mathcal{T}$ du nuage de points le label 0 si celui-ci est vide et 1 s'il est plein. Pour cela, on minimise la fonction énergie (4.4) par coupe de graphe. On notera $G = (V, E)$ le graphe dual dont les sommets $V = \{s, t\} \cup \mathcal{T}$ sont la source s, le puits t et les tétraèdres de $\mathcal{T}$. Les arrêtes E sont les triangles à l'interface entre toutes les paires de tétraèdres adjacents mais aussi celles qui connectent chaque tétraèdre à s et t. Chaque arrête est pondérée de manière à ce que le coût d'une coupe s-t dans le graphe dual ait la même valeur que la fonction énergie (4.4) après avoir attribué les valeurs des labels $\mathbf{x}$. Pour minimiser cette énergie, nous devons donc minimiser la somme des poids des arrêtes coupées (équation 4.5).

$$\sum_{(i,j)\in E} c_{i,j} x_{i,j} \tag{4.5}$$

où $c_{i,j}$ est le poids de l'arrête (i, j) et $x_{i,j}$ est un Booléen indiquant si l'arrête est coupée ($1 =$ coupée, $0 =$ non coupée). De plus, x_i (le label du tétraèdre i) indique si le sommet i est relié à $s\,(x_i = 0)$ ou à $t\,(x_i = 1)$ après une coupe. On a alors : $x_s = 0$, $x_t = 1$ et $\forall (i, j) \in E$, $x_{i,j} = |x_i - x_j|$.

Algorithme distribué

Pour paralléliser l'étape d'optimisation de la fonction énergie, nous proposons de découper le nuage de points d'entrée $\mathcal{P}$ en tuiles. Nous noterons $\mathcal{P}_K = (\mathcal{P}_k)_{k\in K}$ sa décomposition en $|K|$ sous-ensembles disjoints $\mathcal{P}_k$, où K représente l'ensemble des indices des tuiles.

La triangulation de Delaunay $\mathcal{T}$ du nuage de points peut donc être décomposée elle-aussi en $|K|$ sous-graphes $G_k = (V_k, E_k)$ en utilisant l'algorithme et la structure de [71]. Certains tétraèdres seront *partagés* par deux tuiles donc le nœud qui les représente apparaîtra dans deux sous-graphes. Le problème principal est alors de garantir que les tétraèdres à l'interface entre deux tuiles obtiendront le même label dans les deux sous-graphes. Pour tout $k \in |K|$, on dénote respectivement par $x^k_{i,j}$ et x^k_i les valeurs de $x_{i,j}$ et x_i dans le sous-graphe k. En outre, on définit $c^k_{i,j} = \frac{c_{i,j}}{|\{k:(i,j)\in E_k\}|}$: le poids de l'arrête (i, j) normalisé par le nombre de tuiles auxquelles cette arrête appartient. Avec cette décomposition, la fonction optimisée devient :

$$f(\mathbf{x}) = \sum_{(i,j)\in E} c_{i,j} x_{i,j} = \sum_{k\in K} \sum_{(i,j)\in E_k} c^k_{i,j} x^k_{i,j} \tag{4.6}$$

avec la condition que les labels x^k_i doivent avoir la même valeur dans tous les sous-graphes

dans lesquels ils appartiennent :

$$\forall\, k, l \in K^2, \quad \forall\, i \in V_k \cap V_l, \quad x_i^k - x_i^l = 0 \tag{4.7}$$

Pour transformer ce problème en sous-problèmes en utilisant notre structure de graphe tuilé, on optimise par coupe de graphe chaque sous-graphe G_k par rapport aux variables $x_{i,j}^k$. Pour garantir la condition exprimée par l'équation 4.7, on ajoute un terme de pénalité à l'énergie en utilisant la dualité Lagrangienne :

$$\begin{aligned} L(\mathbf{x}, \boldsymbol{\lambda}) = & \sum_{k \in K} \sum_{(i,j) \in E_k} c_{i,j}^k x_{i,j}^k \\ & + \sum_{k \in K} \sum_{l > k} \sum_{i \in V_k \cap V_l} \lambda_i^{k,l} (x_i^k - x_i^l) \end{aligned} \tag{4.8}$$

avec $\boldsymbol{\lambda} = \{\lambda_i^{k,l} : (k,l) \in K^2,\ i \in V_k \cap V_l\}$. En définissant, pour chaque index de tuile $k \in K$:

$$\begin{aligned} L_k(\mathbf{x}^k, \boldsymbol{\lambda}) = & \sum_{(i,j) \in E_k} c_{i,j}^k x_{i,j}^k \\ & + \sum_{l \neq k} \sum_{i \in V_k \cap V_l} \left\{ \begin{matrix} 1 & \text{if } l > k \\ -1 & \text{otherwise} \end{matrix} \right\} \lambda_i^{k,l} x_i^k \end{aligned} \tag{4.9}$$

On a alors :

$$L(\mathbf{x}, \boldsymbol{\lambda}) = \sum_{k \in K} L_k(\mathbf{x}^k, \boldsymbol{\lambda}) \tag{4.10}$$

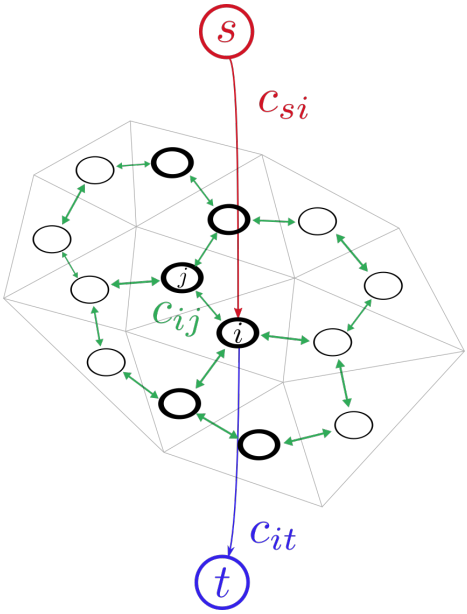
La Figure 4.13 permet de visualiser la mise à jour des poids du graphe dual sur un cas simple avec une seule séparation. On définit la fonction duale de Lagrange :

$$g(\boldsymbol{\lambda}) = \min_{\mathbf{x}} L(\mathbf{x}, \boldsymbol{\lambda}) \tag{4.11}$$

