COCKROACH: AN OPEN-SOURCE TOOL FOR POINT CLOUD PROCESSING IN CAD

ANDREA SETTIMI[1], PETRAS VESTARTAS[2] and JULIEN GAMERRO[3] and YVES WEINAND[4]
[1,2,3,4]*École Polytechnique Fédérale de Lausanne (EPFL).*
[1]*andrea.settimi@epfl.ch, 0000-0001-5020-7331*
[2]*petras.vestartas@epfl.ch, 0000-0002-4428-1110*
[3]*julien.gamerro@epfl.ch, 0000-0001-7802-5345*
[4]*yves.weinand@epfl.ch, 0000-0002-8088-6504*
[1,2]*These authors have equally contributed to the research.*

Abstract. In the architecture, engineering and construction (AEC) sector, the use of point cloud data is not a novelty. Usually employed to retrieve data for inspecting construction sites or retrofitting pre-existing buildings, sensors like LiDAR cameras have been known to practitioners such as architects and engineers for a while now. In recent years, the growing interest in 3D data acquisition for autonomous vehicles, robotic and extended reality (XR) applications has brought to the market new compact, performant, and more accessible hardware leveraging different technologies able to provide low-cost sensing systems. Nevertheless, point clouds obtained from such sensors must be processed to extract valuable data for any design or fabrication application. Unfortunately, most advanced point cloud processing tools are written in low-level languages and are hardly accessible to the average designer or maker. Therefore, we present Cockroach: a link between computer-aided design (CAD) modeling software and low-level point cloud processing libraries. The main objective is an adaptation to C# .NET via Grasshopper visual scripting interface and C++ single-line commands in native Rhinoceros workspaces. Cockroach has proved to be a handy design tool in integrating building components with unpredictable geometries such as raw wood or mineral scraps into new design and industrial fabrication processes.

Keywords. Computer-vision; Point-clouds; Data-processing; 3D modeling; CAD interface; Open-source tools; Quality education; Industry innovation and infrastructure; SDG 9.

1. Introduction

In recent years, computer vision hardware has become increasingly accessible (stereo cameras, LiDAR, and AI-based sensors), allowing rapid data acquisition to model the world around us from small-scale objects to large-scale buildings. Given the sparse character of available point cloud processing libraries, which contain a different set of

POST-CARBON, Proceedings of the 27th International Conference of the Association for Computer-Aided Architectural Design Research in Asia (CAADRIA) 2022, Volume 2, 325-334. © 2022 and published by the Association for Computer-Aided Architectural Design Research in Asia (CAADRIA), Hong Kong.

specific functions with other data structures or method calls, development can frequently become cumbersome when adding, compiling, referencing, and converting code among these repositories. In addition, when processing point clouds, the programming language C++ often results as an indispensable language for performance issues. Unfortunately, average designers lack knowledge and time to post-process point clouds. To address these challenges, we propose Cockroach: a point cloud post-processing open-source tool based on C++ (Vestartas and Settimi, 2020). Cockroach acts like an umbrella library able to regroup, compile and make communication possible among the many libraries for point cloud processing, currently Open3D, Cilantro, CGAL, and PCL (Figure 1). The majority of the code is wrapped around C# .NET, specifically tailored to be integrated into the Rhinoceros 3D and Grasshopper environment. The proposed processing tools have proven to be useful in research applications dealing with irregular geometries such as architectural modeling and structural analysis. The current document aims to provide an overview of Cockroach's most recent point cloud processing applications applied to digital fabrication and design.

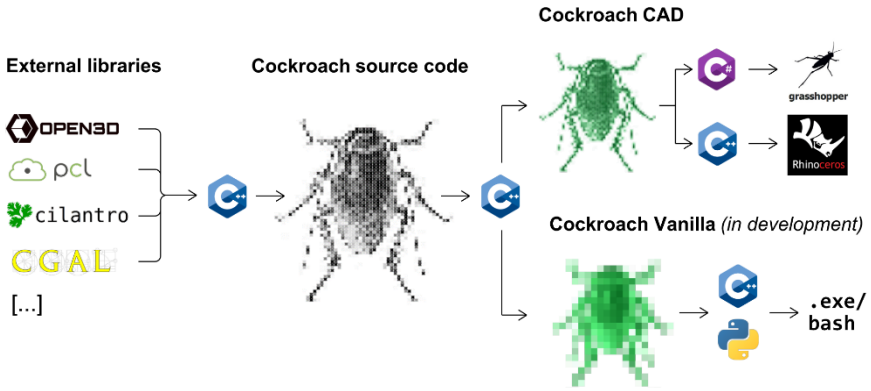

Figure 1: Cockroach architecture, dependencies, and current development branches.

