SENSOR-AWARE FILTERING, REPRESENTATION AND COMPLETION OF 3D SCANS

MASTER’S THESIS

Author
MARC COMINO TRINIDAD

Director:
CARLOS ANDÚJAR GRAN
Department of Computer Science (CS)

Co-Director:
ANTONIO CHICA CALAF
Department of Computer Science (CS)

July 8th, 2015

Computer Graphics Specialization
Master in Innovation and Research in Informatics

FACULTAT D’INFORMÀTICA DE BARCELONA (FIB)
UNIVERSITAT POLITÈCNICA DE CATALUNYA (UPC)
Acknowledgements

In the first place, I would like to express my deep gratitude to Professor Carlos Andújar and to Professor Antonio Chica, my research supervisors, and to Professor Pere Brunet who offered himself to give advice. Professors Andújar and Chica have now been guiding me through my studies for some time, they have already supervised my final degree project and this master’s project and I hope they will also be able to supervise my future PhD. Professor Brunet has a long and successful academic trajectory and is a reference in the field. Their combined insight and expertise greatly assisted the research. I must also offer them my special thanks for their patience and enthusiastic encouragement.

I would also like to show our gratitude to the Visualization, Virtual Reality and Graphics Interaction group, a combined team from the Technical University of Catalonia (UPC) and the University of Girona (UdG) which provided the data that we used during this project.

Finally, I wish to thanks friends and family for their moral support, because with them by my side, there is no stressful situation that I can not overcome.
Abstract (English)

In this work we address different problems related with the representation, registration, filtering and visualization of point-based 3D models.

The trivial representation for a point cloud is a sequential list of points. We claim that significant space savings can be achieved by taking advantage of the spatial coherence between neighbouring points. We thus propose an image-based approach which maps the data points onto the image plane by making use of their spherical coordinates. This results in a compact representation of the point cloud and its appearance attributes.

We also address the problem of registering a collection of 3D scans. We propose a registration pipeline supporting arbitrarily-large datasets and compare it with commercial software. We propose various criteria for comparing and numerically evaluating different registrations.

Robust normal estimation is key in most point-based tasks such as visualization and surface reconstruction. The methods in the literature are able to deal with noisy data under the assumption that every point has the same error. However, when a cloud of points is the result of registering various scans, this is not longer true. We propose and analyze a normal estimation method that is able to deal with this scenario.

Finally, visualizing a massive point cloud is not a trivial task since it might not even fit in memory. A practical solution to this problem is to build panoramas at different points of interest and enabling navigation between them. For this, we propose an efficient method to build panoramas from arbitrary view points by combining data from multiple 3D scans.

As test case, we will use a collection of 3D scans from the Mercat de Sant Antoni, a modernist building in the city of Barcelona that has been undergoing a reshuffling since 2009. The MOVING research group did recently a 3D scan of the Mercat and the archaeological remains that were found, resulting in 3.5 billion points.
Abstract (Catalan)

En aquest treball abordem diferents problemes relacionats amb la representació, el registre, filtrat i visualització de models 3D basats en punts.

La representació trivial per a un núvol de punts és una llista seqüencial de punts. Tanmateix, creiem que es pot aprofitar la coherència espacial de les dades per tal de comprimir-les en gran mesura. Per aquest motiu, proposem una representació basada en imatges que consisteix en fer un mapeig de cada punt a un punt en el pla d’una imatge, mitjançant les seves coordenades esfèriques. Això es tradueix en una representació compacta del núvol de punts i dels seus atributs.

També abordem el problema de registrar una col·lecció d’escaneigs 3D. Proposem un algorisme de registre que suporta conjunts de dades arbitràriament grans i en fem una comparació amb un software comercial. Proposem un conjunt de criteris per comparar i avaluar numèricament diferents registres.

Que les normals estiguin estimades de manera robusta és clau per moltes aplicacions tals com la visualització i la reconstrucció de superfícies. Els mètodes en la literatura són capaços de tractar amb dades amb un soroll uniforme, no obstant això, quan ens trobem amb punts que provenen de diferents escanejos cadascun pot tenir un error diferent. Nosaltres proposarem i analitzarem un algorisme que és capaç de treballar en aquest escenari.

Finalment, visualitzar un núvol de punts massiu no és una tasca fàcil, ja que pot no cabre-hi a memòria. Una solució pràctica a aquest problema és crear panorames en diferents punts d’interès i permetre la navegació entre ells. En conseqüència, proposarem un algorisme eficient i elegant per construir panorames des de punts de vista arbitraris mitjançant la combinació de dades de múltiples escanejos en 3D.

Com a cas de prova, utilitzarem una col·leCCIó d’escanejos en 3D del Mercat de Sant Antoni, un edifici modernista a la ciutat de Barcelona que està sent remodelat des de 2009. El grup de recerca MOVING va fer recentment un escaneig 3D del Mercat i de les restes arqueològiques que es van trobar, del qual se’n van extreure 3.500 milions de punts.
Abstract (Spanish)

En este trabajo abordamos diferentes problemas relacionados con la representación, el registro, filtrado y visualización de modelos 3D basados en puntos.

La representación trivial para una nube de puntos es una lista secuencial de puntos. Sin embargo, creemos que se puede aprovechar la coherencia espacial de los datos para comprimirlos en gran medida. Por este motivo, proponemos una representación basada en imágenes que consiste en hacer un mapeo de cada punto a un punto en el plano de una imagen, mediante sus coordenadas esféricas. Esto se traduce en una representación compacta de la nube de puntos y de sus atributos.

También abordamos el problema de registrar una colección de escaneos 3D. Proponemos un algoritmo de registro que soporta conjuntos de datos arbitrariamente grandes y hacemos una comparación con un software comercial. Proponemos un conjunto de criterios para comparar y evaluar numéricamente diferentes registros.

Que las normales estén estimadas de forma robusta es clave para muchas aplicaciones tales como la visualización y la reconstrucción de superficies. Los métodos en la literatura son capaces de tratar con datos con un ruido uniforme, sin embargo, cuando nos encontramos con puntos que provienen de diferentes escaneos cada uno puede tener un error diferente. Nosotros propondremos y analizaremos un algoritmo que es capaz de trabajar en este escenario.

Finalmente, visualizar una nube de puntos masiva no es una tarea fácil, ya que puede no caber en memoria. Una solución práctica a este problema es crear panoramas en diferentes puntos de interés y permitir la navegación entre ellos. En consecuencia, proponemos un algoritmo eficiente y elegante para construir panoramas desde puntos de vista arbitrarios mediante la combinación de datos de múltiples escaneos en 3D.

Como caso de prueba, utilizaremos una colección de escaneos en 3D del Mercado de San Antonio, un edificio modernista en la ciudad de Barcelona que está siendo remodelado desde 2009. El grupo de investigación MOVING hizo recientemente un escaneo 3D del mercado y de los restos arqueológicos que se encontraron, del que se extrajeron 3.500 millones de puntos.
# Contents

Acknowledgements 1  
Abstract (English) 2  
Abstract (Catalan) 3  
Abstract (Spanish) 4  

1 Introduction 10  
1.1 Justification ........................................ 10  
1.2 Objectives ........................................ 10  

2 State of the Art 11  
2.1 Neighbourhood Computation .......................... 11  
2.2 Normal Estimation .................................. 12  
2.3 Surface Reconstruction ................................ 13  
2.4 Registration ......................................... 15  
2.5 Point Editing ....................................... 16  
2.6 Rendering ........................................... 16  
2.6.1 Visibility Computation ............................ 16  
2.6.2 Splatting .......................................... 18  
2.7 Panoramas ........................................... 19  

3 Data Set Description 20  

4 Data Representation 23  
4.1 Introduction ......................................... 23  
4.2 The Panorama Representation ....................... 23  
4.3 Color Correction .................................... 27  
4.4 Experiments and Results ............................. 29  
4.5 Other Features ...................................... 30  
4.6 Conclusion .......................................... 31  

5 Data Registration 32  
5.1 Introduction ......................................... 32  
5.2 Sub-sampling ........................................ 33  
5.3 Registering ......................................... 37  
5.4 Voxelization ......................................... 37  
5.5 Evaluation .......................................... 38  
5.6 Conclusion .......................................... 43  

6 Normal Estimation 45  
6.1 Introduction ......................................... 45  
6.2 Framework .......................................... 46  
6.3 Segment Weights .................................... 47  
6.3.1 Intuition .......................................... 48
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.4</td>
<td>Uniform Weights</td>
<td>50</td>
</tr>
<tr>
<td>6.5</td>
<td>Quadratic Weights</td>
<td>50</td>
</tr>
<tr>
<td>6.6</td>
<td>Sinusoidal Weights</td>
<td>52</td>
</tr>
<tr>
<td>6.7</td>
<td>Discussion</td>
<td>54</td>
</tr>
<tr>
<td>6.8</td>
<td>Evaluation</td>
<td>55</td>
</tr>
<tr>
<td>6.9</td>
<td>Conclusion</td>
<td>57</td>
</tr>
<tr>
<td>7</td>
<td>Multi-scan Panoramas</td>
<td>59</td>
</tr>
<tr>
<td>7.1</td>
<td>Introduction</td>
<td>59</td>
</tr>
<tr>
<td>7.2</td>
<td>Generation</td>
<td>59</td>
</tr>
<tr>
<td>7.3</td>
<td>Visualization</td>
<td>63</td>
</tr>
<tr>
<td>7.4</td>
<td>Discussion</td>
<td>64</td>
</tr>
<tr>
<td>7.5</td>
<td>Conclusion</td>
<td>65</td>
</tr>
<tr>
<td>8</td>
<td>Future Work</td>
<td>67</td>
</tr>
<tr>
<td>9</td>
<td>References</td>
<td>68</td>
</tr>
<tr>
<td>A</td>
<td>Registration Comparison</td>
<td>70</td>
</tr>
</tbody>
</table>
List of Figures

1  The mercat de Sant Antoni. ................................................................. 20
2  Scans positions. ................................................................. 21
3  Visual representation of the capturing ranges for the Leica ScanStation P20. . . . 24
4  Diagram that shows how the panorama building algorithm works. .................. 25
5  Right: Color image. Left: Intensity image. Discontinuities in the intensity of the colors can be appreciated in the color images whereas they are not present in the intensity image. This is due to the way Cyclone computes the color of each point. 28
6  These images illustrate the procedure that Cyclone carries out in order to estimate the color of the sample points. ................................................................. 28
7  Left: original images. Right: color corrected images. Even very dark and very saturated images become similar in color after applying the correction. ................. 29
8  Some examples of points which an user might want to remove from the clouds. 31
9  Top Left: 23rd scan. Top Right: 28th scan. Bottom Left: 26th scan. Bottom Right: 29th scan. Red stars represent the sensor position, for the 29th it was the center of the dome. The parts of the clouds that overlap most are the arms of the building, which are the furthest from the sensors and, consequently, the most sparse-sampled. ................................. 32
10 Visual comparison of our method versus uniformly sampling. ...................... 33
11 Left: clouds of points sub-sampled with our method. Right: clouds of points sub-sampled uniformly. Top: these points were close the sensor, therefore their density is smaller with out method. Bottom: these points were far from the sensor, therefore their density is higher with out method. For our method we employed a radius of 0.024 meters, increasing this should also increase the difference in densities, at the cost of an increased sub-sampling time. ......................... 35
12 Comparative of point densities. Top charts shows the densities for the same set of points, before and after sub-sampling using our method. Bottom chart shows the densities if we employed an uniform sub-sampling strategy to obtain a set point points of equal size. In order to compute these densities the value used for \( r \) was 0.024 meters. ................................. 36
13 Meshlab align tool interface. ...................................................... 38
14 The grey line represents the underlying “real surface” and the different colors indicate points coming from different scans. On the one hand, top figure illustrates a scenario where the registry is good and all the points lay in the same line, on the other, bottom figure illustrates a scenario where the registry is worse and points are more dispersed. If we subdivide and count the number of occupied sub-voxels we end up with 12 for the former and 18 for the latter and, therefore, less occupied voxels indicate a better registration quality. ................................. 41
15 An approximately planar wall. Left: frontal view. Right: side view. Different grey tones correspond to points coming from different scans. All the scans were registered using the Leica Cyclone [1]. ...................................................... 43
16 Diagram giving an overview of our method for normal estimation. The gray dashed line represents the underlying surface. The red points were taken from sensor A and the yellow points were taken from sensor B. All the points have an associated interval, that goes in the direction of the ray from the sensor to the point, and their width is inversely proportional to the error of the point. ................................. 45
17 Charts showing the resulting weights for different combinations of errors and $k$. Black: error was set to 9mm. Red: error was set to 4.3mm. Purple: error was set to 2.8mm. Blue: error was set to 1.5mm. Green: error was set to 0.4mm. The two vertical bars are placed at $k = 300$ and $k = 1000$. 48

18 Effects of using gaussian weights based on the errors assigned to the points. 49

19 Cloud of points employed to test the normal estimation algorithms. 55

20 Visual result of the first round of experiments. Normals are coded as color, positive $x$, $y$ and $z$ values are codified as red, green, blue, respectively and negative $x$, $y$ and $z$ are codified as cyan, magenta and green, respectively. 56

21 Visual result of the second round of experiments. Normals are coded the same way as in Figure 20. 58

22 Diagram showing how the input clouds are processed to generate the input of the panorama generating algorithm. 60

23 Diagram that shows how the panorama building algorithm works. 61

24 Different captures of the panorama. 64

25 Left: panorama generated from the viewpoint of a sensor. Right: Splat points in white. Splats next to columns or railings are of type (2) whereas splat forming concentric lines are of type (1). 65

26 Different layers of a panorama generate form a point of view over the building. 66

27 Distributions of distances between points registered with the Cyclone transformations and with the transformations resulting from our registering approach. 70

28 Distributions of distances between points registered with the Cyclone transformations and with the transformations resulting from our registering approach. 71

29 Distributions of distances between points registered with the Cyclone transformations and with the transformations resulting from our registering approach. 72

30 Distributions of distances between points registered with the Cyclone transformations and with the transformations resulting from our registering approach. 73

31 Distributions of distances between points registered with the Cyclone transformations and with the transformations resulting from our registering approach. 74
# List of Tables

<table>
<thead>
<tr>
<th></th>
<th>Details of the information available for each sample taken from the laser scanner.</th>
<th>21</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Sample associated error as a function of its intensity and depth.</td>
<td>22</td>
</tr>
<tr>
<td>3</td>
<td>Number of samples along $\theta$ and $\psi$ taking into account the scanning spacings.</td>
<td>24</td>
</tr>
<tr>
<td>4</td>
<td>Compressing ratios obtained by codifying a point cloud as an image.</td>
<td>29</td>
</tr>
<tr>
<td>5</td>
<td>Compressing ratios obtained by codifying depth instead of $(x,y,z)$.</td>
<td>30</td>
</tr>
<tr>
<td>6</td>
<td>For each scan, difference in translation (length in centimetres) and the difference in rotation (offset angle) between the Cyclone registration matrix and the matrix yielded by our approach.</td>
<td>39</td>
</tr>
<tr>
<td>7</td>
<td>Results for the Welch’s t-test comparing the average occupation of sub-voxels.</td>
<td>42</td>
</tr>
<tr>
<td>8</td>
<td>Summary of the weights obtained by the different approximations to the gaussian distribution.</td>
<td>54</td>
</tr>
</tbody>
</table>
1 Introduction

1.1 Justification

In recent years point-based geometry experienced a major renaissance as an alternative surface representation. There are two main reasons behind this: first, point clouds do not require to store nor to maintain globally consistent topological information, which can become a problem for nowadays very large polygonal-meshes. This makes this representations much more flexible when it comes to handling highly complex or dynamical changes of shape. Second, modern digital scans and photography based systems generate huge volumes of points samples which can be regarded as the discrete building pieces of 3D objects.

More arguments against the use of triangle meshes are that algorithms that work with them require topologically consistent two-manifold surfaces and, as a consequence, manifold-extraction or topology cleanup steps are necessary for mesh generation methods. Also, when the number of triangles exceeds the number of pixels on the screen, rendering such massive data sets leads to triangles whose projected area is less than one pixel.

