EPITA–École Pour l'Informatique et les Techniques Avancées
CSI–Calcul Scientifique et Image
LRDE–Laboratoire de Recherche et Développement de l'EPITA

Improving point cloud support of *ContextCapture*^{TM} : Scan Finder, Point Cloud Visibility and Point Cloud Compression

Alexandre Gbaguidi Aïsse

Supervised by Cyril Novel

Submitted in part fulfilment of the requirements for the degree of
Software Engineer of EPITA, Paris, August 2018.

Abstract

Text of the Abstract.

Acknowledgements

I would like to express (whatever feelings I have) to:

- My supervisor
- My second supervisor
- Other researchers
- My family and friends

Dedication

Dedication here.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The need of 3D realistic models is increasingly present in several fields such as architecture, digital simulation or civil and structural engineering. One way to obtain a 3D model might be to build it by hand using specialized modeling softwares. In this case, the realistic aspect of the model could be doubtful. A more reliable way is to use *photogrammetry* or *surface reconstruction* or both at the same time. *ContextCapture*$^{\boldsymbol{TM}}$ is a reality modeling software that can produce highly detailed 3D models. It creates models of all types or scales from simple photographs of a scene to point clouds or both of them thanks to its hybrid processing. The usage of point clouds gained wide popularity because of the emergence of devices such as optical laser-based range scanners, structured light scanners, LiDAR scanners, Microsoft Kinect, etc. Actually, it is only in the past two years that *ContextCapture*$^{\boldsymbol{TM}}$ has started to support point clouds and the common goal between each part of this internship is to improve *ContextCapture*$^{\boldsymbol{TM}}$ point cloud support.

The general problem being solved by *surface reconstruction* is: given a point cloud P assuming to lie near an unknown shape S, construct a digital representation D approximating S. In order to reconstruct point clouds acquired from static[1] LiDAR scanners, possibly multi-scan[2], *ContextCapture*$^{\boldsymbol{TM}}$ needs to know the position of each scanner in the scene, on the one hand, and the attribution of each point to a scanner on the other. The scanners location information is not always present in point clouds metadatas, for instance LAS file format does not provide it. Moreover, some users of *ContextCapture*$^{\boldsymbol{TM}}$ sometimes lost this information due to a prior export with no metadata. Detecting the positions of multiple scanners exclusively from a point cloud is a subject not identified as inter-

[1]As opposed to mobile LiDAR scanners.
[2]Point cloud made of several laser scans.

esting by academics, and thus not tackled since this information is almost always available from the outset. The same holds true on the industry side, it is only recently that scanner position information is relevant for a few applications like *ContextCapture^{TM}* . This is the main contribution presented in this internship report: a method able to detect automatically multiple scanners in a single point cloud without prior knowledge of the scene.

Knowing scanners position is one step toward supporting LAS point cloud format. *ContextCapture^{TM}* still needs to know for each point which scanner sees it best[3]. This enters into the realm of visibility of point clouds. One way to retrieve visibility of point clouds is to reconstruct the surface and use the underlying mesh to compute visibility. But to reconstruct the surface we need to orient the normals; a chicken-and-egg problem. After some literature review on the subject, we did not find accurate method for LiDAR point clouds. Also, most papers try to find which points are visible from a precise viewpoint but in our case, as different scanners can see the same points, we want to know for each point which scanner best sees it. We introduce a custom point cloud visibility method that serves our purpose and works well with LiDAR point clouds, regardless of the sampling density.

ScanFinder and PointCloudVisibility are two contributions which serves mainly the same purpose: expand *ContextCapture^{TM}* input point cloud formats. Another improvement made in *ContextCapture^{TM}* is point cloud compression. *ContextCapture^{TM}* provides a cloud service which gives the opportunity for people not having any clusters or high-performance machine to do the job; reconstructing a large surface requires effective machines. The problem is that, point clouds can be very huge, up to one hundred (100) gibabyte and more. And if something happens while uploading, the upload restarts from scratch. Being able to divide by two point cloud sizes and then reduce uploading time is an interesting point for *ContextCapture^{TM}* cloud services. Point cloud compression can be adressed in two ways: geometric compression [GKIS05, SK06] or pure arithmetic compression regardless of the kind of file being compressed. We compared different compressor such as Brolti, LZMA (7Zip), Zip before integrating one of them into the product.