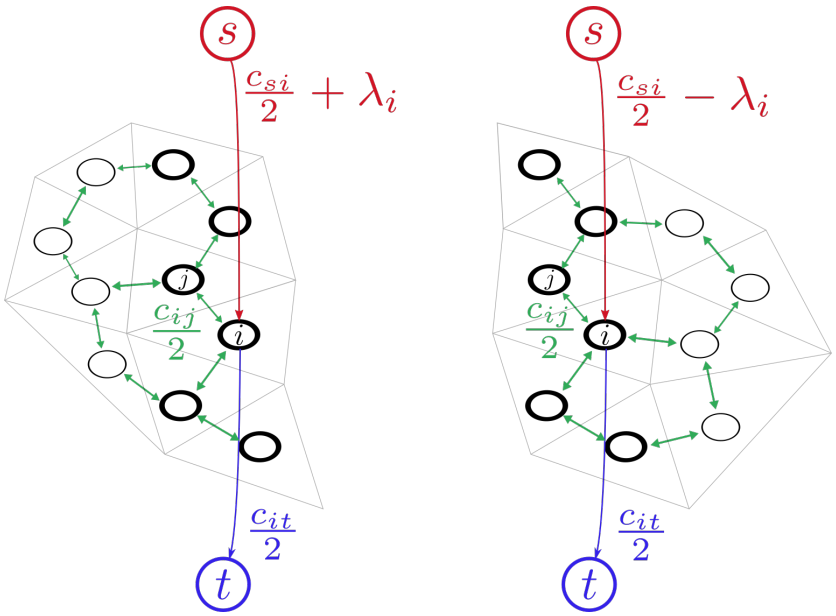
Comme la fonction duale de Lagrange $g(\boldsymbol{\lambda})$ (équation 4.11) est concave [75], on peut trouver une solution optimale au problème (4.6+4.7) en maximisant $g(\boldsymbol{\lambda})$ grâce à une méthode ascendante. On itère alors les phases de :

- Résolution des sous-problèmes d'optimisation par coupe de graphe $L_k(\mathbf{x}^k, \boldsymbol{\lambda})$ for $\mathbf{x}^k$.
- Mise à jour de $\boldsymbol{\lambda}$ pour aller dans la direction de plus forte montée.

Nous montrons expérimentalement que l'implémentation de cet algorithme avec *Spark* [53] et son exécution sur un cluster d'ordinateurs permet de diminuer le temps de calcul (pas de manière linéaire cependant) en augmentant le nombre de noeuds. Cet algorithme garantit l'étanchéité du modèle grâce à la segmentation unique des tétraèdres de la triangulation. Enfin, le modèle produit par la méthode distribuée converge vers un modèle proche de celui produit par l'algorithme de référence.



(a) Graphe dual avant séparation



(b) Sous-graphes duaux après séparation

Figure 4.13: Mise à jour des poids du graphe dual après une séparation. On analyse le noeud i en particulier. Les noeuds en gras sont ceux qui ont été dupliqués au cours de la séparation. s et t représentent la *source* et le *puits*.

Évaluation

Évaluer la reconstruction de surface est un problème difficile par nature étant donné que la seule vérité terrain est la surface réelle elle-même. Comme l'objectif est de produire un modèle numérique représentant cette surface, il est impossible de calculer directement l'"écart" entre les deux. Deux alternatives ont alors été envisagées.

Utilisation de données synthétiques

La ville et eurométropole de Strasbourg a financé la production d'un maillage de son territoire [78] et nous avons utilisé ce jeu de données pour définir des métriques d'évaluation propres à ce cas de figure.

Notre protocole repose premièrement sur la disponibilité d'une surface vérité terrain $\mathcal{M}_{GT}$. Deuxièmement, il faut un nuage de points $\mathcal{P}$ représentant une numérisation de $\mathcal{M}_{GT}$ réalisée à l'aide d'un simulateur de LiDAR comme celui présenté dans cette thèse. Enfin, nous avons besoin d'un maillage triangulé $\mathcal{M}_E$ reconstruit grâce à une méthode que l'on souhaite évaluer.

En pratique, seule une partie limitée de la vérité terrain est couverte par l'acquisition. Si nous pouvons attendre d'un algorithme qu'il remplisse les trous entre les points, on ne peut cependant pas espérer qu'il devine la forme de la surface loin de tout point d'acquisition. On définit alors la partie "reconstructible" $\mathcal{M}_{GT}^{\alpha}$ de $\mathcal{M}_{GT}$ au seuil de α. On peut produire $\mathcal{M}_{GT}^{\alpha}$ en supprimant tous les triangles de $\mathcal{M}_{GT}$ pour lesquels les trois sommets se trouvent à une distance supérieure à α d'un point de $\mathcal{P}$. $\mathcal{M}_{GT}^{\alpha}$ est donc la meilleure surface que l'on peut attendre d'un algorithme, au seuil de α. Toute partie *reconstruite* à une distance supérieure à α d'un point de $\mathcal{P}$ sera également volontairement ignorée pour ne pas que les hypothèses faites par les algorithmes impactent l'évaluation. On définira donc $\mathcal{M}_{E}^{\alpha}$ comme la partie pertinente à évaluer de $\mathcal{M}_E$. La Figure 4.14 permet de visualiser le calcul de $\mathcal{M}_{GT}^{\alpha}$ et $\mathcal{M}_{E}^{\alpha}$ pour un α donné.