We have the chance of working with a massive point cloud coming from the 3D scanning of the Mercat de Sant Antoni. Because of this, we are going to extend traditional algorithms and develop new ones in order to deal with large amount of points. Therefore obvious requirements are efficiency in time and memory without renouncing to quality and robustness.

1.2 Objectives

The work done in this project is clearly divided into four different topics, namely representation, registration, normal estimation and visualization of massive points clouds. A common objective for all of them is developing robust algorithms which are able to deal with large amount of points in an efficient way. Following we will give a brief description of particular objectives for every subject.

For the representation section our objectives are:

1. Specifying a new kind of representation that allows us to save storage space.

2. Developing an algorithm that is able to transform a plain dump of points into the new representation.

3. Studying the new representation, analysing its advantages and inconveniences with respect to the dump of points.

For the registering section our objectives are:

1. Developing a registration technique that is able to deal with massive data points.
2. Completing the partial registration provided by the vendor of the scanning device.
3. Performing an alternative complete registration with our method.
4. Developing techniques to numerically evaluate and compare different registrations.

For the normal estimation section our objectives are:

1. Developing a normal estimation algorithm which is able to deal with noisy cloud of points where the error is not uniform.
2. Evaluating the quality of the resulting normals.

For the visualization section our objectives are:

1. Develop an algorithm that is able to generate an arbitrary-viewpoint panorama from a massive point cloud.
2. Evaluating the quality of the panoramas and propose ways to improve it.

2 State of the Art

In this brief state of the art we will outline the work of some authors regarding various topics related to point data sets. Some useful sources of information on this topic are the survey by Kobbelt and Botsch [2], the Point-Based Computer Graphics SIGGRAPH 2004 Course Notes [3] and the Point Based Graphics book [4] by Gross and Pfister.

2.1 Neighbourhood Computation

Estimating local surface properties for point clouds usually requires the computation of a small neighbourhood around each data point. The restricted Delaunay neighbourhood is popular in the computational geometry community, since it is possible to prove many geometric and topological results about it under certain sampling conditions. The restricted Delaunay neighbours are always a subset of the Delaunay ones, and their main drawback is that they are not as efficiently computable as the k-nearest neighbours.

On the other hand, it is easy to construct examples where estimating local surface properties using the k-nearest neighbours will completely fail, for instance, when the sampling is locally uniform. In practice, however, most acquired point sets encountered in graphics applications are locally uniform. Thus Andersson et al. [5] give an upper-bound in order to approximate the restricted Delaunay neighbourhood from the k-nearest neighbours.
Let $S$ be a smooth surface and $P \subset S$ a point sample from $S$. Let $f : S \to \mathbb{R}$ be a function that assigns to every point in $S$ its least distance to the medial axis. The point sample $P$ is called an $\epsilon -$sample of $S$, if every point $x \in S$ has a point in $P$ at distance at most $\epsilon f(x)$. It is also called an $(\epsilon, \delta)$-sample of $S$, if it is an $\epsilon$-sample, and $p - q \leq \delta f(p)$, for all $p, q \in P$.

Then, choosing:

$$k \geq \frac{(\delta(1 + w) + w)^2}{\delta^2(1 - w)^2 - w^4}$$

where $w = \frac{2\epsilon}{1 - \epsilon}$, guarantees that the $k$ nearest neighbours of a sample point $p$ include all its restricted Delaunay neighbors.

### 2.2 Normal Estimation

There is much of work done in order to estimate the normal on manifold representations of surfaces. However we are concerned with estimating the normal at each point in a point cloud, given to us only as an unstructured set of 3D points. Once we have a way to compute the neighbourhood $N_p$ of a given data point $p$, Hoppe et al. [6] propose a way to estimate the normal of the surface going through $p$ based on estimating the normal of a plane tangent to the surface, which in turn becomes a least-square plane fitting estimation problem.

The solution for estimating the surface normal is therefore reduced to performing a Principal Components Analysis on the covariance matrix created from $N_p$. Namely, we analyse the eigenvectors and eigenvalues of this matrix, which is composed as follows:

$$c_p = \frac{1}{|N_p|} \sum_{q \in N_p} q$$

$$cov = \sum_{q \in N_p} (q - c_p) \times (q - c_p)$$

where $\times$ denotes the outer product vector operator. After applying PCA, the eigenvector with smallest eigenvalue is kept as the normal.

Mitra et al. [7] say that the accuracy of the normal estimation using Hoppe’s method depends on the noise in the point cloud, the curvature of the underlying manifold, the density and the distribution of the samples and the neighborhood size. They study the contribution of each of these factors on the process and this allows them to find the optimal neighborhood size to be used in the method.
2.3 Surface Reconstruction

Hoppe et al. [6] design an algorithm for surface reconstruction from clouds of points which is based on estimating the signed geometric distance to the unknown surface and then applying a variant of marching cubes in order to get the reconstructed geometry.

For this, the first step is estimating the tangent plane at each data point by making use of Equations 2 and 3. Then the tangent planes are consistently oriented by:

1. A Riemannian Graph = \( G(V, E) \) is built where \( V = \{c\} \) is the set of tangent planes centers and \( E = \{e = (c_i, c_j)\} \) is the set of edges that fulfil the property that \( c_j \in k\text{-}neighbourhood(c_i) \).
2. Each edge \( e = (c_i, c_j) \) is assigned a weight \( 1 - |\text{normal}(c_i) \cdot \text{normal}(c_j)| \).
3. A Minimum Spanning Tree is computed over this graph.
4. An initial plane is selected such that its center has the largest \( z \) coordinate and its normal is forced to point toward the +\( z \) axis.
5. The orientation of this plane is propagated in depth-first order following the MST.

Following, a voxelization of the cloud of points is built by discretely sampling the space following a 3D grid and saving for each point its distance to the closest tangent plane. Then a variation of the marching cubes algorithm is applied in order to obtain a simplicial surface which is then simplified using an edge-collapse-like algorithm.

Alexa et al. [8] propose using moving least-squares (MLS) surfaces for point-based methods, which relies on the idea that a given point set implicitly defines a surface. The procedure requires a monotonically decreasing weight function \( \theta \) to be chosen and the usual choice is a Gaussian \( \theta(d) = \exp\left(-\frac{d^2}{h^2}\right) \), which results in \( C^\infty \) continuous MLS surfaces. \( h \) is the globally estimated sample spacing and can be used to control the degree of smoothing.

Let points \( p_i \in \mathbb{R}^3 \) be sampled from the target surface \( S \) (they may contain some noise). Following a method is described that makes it possible to project a point \( r \in \mathbb{R}^3 \) onto the two-dimensional surface \( S_p \) that approximates the sampled points. For this, the following 3 steps procedure is carried out:

1. By minimizing a local weighted sum of square distances to the samples points, a plane \( H_r = \{x|n^T x - D = 0, x \in \mathbb{R}^3\}, n \in \mathbb{R}^3, \|n\| = 1 \) is computed. The objective function to minimize is given by:

\[
\sum_{i}^{N} (n^T p_i - D)^2 \theta(||p_i - q||)
\]

where \( q \) is the projection of \( r \) onto \( H_r \).
2. The reference domain for \( r \) is used to compute a local bivariate polynomial approximation to the surface in a neighbourhood of \( r \) by a weighted least-squares fit to the sampled points. Let \( q_i \) be the projection of \( p_i \) onto \( H_r \), and \( h_i \) the height of \( p_i \), the function to minimize is defined as:

\[
\sum_{i=1}^{N} \left( g(x_i, y_i) - h_i \right)^2 \theta(\|p_i - q\|)
\]

where \((x_i, y_i)\) is the representation of \( q_i \) in a local coordinate system in \( H_r \) and \( g \) is the chosen polynomial approximation.

3. The projection \( \mathcal{P} \) of \( r \) onto the surface that approximates the sample points is given by:

\[
\mathcal{P}(r) = q + g(0,0)n = r + (t + g(0,0))n
\]

The major drawback of MLS surfaces is the computationally involved non-linear optimization problem for finding the reference frame \( H_r \).

Finally, Fuhrmann and Goesele [9] introduce a new method for surface reconstruction from oriented, scale-enabled sample points which operates on large, redundant and potentially noisy point set. Sample points are assumed to contain position, normal and scale information and optionally some confidence information. The surface is approximated by the zero set of an implicit function which is the result of the weighted sum of a set of basis functions parameterized by each sample point:

\[
F(x) = \frac{\sum_i c_i w_i(x) f_i(x)}{\sum_i c_i w_i(x)}
\]

Function \( f_i \) and weight \( w_i \) are parameterized by the ith sample position \( p_i \), normal \( n_i \) and scale \( s_i \). If the confidence \( c_i \) is omitted it is assumed to be 1. Without loss of generality, \( f_i \) and \( w_i \) are defined as one parameter family of functions depending only on the scale of the sample. The position \( p_i \) and normal \( n_i \) are considered by translating and rotating the input coordinate \( x \)

\[
x_i = R(n_i) \cdot (x - p_i)
\]

After this transformation there is still one degree of freedom not fixed and, therefore, it is important that \( f_i \) and \( w_i \) are defined in such a way that are rotation invariant around the normal.

As basis function \( f \) they use the derivative of the Gaussian \( f_x \) in the direction of the normal with \( \sigma = s_i \) set to the scale of the sample. Normalized Gaussians \( f_y, f_z \) are used orthogonal to the normal in \( y \) and \( z \) direction:

\[
f_x(x) = \frac{x}{\sigma^2} \exp \left( \frac{-x^2}{2\sigma^2} \right)
\]

\[
f_y(x) = f_z(x) = \frac{1}{\sigma \sqrt{2\pi}} \exp \left( \frac{-x^2}{2\sigma^2} \right)
\]
As weighting function \( w \) they use a polynomial that has compact support, falls smoothly off to zero and gives more weight to the regions in front of the surface:

\[
    w(x_i) = w_x(x) \cdot w_{yz}(\sqrt{y^2 + z^2})
\]

with a non-symmetric component in the \( x \)-direction

\[
    w_x(x) = \begin{cases} 
    \frac{x^2}{9\sigma^2} + \frac{2x}{3\sigma} + 1 & x \in [-3\sigma, 0) \\
    \frac{2x^3}{27\sigma^3} - \frac{x^2}{3\sigma^2} + 1 & x \in [0, 3\sigma) \\
    0 & \text{otherwise}
    \end{cases}
\]

and a rotation invariant component in \( y \)- and \( z \)-directions

\[
    w_{yz}(x) = \begin{cases} 
    \frac{2x^3}{27\sigma^3} - \frac{x^2}{3\sigma^2} + 1 & x < 3\sigma) \\
    0 & \text{otherwise}
    \end{cases}
\]

After this, all the input samples are put inside an octree, which is then modified to either contain only inner nodes (which have exactly 8 children) or leaf nodes (which have no children). Leaf nodes are sampled on the 8 corners of the voxel they define by making use of \( F \). Finally, a variant or marching cubes works with these samples in order to extract an isosurface.

### 2.4 Registration

The general objective of registration is estimating the optimal rotation and translation that aligns (or registers) what we call the model shape and the data shape. Basically, the model shape is the target geometry and the data shape is the geometry that we want to align with the target.

One classical algorithm that performs this task is Iterative Closest Points[10]. Let the data shape be \( P = \{p_i\}_{i=1..N_P} \) a set of points and let the model shape be \( X = \{x_i\}_{i=1..N_X} \) a set of geometry. ICP is an iterative two-step algorithm that tries compute a rigid movement \((R, \Delta_t)\) that minimizes the objective function:

\[
    E = \sum_{i=1}^{N_p} ||Rp_i + \Delta_t - C(p_i, X)||^2
\]

where \( C(p_i, X) \) is the correspondence of \( p_i \) in \( X \), namely a point \( q_i \) such that \( \forall q \in X, d(p_i, q_i) \leq d(p_i, q) \). Note that \( d(a,b) \) denotes the euclidean distance between \( a \) and \( b \).

Particularly, the algorithm consists in:

1. Given fixed correspondences \( C(p_i, X) \), \( \forall p_i \in P \), find \( \arg\min_{R, \Delta_t} \sum_{i=1}^{N_p} ||Rp_i + \Delta_t - q_i||^2 \).
2. Move the points according to \((R, \Delta_t)\) and recompute \( C(p_i, X) \) as the points \( q_i \) such that \( \forall q \in X, d(p_i, q_i) \leq d(p_i, q) \).
3. Iterate (1) and (2) until \( E \) becomes smaller than a given threshold. Output the accumulated transformation.
2.5 Point Editing

One of the major advantages of point-based representations in comparison with triangular meshes (in general, polygonal meshes) is their ability to be easily restructured without having to care for any manifold conditions. Because of this, applications that require frequent resampling will benefit from this representations.

Pauly et al. [11] presented a free-form shape modeling framework for point-based representations that used a proxy geometry mixing unstructured point clouds with the implicit surface definition of the moving least squares approximation, in order to be able to exploit the advantages of implicit and parametric surface models. In particular, they are able to support boolean operations and free-form deformations.

Zwicker et al. [12] introduced a system for efficient 3D appearance and shape editing of point-based models. For this, they devised a point cloud parameterization scheme and a dynamic resampling policy based on a continuous reconstruction of the model surface.

More recently, Calderon et al. [13] proposed a complete framework for the morphological analysis on point clouds. By introducing a new model for the structuring element, based on a signed scalar field representation, and substituting the Minkowski sum with a new projection procedure, they are able to simulate dilations and erosions without the need of any kind of topological information.

2.6 Rendering

2.6.1 Visibility Computation

Once a surface is reconstructed from the points, it is certainly possible to determine which of the points are visible, which implies that a point cloud inherently contains in formation from which it is possible to extract the visibility of the points. This is the baseline for the work of Katz et al. [14] which present their HPR operator in order to approach this problem.

Given a set of points $P$ and a camera viewpoint $C$, the Hidden Point Removal operator (HPR) works as follows:

1. Translate the system in such a way that $C$ becomes the new origin.
2. Compute the spherical flipping of $\forall p_i \in P$ with respect to an sphere of radius $R$:

$$\hat{p}_i = p_i + 2(R - \|p_i\|) \frac{p_i}{\|p_i\|}$$ (15)

3. Given $\hat{P} = \{\hat{p}_i\}$
4. Set as visible these points in the convex hull of $\hat{P} \cup C$. 

16
The work also defines the notion of \( \epsilon \)-visibility saying that a point \( p \) is \( \epsilon \)-visible if \( \exists q \in \mathbb{R}^D \) such that \( |q - p| < \epsilon \) and \( q \) is visible from \( C \), and gives some relations between the value of \( \epsilon \) and \( R \).

Machado et al. [15] improve the previous method by porting part of the computation to the GPU. In particular, the most expensive step turns out to be the computation of the convex hull which is now done using the following approximated algorithm, which restricts the computations to a smaller set of points:

1. An enclosing sphere \((C, r)\) is computed defining \( C \) to be the centroid of the cloud of points and \( r \) to be the distance to the furthest point from \( C \).

2. Let \( O \) be the position of the observer. This sphere is split into sectors by dividing the horizontal and vertical angles of the viewing frustum regularly in a grid-like manner. The region of the frustum containing the enclosing sphere will be symmetrical, covering an angle:

\[
\Delta \phi = 2 \sin^{-1} \frac{r}{|O - C|}
\]  

(16)

3. Using spherical coordinates, the frustum will then correspond to ranges:

\[
\psi \in \left[ -\frac{\Delta \phi}{2}, \frac{\Delta \phi}{2} \right]
\]  

(17)

\[
\theta \in \left[ \frac{\pi}{2} - \frac{\Delta \phi}{2}, \frac{\pi}{2} + \frac{\Delta \phi}{2} \right]
\]  

(18)

In order to produce \( k \) sectors, these angular ranges are regularly sampled \( \sqrt{k} \) times in each direction.

4. Sector representatives are computed by, first, spherically flipping all the points in the sector and then finding the point which is the maximum in the direction pointing to the center of the sector.