This report is organised as follows. Chapter 2 present Bentley Systems, Acute3D, *ContextCapture^{TM}* and how the achieved work is positioned in the company's business line. After introducing in Chapter 3 some useful definitions for a better understanding of the report, we describe the achieved work of Scan Finder, Point Cloud Visibility and Point Cloud Compression respectively in Chapter 4, Chapter 5 and Chapter 6. Note that in each chapter, we recall the context, the issue addressed and the expected

[3]There is no need to know exactly which scanner generated it.

result before going into details. Finally Chapter 7 summarizes all the work, evaluates my contributions to *ContextCapture*$^{\boldsymbol{TM}}$ and assess what this experience has brought to me.

Chapter 2

The company

This chapter shed more light on *ContextCapture*$^{\boldsymbol{TM}}$, the software I contributed to during these six (6) months internship as well as Bentley Systems, the company.

2.1 Bentley Systems

Bentley Systems is an American-based software development company founded by Keith A. Bentley and Barry J. Bentley in 1984.

For a bit of history[1], they introduced the commercial version of PseudoStation in 1985, which allowed users of Intergraph's VAX systems to use low-cost graphics terminals to view and modify the designs on their Intergraph IGDS (Interactive Graphics Design System) installations. Their first product was shown to potential users who were polled as to what they would be willing to pay for it. They averaged the answers, arriving at a price of $7,943. A DOS-based version of MicroStation was introduced in 1986. Later the two other brothers joined them in the business. Today, Bentley Systems is considered to have four (4) founders: Greg Bentley (CEO), Keith A. Bentley (EVP, CTO), Barry J. Bentley, Ph.D. (EVP) and Raymond B. Bentley (EVP).

At its core, Bentley Systems is a software development company that supports the professional needs of those responsible for creating and managing the world's infrastructure, including roadways, bridges, airports, skyscrapers, industrial and power plants as well as utility networks. Bentley delivers solutions for the entire lifecycle of the infrastructure asset, tailored to the needs of the various professions –

[1] Derived from [ben]

the engineers, architects, planners, contractors, fabricators, IT managers, operators and maintenance engineers – who will work on and work with that asset over its lifetime. Comprised of integrated applications and services built on an open platform, each solution is designed to ensure that information flows between workflow processes and project team members to enable interoperability and collaboration.

Bentley's commitment to their user community extends beyond delivering the most complete and integrated software – it pairs their products with exceptional service and support. Access to technical support teams 24/7, a global professional services organization and continuous learning opportunities through product training, online seminars and academic programs define their commitment to current and future generations of infrastructure professionals.

With their broad product range, strong global presence, and pronounced emphasis on their commitment to their neighbors, Bentley is much more than a software company – they are engaged functioning members of the global community. Their successes are determined by the skills, dedication, and involvement of extraordinary Bentley colleagues around the world.

Bentley has more than 3,500 colleagues in over 50 countries, and is on track to surpass an annual revenue run rate of $700 million. Since 2012, Bentley has invested more than $1 billion in research, development, and acquisitions.

2.2 Acute3D

Acquired by Bentley Systems in February 2015, Acute3D is now developing *ContextCapture*^{TM} , as part of Bentley Systems' Reality Modeling solutions.

Acute3D is a technological software company created in January 2011 by Jean-Philippe Pons and Renaud Keriven, by leveraging on 25 man-years of research at two major European research institutes, École des Ponts ParisTech and Centre Scientifique et Technique du Bâtiment. It won the French "most innovative startup" Awards. In 2011, Acute3D signed an industrial partnership with Autodesk, while keeping to advance its R&D work on city-scale 3D reconstruction. In 2012, Acute3D signed industrial partnerships with other industry leaders, including Skyline Software Systems and InterAtlas. In parallel, it started to commercialize its own Smart3DCapture® (now replaced by *ContextCapture*^{TM}) standalone software solution, optimized for highly detailed and large-scale automatic 3D reconstruction from photographs. In 2015, Acute3D is acquired by Bentley Systems, and becomes part of their end-to-end Reality Modeling solutions.