Nous proposons d'évaluer la qualité de la reconstruction $\mathcal{M}_E$ en calculant la distance moyenne entre des échantillonnages de type *Poisson-disk* de $\mathcal{M}_{E}^{\alpha}$ et $\mathcal{M}_{GT}^{\alpha}$. En notant $\mathcal{P}^{R}_{\mathcal{M}_{GT}^{\alpha}}$ and $\mathcal{P}^{R}_{\mathcal{M}_{E}^{\alpha}}$ les échantillonnages de type *Poisson-Disk* de rayon R correspondant à $\mathcal{M}_{GT}^{\alpha}$ and $\mathcal{M}_{E}^{\alpha}$, et grâce à la définition de la distance point-vers-maillage (équation 4.12), on calcule alors deux métriques (équations 4.13 and 4.14).

$$\forall p \in \mathbb{R}^3,\ d\left(p, \mathcal{M}\right) = \min_{q \in \mathcal{M}} d(p, q) \tag{4.12}$$

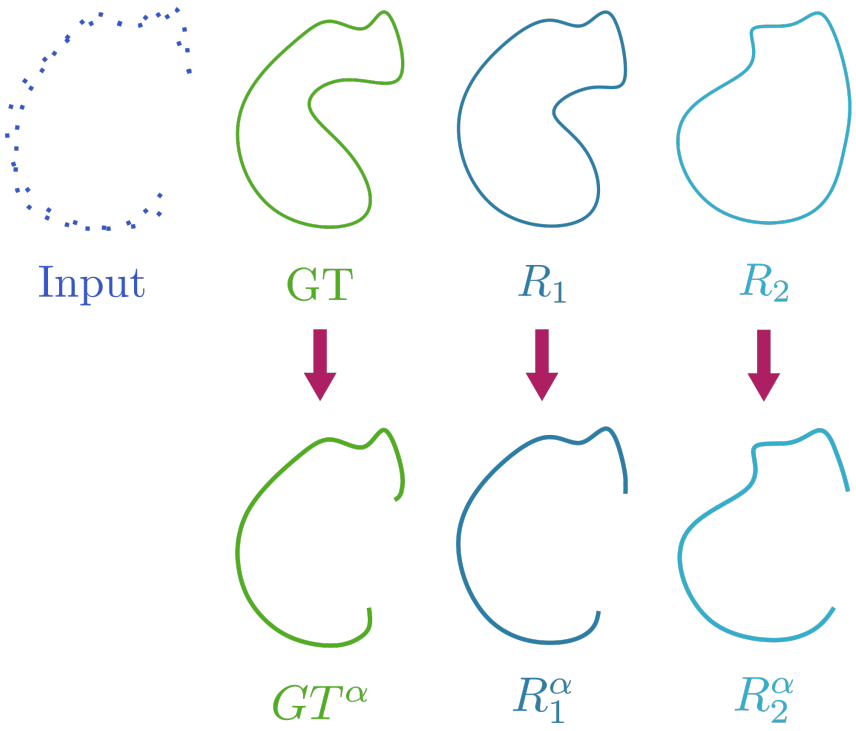


Figure 4.14: Visualisation du calcul de la partie "reconstructible" des différents maillages pour un α proche de la valeur de l'espacement moyen entre deux points du nuage. À partir du nuage de points, du maillage vérité terrain et de deux maillages reconstruits R_1 et R_2, on calcule la partie reconstructible de la vérité terrain, et les parties pertinentes à évaluer R_1^α et R_2^α en enlevant les triangles éloignés de plus de α d'un point du nuage de points.

$$\text{Précision moyenne} : \frac{1}{|\mathcal{P}^R_{\mathcal{M}_E^\alpha}|} \sum_{p \in \mathcal{P}^R_{\mathcal{M}_E^\alpha}} d\left(p, \mathcal{M}_{GT}^\alpha\right) \tag{4.13}$$

$$\text{Rappel moyen} : \frac{1}{|\mathcal{P}^R_{\mathcal{M}^\alpha}|} \sum_{p \in \mathcal{P}^R_{\mathcal{M}_{GT}^\alpha}} d\left(p, \mathcal{M}_E^\alpha\right) \tag{4.14}$$

La précision décrit à quel point la surface reconstruite est en accord avec la vérité terrain. Le rappel donne une information d'exhaustivité de la reconstruction c'est-à-dire qu'il permet de savoir à quel point les morceaux de surface de la vérité terrain sont également présents dans la reconstruction.

En calculant ces deux grandeurs pour différentes valeurs de α, on obtient deux *courbes* qui permettent d'analyser les performances des algorithmes *localement* (pour les faibles valeurs de α) et plus *globalement* (pour les hautes valeurs de α). Les différents comportements observés pour les différentes valeurs de α valident la pertinence de ce raisonnement.