5. The representative of each sector is refined by comparing it to the representatives of the eight sectors sharing an edge or a vertex in the angular grid. This is because, for a given sector, the neighbouring representatives might be more extremal for its direction than its own representative. This is repeated until no better estimate is found for any sector taking care of avoiding computations for empty sectors.

6. The remaining representatives of this process are used to compute the convex hull.

Another advantage of this algorithm is that the computation for each sector can be easily parallelized in GPU and authors claim that makes it suitable for real-time rendering of point clouds with millions of points.
2.6.2 Splatting

Surface splats have first been proposed for rendering purposes by Zwicker et al. [16]. In order to bridge the gaps between neighbouring point samples, points $p_i$ are associated with a normal vector $v_i$, two tangential axes $u_i$ and $v_i$ with radii $r_{u_i}$ and $r_{v_i}$. A locally optimal approximation to the curvature of the underlying surface is achieved if the two axes are aligned to the principal curvature directions of the underlying surface and the radii are inversely proportional to the corresponding minimum and maximum curvatures.

Splat-based representations can be considered superior to triangle meshes if we take into account that, in differential geometry, an ellipse is the best local linear approximant to a smooth surface. Also, same as for triangles, sampling density can be adapted to the surface curvature but, on the other hand, since splats do not have to join continuously they still provide the same topological flexibility as pure point clouds.

Mutual overlap of splats in object-space guarantees a hole-free rendering in image-space however a naive approach might cause shading discontinuities. Zwicker et al. [16] for this proposed a high quality anisotropic anti-aliasing method based on the Elliptical Weighted Average (EWA) filter. Each splat is assigned a radially symmetric Gaussian filter kernel, such that a continuous surface signal in object-space is reconstructed by a respectively weighted averaging of splat data.

Nevertheless this implementation was completely software based which limited the real-time interactivity. Rusinkiewicz and Levoy [17] proposed a hierarchical structure, based on a pre-computed bounding-sphere tree, that can be used for visibility culling, level-of-detail control, and rendering. Each node of the tree contains the sphere center and radius, a normal, the width of a normal cone and optionally a color. Following a more detailed explanation is given on the capabilities of this structure:

1. As the tree is traversed, frustum culling is performed by testing each sphere against the planes of the view frustum. If the sphere lies completely outside, the subtree is discarded, it it lies completely inside, frustum culling is not applied on the subtrees.

   Backface culling is applied in a similar fashion: if the cone of normals faces entirely away from the viewer, the node and its subtree are discarded. If it entirely faces towards the viewer, no checking is performed on the subtrees.

2. Projected size on the screen is used to determine when to expand a node, more specifically a node is subdivided if the area of the sphere, projected onto the viewing plane, exceeds a threshold.

3. Once a leaf node has been reached or the recursion has stopped a splat of the size of the projected diameter of the current sphere is rendered. Its color is obtained from a lighting calculation based on the current per-sphere normal and color.

   When two splats are projected to the same pixel, only closely overlapping splats should be blended (the rest are assumed to belong to more distant surfaces and should be ignored).
Rusinkiewicz and Levoy [17] achieves this through some kind of deferred shading which is usually referred as visibility splatting, i.e. all objects are rendered slightly shifted away from the viewer and, in a second pass, the depth test discards all splats that are further than this shift.

Elliptical splats can be achieved by rendering just one vertex per splat and making use of the programmable pipeline. In the vertex shader, the projected size is computed and this triggers the rasterization of an image space square which is then turned into an ellipse in the fragment shader by discarding the pixels outside of it.

In order to represent sharp features by point-sampled geometries, Pauly et al. [11] proposed that all splats that sample these features to be clipped against one (edges) or two (corners) clipping lines that are specified in their local tangent frames. This can be easily integrated in fragment shaders as a per pixel-clipping test.

Finally, say that since current graphics hardware is highly optimized for triangle rendering, points or splats still cannot keep up with triangles in terms of effective rendering performance.

2.7 Panoramas

The rendering quality and scene complexity are sometimes limited because of the real-time constraint. Image-based rendering approaches can help in these cases in order to reduce the scene complexity while maintaining a fairly good rendering quality.

Chen [18] proposes an architecture where environment maps, in particular 360-degree cylindrical panoramic images, are used to compose a scene. The environment maps are orientation-independent images, which allow the user to look around in arbitrary view directions through the use of real-time image processing. These panoramic images can be created with computer rendering or specialized panoramic cameras. Also, nowadays many mobile device cameras can stitch together overlapping photographs to generate them.

The panoramas are digitally warped on-the-fly to simulate camera panning and zooming. Walking in a space is currently achieved by “hopping” to different panoramic points. More modern approaches, such as Benedetto et al. [19], consists in precomputing video sequences between view-points and playing them during the transitions.
3 Data Set Description

The data set that will be used to carry out all the experiments related to this project is comprised of 31 different 3D scans of the building known as Mercat de Sant Antoni (Figure 1a). This building has an extension of 5214 m$^2$ and can be found in the crossing between the streets Comte Urgell, Tamarit, Comte Borell and Manso (Figure 1b).

(a) Mercat de Sant Antoni.   (b) Comte Urgell, Tamarit, Comte Borell and Manso crossing.

Figure 1: The mercat de Sant Antoni.

The Mercat de Sant Antoni is a modernist building that has a steel structure composed of four big arms that converge in a big octagonal dome 28 meters high. This dome is supported by eight big steel columns. Since 2009 the whole building is undergoing a reshuffling which uncovered some archaeological remains of the medieval wall of Barcelona.

In order to keep a record of the works done in the building, the Visualization, Virtual Reality and Graphics Interaction group (ViRVIG from now on) was asked to perform a 3D scanner of it. For this task, they employed the Leica ScanStation P20 whose specification sheet can be found in the Annex.

The scanning positions are shown Figure 2. Out of the 31 scans, the first one was taken with a spacing of 1.6 mm (at 10 meters) between samples, which took close to an hour, and the rest were taken at a smaller spacing of 3.1 mm at 10 meters. Also, all the scans have a field of view of 360° horizontally and 270° vertically, except scans 23, 24, 25, 26, 27, 30 and 31 which have a field of view of 180° horizontally.

Each scan’s data is contained on a pts file, which is an ASCII file where each row represents a sample. Particularly, for each sample we have information about its three dimensional position, coded as $(x, y, z)$, the intensity of its color and its color, coded as $(r, g, b)$. Following we give a more in-depth description of this data:
1. The maximum range of the scanner is 120 m and, therefore, none of the values for the coordinates will exceed that. It is also important to note that, in this case, the height dimension is coded as the $z$.

2. The intensity field is a value that indicates how much infrared light was reflected from the ray that captured the sample. All the values range from -2048 to 2047, probably to make them fit in a 12 bit representation.

3. The scan determines the $(r, g, b)$ color \textit{a posteriori}. Once all the samples have been captured, multiple pictures of the scene are taken and, from them, the sample color is approximated. This may cause color inconsistencies specially if there are dynamic elements.

The details of the information stored for each sample are summarised in Table 1.

<table>
<thead>
<tr>
<th>Coordinates $(x, y, z)$</th>
<th>Type</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>float</td>
<td>(-120, 120)</td>
</tr>
<tr>
<td>Intensity</td>
<td>integer</td>
<td>(-2048, 2048)</td>
</tr>
<tr>
<td>Color $(r, g, b)$</td>
<td>unsigned char</td>
<td>(0, 255)</td>
</tr>
</tbody>
</table>

\textit{Table 1: Details of the information available for each sample taken from the laser scanner.}

Each of the \texttt{pts} files contains between 13 and 420 million samples and overall they sum up to more or less 3500 million samples which weight around 160 GB.

Moreover, in Table 2 we show the measurement error in depth given the sample position and intensity, according the machine’s specification.

Finally, the Leica Cyclone [1] tool was used to obtain a registration of the point clouds corresponding the different scans. In order to do so, some marks were set around the scenario during
the scanning process and were identified manually in the point clouds. These marks served to provide a first coarse alignment of all the scans and some optimization procedure took care of refining it.

The result of this process was a set of transformations that align scans ranging from 1 to 22 and scans ranging from 23 to 31 separately. Nevertheless, later it was found that the transformation corresponding to the 2nd scan was not coherent with the rest and had to be discarded.

<table>
<thead>
<tr>
<th>Distance (m)</th>
<th>Black(10%)</th>
<th>Gray(28%)</th>
<th>White(100%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0.8 mm</td>
<td>0.5 mm</td>
<td>0.4 mm</td>
</tr>
<tr>
<td>25</td>
<td>1.0 mm</td>
<td>0.6 mm</td>
<td>0.5 mm</td>
</tr>
<tr>
<td>50</td>
<td>2.8 mm</td>
<td>1.1 mm</td>
<td>0.7 mm</td>
</tr>
<tr>
<td>100</td>
<td>9.0 mm</td>
<td>4.3 mm</td>
<td>1.5 mm</td>
</tr>
</tbody>
</table>

*Table 2: Sample associated error as a function of its intensity and depth.*
4 Data Representation

4.1 Introduction

As described in the previous section, our data set is comprised of 31 ASCII files containing information about 3,487,095,733 points which weight 157.3 GB. One can think in many scenarios where the magnitude of this data is not friendly, for instance for storage, for moving the data between different machines, for streaming the data or even for reading it from disk to serve it to some application.

Reducing the size of this data is convenient even for visualization since it is not the same reading 160 GB from disk than reading a tenth of it. A first step in this direction is converting the files into a binary format:

- The \((x, y, z)\) coordinates can be stored using 3 floats (3 x 32 bits).
- The intensity value can be stored using 1 signed short (16 bits).
- The \((r, g, b)\) values can be stored using 3 unsigned chars (3 x 8 bits).

Overall, changing to described binary format reduces the size to 55.20 GB (64.9% reduction). But this approach is kind of trivial and one could think of many ways of, for instance, exploiting the spatial and color continuity on the data.

Another handicap that of binary representation has is that it is not very friendly for point editing. Namely, it is very difficult to identify in the list a region of points that we may want to edit. Later, we will show some application where this might be useful and desirable.

4.2 The Panorama Representation

In this section we will explain how we approach the problem of representing the data extracted from a single scan in a compact and friendly way. The data in each of our 31 scans was taken by uniformly sampling a sphere around the Leica sensor (Figure 3). Our idea is mapping each point to a 2D plane, using its spherical coordinates, and building an image layer for each piece of information. Then we will use available image compression techniques to exploit the spatial coherence of the data and reduce the storage requirements.

Basically we want to transform our 31 scans to \(31 \times 7\) image layers (3 layers for \((x, y, z)\), one for intensity and 3 for \((r, g, b)\)). This is something that can be done in a quite straightforward way because of how the scan data was taken (there is at most one sample along every ray that starts at the scanning position).

Now, imagine we take a sphere described by polar coordinates \((\theta, \psi)\) and unroll it such that we end up with a rectangle with the \(\theta\) along its width and the \(\psi\) along its height. In our case we
can do the same mapping for our points if we compute their polar coordinates. Then we will want to discretize this rectangle in regular intervals so that we are able to store the resulting information into an image.

In order to compute the optimal resolution for this discretization we must take into account multiple facts:

1. The spacing between points can be either 1.6 mm or 3.1 mm at 10 meters.
2. The $\theta$ value can range between 0 and 360º. The length of a 360º arc on a sphere of 10m is 62.83m.
3. The $\psi$ value can range between 0 and 135º. The length of a 135º arc on a sphere of 10m is 23.56m.

Using this information we can easily obtain the target resolution information in Table 3.

<table>
<thead>
<tr>
<th>Spacing</th>
<th>Nº of samples along $\theta$</th>
<th>Nº of samples along $\psi$</th>
<th>Total Samples</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.6 mm</td>
<td>$\approx 39269$</td>
<td>$\approx 14726$</td>
<td>$\approx 580$ million</td>
</tr>
<tr>
<td>3.1 mm</td>
<td>$\approx 20268$</td>
<td>$\approx 7600$</td>
<td>$\approx 154$ million</td>
</tr>
</tbody>
</table>

Table 3: Number of samples along $\theta$ and $\psi$ taking into account the scanning spacings.

Now, if we take a look at the actual number of samples in our scans we will see that they are slightly inferior to this. The scan taken with spacing of 1.6mm has 420 million points instead of 580, between the others, the scans that were taken in an open space lack data corresponding to the sky and the rest ranges between 130 to 150 million points. This means that the theoretical scan resolution is higher than the reported in practice.

In the end, we decided to generate our images at a lower resolution because of the reason stated above but also because we want GPU friendly resolutions (in the end we are also inter-
ested in rendering them) and because the points contain some noise meaning that the $\theta$ and $\phi$ steps between them is not uniform.

Following we will explain how the proposed algorithm works. Figure 4 shows an overview:

1. Take one of the 31 scans, read its points and map them to a unit sphere. Recall that overlapping points should not happen.

2. Build a kd-tree with the mapped points to speed up neighbourhood searches.

3. Build multiple image layers of resolution $(2x, x)$, where $x$ is a user specified parameter.

4. Begin an iteration traversing each pixel of the image layers. For each pixel, compute its corresponding polar coordinate $(\theta, \psi)$.

Figure 4: Diagram that shows how the panorama building algorithm works.
(a) Notice that not every scan covers all the $\theta$ range (before it was mentioned that some scans just cover 180°) and that, in this cases, the $\theta$ values can cover an arbitrary range (meaning that they may not always range from 0 to 180 but, for instance, from 23 to 203 or to from 300 to 120 and so on). Because of this we precompute the covered range and the necessary offset so that the image ends up centred.

(b) In general not every scan covers all the $\psi$ range either. The possible values range from 45° to 180°. We use an offset so that the minimum $\psi$ corresponds to the bottom row of the images.

5. Let $p_{q}$ be the point resulting from converting the $(\theta, \psi)$ polar coordinates to $(x, y, z)$. The neighbourhood $\mathcal{N}_{p_{q}}$ is computed by performing a search around $p_{q}$ on the mapped cloud using the kd-tree. As search radius we use the length of the range of spanned $\theta$ divided by 2x.

6. If $|\mathcal{N}_{p_{q}}| = 0$ it means that there was no sample that represents well enough the current position and therefore the pixel is filled with a key color that depends on the layer being filled (this will be explained in more detail later).

7. Usually, when $\mathcal{N}_{p_{q}}$ is not empty it happens that $|\mathcal{N}_{p_{q}}| > 1$. This is because the scanner sampling was not perfectly uniform and because we usually use a resolution different than the optimal. In this case we average all the retrieved points using something similar to a bilateral filtering. For this we first select the sample point which is closest to the queried point and call it candidate $p_{c}$.

8. Let $d(p_{1}, p_{2})$ denote the distance between point $p_{1}$ and point $p_{2}$. We compute the weight of each $p \in \mathcal{N}_{p_{q}}$ as:

$$w_{p} = e^{-(d(p, p_{c})^2k)2} \cdot e^{-(d(p, p_{q})^2k)2}$$

(a) The order of magnitude $d(p, p_{c})$ and $d(p, p_{q})$ is of millimetres but they are measured in meters. If we used this value ($\approx 10^{-3}$) the squaring and exponentiation would bring them close to 0. Therefore we use $k = 1000$ avoid this.

(b) $e^{-(d(p, p_{q})^2k)2}$ penalises points that are very far from the query point $p_{q}$.

(c) $e^{-(d(p, p_{c})^2k)2}$ penalises points that are very far from the candidate point $p_{c}$.

9. Different layers are saved as different images. In particular $(x, y, z)$ is saved to one image, intensity to another and $(r, g, b)$ to another since this way the spatial coherence in the data is maximized.

The codification used for the different images:

- $(x, y, z)$: these three values range from -120 to 120. Using a three-channelled 16-bit they can be mapped to the range 0 - 65535. This means that we lose some precision as we can only represent with a detail of approximately 4 mm. We use a key color to denote “no information” the value 0 (which maps to 32767) since, because of the scan specifications, there can be no sample captured that close.
• Intensity: can be coded without any loss of precision mapping using a single channel 16-bit image since the values range from -2048 to 2047. We use a key color to denote “no information” the value -2048 (which maps to 0).