2.3 *ContextCapture*^{TM}

With *ContextCapture*^{TM} , you can quickly produce even the most challenging 3D models of existing conditions for infrastructure projects of all types. Without the need for expensive, specialized equipment, you can quickly create and use these highly detailed, 3D reality meshes to provide precise real-world context for design, construction, and operations decisions for use throughout the lifecycle of a project.

Hybrid processing in ContextCapture enables the creation of engineering-ready reality meshes that incorporate the best of both worlds – the versatility and convenience of high-resolution photography supplemented, where needed, by additional accuracy of point clouds from laser scanning.

Develop precise reality meshes affordably with less investment of time and resources in specialized acquisition devices and associated training. You can easily produce 3D models using up to 300 gigapixels of photos taken with an ordinary camera and/or 500 million points from a laser scanner, resulting in fine details, sharp edges, and geometric accuracy.

Extend your capabilities to extract value from reality modeling data with ContextCapture Editor, a 3D CAD module for editing and analyzing reality data, included with ContextCapture. ContextCapture Editor enables fast and easy manipulation of meshes of any scale as well as the generation of cross sections, extraction of ground and breaklines, and production of orthophotos, 3D PDFs, and iModels. You can integrate your meshes with GIS and engineering data to enable the intuitive search, navigation, visualization, and animation of that information within the visual context of the mesh to quickly and efficiently support the design process.

Chapter 3

Basic Concepts

This chapter introduces the necessary concepts and definitions for a better understanding of the report.

3.1 Point Cloud

Definition 3.1 *: Point*

A point p_i *is a tuple* $\langle x_i, y_i, z_i, \vec{n}_i \rangle$ *where:*

- i *is an unique integer identifying* p_i*,*
- x_i*,* y_i *and* z_i *are the coordinates of* p_i*,*
- $\vec{n}_i$ *is the normal at* p_i*.*

Definition 3.2 *: Point Cloud*

A point cloud P *of size* n *is a set* $P = \{p_i \mid i \in [0, n]\}$

3.2 Some algorithms

3.3 Principal Component Analysis (PCA)

PCA is a common algorithm of data analysis. It is used to project data into a space with reduced dimension while keeping the most relevant information – the directions where there is the most variance, where the data is most spread out.

Definition 3.3 *: Principal Component Analysis (PCA)*

3.3.1 Least Square Solving

Linear

Non-linear

Talk about Ceres solver?

3.3.2 RANSAC

3.4 Geometric operations

3.4.1 Translation

3.4.2 Rotation

3.4.3 Projection ??

Chapter 4

Scan Finder

This chapter describes *Scan Finder*, an algorithm that can be used to retrieve all scanners position of a point cloud, if there are any, regardless of its nature. Firstly, Section 4.1 reviews the context which brings this need and describes some characteristics of the expected solution. Then, Section 4.2 reminds that there is no previous work related to this subject. Finally, Section 4.3 and Section 4.4 show two different approaches to solve the problem and discuss the results.

4.1 Specifications

4.1.1 Context

ContextCapture^{TM} started to support point cloud reconstruction just two (2) years ago. Today it even provides a hybrid processing mode which gives the opportuniy to supplement photos of a scene with point clouds in order to have a better precision. However, a recurring problem observed among *ContextCapture*^{TM} users is the impossibility to use the software after losing some metadata, specifically, scanners location. This is particulary problematic because usually, *ContextCapture*^{TM} users subcontract point cloud production to private companies (for several thousand euro) which can charge them again for new exports (provided that they still have the point clouds). To enable customers to use their *defective* point clouds, the graphic interface of *ContextCapture*^{TM} allows to specify scanners location by hand by positioning them in a 3D representation of the point cloud. But, not surprisingly, 3D models reconstructed this way often contains errors.