Utilisation de données réelles

Une continuation assez naturelle du travail réalisé sur l'évaluation de la reconstruction de surface était de mettre en place un protocole remplissant le même objectif mais sans avoir

recours à des données synthétiques. On appelle "*données réelles*" des données acquises dans le monde réel avec de vrais capteurs. Ceci inclut les images RGB-D, les acquisitions LiDAR, etc. Nous allons évaluer la reconstruction de scènes réelles à partir de données réelles de qualité significativement meilleure que celles qui ont servi à la reconstruction. Même si cette idée est assez classique, la contribution majeure de ce travail réside dans la manière dont nous proposons de calculer la différence entre la surface reconstruite et les données réelles de haute qualité en relevant les incompatibilités et les morceaux de surface manquants. De plus, pour la première fois, nous proposons d'utiliser les positions du capteur pour l'évaluation. Associées aux points de haute qualité, elles permettent de définir un rayon le long duquel l'espace est censé être vide. Tout morceau de surface se trouvant sur le chemin du rayon laser sera donc pénalisé car incompatible avec cette information. Un morceau de surface sera considéré manquant si aucune intersection n'est trouvée entre le rayon laser et la surface reconstruite ou si une telle intersection est trouvée à une distance supérieure à un certain seuil du point de haute qualité correspondant, et située *après*, le long du rayon. Cela nous permet de classer les intersections trouvées dans des zones *observées*. Les intersections correctes correspondent à des *Vrais positifs*, celles incorrectes sont des *Faux positifs* et les morceaux de surface manquants sont des *Faux négatifs*. On peut alors calculer la *précision* et le *rappel* et combiner ces deux métriques à la *distance moyenne* des intersections correctes aux points de haute qualité. La Figure 4.15 permet de visualiser ces définitions sur un exemple simple.

Ce protocole d'évaluation, basé sur l'information de visibilité venant des positions du capteur, a été testé sur trois jeux de données différents, contenant chacun trois scènes. Les métriques permettent de confirmer des intuitions visuelles, participant ainsi à valider la pertinence de ce mode d'évaluation.

Conclusion

Cette thèse de doctorat contribue à deux aspects fondamentaux de la reconstruction de surface. Premièrement, un algorithme distribué de bout en bout a été proposé afin de répondre à la demande de traitement des nuages de points large échelle. Il offre notamment la garantie de reconstruire une surface étanche. Deuxièmement, nous avons présenté deux protocoles d'évaluation permettant de quantifier la qualité d'une surface reconstruite. L'un nécessite d'avoir accès à un maillage vérité terrain alors que l'autre fait uniquement usage de données provenant d'un capteur.

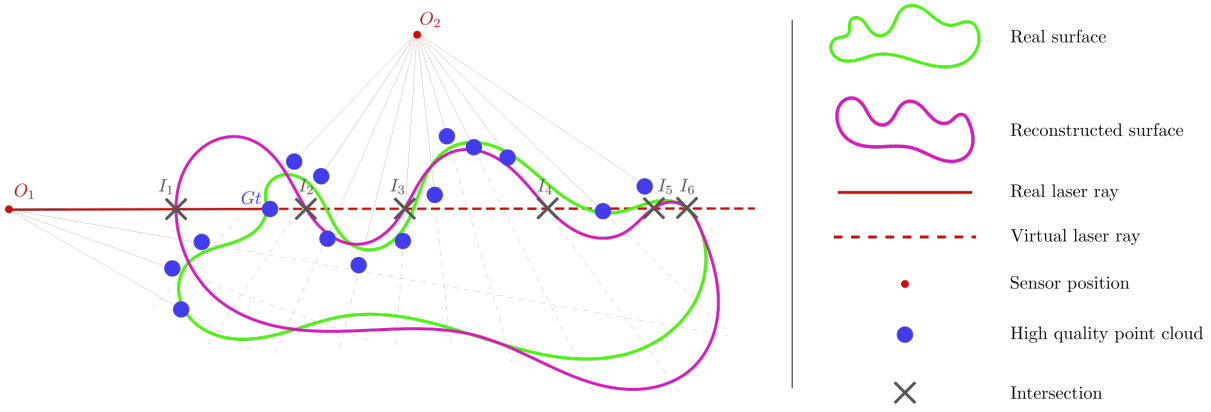


Figure 4.15: Exemple simple pour visualiser les définitions des métriques. La surface réelle a été scannée depuis deux positions : O_1 et O_2. Un rayon laser réel particulier (le plus épais) a été lancé depuis O_1 et il a intersecté la surface réelle au point Gt. Il est à noter que la position de l'intersection peut être bruitée, d'où le décalage entre la surface réelle et le nuage de points de haute qualité. On calcule toutes les intersections entre le rayon virtuel associé (i.e. l'extension du rayon laser réel) et la surface reconstruite. Dans le cas présent, on trouve six intersections $I_{1,\dots,6}$. L'intersection la plus proche de Gt se trouve être I_2 donc la métrique de "**distance rayon**" pour ce rayon particulier est la distance (Gt, I_2). Si $(Gt, I_2) < d_{max}$, Gt doit être compté comme un **Vrai Positif**, sinon il ne sera pas pris en compte dans cette évaluation puisqu'il se situe *après* Gt. Par ailleurs, on rencontre une intersection I_1 *sur le chemin vers* l'intersection la plus proche I_2, ce qui est compté comme un **Faux positif**.

Perspectives

L'algorithme de reconstruction de surface distribué de bout en bout constitue une avancée importante par rapport aux approches basées sur la triangulation de Delaunay. Plusieurs pistes d'améliorations sont toutefois envisageables. En particulier, il n'y a pas de garantie théorique sur la convergence de l'algorithme vers une segmentation unique des tétraèdres parmi toutes les tuiles. Pour garantir l'étanchéité du modèle en cas de désaccord, un tétraèdre se verra octroyer le label d'une tuile particulière. Dans le cas où il est partagé par un nombre impair de tuiles, il pourrait être judicieux de prendre cette décision par vote majoritaire entre les tuiles concernées. Une autre possibilité consiste à réduire le nombre de désaccords en dupliquant plus de tétraèdres que ceux définis par la structure de triangulation de Delaunay de [71]. Ceci ajouterait plus de contexte rendant la prise de décision plus fiable. En revanche, l'empreinte mémoire et le temps de calcul à chaque itération se verraient être augmentés à cause de ces tétraèdres dupliqués supplémentaires.