• \((r, g, b)\): these three values can be coded in a standard 3-channel 8-bit image without any loss of precision. We use a key color to denote “no information” the value 0.

• Depth: this values ranges from 0 to 120. Using a single-channelled 16-bit it can be mapped to the range 0 - 65535. We can represent a precision up to 2 mm . We use a key color to denote “no information” the value 0.

Let \(N\) denote the number of points in the scan we are currently processing and \(M = 2x^2\). The overall cost of this algorithm is \(O(N \log(N))\) to build the kd-tree plus the cost of filling the image layers which is \(O(M \log(N))\) (assuming that the size of the neighbourhood retrieved for each point is a constant). In practice, usually \(N \approx M\) and the constant associated to the second term seems much bigger.

An optimization of this algorithm could consist in a 2 pass algorithm as follows: one pass where each point is annotated in the pixels it influences \(O(N)\) and a second pass computing the actual values \(O(M)\). Nevertheless, later in this document we will give more insight on why this architecture is desirable.

Finally, remark that this algorithm was implemented using the tools available in the Point Cloud Library [20].

4.3 Color Correction

We discussed before that the color of the samples is taken after the scanning process has finished and that this might cause artifacts due to dynamic objects present in the scene. Also, the scan takes multiple pictures and combine them to generate the color of the samples, but these pictures potentially were taken with different illumination conditions. Color cameras usually try to adjust the settings controlling exposure (lens aperture, shutter speed) to the individual lighting conditions of each shot. This means that automatic exposure causes image colors to vary across different shots and it turns out that some of them get easily over-exposed (see Figure 6). Because of this, in the generated panoramas there appear notorious discontinuities that are not present in the intensity values (see Figure 5).

Moreover, the color is not consistent between different scans either, which is not desirable because, at some point, we want to combine them. Consequently, we propose a very basic color correction strategy that has proved to be very effective. Our approach consist in computing the intensity of the color as the mean of the three channels and, then, substituting this by the scan reported intensity:

\[
I = \frac{1}{3}(R + G + B)
\]
Figure 5: Right: Color image. Left: Intensity image. Discontinuities in the intensity of the colors can be appreciated in the color images whereas they are not present in the intensity image. This is due to the way Cyclone computes the color of each point.

\[ R = R \times \frac{\text{scanner intensity}}{I} \]  
\[ G = G \times \frac{\text{scanner intensity}}{I} \]  
\[ B = B \times \frac{\text{scanner intensity}}{I} \]  

The scan reported intensity is a number proportional to the reflected infrared light from the ray used to capture the sample. Therefore it is neither affected by dynamic objects nor sensible to the illumination conditions.

Figure 6: These images illustrate the procedure that Cyclone carries out in order to estimate the color of the sample points.

Using this approach, we theorise that we are able to estimate the albedo of the surfaces. This is very convenient because we can, later on, illuminate the scene using empirical models. In Figure 7 we show a comparison of original and corrected images and how even very saturated and very dark images become very similar between themselves.
4.4 Experiments and Results

It is somehow difficult to compare the storage reduction between the original scans and the new representation because there is not a direct mapping between points contained in the two (in the previous section we saw how there is a mismatch between the theoretical amount of data and the amount of data present in the scans). Therefore what we will compare is the weight of the images saved to a binary dump and to different image formats and this will give us an idea of the achievable reduction rate.

<table>
<thead>
<tr>
<th></th>
<th>Binary</th>
<th>PNG</th>
<th>Saving</th>
<th>PNG + JPG</th>
<th>Saving</th>
</tr>
</thead>
<tbody>
<tr>
<td>Position</td>
<td>32.3 GB</td>
<td>2.1 GB</td>
<td>93.50%</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Intensity</td>
<td>5.4 GB</td>
<td>3.9 GB</td>
<td>27.78%</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Color</td>
<td>8.1 GB</td>
<td>3.0 GB</td>
<td>62.96%</td>
<td>658 MB</td>
<td>91.88%</td>
</tr>
<tr>
<td>Overall</td>
<td>45.8 GB</td>
<td>9 GB</td>
<td>80.35%</td>
<td>6.66 GB</td>
<td>85.46%</td>
</tr>
</tbody>
</table>

Table 4: Compressing ratios obtained by codifying a point cloud as an image.

We have generated panorama representations of resolutions 16384 × 8192 for the 360º scans and 8192 × 8192 for the 180º scans. We have dumped them to binary files, compressed them using png format (lossless) and finally we have also compressed the color layer in jpg format (lossy). In Table 4 we depict the compressing rates obtained.
Moreover, we can further reduce this: before we gave some numbers for a depth layer, this is because we can encode \((x, y, z)\) as \((r, \theta, \psi)\) where \(\theta\) and \(\psi\) are given by the coordinates of a pixel and then we only need to encode \(r\) or the depth information. We also saw that this is beneficial because we can use one more bit for precision. Knowing this, in Table 5 we depict the reduction obtained by this change.

<table>
<thead>
<tr>
<th></th>
<th>Binary</th>
<th>PNG + JPG</th>
<th>Saving</th>
</tr>
</thead>
<tbody>
<tr>
<td>Saving compared to using Position</td>
<td>66.67%</td>
<td>54.63%</td>
<td></td>
</tr>
<tr>
<td>Overall using Depth</td>
<td>24.27 GB</td>
<td>5.51 GB</td>
<td>77.30%</td>
</tr>
<tr>
<td>Overall compared to using Position</td>
<td>47.01%</td>
<td>17.27%</td>
<td></td>
</tr>
</tbody>
</table>

*Table 5: Compressing ratios obtained by codifying depth instead of \((x, y, z)\).*

Finally, the intensity channel is aimed to be used to correct the color channel. If we perform the correction then we can discard the intensity channel. In our experiments the corrected color images increased their size to 720 MB (when coded as jpg), adding the 952.7 MB coming from the depth channel we ended up with 1.673 GB, which represent a 96.35% saving compared to the 45.8 GB of the initial binary representation. If we also account for the estimated saving of converting from the ASCII format to the binary format (64.91%) in the end we obtain a storage saving of approximately a 98.72%.

### 4.5 Other Features

Sometimes the scanner captures dust motes in the air and places points where there should not be. Other times, it captures structures that are there but that are of no interest, such as trees in this case. Some examples of these are shown in Figure 8. If we wanted to erase them using the original representation we would have a very hard and tedious time. One could also represent them in 3D but then selection techniques need to be implemented. Instead, in the proposed panorama representation, doing this kind of operation is trivial, the user only needs to open the images with any editing tool and paint these areas with the key colors.

In general, our representation is equivalent to parameterizing the point cloud, assigning texture coordinates and a texture each point. Because of this, any texturing technique in the literature can be directly applied. Some examples are modifying the color texture, editing it or simply substituting it for a totally different one, using a lower resolution texture, etc. This is similar to what Xianfeng *et al.* do with their Geometry images [21].

Another advantage is that if we only wanted to show the scene from the point of view from where the scan was taken, rendering our panorama representation is as easy as texture a sphere and also much more efficient. Moreover, if then we want to apply screen space techniques, such as screen space ambient occlusion, we only need to perform a single rendering pass. Finally, uniformly down-sampling and up-sampling the cloud becomes trivial since they are equivalent to reducing or increasing the resolution of the panoramas.
4.6 Conclusion

We have proposed an alternative representation for point clouds taken from a single scanning position. Sensors such as Leica ScanStation P20 capture points by sweeping a 3D sphere and uniformly launching rays. We propose mapping the spherical coordinates of each point to a \((x,y)\) position in an image. Different information is stored in different image layers which are packed together according to their consistency. Compression is trivially achieved by using well-known image compression algorithms.

Particularly, we store position, color and intensity in three separate png (loss-less compression) images. The values for each position \((x,y)\) are computed by transforming these into spherical coordinates and searching the nearby points in the cloud. The resulting points are pondered and averaged using bilateral filtering. This process achieves 80.35% saving with respect to a binary representation.

Further compression rates can be achieved if lossy compression techniques are applied to the color (such as jpg coding). Moreover, \((x,y,z)\) position coordinates can be encoded in a single depth layer. Also the purpose of the intensity layer is color correction and, therefore, it can disposed of after fulfilling its role. With these changes a 87.97% storage saving is achieved compared to the binary representation.

Each point in the position image has a corresponding color in the color image. Namely, this is similar to assigning texture coordinates to each point. Thanks to this, our representation allows the user to elegantly apply most of the texturing techniques in the literature to the points, which include level of detail, editing and image-based rendering techniques.
5 Data Registration

5.1 Introduction

The Leica Cyclone [1] software provided us a partial registration of the different scans. Nevertheless, we need an additional registration strategy for 3 reasons:

1. Scans 1 to 22 and 23 to 31 are registered independently and therefore we need to register these two sets.
2. The transformation for the 2nd scan is not coherent with the rest and we need a new one.
3. It turns out that the registration performed by the Cyclone consists only in a translation plus rotation around the $y$ axis. We are concerned about this fact because one degree of freedom might be not enough obtain an accurate register.

Figure 9: Top Left: 23rd scan. Top Right: 28th scan. Bottom Left: 26th scan. Bottom Right: 29th scan. Red stars represent the sensor position, for the 29th it was the center of the dome. The parts of the clouds that overlap most are the arms of the building, which are the furthest from the sensors and, consequently, the most sparse-sampled.

For this task we want to use the version of the Iterative Closest Point algorithm [10] implemented in the Point Cloud Library [20]. However, even if we have discussed an efficient way
to store our point clouds, once decompressed they can not fit entirely in memory. This means that, in order to perform the registry, we must work with sub-sampled clouds.

Once we have performed the sub-sampling and consecutive registry we will perform an evaluation and comparison of the Cyclone registry and the one resulting from our approach. For this, we will use techniques based on the fractal dimension computation through box counting [22].

5.2 Sub-sampling

A very naive approach to sub-sampling is to traverse the list of points and keep a fraction of them. In our case this is not the best approach because the sample density is inversely proportional to the distance from the sensor, that is surfaces that are close to the sensor have been sampled with a higher density than surfaces that are very far from the sensor.

Why is this a problem? The issue here is that the parts of the point clouds that have a biggest overlapping between them are, in general, far from the sensors (this happens higher degree in the ground-level scans but also in the underground ones). We depict this in Figure 9. Uniformly sampling the cloud will reduce the quantity of points everywhere in an equal manner, but what would be desirable is a sampling strategy that uniformizes the sample density.

![Figure 10: Visual comparison of our method versus uniformly sampling.](image)

Given a radius $r$ and a cloud $C$, let the density ($d_p$) at a point $p$ be the number of neighbouring points in $C$ within the radius $r$. Our approach aims to obtain the sub-sampled cloud $C_s$ from the cloud $C$ in such a way that we end up having similar densities for all the points
\( p \in C_s \). For this, we will assign to each point in the original cloud a sampling probability \( P(p) \), let us derive these:

1. Let \( p_m \) be the point with smallest density in the original cloud \( C_o \) within radius \( r \).
2. Let \( d_{p_m} \) be the density of \( p_m \).
3. Let \( P(p_m) \) be the probability assigned to \( p_m \) (we will see how to compute this later).
4. Let \( p \) be an arbitrary point in \( C_o \) with density \( d_p \). We want to compute \( P(p) \) in such a way that the average picked points within the sphere of radius \( r \) centred at \( p_m \) is the same as average picked points in the sphere of radius \( r \) centred at \( p \). This is quite easy to do applying the formula:

\[
P(p) = P(p_m) \times \frac{d_{p_m}}{d_p}
\]  

(24)

It is quite intuitive to see the reason why this works if we assume that the density between points that are relatively close is quite similar and, in general, this happens. Following we will show how to compute \( P(p_m) \):

1. Let \( \rho \) be the fraction of the cloud that we want to keep: \( |C_s| = \rho |C_o| \).
2. Assign an arbitrary value in the range \((0,1)\) to \( P(p_m) \).
3. Compute all the \( P(p) \) using the previously chosen \( P(p_m) \).
4. Note that if \( \sum_{c \in C_o} P(p) \approx |C_s| \) we would have finished since sub-sampling the cloud \( C_o \) with these probabilities would yield a cloud with size \( \approx |C_s| \).
5. In the general case \( \sum_{c \in C_o} P(p) \neq |C_s| \) and, therefore, we must modulate the \( P(p) \). To do this simply apply the formula:

\[
\forall p \in C_o, P'(p) = \frac{P(p) \times |C_s|}{\sum_{p \in C_s} P(p)}
\]  

(25)

Next we will explain how to determine \( r \): \( r \) could perfectly be a user defined based on some knowledge on the density of the point clouds. However, different point cloud have different average point densities, for instance imagine scanning a wall from close or from far, both would yield the same number of points but the inter-point-distance for the former would be much closer than for the latter. Our approach estimates the surface area captured by then scan, then estimates the point density and this, combined with a user-defined parameter, yields the final \( r \).

To estimate the surface area captured by the scan we compute the mean distance sensor-to-point \((md)\) and the minimum and maximum point height \((\min_y \) and \( \max_y \) respectively). All these measure are taken by first discarding 2.5\% of the samples on each extreme. Then we compute the are of a cylindrical section given by the formula:
\[ A = \frac{2\pi \theta}{360} \cdot md \cdot (\text{max}_y - \text{min}_y) \]  

(26)

Figure 11: Left: clouds of points sub-sampled with our method. Right: clouds of points sub-sampled uniformly. Top: these points were close to the sensor, therefore their density is smaller with our method. Bottom: these points were far from the sensor, therefore their density is higher with our method. For our method we employed a radius of 0.024 meters, increasing this should also increase the difference in densities, at the cost of an increased sub-sampling time.

Where the spanned range of \( \theta \) can be either 180° or 360° as explained in previous sections. Next, the surface density can be estimated as \( \frac{|C|}{A} \), the linear density is the square root of this \( (\sqrt{\frac{|C|}{A}}) \) and its inverse \( (\sqrt{\frac{A}{|C|}}) \) gives the average inter-point-distance. Therefore, we compute \( r \)
as \( r = k \sqrt{\frac{A}{|C_s|}} \) with \( k \) being a user defined parameter.

In Figure 10 we show visually, for one of the scans, how using our sub-sampling method the sample density becomes much more uniform everywhere than using a naive uniform sampling. Usually \( d_{pm} = 1 \) because there are always some points which lay very far from the rest. If this happens, ideally the sample density in the sub-sampled cloud should tend to be 1 everywhere, making it a so-called Poisson disk sample within the chosen radius \( r \). In practice, if we take a look at Figure 12 we are very close to attaining it.

![Density previous to Sub-Sampling](image1.png) ![Density after to Sub-Sampling](image2.png)

(a) Point densities before sub-sampling. Min: 1, Avg: 10.79, Max: 7453

(b) Point densities after sub-sampling with our approach. Min: 1, Avg: 1.133, Max: 5

![Density with Uniform Sub-Sampling](image3.png)

(c) Point densities before sub-sampling. Min: 1, Avg: 26.89, Max: 473

Figure 12: Comparative of point densities. Top charts shows the densities for the same set of points, before and after sub-sampling using our method. Bottom chart shows the densities if we employed an uniform sub-sampling strategy to obtain a set point points of equal size. In order to compute these densities the value used for \( r \) was 0.024 meters.
5.3 Registering

We decided to sub-sample the clouds to $\frac{1}{10}$th of the original ones, approximately 90 million points which fit well in memory. After doing this we proceeded to register them following an incremental two-steps approach:

- Our approach is incremental because Iterative Closest Points [10] computes the transformation to match one point cloud with another and, therefore, it can only register two scans at a time. Therefore we took one point cloud as the base one and registered another with this, then took the result of combining both and registered a third one, and so on.

- Our approach has two steps because Iterative Closest Points [10] works very well when the two point clouds are already partially registered. In this case, this algorithm is able to fine tune the register and make the two clouds match almost perfectly. Therefore we had to use an external tool to compute a coarse registering.

- Finally, what worked best was to register the ground-level scans and the underground scans separately and then register the two resulting point clouds together.