This is how the need for an algorithm to automatically find scanners positions in a point cloud is born. In addition, with such algorithm, *ContextCapture*^{TM} will have the possibility to enhance the set of supported file formats. Currently, it supports file formats such as *PTX*, *e57*, *PLY*, *POD*, each of them being able to store scanners positions. But not all file formats are able to do it. For instance, *LAS* file format is not currently accepted as input because it does not provide any means of storing scanners location in metadata. Supporting more input file formats is also an interesting point for *ContextCapture*^{TM} users.

4.1.2 Objective

In summary, the purpose here is to improve *ContextCapture*^{TM} point cloud support in two ways: support more file formats and allow users who lose scanners location to still use their point clouds. To do so, the algorithm must:

- take as input any point cloud file format
- take as input monoscan and multiscan point clouds
- be invariant to differences between scanners, such as: density, noise, rotation angle
- find all scanners locations
- have a reasonnable running time

Let us precise here that as a first step, the algorithm is expected to work only with static point clouds. It would be difficult to have the same approach with static and mobile point clouds. Extending it to mobile point clouds is certainly the next step.

4.2 Related Work

As said in the introduction, to the best of our knowledge, there have been no previous work on the subject.

"Detecting the positions of multiple scanners exclusively from a point cloud is a subject not identified as interesting by academics, and thus not tackled since this information is almost always available

from the outset. The same holds true on the industry side, it is only recently that scanner position information is relevant for a few applications like ContextCapture™ *."*

The closest thing to it, [TM15, SQLG15, KGC15, KC15, NS17] use learning techniques to estimate the point of view of one particular photo based on other photos. But this belongs more to the photogrametry domain and therefore is not applicable in a point cloud context.

4.3 The grid-pattern method

This section explains one approach we tried in order to solve the problem. Before explaining the method in detail, let us first give the intuition behind this idea.

4.3.1 Overview

To understand this approach, it is important to know how a LiDAR scanner works. The next paragraph is inspired by [lid].

LiDAR is an acronym of Light Detection and Ranging. It is a remote sensing technology which uses the pulse from a laser to collect measurements. The principle is simple: it works in a similar way to Radar and Sonar but uses light waves from a laser, instead of radio or sound waves. A LiDAR system calculates how long it takes for the light to hit an object or surface and reflect back to the scanner and then, uses the velocity of light[1] to calculate the distance. LiDAR systems can fire around 1,000,000 pulses per second. The key point here is that not only a LiDAR scanner has a constant rotation angle when turning on itself but it also have a constant vertical rotation angle when scanning the environment.

These constant vertical and horizontal rotation angles reveal some grid patterns on surfaces perpendicular to the ground[2]. Figure 4.1 shows an example of a grid pattern that can be found in point clouds. The purpose here is to find, for a single grid pattern, a relation between all constant rotation angles and the scanner's position. Therefore, this relation can be applied with all possible grid patterns in the point clouds, leading to a problem with a huge set of constraints to solve. The idea is that only the real scanner's location is able to explain in the best way these rotation angles.

[1]The velocity, or speed of light is 299,792,458 metres per second.
[2]To be precise, perpendicular to the surface on which the scanner is installed.