Dans le but d'améliorer la capacité de l'algorithme à converger, réduisant ainsi le temps de calcul, il serait judicieux de trouver une stratégie de mise à jour des multiplicateurs de Lagrange plus efficace.

Les contributions à l'évaluation de la reconstruction de surface peuvent elles aussi êtres approfondies. Le protocole d'évaluation basé sur les données synthétiques a été

testé sur un maillage représentant une vraie scène (la ville de Strasbourg, en France). Il est théoriquement possible de l'utiliser avec un maillage représentant une scène imaginaire. Nous pourrions alors envisager d'utiliser des scènes créées à la main afin de diversifier le type d'environnement sur lesquels nous évaluons les algorithmes.

Nous pourrions même en faire un "benchmark" ouvert en incluant des géométries d'acquisition plus complexes mais aussi en proposant d'autres dispositifs comme des LiDAR terrestres ou des drones. Cela serait aussi l'occasion d'incorporer les nouvelles métriques basées sur les données réelles que nous avons définies, permettant ainsi de créer une plateforme d'évaluation multi-scène, multi-données et multi-métriques.

Enfin, les algorithmes d'apprentissage machine connaissent une popularité grandissante même s'ils sont souvent limités à la reconstruction d'objets fermés simples. Dans ce contexte, ces travaux sur l'évaluation pourraient devenir un moyen de produire davantage de données d'entraînement et d'apprendre plus efficacement en utilisant les métriques présentées.

References

[1] P. Cignoni, C. Rocchini, R. Scopigno, Metro: measuring error on simplified surfaces, in: Computer graphics forum, Vol. 17, Wiley Online Library, 1998, pp. 167–174, issue: 2.

[2] R. Lange, P. Seitz, Solid-state time-of-flight range camera, IEEE Journal of Quantum Electronics 37 (3) (2001) 390–397. doi:10.1109/3.910448.

[3] J. Geng, Structured-light 3D surface imaging: a tutorial, Advances in Optics and Photonics 3 (2) (2011) 128–160. doi:10.1364/AOP.3.000128.

[4] B. Lohani, S. Ghosh, Airborne LiDAR technology: a review of data collection and processing systems, Proceedings of the National Academy of Sciences, India Section A: Physical Sciences 87 (4) (2017) 567–579, publisher: Springer.

[5] Y. Furukawa, C. Hernández, Multi-view stereo: A tutorial, Foundations and Trends® in Computer Graphics and Vision 9 (1-2) (2015) 1–148, publisher: Now Publishers Inc. Hanover, MA, USA.

[6] O. Ozyesil, V. Voroninski, R. Basri, A. Singer, A survey of structure from motion, arXiv preprint arXiv:1701.08493 (2017).

[7] P. Cignoni, C. Montani, C. Rocchini, R. Scopigno, External memory management and simplification of huge meshes, IEEE Transactions on Visualization and Computer Graphics 9 (4) (2003) 525–537. doi:10.1109/TVCG.2003.1260746.

[8] L. C. Kinsey, Surfaces, in: Topology of Surfaces, Springer New York, New York, NY, 1993, pp. 56–93. doi:10.1007/978-1-4612-0899-0_4.
URL https://doi.org/10.1007/978-1-4612-0899-0_4

[9] M. Berger, A. Tagliasacchi, L. M. Seversky, P. Alliez, G. Guennebaud, J. A. Levine, A. Sharf, C. T. Silva, A Survey of Surface Reconstruction from Point Clouds, Comput. Graph. Forum 36 (1) (2017) 301–329. doi:10.1111/cgf.12802.
URL https://doi.org/10.1111/cgf.12802

[10] A. Khatamian, H. Arabnia, Survey on 3D Surface Reconstruction, Journal of Information Processing Systems 12 (2016) 338–357. doi:10.3745/JIPS.01.0010.

[11] M. Kazhdan, M. Bolitho, H. Hoppe, Poisson surface reconstruction, in: Proceedings of the fourth Eurographics symposium on Geometry processing, Eurographics Association, 2006, pp. 61–70.

[12] W. E. Lorensen, H. E. Cline, Marching Cubes: A High Resolution 3D Surface Construction Algorithm, in: Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '87, ACM, New York, NY, USA, 1987, pp. 163–169. doi:10.1145/37401.37422.
URL http://doi.acm.org/10.1145/37401.37422

[13] M. Kazhdan, H. Hoppe, Screened poisson surface reconstruction, ACM Transactions on Graphics 32 (3) (2013) 1–13. doi:10.1145/2487228.2487237.

[14] M. Kazhdan, Reconstruction of solid models from oriented point sets, in: Proceedings of the third Eurographics symposium on Geometry processing, Eurographics Association, 2005, p. 73.

[15] S. Peng, C. M. Jiang, Y. Liao, M. Niemeyer, M. Pollefeys, A. Geiger, Shape As Points: A Differentiable Poisson Solver, in: Advances in Neural Information Processing Systems (NeurIPS), 2021.

[16] Z. Chen, H. Zhang, Learning implicit fields for generative shape modeling, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 5939–5948.

[17] C. R. Qi, H. Su, K. Mo, L. J. Guibas, Pointnet: Deep learning on point sets for 3d classification and segmentation, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 652–660.

[18] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, A. Geiger, Occupancy networks: Learning 3d reconstruction in function space, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 4460–4470.

[19] S. Peng, M. Niemeyer, L. Mescheder, M. Pollefeys, A. Geiger, Convolutional occupancy networks, in: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16, Springer, 2020, pp. 523–540.

[20] T. Groueix, M. Fisher, V. G. Kim, B. C. Russell, M. Aubry, A Papier-Mâché Approach to Learning 3D Surface Generation, in: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018.