For the coarse-grained register we used the Align tool provided by the Meshlab [23] software which can be seen in Figure 13a. In particular we used what is called “Point-based gluing” which consists in letting the user identify at least 4 points that are approximately the same ones in the two input clouds and, then, the software automatically estimates the transformation that makes these points match (Figure 13b).

For the fine-grained register we used the ICP implementation given by the PCL [20]. In particular, we configured the algorithm to work with a point-to-plane metric. This metric takes as input a base cloud with normal estimations at each of its points and a target cloud. Then, the Levenberg Marquardt optimization [24] is used to find the transformation that minimizes the point-to-plane distance between the point correspondences found between the two clouds.

The final result of this procedure is a set of 31 matrices that register all of our scans.

Finally, recall that Cyclone [1] tool gave the ground-level scans and underground scans registered separately. Using a similar approach to the described one we were able to find the missing transformation to register these two clouds. Also the missing transformation for the 2nd scan.

5.4 Voxelization

After the registration we have all the 31 clouds in the same reference system. This means that we can treat it as a single gigantic cloud of 3500 million points, but this is not convenient. Instead, we decided to “voxelize” it and generate smaller clouds representing $1 \ m^3$ each.

The representation explained in Section 4 was only aimed to represent and store one cloud at a time. Now we are dealing with points from different clouds and, therefore, we decided
to store these voxels in plain binary format. Moreover, we added an additional field per point which indicates the index of their original cloud.

Finally note that, in fact, we need to generate two voxelizations, one for each registry.

5.5 Evaluation

At this point we have two sets of 31 transformations, the ones given by Cyclone (plus some fine tuning) and the ones given by our procedure. In this section we will see how we determine which approach is better and give some hints about which might be the reasons behind this.

For starters, in Annex A we present different charts that show the distribution of distances (in meters) between the points registered with one approach or the other. A summary of this charts can be found in Table 6 where we show, for each scan, the difference in translation (its length) and the difference in rotation (offset angle) between registration matrices. In particular,
Table 6: For each scan, difference in translation (length in centimetres) and the difference in rotation (offset angle) between the Cyclone registration matrix and the matrix yielded by our approach.

we compute the offset angle in the following way; let $p$ be an arbitrary point, let $R_1$ be the registration rotation induced by one of the methods and $R_2$ be the other one:

$$p_1 = R_1 \ast p$$  \hspace{1cm} (27)$$

$$p_2 = R_2 \ast p$$  \hspace{1cm} (28)$$

$$p_2 = R_2 \ast R_1^{-1} p_1$$  \hspace{1cm} (29)$$

Now assume we chose $p$ so that $p_1$ is given by an unitary vector from the origin. Then we compute the offset angle $\alpha$ as:

$$\alpha = \cos(p_2 \cdot p_1)$$  \hspace{1cm} (30)$$

There are few observations that can be made from this data:

1. The chart for the 23rd scan is not present. This is because this one was taken as the reference scan for both registers and therefore in both cases the transformation is equal to the identity matrix.

2. The error for the 23rd, 24th and 25th scan is smaller than the error for scans ranging from 26 to 31. This can be explained because our registering approach is incremental, meaning
that the registering error is accumulative and should increase as we get far from the 23rd scan. In the other hand, we suspect that the Cyclone tool has some global optimization step that our approach lacks, which means that the error in this other registry should not be accumulative.

3. Also, the distance distributions between different scans can vary a lot. This is also explained by the difference in registration steps.

4. In general, the differences in distance for the scans ranging from 1 to 22 is much smaller than for the scans ranging from 23 to 31. This is because the underground scans have much more overlapping between them than the ground-level ones, therefore the formers are much easier to register.

Up to this point we have shown that the two registries are enough different to consider comparing them. Following we will explain the method we used in order to compute this comparison:

There exist one method to estimate the fractal dimension of a cloud of points called Box Counting [22]. This consist in:

1. Computing the bounding cube of a cloud of points.
2. Recursively perform a uniform subdivision of the box. That is each time consider units of space (cubes) eight times smaller than the previous ones.
3. While computing the subdivision, annotate which cubes contain any point.
4. Plot the the number of occupied points (in log scale) against the subdivision level.
5. Report the slope of the regression line going through these points.

More particularly, what we want to do is, using the voxelization of the cloud that we previously computed, sub-divide each voxel and compare the number of occupied sub-voxels on one registry with the number of occupied sub-voxels on the other. Intuitively, as can be seen in Figure 14, a smaller number of occupied sub-voxels will indicate a better registry.

Let an active voxel be a voxel, in one of the registries, that contains at least some points. We found afterwards that the number of voxels that were active ones in both registries was very different than the total number of active voxels in on or the other. More specifically, the Cyclone registry had 83869 active voxels, the registry yielded by our approach had 86148, but their intersection was only 77907. This is due to the differences between both registries but does not make the occupation values for these voxels comparable.

Instead, using voxels of $10m^3$ is much better. This way the Cyclone registry has 535 active voxels, our registry has 538 and their intersection is 534 voxels. In conclusion, now we have a much more solid grounding in order to compare both registries. However, again because the difference in registries is significant, directly comparing homologous active voxels does not
Figure 14: The grey line represents the underlying “real surface” and the different colors indicate points coming from different scans. On the one hand, top figure illustrates a scenario where the registry is good and all the points lay in the same line, on the other, bottom figure illustrates a scenario where the registry is worse and points are more dispersed. If we subdivide and count the number of occupied sub-voxels we end up with 12 for the former and 18 for the latter and, therefore, less occupied voxels indicate a better registration quality.

make much sense since we would not be comparing the same set of points. Alternatively, we think that it makes much more sense to compare the distribution of occupied sub-voxels.

We decided to divide each $10m^3$ voxel up to 12 times, yielding 12 occupation values per voxel. After this, for each level we had two populations of 535 and 538 values and we wanted to apply statistic tests in order to determine whether the average occupied number of sub-voxels was the same for both registries or not.

The common t-test assumes that the means of the different samples are normally distributed. By the central limit theorem, means of samples from a population with finite variance approach a normal distribution regardless of the distribution of the population. Sample means are basically normally distributed as long as the sample size is at least 20 or 30 and our population sizes are around 500 therefore this condition is fulfilled. Nevertheless there are two other assumptions: equal population sizes, which we have been that is not fulfilled, and equal variances.

The F test implemented in R [25] was used to test the homoscedasticity (equality of variances) of the two populations for a given subdivision level. For levels ranging from 1 to 4 and 12 there was no statistical proof that the variances of the two populations was different. For levels ranging from 5 to 11 it was proven that the variances were different. In conclusion, it is much better to use the Welch’s t-test which is robust to these conditions.

In Table 7 we can see the results for the Welch’s t-test between the two populations, for
Table 7: Results for the Welch’s t-test comparing the average occupation of sub-voxels.

<table>
<thead>
<tr>
<th>Level of Subdivision</th>
<th>Mean Occupied Voxels</th>
<th>Significative Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cyclone</td>
<td>Our Registry</td>
<td>p-value</td>
</tr>
<tr>
<td>1</td>
<td>4.59</td>
<td>4.61</td>
</tr>
<tr>
<td>2</td>
<td>21.49</td>
<td>21.80</td>
</tr>
<tr>
<td>3</td>
<td>95.60</td>
<td>98.32</td>
</tr>
<tr>
<td>4</td>
<td>408.04</td>
<td>437.15</td>
</tr>
<tr>
<td>5</td>
<td>1684.80</td>
<td>1902.86</td>
</tr>
<tr>
<td>6</td>
<td>6822.96</td>
<td>8235.28</td>
</tr>
<tr>
<td>7</td>
<td>26806.31</td>
<td>34500.32</td>
</tr>
<tr>
<td>8</td>
<td>100081.4</td>
<td>133663.1</td>
</tr>
<tr>
<td>9</td>
<td>340174.4</td>
<td>440893.3</td>
</tr>
<tr>
<td>10</td>
<td>987437.3</td>
<td>1153572.6</td>
</tr>
<tr>
<td>11</td>
<td>2236420</td>
<td>2371066</td>
</tr>
<tr>
<td>12</td>
<td>3650644</td>
<td>3713339</td>
</tr>
</tbody>
</table>

each subdivision level. Overall, the Cyclone registry appears to be better than the one done with our approach. The differences become significative starting from subdivision level 5 ($\approx 31$ cm) which means that previous levels were too coarse. The maximum significance is achieved at level 8 ($\approx 4$ mm) which seems reasonable because at a thinner scale points start to become isolate on their voxels and, in consequence, the numbers between both registries tend to be similar.

There are mainly three reasons which might explain why our registry is worse:

1. Our registry is incremental, there is no global optimization step.

2. The Cyclone registry used marks to perform the coarse grained alignment of the clouds. We had to perform this manually since they were not identifiable on the sub-sampled scans.

3. The Cyclone transformations only consider rotation along the $y$ axis. Our first impression was that this was a drawback of their method. Nonetheless, perhaps their scanners compensate this with accelerometers. If this was the case, any rotation in the $x$ and $z$ axis would induce errors and this is extra knowledge that we did not take into account.

Regardless of this result there are two positive aspects: we have been able to generate a registry which is very similar to the one provided by a commercial tool such as the Leica Cyclone [1] (using just free software tools) and we have been able to complete the partial Cyclone registry.

To conclude this section now we want to estimate how good the Cyclone registry is. For this we have identified a set of voxels which contain points that belong to a wall of the building (Figure 15). If this wall was perfectly planar, theoretically its corresponding fractal dimension, estimated with box counting method, should be $\frac{2}{3}$. This is because every time we subdivide
the space in 8, the number of occupied voxels should quadruplicate itself in average, hence the
\[ \text{slope} = \frac{\log(4)}{\log(8)} = \frac{2}{3}. \]

In practice, this wall is formed by 20 voxels of 1 m³ the fractal dimension of which range
from 0.69 to 0.71 with average of 0.697. But, if we do the same just taking into account the
points from a single scan we obtain fractal dimensions that range from 0.65 to 0.67 with average
0.658 which is much closer to 0.66. In summary, using more scans increases the number of
occupied voxels and, therefore, the Cyclone registry is not perfect either.

5.6 Conclusion

We have proposed a full pipeline to register multiple large 3D scans. Our algorithm has 3
notorious properties:

1. Scalable: it is robust to working with massive point clouds through smart subsampling.
The sampling rate can be adapted to work with any memory budget and we have proven
this by computing a full registration of 31 scans using commodity hardware.

2. Parallelizable: the sub-sampling and normal estimation of each scan can be done in par-
allel if there are enough resources available. The only sequential step of our algorithm is
the incremental registration.

3. Open source: we have implemented the whole pipeline using only open source software
such as Meshlab [23] and the Point Cloud Library [20].

Also, we have proposed a smart strategy to sub-sample point clouds in such a way that the
sample density of the resulting clouds is approximately uniform within a given radius. This

Figure 15: An approximately planar wall. Left: frontal view. Right: side view. Different grey tones correspond
to points coming from different scans. All the scans were registered using the Leica Cyclone [1].
is specially good for registering when the input clouds are large and the major part of the overlapping happens far from the sensor.

Moreover, we have designed a strategy for evaluation and comparison of registries based on fractal dimension estimation by Box Counting [22]. Even if our registry was proven to be a little worst than the one done by a commercial tool such as the Leica Cyclone [1], we were also able to prove that the latter is not perfect either.
6 Normal Estimation

6.1 Introduction

A robust method for estimating the normal at each point in a point cloud is key to many applications such as visualization (in order to perform a coherent back-face culling), splatting, surface reconstruction, etc. We believe that a crucial aspect to this problem is taking into account the noise present in the data. Hoppe et al. [6] compute the normal at each point as the normal to the fitting plane obtained by applying the total least square method to the \( k \)-nearest neighbourhood of the point. This method is robust to the presence of noise due to the inherent low pass filtering. Furthermore, Mitra et al. [7] analyze the noise in the data in order to estimate the optimal neighbourhood size.

Both of these methods perform a least square fitting and, therefore, they assume that the points are normally distributed around the fitted plane. However, when we have a cloud that is the result of combining multiple registered scans, the noise associated to points that are relatively close can greatly differ. The cited methods are not flexible enough because they are unable to take into account that the variance (as a function of the error) for each points might be different and also that the error is directional (mostly in the direction form the sensor to the point). To the knowledge of the authors, there is no method in the literature that takes into account this fact.

![Diagram giving an overview of our method for normal estimation. The gray dashed line represents the underlying surface. The red points were taken from sensor A and the yellow points were taken from sensor B. All the points have an associated interval, that goes in the direction of the ray from the sensor to the point, and their width is inversely proportional to the error of the point.](image)

Figure 16: Diagram giving an overview of our method for normal estimation. The gray dashed line represents the underlying surface. The red points were taken from sensor A and the yellow points were taken from sensor B. All the points have an associated interval, that goes in the direction of the ray from the sensor to the point, and their width is inversely proportional to the error of the point.

Thanks to the information in Table 2 we are able to interpolate the error at each point in our cloud and take this into account to compute the normals. Our approach consists in extending Hoppe et al. [6]’s method to deal with small gaussian intervals instead of points (see Figure 16). Basically, when the scan captures a point there might a be an error in depth which depends on the distance to the point and its reported intensity. We therefore consider that, instead of
points, we are dealing with probability distributions which are gaussians centred at each point and with a variance proportional to the error associated with the point.

6.2 Framework

Given a cloud of 3D points $C$, the aim is to estimate the normal for each point $p \in C$. For a given $p$ we know the position $p_s$ of the sensor $s \in S$ that captured it, its associated error $\sigma_p$ and we can compute its neighbourhood denoted as $N(p)$.

The first step is computing the centroid $c_p$ of $N(p)$ and for this let us consider for each $p' \in N(p)$ the segment centred at $p'$ in the direction $\overrightarrow{p_p'}$ and length $2\sigma_{p'}$. We will compute the $c_p$ as the mean of the properly weighted points along these segments:

$$c_p = \frac{1}{\sum_{p' \in N(p)} W_{p'}} \sum_{p' \in N(p)} W_{p'} \int_{-\sigma_{p'}}^{\sigma_{p'}} N_{p'}(p' + \overrightarrow{p_s p'}) dx$$

(31)

Here $W_p$ is the weight that we want to assign to the segment corresponding to $p$. On the other hand, $N_p(x)$ is the relative weight that we want to assign to each point along the segment and it must satisfy the normalization condition:

$$\int_{-\sigma_p}^{\sigma_p} N_p(x) dx = 1$$

(32)

Solving the integrals yields:

$$c_p = \frac{1}{\sum_{p' \in N(p)} W_{p'}} \sum_{p' \in N(p)} W_{p'} \left( \int_{-\sigma_{p'}}^{\sigma_{p'}} N_{p'}(x) dx + \int_{-\sigma_{p'}}^{\sigma_{p'}} N_{p'}(x) p' dx \right)$$

$$= \frac{1}{\sum_{p' \in N(p)} W_{p'}} \sum_{p' \in N(p)} W_{p'} \left( \int_{-\sigma_{p'}}^{\sigma_{p'}} N_{p'}(x) p' dx + \int_{-\sigma_{p'}}^{\sigma_{p'}} N_{p'}(x) \overrightarrow{p_s p'} dx \right)$$

$$= \frac{1}{\sum_{p' \in N(p)} W_{p'}} \sum_{p' \in N(p)} W_{p'} \left( p' \int_{-\sigma_{p'}}^{\sigma_{p'}} N_{p'}(x) dx + \overrightarrow{p_s p'} \int_{-\sigma_{p'}}^{\sigma_{p'}} N_{p'}(x) dx \right)$$

$$= \frac{1}{\sum_{p' \in N(p)} W_{p'}} \sum_{p' \in N(p)} W_{p'} \left( p' + \overrightarrow{p_s p'} \int_{-\sigma_{p'}}^{\sigma_{p'}} N_{p'}(x) dx \right)$$

(33)

Furthermore, if we assume that $N_p(x)$ is symmetric about the origin, it is not hard to see that the $\int_{-\sigma_p}^{\sigma_p} N_p(x) dx$ term becomes zero, yielding:
\[
c_p = \frac{1}{\sum_{p' \in \mathcal{N}(p)} W_{p'}} \sum_{p' \in \mathcal{N}(p)} W_{p'} p' \tag{34}
\]

If we take into account that the centroid of the cloud of segments is the centroid of the segment's centroids, the previous result makes sense because the centroid of the segment for \( p \) is \( p \) itself if we use the previous constraints on \( \mathcal{N}_p(x) \).