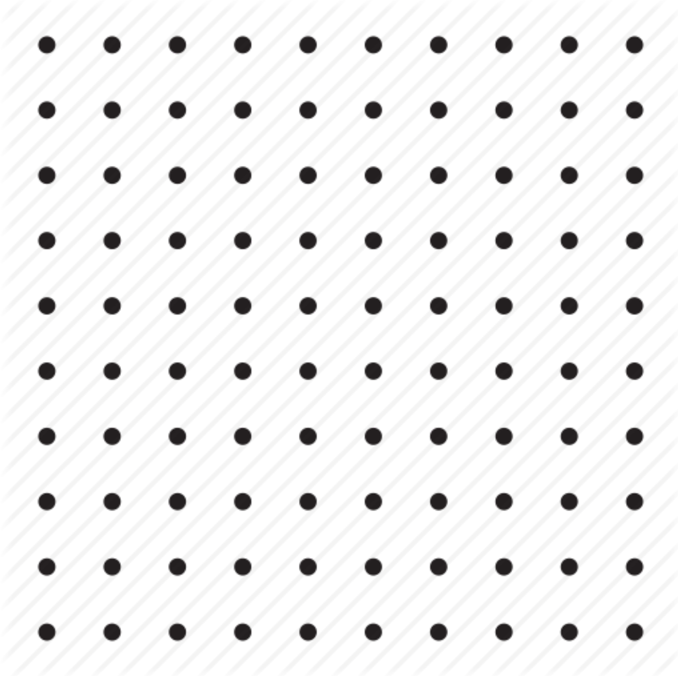

Figure 4.1: Example of a grid pattern. As you can see there is a constant offset between each vertical line and each horizontal lines. Note that the offset between columns is not necessary the same than between lines.

This algorithm can be divided into two parts. The first step is to find *accurate* grid patterns in point clouds. Once this is done, the second step is to build the equation that will be solved. These two parts are explained in the following subsections.

4.3.2 Grid-pattern matching

4.3.3 Equation to solve

4.3.4 Results and discussions

Not multiscan.

4.4 The elliptic method

4.4.1 Overview

4.4.2 Clustering high-density area

4.4.3 Fitting ellipse

4.4.4 Equation to solve

4.4.5 Results and discussions

Chapter 5

Point Cloud Visibility

5.1 Specifications

5.1.1 Problem being addressed

5.1.2 Objective

5.1.3 Scope

5.2 Related work

5.3 Direct Visibility of Point Sets

5.3.1 Overview

5.3.2 Implementation

5.3.3 Results and discussions

5.4 Visibility of Noisy Point Cloud Data

5.4.1 Overview

5.4.2 Implementation

5.4.3 Results and discussions

Chapter 6

Point Cloud Compression

6.1 Specifications

6.1.1 Problem being addressed

6.1.2 Objective

6.1.3 Scope

6.2 Related work

6.3 A custom arithmetic approach

6.3.1 Overview

6.3.2 Implementation

6.3.3 Comparison with Brotli, 7Z and Zip

6.4 Integration

Chapter 7

Conclusion

7.1 Summary of Internship Achievements

Summary.

7.2 Applications

Applications.

7.3 Future Work

Future Work.

Bibliography

[ben] Bentley systems by wikipedia. `https://en.wikipedia.org/wiki/Bentley_Systems#History`. Accessed: 2018-07-06.

[GKIS05] Stefan Gumhold, Zachi Kami, Martin Isenburg, and Hans-Peter Seidel. Predictive point-cloud compression. In *ACM SIGGRAPH 2005 Sketches*, page 137. ACM, 2005.

[KC15] Alex Kendall and Roberto Cipolla. Modelling uncertainty in deep learning for camera relocalization. *arXiv preprint arXiv:1509.05909*, 2015.

[KGC15] Alex Kendall, Matthew Grimes, and Roberto Cipolla. Posenet: A convolutional network for real-time 6-dof camera relocalization. In *Proceedings of the IEEE international conference on computer vision*, pages 2938–2946, 2015.

[lid] What is lidar? `https://www.3dlasermapping.com/what-is-lidar-and-how-does-it-work/`. Accessed: 2018-07-06.

[NS17] Yoshikatsu Nakajima and Hideo Saito. Robust camera pose estimation by viewpoint classification using deep learning. *Computational Visual Media*, 3(2):189–198, 2017.

[SK06] Ruwen Schnabel and Reinhard Klein. Octree-based point-cloud compression. *Spbg*, 6:111–120, 2006.

[SQLG15] Hao Su, Charles R Qi, Yangyan Li, and Leonidas J Guibas. Render for cnn: Viewpoint estimation in images using cnns trained with rendered 3d model views. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2686–2694, 2015.

[TM15] Shubham Tulsiani and Jitendra Malik. Viewpoints and keypoints. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1510–1519, 2015.