[21] P. Labatut, J.-P. Pons, R. Keriven, Robust and efficient surface reconstruction from range data, in: Computer graphics forum, Vol. 28, Wiley Online Library, 2009, pp. 2275–2290, issue: 8.

[22] F. Lafarge, P. Alliez, Surface reconstruction through point set structuring, in: Proc. of Eurographics, Girona, Spain, 2013.

[23] L. Caraffa, M. Brédif, B. Vallet, 3D Watertight Mesh Generation with Uncertainties from Ubiquitous Data, in: S.-H. Lai, V. Lepetit, K. Nishino, Y. Sato (Eds.), Computer Vision – ACCV 2016, Lecture Notes in Computer Science, Springer International Publishing, 2016, pp. 377–391.

[24] R. Kolluri, J. R. Shewchuk, J. F. O'Brien, Spectral Surface Reconstruction from Noisy Point Clouds, in: Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing, SGP '04, ACM, New York, NY, USA, 2004, pp. 11–21, event-place: Nice, France. doi:10.1145/1057432.1057434.
URL http://doi.acm.org/10.1145/1057432.1057434

[25] C. Holenstein, R. Zlot, M. Bosse, Watertight surface reconstruction of caves from 3D laser data, in: 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2011, pp. 3830–3837.

[26] R. Sulzer, L. Landrieu, R. Marlet, B. Vallet, Scalable Surface Reconstruction with Delaunay-Graph Neural Networks, in: Computer Graphics Forum, Vol. 40, Wiley Online Library, 2021, pp. 157–167, issue: 5.

[27] Y. Ohtake, A. Belyaev, M. Alexa, G. Turk, H.-P. Seidel, Multi-level partition of unity implicits, ACM Transactions on Graphics 22 (3) (2003) 463–470. doi:10.1145/882262.882293.

[28] F. Calakli, G. Taubin, SSD: Smooth signed distance surface reconstruction, in: Computer Graphics Forum, Vol. 30, Wiley Online Library, 2011, pp. 1993–2002, issue: 7.

[29] J. J. Park, P. Florence, J. Straub, R. Newcombe, S. Lovegrove, Deepsdf: Learning continuous signed distance functions for shape representation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 165–174.

[30] A. Hornung, L. Kobbelt, Robust reconstruction of watertight 3 d models from non-uniformly sampled point clouds without normal information, in: Symposium on geometry processing, Citeseer, 2006, pp. 41–50.

[31] L. Nan, P. Wonka, Polyfit: Polygonal surface reconstruction from point clouds, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 2353–2361.

[32] R. Schnabel, R. Wahl, R. Klein, Efficient RANSAC for point-cloud shape detection, in: Computer graphics forum, Vol. 26, Wiley Online Library, 2007, pp. 214–226, issue: 2.

[33] P. Lancaster, Moving Weighted Least-Squares Methods, in: B. N. Sahney (Ed.), Polynomial and Spline Approximation: Theory and Applications, NATO Advanced Study Institutes Series, Springer Netherlands, Dordrecht, 1979, pp. 103–120. doi:10.1007/978-94-009-9443-0_7.
URL https://doi.org/10.1007/978-94-009-9443-0_7

[34] D. Shepard, A two-dimensional interpolation function for irregularly-spaced data, in: Proceedings of the 1968 23rd ACM national conference, ACM '68, Association for Computing Machinery, New York, NY, USA, 1968, pp. 517–524. doi:10.1145/800186.810616.
URL https://doi.org/10.1145/800186.810616

[35] Z.-Q. Cheng, Y.-Z. Wang, B. Li, K. Xu, G. Dang, S.-Y. Jin, A survey of methods for moving least squares surfaces, in: Proceedings of the Fifth Eurographics / IEEE VGTC conference on Point-Based Graphics, SPBG'08, Eurographics Association, Los Angeles, CA, 2008, pp. 9–23.

[36] D. Levin, The Approximation Power Of Moving Least-Squares, Mathematics of Computation 67 (Feb. 2000). doi:10.1090/S0025-5718-98-00974-0.

[37] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, C. T. Silva, Point set surfaces | Proceedings of the conference on Visualization '01 (2001).
URL https://dl.acm.org/doi/abs/10.5555/601671.601673

[38] D. Levin, Mesh-Independent Surface Interpolation, Geometric Modeling for Scientific Visualization 3 (Jan. 2003). doi:10.1007/978-3-662-07443-5_3.

[39] N. Amenta, Y. Kil, Defining point-set surfaces, ACM Trans. Graph. 23 (2004) 264–270. doi:10.1145/1015706.1015713.

[40] V. Fiorin, P. Cignoni, R. Scopigno, Out-of-core MLS Reconstruction, in: Proceedings of the Ninth IASTED International Conference on Computer Graphics and Imaging, CGIM '07, ACTA Press, Anaheim, CA, USA, 2007, pp. 27–34, event-place: Innsbruck, Austria.
URL http://dl.acm.org/citation.cfm?id=1710707.1710713

[41] R. Allegre, R. Chaine, S. Akkouche, A Dynamic Surface Reconstruction Framework for Large Unstructured Point Sets., in: PBG@ SIGGRAPH, 2006, pp. 17–26.

[42] R. Chaine, A geometric convection approach of 3-D reconstruction, in: Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing, SGP '03, Eurographics Association, Aachen, Germany, 2003, pp. 218–229.

[43] H. Hoppe, T. Derose, T. Duchamp, J. Mcdonald, W. Stuetzle, Surface reconstruction from unorganized points. SIGGRAPH Computer Graphics, 26(2), 71-78, Computer Graphics (Proc. SIGGRAPH 86) 20 (Jan. 1996). doi:10.1145/142920.134011.