Following, let \( p'_c = p' - c_p, \forall p' \in \mathcal{N}(p) \), the covariance matrix for the centred cloud can be estimated as:

\[
cov = \sum_{p' \in \mathcal{N}(p)} W_{p'}^2 \int_{-\sigma_{p'}}^{\sigma_{p'}} \mathcal{N}_p^2(x)(p'_c + \overline{s_p} p' x)(p'_c + \overline{s_p} p' x)^T dx
= \sum_{p' \in \mathcal{N}(p)} W_{p'}^2 \left( p'_c p'_c^T \int_{-\sigma_{p'}}^{\sigma_{p'}} \mathcal{N}_p^2(x) dx + (\overline{s_p} p'_c \overline{s_p} p'_c)^T \int_{-\sigma_{p'}}^{\sigma_{p'}} \mathcal{N}_p^2(x) x^2 dx + (\overline{s_p} p'_c)^T (\overline{s_p} p'_c) \int_{-\sigma_{p'}}^{\sigma_{p'}} \mathcal{N}_p^2(x) x^2 dx \right)
\tag{35}
\]

Solving the integrals yields:

\[
cov = \sum_{p' \in \mathcal{N}(p)} W_{p'}^2 \left( p'_c p'_c^T \int_{-\sigma_{p'}}^{\sigma_{p'}} \mathcal{N}_p^2(x) dx + (\overline{s_p} p'_c \overline{s_p} p'_c)^T \int_{-\sigma_{p'}}^{\sigma_{p'}} \mathcal{N}_p^2(x) x^2 dx + (\overline{s_p} p'_c)^T (\overline{s_p} p'_c) \int_{-\sigma_{p'}}^{\sigma_{p'}} \mathcal{N}_p^2(x) x^2 dx \right)
\tag{36}
\]

Following, if \( \mathcal{N}_p(x) \) is symmetric about the origin so is \( \mathcal{N}_p^2(x) \). Therefore, it is not hard to see that the \( \int_{-\sigma_{p'}}^{\sigma_{p'}} \mathcal{N}_p^2(x) x^2 dx \) term becomes zero, yielding:

\[
cov = \sum_{p' \in \mathcal{N}(p)} W_{p'}^2 \left( p'_c p'_c^T \int_{-\sigma_{p'}}^{\sigma_{p'}} \mathcal{N}_p^2(x) dx + (\overline{s_p} p'_c \overline{s_p} p'_c)^T \int_{-\sigma_{p'}}^{\sigma_{p'}} \mathcal{N}_p^2(x) x^2 dx \right)
\tag{37}
\]

### 6.3 Segment Weights

At this point we are thinking about assigning a weight to each point/segment that reflects how “trustable” it is. For this, we propose using some gaussian weights with the form:

\[
W_p = e^{-\sigma_p^2 x^2} \tag{38}
\]
First of all, recall that $\sigma_p$ (the error associated to the point) is given in meters but has the magnitude of millimeters. Moreover, $\sigma_p$ can range from 0 to $9 \times 10^{-3}$ and one can check that, if we ignore $k$, the weight for both extremal values is practically 1. Because of this, we introduce a constant $k$ the function of which is to enlarge the error values. The problem is how to pick the adequate value for $k$ and for this we show in Figure 17 how this value affects to the weights computed for different errors.

![Figure 17: Charts showing the resulting weights for different combinations of errors and $k$. Black: error was set to 9mm. Red: error was set to 4.3mm. Purple: error was set to 2.8mm. Blue: error was set to 1.5mm. Green: error was set to 0.4mm. The two vertical bars are placed at $k = 300$ and $k = 1000$.](image)

Basically, choosing a value of $k$ determines the influence of different ranges of errors. On the one hand, for $k$ that are very small it is practically as if all weights were the same. On the other hand, for large values of $k$ the weight assigned to large errors becomes practically null. Judging from Figure 17 one reasonable choice would be $k = 300$ but in the evaluation section we will see that, in practice, it is better to penalise harder large errors.

### 6.3.1 Intuition

In our point cloud there are points coming from 31 different data sets which means that, potentially, we can have regions of the cloud with a lot more noisy points than reliable points. For instance, this can happen if a surface is captured by a relatively close sensor and by many far away ones.

One intuition as to why our approach might work better, in these cases, than simply using uniform weights is that, for a proper value of $k$, weights are set such that very noisy points are almost ignored and the correct normal can be estimated by giving more importance to reliable points. This is illustrated in Figure 18.
Figure 18: Effects of using gaussian weights based on the errors assigned to the points.
6.4 Uniform Weights

The most basic approach is choosing a constant function for the weights $N_p$, in particular:

$$N_p = \frac{1}{2\sigma_p} \quad (39)$$

Given the previously defined framework, all we need to do now is solving the following integrals:

$$\int_{-\sigma_p}^{\sigma_p} N_p^2(x) dx = \int_{-\sigma_p}^{\sigma_p} \left(\frac{1}{2\sigma_p}\right)^2 dx = \int_{-\sigma_p}^{\sigma_p} \frac{1}{4\sigma_p^2} dx \quad (40)$$

$$\int_{-\sigma_{p'}}^{\sigma_{p'}} N_{p'}^2(x)x^2 dx = \int_{-\sigma_{p'}}^{\sigma_{p'}} \left(\frac{1}{2\sigma_{p'}}\right)^2 x^2 dx = \int_{-\sigma_{p'}}^{\sigma_{p'}} \frac{1}{4\sigma_{p'}^2} x^2 dx \quad (41)$$

Which trivially yield:

$$\int_{-\sigma_p}^{\sigma_p} \frac{1}{4\sigma_p^2} dx = \frac{1}{4\sigma_p^2} \int_{-\sigma_p}^{\sigma_p} dx = \frac{1}{4\sigma_p^2} [x]_{-\sigma_p}^{\sigma_p} = \frac{1}{2\sigma_p} \quad (42)$$

$$\int_{-\sigma_{p'}}^{\sigma_{p'}} \frac{1}{4\sigma_{p'}^2} x^2 dx = \frac{1}{4\sigma_{p'}^2} \int_{-\sigma_{p'}}^{\sigma_{p'}} x^2 dx = \frac{1}{4\sigma_{p'}^2} \left[\frac{x^3}{3}\right]_{-\sigma_{p'}}^{\sigma_{p'}} = \frac{\sigma_{p'}}{6} \quad (43)$$

In summary:

$$\text{cov} = \sum_{p' \in \mathcal{N}(p)} W_{p'}^2 \left(\frac{1}{2\sigma_{p'}} p_{p'} c^T + \frac{\sigma_{p'}}{6} (p_{p'}^T p_{p'}) (p_{p'}^T p_{p'})^T\right) \quad (44)$$

6.5 Quadratic Weights

From now on, our aim will be finding a way of giving more importance to points near the center of the segment. The ideal solution would be using gaussian weights but their integral is not trivial and, therefore, our approach will be using functions to approximate them.

To begin with, one basic option is using a quadratic polynomial for the $N_p(x)$:

$$N_p(x) = ax^2 + bx + c \quad (45)$$
Particularly, we impose the conditions that these weights should become 0 at $\pm \sigma_p$:

$$0 = a\sigma^2 - b\sigma + c \quad (46)$$

$$0 = a\sigma^2 + b\sigma + c \quad (47)$$

Also recall the normalization condition:

$$\int_{-\sigma_p}^{\sigma_p} N_p(x) dx = \int_{-\sigma_p}^{\sigma_p} ax^2 + bx + c dx = 1 \quad (48)$$

From 46 and 47 we can conclude that $b = 0$ and $c = -a\sigma^2$. Therefore $N_p = ax^2 - a\sigma^2$.

$$\int_{-\sigma_p}^{\sigma_p} N_p(x) dx = \int_{-\sigma_p}^{\sigma_p} ax^2 - a\sigma^2 dx = a \left[ \frac{x^3}{3} \right]_{-\sigma_p}^{\sigma_p} - a\sigma^2 \left[ x \right]_{-\sigma_p}^{\sigma_p} =$$

$$a \frac{\sigma^3}{3} - a\sigma^2 \sigma_p - (-a \frac{\sigma^3}{3} + a\sigma^2 \sigma_p) =$$

$$a \frac{2\sigma^3}{3} - a2\sigma^3 = -a \frac{4\sigma^3}{3} = 1 \quad (49)$$

$$a = -\frac{3}{4\sigma^3}, \quad c = \frac{3}{4\sigma} \quad (50)$$

In conclusion:

$$N_p(x) = -\frac{3}{4\sigma^3} x^2 + \frac{3}{4\sigma} \quad (51)$$

Given the previously defined framework, all we need to do now is solving the following integrals:

$$\int_{-\sigma_p}^{\sigma_p} N_p^2(x) dx = \int_{-\sigma_p}^{\sigma_p} \left( -\frac{3}{4\sigma^3} x^2 + \frac{3}{4\sigma} \right)^2 dx \quad (52)$$

$$\int_{-\sigma_p}^{\sigma_p} N_p'(x) x^2 dx = \int_{-\sigma_p}^{\sigma_p} \left( -\frac{3}{4\sigma^3} x^2 + \frac{3}{4\sigma} \right)^2 x^2 dx \quad (53)$$

Which yield:
\[ \int_{-\sigma_p}^{\sigma_p} \mathcal{N}_p^2(x) \, dx = \int_{-\sigma_p}^{\sigma_p} \left( -\frac{3x^2}{4\sigma_p^3} + \frac{3}{4\sigma_p} \right)^2 \, dx = \int_{-\sigma_p}^{\sigma_p} \frac{9x^4}{16\sigma_p^6} - \frac{9x^2}{8\sigma_p^4} + \frac{9}{16\sigma_p^2} \, dx = \]

\[ \frac{9}{16\sigma_p^6} \left[ \frac{x^5}{5} \right]_{-\sigma_p}^{\sigma_p} - \frac{9}{8\sigma_p^4} \left[ \frac{x^3}{3} \right]_{-\sigma_p}^{\sigma_p} + \frac{9}{16\sigma_p^2} [x]_{-\sigma_p}^{\sigma_p} = \]

\[ \frac{9}{16\sigma_p^6} \left( \frac{2\sigma_p^7}{7} \right) - \frac{9}{8\sigma_p^4} \left( \frac{2\sigma_p^5}{5} \right) + \frac{9}{16\sigma_p^2} \left( \frac{2\sigma_p^3}{3} \right) = \]

\[ \frac{9\sigma_p^7}{56} - \frac{9\sigma_p^5}{20} + \frac{3\sigma_p^3}{8} = \frac{3\sigma_p^3}{35} \quad (55) \]

In summary:

\[ \text{cov} = \sum_{p' \in \mathcal{N}(p)} W_{p'}^2 \left( \frac{3}{5\sigma_{p'}^2} p_{p'} p_{p'}^T + \frac{3\sigma_{p'}^4}{35} (p_{p'} p_{p'})^T \right) \quad (56) \]

### 6.6 Sinusoidal Weights

A slightly better approximation to a gaussian is the one given by the formula:

\[ \mathcal{N}_p(x) = 1 + \cos \left( \frac{x\pi}{\sigma_p} \right) \quad (57) \]

Note that this formula banishes at \( \pm \sigma_p \), and but we must make it fulfill the normalization condition:
\[
\int_{-\sigma_p}^{\sigma_p} N_p(x)dx = \int_{-\sigma_p}^{\sigma_p} 1 + \cos \left( \frac{x\pi}{\sigma_p} \right) dx = \int_{-\sigma_p}^{\sigma_p} 1 + \int_{-\sigma_p}^{\sigma_p} \cos \left( \frac{x\pi}{\sigma_p} \right) dx = [x]_{-\sigma_p}^{\sigma_p} + \left[ \frac{\sigma_p}{\pi} \sin \frac{x\pi}{\sigma_p} \right]_{-\sigma_p}^{\sigma_p} = 2\sigma_p
\]

(58)

Therefore:

\[
N_p(x) = \frac{1 + \cos \left( \frac{x\pi}{\sigma_p} \right)}{2\sigma}
\]

(59)

Given the previously defined framework, all we need to do now is solving the following integrals:

\[
\int_{-\sigma_p}^{\sigma_p} N_p^2(x)dx = \int_{-\sigma_p}^{\sigma_p} \left( \frac{1 + \cos \left( \frac{x\pi}{\sigma_p} \right)}{2\sigma} \right)^2 dx
\]

(60)

\[
\int_{-\sigma_p'}^{\sigma_p'} N_p^2(x)x^2dx = \int_{-\sigma_p'}^{\sigma_p'} \left( \frac{1 + \cos \left( \frac{x\pi}{\sigma_p} \right)}{2\sigma} \right)^2 x^2 dx
\]

(61)

Which yield:

\[
\int_{-\sigma_p'}^{\sigma_p'} N_p^2(x)dx = \int_{-\sigma_p'}^{\sigma_p'} \left( \frac{1 + \cos \left( \frac{x\pi}{\sigma_p} \right)}{2\sigma} \right)^2 dx = \frac{1}{4\sigma^2} \int_{-\sigma_p'}^{\sigma_p'} \left( 1 + \cos \left( \frac{x\pi}{\sigma_p} \right) \right)^2 dx =
\]

\[
\frac{1}{4\sigma^2} \left( \int_{-\sigma_p'}^{\sigma_p'} dx + 2 \int_{-\sigma_p'}^{\sigma_p'} \cos \left( \frac{x\pi}{\sigma_p} \right) dx + \int_{-\sigma_p'}^{\sigma_p'} \left( \cos \left( \frac{x\pi}{\sigma_p} \right) \right)^2 dx \right) =
\]

\[
\frac{1}{4\sigma^2} \left( [x]_{-\sigma_p'}^{\sigma_p'} + 2 \left[ \frac{\sigma_p}{\pi} \sin \frac{x\pi}{\sigma_p} \right]_{-\sigma_p'}^{\sigma_p'} + \left[ \frac{\sigma_p}{\pi} \sin \frac{x\pi}{\sigma_p} \right]_{-\sigma_p}^{\sigma_p} \right) = \frac{1}{4\sigma^2} \left( 2\sigma_p + \sigma_p \right) = \frac{3}{4\sigma}
\]

(62)
\[
\int_{-\sigma_p}^{\sigma_p} p^2 x^2 \, dx = \int_{-\sigma_p}^{\sigma_p} \left( \frac{1 + \cos \left( \frac{\pi x}{\sigma_p} \right)}{2 \sigma} \right)^2 x^2 \, dx = \frac{1}{4 \sigma^2} \int_{-\sigma_p}^{\sigma_p} \left( \frac{1 + \cos \left( \frac{\pi x}{\sigma_p} \right)}{2 \sigma} \right)^2 x^2 \, dx =
\]

\[
\frac{1}{4 \sigma^2} \left( \int_{-\sigma_p}^{\sigma_p} x^2 \, dx + 2 \int_{-\sigma_p}^{\sigma_p} \cos \left( \frac{\pi x}{\sigma_p} \right) x^2 \, dx + \int_{-\sigma_p}^{\sigma_p} \cos \left( \frac{\pi x}{\sigma_p} \right)^2 x^2 \, dx \right) =
\]

\[
\frac{1}{4 \sigma^2} \left[ \frac{x^3}{3} |_{-\sigma_p}^{\sigma_p} + 2 \left[ \frac{\sigma_p \left( \pi^2 x^2 - 2 \sigma_p^2 \right) \sin \left( \frac{\pi x}{\sigma_p} \right) + 2 \pi \sigma_p x \cos \left( \frac{\pi x}{\sigma_p} \right)}{\pi^3} \right] \right] =
\]

\[
\frac{1}{4 \sigma^2} \left( \frac{2 \sigma_p^3}{3} - \frac{8 \sigma_p^3}{\pi^2} + \frac{2 \pi^2 + 3}{6 \sigma_p^2} \right) = \frac{2 \pi^2 - 15}{8 \pi^2} \sigma_p \approx \frac{3 \sigma_p'}{50} \quad (63)
\]

In summary:

\[
cov = \sum_{p' \in N(p)} W' \left( \frac{3}{4 \sigma_p'} p_c' p_c^T + \frac{2 \pi^2 - 15}{8 \pi^2} \sigma_p' \left( p_s' p' \right) \left( p_s' p' \right)^T \right) \approx \sum_{p' \in N(p)} W' \left( \frac{3}{4 \sigma_p'} p_c' p_c^T + \frac{3 \sigma_p'}{50} \left( p_s' p' \right) \left( p_s' p' \right)^T \right) \quad (64)
\]

### 6.7 Discussion

It turns out that, independently of the function the we used to approximate the guassian function, we always ended up with the similar expressions for the covariance matrix. In fact, they only differed by some constants. In Table 8 we show a summary of these weights.