2. State-of-the-art

2.1. OPEN-SOURCE INSIDE CAD ENVIRONMENT

The following point cloud processing software review is based on open-source applications targeting either standalone or hybrid CAD and visual programming interfaces such as Rhinoceros 3D. Volvox (Zwierzycki et al., 2016) was one of the first tools available to post-process heavy point cloud data to link the scanning methods with architectural and engineering design. Beforehand, this data was not accessible to designers due to the overly complicated workflows of closed commercial applications required to import and export geometry. The DURAARK project (Tamke et al., 2016) revealed five critical steps of point-cloud processing: 1) retrieve data, 2) allocation, 3) analysis and verification, 4) focus and abstraction and 5) design. The third and fourth points pose a technical problem when higher-level computer languages, e.g. C#, VB,

IronPython, are insufficient to post-process point clouds. The lower-level languages result in a faster execution time, memory management and, more importantly, are used in many research fields associated with open-source point cloud processing tools. Therefore, it is more accessible for integration and access to existing development teams. Point Cloud Components (Ervine and Girot, 2014) also addresses the problem of point cloud manipulation using a Rhino and Grasshopper interface. One possible reason why these tools are not commonly used and developed is that point cloud processing remains a daunting task, as the available tools are more akin to quantitative engineering analysis than to interactive design. The developed methods are limited to cropping, merging, subsampling, file transfer, and Delaunay meshing. Lastly, small-scale open-source projects, such as Tarsier (Newnham and Gwyllim, 2016), can mainly be used for visualization, while most users apply these methods for analysis, reverse-engineering 3D modeling, and visualization. The remaining question is how these tools can be used for fabrication or assembly, especially for relatively small and medium scale scans. Methods like registration, point-set alignment, abstraction to geometric primitives, and clustering techniques are missing features in the reviewed plug-ins.

2.2. OPEN-SOURCE OUTSIDE CAD ENVIRONMENT

Aside from CAD environments, there is a vast literature of free-to-use point cloud processing software written and used in less accessible languages and less interacting interfaces than a modeling environment, with few exceptions given.

Point Cloud Library (PCL) (Rusu and Cousins, 2011) is the most widely known open-source software for point cloud processing. The library has been employed in numerous research works, especially in autonomous robotic vehicles. It integrates several state-of-the-art algorithms, including filtering, feature estimation, surface reconstruction, registration, model fitting, and segmentation. PCL is implemented in C++, and it is organized in separately compilable modules which can run on Linux, macOS, Windows, and Android. Although extremely powerful, PCL presents many challenges (e.g., compiling issues, C++ intrinsic code complexity), making it hardly accessible for the broader public of developers, not to mention average designers. Although hard to maintain and fragmented in various projects, multiple wrappers written in high-level languages like Python exist (Caron et al., 2020). Open3D (Zhou et al., 2018) is the most recent point cloud processing library. Natively written in C++, it offers a lean, well-documented, and modern Python application programming interface (API). To this day, this represents the most maintained library for point cloud processing. In its recent release (0.13.0), the library features advanced GPU acceleration, real-time 3D reconstruction pipelines, and a web visualizer. Cilantro (Zampogiannis et al., 2018) is another lean and fast C++ library dealing with point cloud data. Besides more ordinary functionalities (I/O utilities, resembling algorithms, normal estimation, convex hulling, etc.), it implements multiple clustering techniques such as connected-component, mean-shift, general dimension k-means, and spectral clustering techniques. There are several open-source point cloud processing software that tends to be mainly used with desktop graphical user interface (GUI). Cloud Compare (Girardeau-Montat, 2020) is an open-source library particularly famous thanks to its well-designed GUI and robust functions to manipulate and register clouds. It also features a valuable plug-in system for contributors to deploy their work on self-

contained modules. In this GUI category, MeshLab (Cignoni et al. 2008) offers a more limited set of point cloud tools but a very user-friendly interface. Despite the high level of user accessibility for GUIs, such interfaces often lack tools to automate processing tasks. Nonetheless, Geometry-central (Sharp et al., 2019a), a mesh library with several point-set processing possibilities visualized by Polyscope GUI (Sharp et al., 2019b), provides parametrization of point-cloud. Point Data Abstraction Library (PDAL) (Bell et al. 2020) also proposes a proper level of automation with a slightly lower-end interface. Thanks to its JSON layout, this library offers the possibility to stage multiple processing operations and automatically execute them on different point clouds. Although its very core is written in C++, PDAL provides various interfaces to meet with the different skill levels of its user base: JSON, command line from the terminal, Python, and native C++ API. A more extensive geometric library like CGAL (The CGAL Project v. 5.3, 2021) also provides packages with point cloud data structure for essential processing functions such as normal estimation, up/down-sampling, smoothing, but also more advanced ones, such as point cloud re-structuring, multiple registration techniques, polygonal surface reconstruction from a point cloud, classification methods from unordered point clouds, and a shape detection algorithm for point clouds. In particular for registration functions, CGAL implements them from two open-source libraries: OpenGR (Mellado, 2018) for global alignment and Libpointmatcher (Pomerlau et al., 2013) for the Iterative Closest Point (ICP) algorithm. Registration methods for aligning point cloud fragments are illustrated and further discussed in the following sections.