[44] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, G. Taubin, The ball-pivoting algorithm for surface reconstruction, IEEE Transactions on Visualization and Computer Graphics 5 (4) (1999) 349–359. doi:10.1109/2945.817351.

[45] R. Pajarola, Stream-processing points, in: VIS 05. IEEE Visualization, 2005., 2005, pp. 239–246, iSSN: null. doi:10.1109/VISUAL.2005.1532801.

[46] M. Bolitho, M. Kazhdan, R. Burns, H. Hoppe, Multilevel Streaming for Out-of-core Surface Reconstruction, in: Proceedings of the Fifth Eurographics Symposium on Geometry Processing, SGP '07, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 2007, pp. 69–78, event-place: Barcelona, Spain.
URL http://dl.acm.org/citation.cfm?id=1281991.1282001

[47] D. E. Knuth, The art of computer programming, Vol. 3, Addison-Wesley, 1997.

[48] J. S. Vitter, External memory algorithms and data structures: dealing with massive data (Jun. 2001).
URL https://doi.org/10.1145/384192.384193

[49] P. Pacheco, Parallel programming with MPI, Morgan Kaufmann, 1997.

[50] W. Gropp, W. D. Gropp, E. Lusk, A. Skjellum, A. D. F. E. E. Lusk, Using MPI: portable parallel programming with the message-passing interface, Vol. 1, MIT press, 1999.

[51] M. Randrianarivony, G. Brunnett, Parallel Implementation of Surface Reconstruction From Noisy Samples (Sep. 2002).

[52] J. Dean, S. Ghemawat, MapReduce: simplified data processing on large clusters, Communications of the ACM 51 (1) (2008) 107. doi:10.1145/1327452.1327492.
URL http://portal.acm.org/citation.cfm?doid=1327452.1327492

[53] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, I. Stoica, Spark: Cluster Computing with Working Sets, in: Proceedings of the 2Nd USENIX Conference on Hot Topics in Cloud Computing, HotCloud'10, USENIX Association, Berkeley, CA, USA, 2010, pp. 10–10, event-place: Boston, MA.
URL http://dl.acm.org/citation.cfm?id=1863103.1863113

[54] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauly, M. J. Franklin, S. Shenker, I. Stoica, Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing, 2012, pp. 15–28.

URL https://www.usenix.org/conference/nsdi12/technical-sessions/presentation/zaharia

[55] M. Bolitho, M. Kazhdan, R. Burns, H. Hoppe, Parallel Poisson Surface Reconstruction, in: G. Bebis, R. Boyle, B. Parvin, D. Koracin, Y. Kuno, J. Wang, J.-X. Wang, J. Wang, R. Pajarola, P. Lindstrom, A. Hinkenjann, M. L. Encarnação, C. T. Silva, D. Coming (Eds.), Advances in Visual Computing, Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2009, pp. 678–689.

[56] K. Zhou, M. Gong, X. Huang, B. Guo, Highly Parallel Surface Reconstruction (Apr. 2008).
URL https://www.microsoft.com/en-us/research/publication/highly-parallel-surface-reconstruction/

[57] J. Manson, G. Petrova, S. Schaefer, Streaming Surface Reconstruction Using Wavelets, in: Proceedings of the Symposium on Geometry Processing, SGP '08, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 2008, pp. 1411–1420, event-place: Copenhagen, Denmark.
URL http://dl.acm.org/citation.cfm?id=1731309.1731324

[58] J. Han, S. Shen, Distributed surface reconstruction from point cloud for city-scale scenes, in: 2019 International Conference on 3D Vision (3DV), IEEE, 2019, pp. 338–347.

[59] G. Cuccuru, E. Gobbetti, F. Marton, R. Pajarola, R. Pintus, Fast low-memory streaming MLS reconstruction of point-sampled surfaces, in: Proceedings - Graphics Interface, 2009, pp. 15–22. doi:10.1145/1555880.1555893.

[60] Chemnitzer Linux Cluster (CLiC), archive Location: Worldwide Last Modified: 2020-06-02 Library Catalog: www.tu-chemnitz.de.
URL https://www.tu-chemnitz.de/urz/

[61] F. Williams, T. Schneider, C. Silva, D. Zorin, J. Bruna, D. Panozzo, Deep Geometric Prior for Surface Reconstruction, 2019, pp. 10130–10139.
URL http://openaccess.thecvf.com/content_CVPR_2019/html/Williams_Deep_Geometric_Prior_for_Surface_Reconstruction_CVPR_2019_paper.html

[62] F. Ter Haar, P. Cignoni, P. Min, R. Veltkamp, A comparison of systems and tools for 3D scanning, 3D Digital Imaging and Modeling: Applications of Heritage, Industry, Medicine and Land (2005).

[63] M. Berger, J. A. Levine, L. G. Nonato, G. Taubin, C. T. Silva, A benchmark for surface reconstruction, ACM Transactions on Graphics 32 (2) (2013) 1–17. doi:10.1145/2451236.2451246.

[64] J. Süßmuth, Q. Meyer, G. Greiner, Surface reconstruction based on hierarchical floating radial basis functions, in: Computer Graphics Forum, Vol. 29, Wiley Online Library, 2010, pp. 1854–1864, issue: 6.

[65] L. Winiwarter, A. M. E. Pena, H. Weiser, K. Anders, J. M. Sánchez, M. Searle, B. Höfle, Virtual laser scanning with HELIOS++: A novel take on ray tracing-based simulation of topographic full-waveform 3D laser scanning, Remote Sensing of Environment 269 (2022) 112772. doi:https://doi.org/10.1016/j.rse.2021.112772. URL https://www.sciencedirect.com/science/article/pii/S0034425721004922

[66] S. Manivasagam, S. Wang, K. Wong, W. Zeng, M. Sazanovich, S. Tan, B. Yang, W.-C. Ma, R. Urtasun, LiDARsim: Realistic LiDAR Simulation by Leveraging the Real World, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020.