<table>
<thead>
<tr>
<th></th>
<th>$p_c p_c^T$</th>
<th>$(p_s p)(p_s p)^T$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uniform</td>
<td>0.5 $\frac{1}{\sigma_p}$</td>
<td>0.167 $\sigma_p$</td>
</tr>
<tr>
<td>Quadratic</td>
<td>0.6 $\frac{1}{\sigma_p}$</td>
<td>0.086 $\sigma_p$</td>
</tr>
<tr>
<td>Sinusoidal</td>
<td>0.75 $\frac{1}{\sigma_p}$</td>
<td>0.060 $\sigma_p$</td>
</tr>
</tbody>
</table>

\textbf{Table 8: Summary of the weights obtained by the different approximations to the gaussian distribution.}

Even if we have not been able to find a formal proof, we think that we have enough insight to say that, in general, the covariance matrix, when using the proposed segments instead of points, can be expressed as:
\[
\text{cov} = \sum_{p' \in N(p)} W^2(p') \left( \frac{1}{\sigma_{p'}} p'_c p'^T_c + \beta \sigma_{p'} (\overrightarrow{p_p} \overrightarrow{p'_s})^T \right)
\] (65)

Note that, when \( \text{beta} = 0 \) we are precisely in the case where we completely ignore directionality and only consider weighted points instead of segments. Therefore \( \alpha \) and \( \beta \) are two parameters that modulate the interaction between the term given by just the centered points \((p_c p'^T_c)\) and the term given by the unit segment from the point to the sensor \((\overrightarrow{p_p} \overrightarrow{p'_s})^T\).

Moreover note that the units match: on the one hand, the length of the vector that defines \( p_c \) is of magnitude \( 10^{-3} \)m and, consequently, \( p_c p'^T_c \) is a matrix the elements of which have an order of magnitude of \( 10^{-6} \)m. On the other hand, the vector \( (\overrightarrow{p_p} \overrightarrow{p'_p})^T \) is unitary and, consequently, its length is 1m and \( (\overrightarrow{p_p} \overrightarrow{p'_p})^T \) is a matrix the elements of which have an order of magnitude of 1m. Next, \( \sigma_p \) is about some millimetres hence of order of magnitude \( 10^{-3} \). In conclusion, the coefficients of \( \frac{1}{\sigma_{p'}} p'_c p'^T_c \) and the coefficients of \( \sigma_{p'} (\overrightarrow{p_p} \overrightarrow{p'_s})^T \) have the same order of magnitude.

### 6.8 Evaluation

All the experiments explained in this section where done using a subset of the cloud registered using the Cyclone transformations (Figure 19) and a neighbourhood of at least 50 points. In the case that the radius of the neighbourhood did not exceed 9 mm (which happened in about a 0.1% of the cases), we added to it all the points within this distance. This was done to prevent directionality terms from distorting the clouds too much.

![Figure 19: Cloud of points employed to test the normal estimation algorithms.](image)

(a) Different grey tones indicate points coming from different scans.  
(b) Errors associated to the points visualized in grey-scale. The brighter, the bigger the error.
Following Hoppe et al. [6], once we have computed the covariance matrix for $\mathcal{N}(p)$ we must perform Principal Components Analysis on it and keep as the normal the eigenvector corresponding to the smallest eigenvalue. This is the same as estimating the normal of the tangent plane to the surface, estimated by least-square plane fitting.

(a) Normals estimated using Hoppe et al. [6]'s method e.g. all points are weighted evenly.

(b) Normals estimated using our approach with $k = 1000$, $\alpha = 1$ and $\beta = 0$.

(c) Normals estimated using our approach with $k = 300$, $\alpha = 1$ and $\beta = 0$.

Figure 20: Visual result of the first round of experiments. Normals are coded as color, positive $x$, $y$ and $z$ values are codified as red, green, blue, respectively and negative $x$, $y$ and $z$ are codified as cyan, magenta and green, respectively.

Another way to see this is imagining that we fit an ellipsoid to $\mathcal{N}(p)$ and retrieve the direction in which it is less prominent. Therefore, a measure of the goodness of normal estimating algorithm is to check how flat are these ellipsoids. In fact, this can be easily computed as the
ratio between the smallest eigenvalue value and the second smallest one. One could say that
the smallest this ratio turns to be, the better the algorithm is.

Our first experiment was comparing Hoppe et al. [6]’s approach with our approach taking
\( k = 1000, \alpha = 1 \) and \( \beta = 0 \). The goodness measure described above indicated that 56% of the
normals improved their quality significantly.

The second experiment compared our approach using \( k = 1000, \alpha = 1 \) and \( \beta = 0 \) and out
approach using \( k = 300, \alpha = 1 \) and \( \beta = 0 \). The goodness measure described above indicated that
55% of the normals worsened their quality significantly.

In Figure 20 we can see the results of these two experiments in a visual way. The improve-
ment using our approach with increasing \( k \) is notorious. It seems that using higher values for
\( k \) gives better results and this can be explained because the high \( k \) values makes big errors be
ignored (recall Figure 17).

All that is left is checking what happens when \( \beta \geq 0 \). It turns out that the goodness measure
described in the beginning of this section does not give reliable results any longer. The expla-
nation of this is that taking into account directionality does not usually flatten your ellipsoids.
In fact this is expected because what we are aiming to do with directionality is “smoothing” or
“blurring” the cloud, which intuitively goes against flattening.

Therefore, for directionality, we will rely on visual evaluation of the normals and it turns
out, as can be seen in Figure 21, that it has little effect. In fact, there is such a small number
of normals that significantly change that it is impossible to say which approach is better. We
argue that this happens because we are considering directionality of order of magnitude of
millimetres and, as we have seen in previous sections, register errors are usually of the order of
tens of centimetres.

6.9 Conclusion

We have proposed a generalization of the algorithm described by Hoppe et al. [6] for normal
estimation. Our approach is able to take into account arbitrary errors assigned to points in
an elegant way. We have also proven numerically and visually the improvement in estimated
normals with respect to the original algorithm.

In particular, we developed a framework for normal estimation that consists in a set of
weights \( W_p \) that take into account the error associated to a point and \( N_p \) that take into account
its directionality. These two weights ponder the points when they are used for the covariance
matrix computation, which is then used to extract the normal through PCA.

For \( W_p \) we have chosen gaussian weights parameterized by a parameter \( k \). We have given
insight on the importance and effects of this parameter and on the reasoning to appropriately
choose it. We have also shown its effect visually and have proposed a value for it.

For \( N_p \) we have proposed several approximations to the gaussian form that ease the calcula-
tions done with it. We have solved these calculations and give a general form to the expression that yields the covariance matrix. This form combines the outer product of the centered points \( (p_c p_c^T) \) and the outer product of the unit vector from a point to the sensor \((\vec{p_s} \vec{p})(\vec{p_s} \vec{p})^T\) modulated by the parameters \(\alpha\) and \(\beta\), respectively. We have also explained the rationale behind these parameters.

(a) Normals estimated using our approach with \(k = 1000, \alpha = 1\) and \(\beta = 0\).

(b) Normals estimated using our approach with uniform weights \((k = 300, \alpha = 0.5\) and \(\beta = 0.167)\).

(c) Normals estimated using our approach with quadratic weights \((k = 1000, \alpha = 0.6\) and \(\beta = 0.086)\).

(d) Normals estimated using our approach with sinusoidal weights \((k = 300, \alpha = 0.75\) and \(\beta = 0.060)\).

Figure 21: Visual result of the second round of experiments. Normals are coded the same way as in Figure 20.

In the end, we have provided visual and numerically evaluation of our method. \(W_p\) alone improves significantly the quality of the estimated normals. We have to work more with the \(N_p\) to extract and determine their full potential.
7 Multi-scan Panoramas

7.1 Introduction

This is the section where we will wrap up all the work done in previous ones. Current hardware is not able to perform a real-time rendering of a cloud with over 3500 millions points. Moreover, an easy calculation can be done to determine that they would occupy around 50 GB of main memory which does not fit in most nowadays commodity computers. This means that we need to resort to other strategies if we want to visualize our cloud.

Some image-based techniques, such as panoramas, were specifically designed for this purpose. Chen [18] computes panoramas at different points of a given scene and then uses them to represent it. This model allows the user to perform an omnidirectional inspection of the scene at the points where panoramas were generated and navigation through the scene by hoping between the panoramas.

In this section we are concerned with the generation of panoramas from scenes represented by massive point clouds. In particular, we will use the a similar approach to the one described in Section 4 but extending it to be able to deal with the voxelized registered cloud generated in Section 5. Furthermore, we will use the techniques described in Section 6 to compute robust normals that will be used for “back-face point culling” and directional splatting.

7.2 Generation

In this section we will explain how we approach the problem of generating panoramas for a scene represented by a point cloud, from arbitrary points of view. In particular, we have designed our algorithm in such a way that it is able to work with massive point clouds. Recall that our data set has over 3500 million point clouds.

The procedure is very similar to the one described in Section 4. However, before we were able to hold the whole point cloud in main memory, now this is unfeasible. We will address this by traversing the “layers” of a 1$m^3$ voxelization of our point cloud, which have a small enough quantity of points. In the case this was not enough, we could always compute a finer voxelization.

We have pointed out before that the Leica scans capture the physical scene by throwing rays while uniformly traversing a sphere. This causes that, from the points of view of the sensor, the screen space density of points appears to be uniform. Another difference with respect to the algorithm in Section 4 is that now we are generating panoramas from arbitrary points of view and this means that the screen-space density of points is not uniform, which causes gaps to appear in the images. Because of this, we are forced to implement a splatting strategy.

Figure 22 shows a diagram for the preprocessing needed to prepare the input data for our algorithm. Basically, as explained before, we take the 31 registered scans (taking Cyclone transforms) and voxelize them in voxels representing 1$m^3$. Then we proceed to the normal estima-
Figure 22: Diagram showing how the input clouds are processed to generate the input of the panorama generating algorithm.

1. For a given voxel $V$ generate a cloud with its points and the points of its neighbouring voxels.

2. For each $p \in V$ compute $N(p)$ and estimate its normal. In this case we used the approach described in Section 6 with $k = 1000$, $\alpha = 1$ and $\beta = 0$. Furthermore, we orient the normal consistently towards the sensor that from which $p$ was taken.

3. Store different files the normals for different voxels.

A part from the points and the normals, our algorithm takes 3 additional user defined parameters:

1. The resolution of the panorama. We will call this parameter $x$.

2. The splatting radius in meters. We will call this parameter $s_r$. 
Following we will explain how the proposed algorithm works. Figure 23 shows an overview.

1. Take the voxelization and traverse it by layers. That is, starting from the viewpoint, take the $27 (3^3)$ surrounding voxels for the first iteration, then the 98 voxels surrounding these $(5^3 - 3^3)$ for the second, then the next 218 $(7^3 - 5^3)$, etc. Take into account that some of these might be empty and that the algorithm should finish when the bounding box of the point cloud has been trespassed.

2. Let $p_v$ be the viewpoint from which we are generating the panorama and let $C$ be the cloud formed by all the points in the current layer of the voxelization. For each point $p \in C$ we
know its normal \( n_p \) and therefore we can perform backface culling by discarding it if \((p - p_v) \cdot n_p > 0\).

3. Take \( C \) and map its points to a unit sphere. Build a kd-tree with the mapped points to speed up neighbourhood searches.

4. Begin an iteration traversing each pixel of the image layers. For each pixel, if it already has information then we skip it, if not we compute its corresponding polar coordinate \((\theta, \psi)\) and proceed to 5.

5. Let \( p_q \) be the point resulting from converting the \((\theta, \psi)\) polar coordinates to \((x, y, z)\). The neighbourhood \( \mathcal{N}_{pq} \) is computed by performing a search around \( p_q \) on the mapped cloud using the kd-tree. As search radius we use \( 2\pi \) divided by \( 2x \).

6. If \(|\mathcal{N}_{pq}| = 0\) we proceed to perform the splatting step. For this we first perform a second search with an increased search radius.

7. If still \(|\mathcal{N}_{pq}| = 0\) it means that there was no sample that represents well enough the current position and therefore the pixel is filled with a key color that depends on the layer being filled.

8. If we accepted all the neighbouring points found in the second search, we would be generating splats aligned with the image. However we want splat oriental by the normal. Therefore, we discard all the points that do not fulfil the following criteria:

   (a) Recall that each \( p \in \mathcal{N}_{pq} \) is a point projected in the unit sphere. The same goes for \( p_q \).

   (b) For a given \( p \in \mathcal{N}_{pq} \), the splat generated by \( p \) lays in the plane defined by \( p \) and \( n_p \). Let the projection of \( p_q \) over this plane be \( \text{proj}_{pq} \). \( \text{proj}_{pq} \) represents the point in the splat that is candidate to contribute to the information of \( p_q \).

   (c) Let \( r_{np} \) be the distance between \( p \) and \( \text{proj}_{pq} \). For \( \text{proj}_{pq} \) to be a valid point in the splat generated by \( p \) it must be within the splatting radius, that is \( r_{np} \leq s_r \).

   (d) Let \( r_p \) be the distance between \( p_q \) and \( \text{proj}_{pq} \). For \( \text{proj}_{pq} \) to be a valid a point that influences \( r_p \) it must be within the original search radius, that is \( r_p < \text{search radius} \).

   A the end of this checking, we redefine \( \mathcal{N}_{pq} \) as the set of \( \text{proj}_{pq} \) for which \( p \) fulfil the criteria. If no point fulfils the criteria then we do the same as in step 7.

9. In this case \(|\mathcal{N}_{pq}| \geq 1\) we average all the retrieved points using something similar to a bilateral filtering. For this we first select the sample point which is closest to the view point and call it candidate \( p_c \).

10. Let \( d(p_1, p_2) \) denote the distance between point \( p_1 \) and point \( p_2 \). We compute the weight of each \( p \in \mathcal{N}_{pq} \) as:

\[
w_p = e^{-d(p, p_c)^2} * e^{-d(p, p_q)^2} \tag{66}\]

   (a) The order of magnitude \( d(p, p_c) \) and \( d(p, p_q) \) is of millimetres but they are measured in meters. If we used this value (\( \approx 10^{-3} \)) the squaring and exponentiation would bring them close to 0. Therefore we use \( k = 1000 \) avoid this.
(b) $e^{-(d(p,q) + k)^2}$ penalises points that are very far from the query point $p_q$.

(c) $e^{-(d(p,p_c) + k)^2}$ penalises points that are very far from the candidate point $p_c$.

11. Once we have finished traversing one layer of the voxelization we go for the next one. Instead of performing an explicit depth test, this front-to-back ordering already gives priority to points that are close to the viewpoint.

12. Different layers are saved as different images. In particular we decided to store $(x,y,z)$ one image, depth to another image, $(r,g,b)$ to another and the normal to another.

For color, position and depth we use the same codifications as the ones explained in Section 4. In particular we also correct color the same way as we have already explained. For normals its $(x,y,z)$ values range from -1 to 1. We use a three-channelled 16-bit and map the three values to the range 0 - 65535. The key color to denote “no information” is the value 0 (which maps to 32767).

Let $N$ denote the number of points in the scan we are currently processing and $M = 2x^2$. Let’s assume that we break the problem in $k$ layers and that we process $\frac{N}{k}$ points in each iteration. The cost of one iteration is $O(\frac{N}{k} \log(\frac{N}{k}))$ to build the kd-tree plus the cost of filling the layers which is $O(M \log(\frac{N}{k}))$ (assuming that the size of the neighbourhood retrieved for each point is a constant). In total, the worst-case cost of the algorithm is $O(N \log(\frac{N}{k})) + kO(M \log(\frac{N}{k}))$.

In this case $N$ is much bigger than $M$. Also, in the best case only one search is performed by pixel (because we skip computations when there is already information present for the pixel) bringing down the cost to $O(N \log(\frac{N}{k}))+O(M \log(\frac{N}{k}))$. Moreover, if $k$ is big enough then $\log(\frac{N}{k})$ becomes negligible.

### 7.3 Visualization

We have developed a small Qt-based application for visuzalization of the generated panoramas. It works both for single-scan panoramas (Section 4) and arbitrary point of view panoramas (Section 7). The application basically renders a sphere, centred at the viewpoint, textures it using the color panorama image and enables omnidirectional inspection of the scene from the viewpoint.

Recall that in Section 4.3 we claimed that correcting the color of panoramas yields a result that approximates the albedo of the scene. Because of this, if we have additional depth and normal information (which can also be added in the form of images) we can illuminate the scene in real time. In particular we have implemented direct lighting and screen space ambient occlusion.

In Figure 24 we can see multiple captures of the panorama visualizing tool. In particular, we show a feature that displays the result of the screen space ambient occlusion in greyscale and multiple illumination settings (basically, moving the direct light around the scene).
7.4 Discussion

We consider that the more remarkable issues with our approach are related to splatting. We have pointed out that splatting is needed because the screen-space point density is approximately uniform only from the point of view of the sensors. Therefore, from an arbitrary viewpoint usually gaps appear and they must be filled. However, this causes three main problems:

1. The splatting radius sometimes is not enough to fill large gaps.
2. Sometimes there are gaps present in the real scene that are filled by splatting.
3. Because of the layered approach, splats fill the frontier between layers.

(3) is easily amendable if when processing layer $l$ we overwrite the splats generated in layer $l-1$. However (1) and (2) are inherent to the data and much harder to address. One option would be computing a value (0/1) per point that would indicate whether it was surrounded by other points, in its original scan, or not. On the one hand, 0 would mean “not surrounded” and it would indicate that the associated point should not generate splats. On the other hand, 1 would mean “surrounded” and it would indicate that the associated point should generate splats.
To better illustrate the problems with splatting, we have generated an arbitrary-viewpoint panorama from the viewpoint of one of the scans (Figure 25). This means that the screen space density of points should be uniform everywhere and, consequently, there should not be splats. However, there are splat points caused by errors of type (1) and (2).

Another important issue in the color matching between points coming from different scans. Even if the approach explained in Section 4.3 homogenises the colors notoriously, a global optimization step is still needed for them to perfectly match.

![Figure 25: Left: panorama generated from the viewpoint of a sensor. Right: Splat points in white. Splits next to columns or railings are of type (2) whereas splat forming concentric lines are of type (1).](image)

### 7.5 Conclusion

We have proposed an algorithm for creating arbitrary-viewpoint, multi-scan panoramas. We take a set of registered point clouds and voxelize it while computing normals for their points. For normal estimation we employ the smart algorithm proposed in section 6. The voxelization is traversed by layers in a front-to-back manner. At each step we consider the points in a single layer and map them into an image using their spherical coordinates. An elegant splatting strategy has been implemented in order to fill the gaps caused by regions with a low point density.

Different information is stored in different image layers which are packed together according to their consistency. The representation of the panorama is very similar as the one explained in Section 4. Additionally we also store an extra image for normals. An example panorama can be seen in Figure 26.

We have also implemented a visualization tool that enables omnidirectional inspection of a panorama. The applied color correction approximates the albedo of the scene and, therefore, we have implemented two illumination strategies: direct lighting and screen space ambient occlusion.

Finally, we have pointed out and analysed the issues caused by splatting. We have also suggested possible ways to address these issues although further research is needed.
(a) Color layer.
(b) Position layer.
(c) Depth layer.
(d) Normal layer.
(e) Different grey values indicate different sensors.

Figure 26: Different layers of a panorama generate form a point of view over the building.
8 Future Work

Throughout this project we have worked on many different topics. From the point of view of cloud representation there are mainly two lines of future work:

1. Comparing our image-based approach to entropy encoding and check which yields better space savings.

2. Extending the approach to arbitrary points clouds. Note that the spherical mapping would not longer work, therefore this task involves basically finding a mapping from arbitrary points to the texture such that the spatial coherence is maximized.

Basically these are two ways of giving more versatility to our methods, namely comparing it to already existing methods and generalizing it to work with other kinds of input.

On the field of registration perhaps is where there is more room for improvement. Many registration algorithms have been designed for scans with large overlaps and whose points are almost at the same distance from the sensors (for instance, this is the case of scans of statues). We have seen that our scans of the Mercat de Sant Antoni do not fulfil this criterion and, consequently, we have to find which tunings work best for our case. Nevertheless, we have a robust registration comparison method which will be useful to evaluate the to evaluate further approaches.

Regarding normal estimation we have already given few hints on what should future work be focused on. On the one hand, we need to further determine and analyze the effects of the directionality of the error. On the other hand, we must see how to incorporate the results of this analysis in the normal computation in order to improve their quality.

Finally, with respect to the arbitrary-viewpoint multi-scan panoramas we must address the problem of splatting. Some hints have been already given in this direction but another totally different approach would be resampling the point cloud. We think that we can develop an algorithm that is similar to the Moving Least Squares[8] but that uses our robust normals to generate a uniform sampling of the cloud. Then, the resampled cloud could be used either for reconstructing the surface or to generate our better quality panoramas.

Also, another line of work related these panoramas is directly doing so from the different scans also represented as panoramas. As of now, the input for our algorithm is a voxelization of the different aligned scans. However, ideally we would like to directly use the proposed image-based representation. For this, we imagine performing something similar to what is done for relief impostors, namely, given an arbitrary ray from the viewpoint, perform a search in the different scans represented as depth images and retrieve the values in the intersections. Sadly, we have had no time experiment with this but it is one of the most interesting ideas for our future work.
9 References


69
A Registration Comparison

Figure 27: Distributions of distances between points registered with the Cyclone transformations and with the transformations resulting from our registering approach.
Figure 28: Distributions of distances between points registered with the Cyclone transformations and with the transformations resulting from our registering approach.
Figure 29: Distributions of distances between points registered with the Cyclone transformations and with the transformations resulting from our registering approach.
Figure 30: Distributions of distances between points registered with the Cyclone transformations and with the transformations resulting from our registering approach.
Figure 31: Distributions of distances between points registered with the Cyclone transformations and with the transformations resulting from our registering approach.
Leica ScanStation P20
Industry’s Best Performing Ultra-High Speed Scanner

Unprecedented performance in ultra-high speed laser scanning

Productivity and Accuracy
An innovative combination of advanced time-of-flight range measurement plus modern Waveform Digitizing (WFD) technology enables the compact Leica ScanStation P20 to achieve ultra-high scan speeds and low-noise performance at extended range (to 120 m). Together with high-accuracy angular measurements and survey-grade tilt compensation, Leica ScanStation P20 delivers unprecedented ultra-high speed scan data quality for as-built and scene surveys.

Scan up to 1 million points per second
Leica ScanStation P20 is the ideal instrument when very short time windows are available for capturing High-Definition Survey™ data or when ultra-high density, full dome scan data is needed for client deliverables.

Unmatched environmental capabilities
Developed and manufactured by Leica Geosystems, Leica ScanStation P20 lets users apply ultra-high speed scanning in operating temperatures ranging from –20° C to +50° C. Moreover, with an Ingress Protection rating of IP54 and a an eye-safe laser class 1 rating, users can reap the benefits of ultra-high speed scanning for even more sites and projects.

“Check & Adjust” for added confidence
Leica ScanStation P20 is the first laser scanner to feature a valuable “Check & Adjust” capability. Instead of sending the instrument to a service center, users can electronically check the accuracy of their ScanStation P20 themselves and automatically adjust instrument parameters to ensure the highest level of performance.

- when it has to be right
# Leica ScanStation P20

## Product Specifications

### General

<table>
<thead>
<tr>
<th>Instrument type</th>
<th>Compact, ultra-high speed pulsed laser scanner with survey grade accuracy, range and field-of-view, integrated camera and laser plummet</th>
</tr>
</thead>
<tbody>
<tr>
<td>User interface</td>
<td>Onboard control, notebook or tablet PC, PDA</td>
</tr>
<tr>
<td>Data storage</td>
<td>Integrated solid-state drive (SSD) or external USB flash drive</td>
</tr>
<tr>
<td>Camera</td>
<td>Auto-adjusting, integrated high-resolution digital camera with zoom video</td>
</tr>
</tbody>
</table>

### System Performance

<table>
<thead>
<tr>
<th>Accuracy of single measurement</th>
<th>3 mm at 50 m; 6 mm at 100 m; 11 mm at 200 m</th>
</tr>
</thead>
<tbody>
<tr>
<td>Angular accuracy</td>
<td>8° horizontal; 8° vertical</td>
</tr>
<tr>
<td>Target acquisition</td>
<td>2 mm standard deviation up to 50 m</td>
</tr>
<tr>
<td>Dual-axis compensator</td>
<td>Selectable on/off, resolution 1°, dynamic range +/- 5°, accuracy 1.5&quot;</td>
</tr>
</tbody>
</table>

### Laser Scanning and Imaging System

| Type                           | Ultra-high speed time-of-flight enhanced by Waveform Digitizing (WFD) technology                                               |
| Wavelength                     | 808 nm (invisible) / 658 (visible)                                                                                           |
| Laser class                    | 1 in accordance with ECE60825:2014                                                                                            |
| Beam divergence                | 0.2 mrad                                                                                                                      |
| Beam diameter at front window  | 2.8 mm                                                                                                                        |
| Range                          | Up to 120 m; 18% reflectivity (minimum range 0.4 m)                                                                            |
| Scan rate                      | Up to 1,000,000 points/s                                                                                                      |
| Range noise**                  | Range Black (100%) / Gray (28%) / White (100%)                                                                               |
|                                | 10 m 0.8 mm rms / 0.5 mm rms / 0.4 mm rms                                                                                  |
|                                | 25 m 1.0 mm rms / 0.6 mm rms / 0.5 mm rms                                                                                  |
|                                | 50 m 2.8 mm rms / 1.1 mm rms / 0.7 mm rms                                                                                  |
|                                | 100 m 9.0 mm rms / 4.3 mm rms / 1.5 mm rms                                                                               |
| Scan time and resolution        | 7 pre-set point spacings (mm at 10 m)                                                                                         |
| (kHz/mm:ss)                    | Quality level                                                                                                               |
|                                | 1 00.20 00.20 00.28 00.33 00.33 0.53 0.53 0.43                                                                            |
|                                | 25 00.33 00.33 00.33 0.53 0.53 0.43                                                                            |
|                                | 12.5 00.58 01.44 03.24 06.46                                                                                               |
|                                | 6.3 01.49 03.25 06.46 13.30                                                                                                 |
|                                | 3.1 03.30 06.47 13.30 26.59                                                                                                 |
|                                | 1.6 13.30 27.04 54.07                                                                                                       |
|                                | 0.8 54.07 14.83                                                                                                              |

### Field-of-View

| Horizontal                     | 360°                                                                                                                         |
| Vertical                       | 270°                                                                                                                         |
| Aiming/Sighting               | Parallax-free, integrated zoom video                                                                                         |
| Scanning optics                | Vertically rotating mirror on horizontally rotating base Up to 50 Hz with internal battery                               |
| Data storage capacity          | 356 GB onboard solid-state drive (SSD) or external USB device                                                               |
| Communications                 | Gigabit Ethernet or integrated Wireless LAN                                                                                |
| Imaging                        | 5 megapixels per each 17° x 17° colour image; streaming video with zoom; auto-adjusts to ambient lighting                   |
| Onboard display                | Touchscreen control with styus, full color VGA graphic display (460 x 480 pixels)                                          |
| Level indicator                | External bubble; electronic bubble in onboard software                                                                       |
| Data transfer                  | Ethernet, WLAN or USB 2.0 device                                                                                            |
| Laser plummet                  | Laser class 1 (ECE60825:2014)                                                                                               |
|                                | Centering accuracy: 1.5 mm at 1.5 m                                                                                          |
|                                | Laser dot diameter: 2.5 mm at 1.5 m                                                                                          |
|                                | Selectable ON/OFF                                                                                                           |

### Electrical

| Power supply                   | 24 V DC, 100 – 240 V AC                                                                                                    |
| Power consumption              | 40 W typical                                                                                                               |
| Battery type                   | Internal: Li-Ion; External: Li-Ion                                                                                         |
| Power ports                    | Internal: 2; External: 1 (simultaneous use, hot swappable)                                                                   |
| Duration                       | Internal > 7 h (2 batteries), External > 8.5 h (room temp.)                                                                 |

### Environmental

| Operating temperature          | -20° C to +50° C / -4° F to 122° F                                                                                         |
| Storage temperature            | -40° C to +70° C / -40° F to 158° F                                                                                         |
| Lighting                       | Fully operational between bright sunlight and complete darkness                                                              |
| Humidity                       | Non-condensing                                                                                                             |
| Dust/Humidity                  | IP54 (IEC 60529)                                                                                                           |

### Physical

| Scanner                        | Dimensions (D x W x H)                                                                                                     |
|                                | 238 mm x 358 mm x 395 mm / 9.4” x 14.1” x 15.6”                                                                        |
| Weight                         | 11.9 kg / 26.2 lbs, nominal (w/o batteries)                                                                               |
| Battery (internal)             | Dimensions (D x W x H)                                                                                                     |
|                                | 40 mm x 72 mm x 77 mm / 1.6” x 2.8” x 3.0”                                                                                   |
| Weight                         | 0.4 kg / 0.9 lbs                                                                                                           |
| Battery (external)             | Dimensions (D x W x H)                                                                                                     |
|                                | 95 mm x 248 mm x 60 mm / 3.7” x 9.8” x 2.4”                                                                                   |
| Weight                         | 1.9 kg / 4.2 lbs                                                                                                           |
| AC Power Supply                | Dimensions (D x W x H)                                                                                                     |
|                                | 170 mm x 85 mm x 42.5 mm / 6.6” x 3.3” x 1.6”                                                                                   |
| Weight                         | 0.86 kg / 1.9 lbs                                                                                                           |
| Mounting                       | Upright or upside down                                                                                                    |

### Additional Accessories & Services

- B/WV scan targets and target accessories
- Range of Customer Care Products (CCPs) that include Support, Hardware & Software maintenance and Extended warranty.
- External battery with changing station, AC power supply and power cable
- Professional charger for internal batteries
- AC power supply for scanner
- Tripod and tripod star
- Upside down mounting adapter

### Control Options

- Full colour touchscreen for onboard scan control
- Remote control: Leica CS10/CS15 controller or any other remote desktop capable device, including iPad, iPhone and other SmartPhones.

### Ordering Information

Contact your local Leica Geosystems representative or an authorized Leica Geosystems dealer.

All specifications are subject to change without notice.

All accuracy specifications are one sigma unless otherwise noted.

** Algorithms fit to plane B&W targets

- Detailed explanation on request

Scanner: Laser class 1 in accordance with ECE60825:2014

Leica Geosystems is a trademark of Leica Geosystems AG.

Illustrations, descriptions and technical data are not binding. All rights reserved. Printed in Switzerland – Copyright Leica Geosystems AG, Heerbrugg, Switzerland, 2013. 7956/5en-05 – 07/14 – galdeca