[67] J. Lu, Z. Li, J. Bai, Q. Yu, Oriented and Directional Chamfer Distance Losses for 3D Object Reconstruction From a Single Image, IEEE Access 10 (2022) 61631–61638, publisher: IEEE.

[68] D. P. Huttenlocher, G. A. Klanderman, W. J. Rucklidge, Comparing images using the Hausdorff distance, IEEE Transactions on pattern analysis and machine intelligence 15 (9) (1993) 850–863, publisher: IEEE.

[69] K. Hildebrandt, K. Polthier, M. Wardetzky, On the convergence of metric and geometric properties of polyhedral surfaces, Geometriae Dedicata 123 (1) (2006) 89–112, publisher: Springer.

[70] Z. Huang, Y. Wen, Z. Wang, J. Ren, K. Jia, Surface Reconstruction from Point Clouds: A Survey and a Benchmark, arXiv preprint arXiv:2205.02413 (2022).

[71] L. Caraffa, P. Memari, M. Yirci, M. Brédif, Tile & Merge: Distributed Delaunay Triangulations for Cloud Computing, in: 2019 IEEE International Conference on Big Data (Big Data), IEEE, 2019, pp. 1613–1618.

[72] P. Strandmark, F. Kahl, Parallel and distributed graph cuts by dual decomposition, in: 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2010, pp. 2085–2092, iSSN: 1063-6919. doi:10.1109/CVPR.2010.5539886.

[73] J.-D. Boissonnat, M. Yvinec, Algorithmic geometry, Cambridge university press, 1998.

[74] P. M. Jensen, N. Jeppesen, A. B. Dahl, V. A. Dahl, Review of Serial and Parallel Min-Cut/Max-Flow Algorithms for Computer Vision, arXiv preprint arXiv:2202.00418 (2022).

[75] S. Boyd, S. P. Boyd, L. Vandenberghe, Convex optimization, Cambridge university press, 2004.

[76] K. Shvachko, H. Kuang, S. Radia, R. Chansler, The Hadoop Distributed File System, in: 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST), IEEE, Incline Village, NV, USA, 2010, pp. 1–10. doi:10.1109/MSST.2010.5496972.
URL http://ieeexplore.ieee.org/document/5496972/

[77] E. Boros, P. L. Hammer, Pseudo-Boolean optimization, Discrete Applied Mathematics 123 (1) (2002) 155–225. doi:https://doi.org/10.1016/S0166-218X(01)00341-9.
URL https://www.sciencedirect.com/science/article/pii/S0166218X01003419

[78] V. et eurometropole de Strasbourg, odata3d_pm3d dataset (2018).
URL https://3d.strasbourg.eu/

[79] A. Warcho\l, The concept of LiDAR data quality assessment in the context of BIM modeling, International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences (2019).

[80] F. Aguilar, J. Mills, Accuracy assessment of LiDAR-derived digital elevation models, The Photogrammetric Record 23 (2008) 148 – 169. doi:10.1111/j.1477-9730.2008.00476.x.

[81] F. J. Aguilar, J. P. Mills, J. Delgado, M. A. Aguilar, J. G. Negreiros, J. L. Pérez, Modelling vertical error in LiDAR-derived digital elevation models, ISPRS Journal of Photogrammetry and Remote Sensing 65 (1) (2010) 103 – 110. doi:https://doi.org/10.1016/j.isprsjprs.2009.09.003.

[82] C. Toth, E. Paska, D. Brzezinska, Quality assessment of lidar data by using pavement markings, in: Proceedings of the ASPRS Annual Conference, Portland, Oregon, USA, Vol. 28, 2008.

[83] G. Vosselman, others, Analysis of planimetric accuracy of airborne laser scanning surveys, International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences 37 (3a) (2008) 99–104.

[84] The CGAL Project, CGAL User and Reference Manual, 5th Edition, CGAL Editorial Board, 2020.
URL https://doc.cgal.org/5.2/Manual/packages.html

[85] P. Labatut, J.-P. Pons, R. Keriven, Efficient Multi-View Reconstruction of Large-Scale Scenes using Interest Points, Delaunay Triangulation and Graph Cuts, in: 2007

IEEE 11th International Conference on Computer Vision, 2007, pp. 1–8, iSSN: 2380-7504. doi:10.1109/ICCV.2007.4408892.

[86] W. Xiao, B. Vallet, M. Brédif, N. Paparoditis, Street environment change detection from mobile laser scanning point clouds, ISPRS Journal of Photogrammetry and Remote Sensing 107 (2015) 38–49, publisher: Elsevier.

[87] T. Schops, J. L. Schonberger, S. Galliani, T. Sattler, K. Schindler, M. Pollefeys, A. Geiger, A Multi-View Stereo Benchmark With High-Resolution Images and Multi-Camera Videos, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

[88] Y. Marchand, B. Vallet, L. Caraffa, Evaluating Surface Mesh Reconstruction of Open Scenes, ISPRS-International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences 43 (2021) 369–376.

[89] D. Cernea, OpenMVS: Multi-View Stereo Reconstruction Library (2020).
URL https://cdcseacave.github.io/openMVS

[90] M. Jancosek, T. Pajdla, Exploiting visibility information in surface reconstruction to preserve weakly supported surfaces, International scholarly research notices 2014, publisher: Hindawi (2014).

[91] H.-H. Vu, P. Labatut, J.-P. Pons, R. Keriven, High accuracy and visibility-consistent dense multiview stereo, IEEE transactions on pattern analysis and machine intelligence 34 (5) (2011) 889–901, publisher: IEEE.