3. Application of the Point-cloud Processing Library Cockroach

Two applications from the Cockroach library are detailed focusing on two key methods that were previously missing in point-cloud processing applications for CAD users: registration and clustering. The registration method takes an example of a tree fork truss fabrication and clustering is explained using an experiment of a stone wall assembly. Nonetheless, all functions can be retrieved in the library repository (Vestartas and Settimi, 2020): 1) crop and cut by box, polyline, mesh or plane, 2) normal estimation, uniform orientation and visualization; 3) cleaning and downsampling techniques like random point removal and voxelization, 4) meshing methods such as Poisson and ball-pivoting surface reconstruction, mesh repairing functions, skeletonization and mesh boolean operations, and 5) several utility functions like point cloud serialization, merging, coloring, properties and normal display.

3.1. REGISTRATION METHODS

Point cloud registration consists in finding a spatial transformation such as scaling, rotation, and translation that aligns two point cloud data sets. Cockroach implements two typical alignment algorithms into one method called Recursive Registration (RR), including 1) global Random Sample Consensus (RANSAC) registration and 2) local-global ICP. First, point clouds (Figure 2A) are downsampled to reduce computation time (Figure 2B). Second, point-cloud normal and geometric features are estimated. Third, distance, edge length, and angle correspondence are computed. Fourth, the previous inputs are given to the RANSAC algorithm to align point clouds globally

(Figure 2C). Fifth, registration is refined using local ICP point-to-plane registration (Figure 2, D). The two transformations are applied to the original high-resolution point clouds (Figure 2E). The five steps are repeated several times until the desired fitting is achieved according to the user's tolerance. The combined methods are illustrated in the tree fork example in Figure 2, showing the registration of two point cloud fragments belonging to the same object.

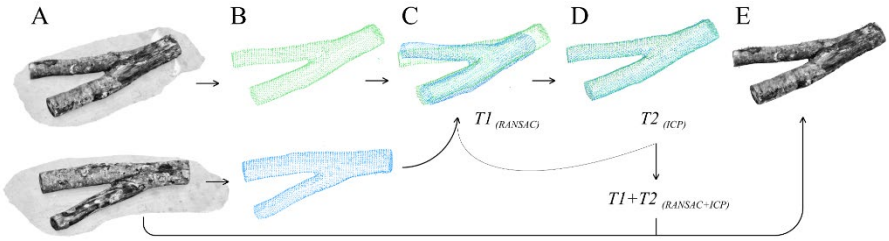

Figure 2: Basic application of the combined registration method to obtain a complete 3D model from two different scans. Without this function, it would be hard to acquire the entirety of the object's geometry without missing parts in the point cloud.

The proposed RR is also employed in alignment problems when the position and orientation of the workpiece, irregular in shape, is not known regarding the fabrication setup (Vestartas, 2021). In our case study developed for timber construction, manufactured elements are scanned twice: 1) before fabrication to obtain a tree stock and create a design model, and 2) after the design is made with a second scan made during fabrication to align tool-paths for cutting (see Figure 3). The previous publication describes that it is enough to use three reference points when mounting on a rig (Mollica, 2016) without pre-scanning. However, our experiment shows that scanning is needed to acquire a precise position of a tree fork because three pre-drilled mounting holes are not precise enough (see Figure 4E). The darker scan shown in Figure 3B is taken without the fabrication rig and was used for a design of a truss (see Figure 4F). When the design was fixed, a second scan is taken for fabrication and the two point clouds were aligned as shown in Figure 3 C-D.

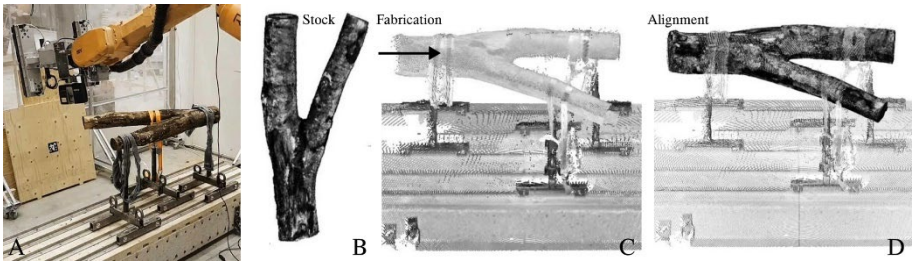

Figure 3: Registration process: a) robot scan, b) point-cloud taken months before for a design phase, c) scan with a mounted object and d) alignment of the two point-clouds.

Afterward, the fabrication can be started because pre-scanned tree-forks are combined with timber joinery tool-paths as illustrated in Figure 4F. The sequence of

fabrication steps are as follows: a) mounting, b) scanning and alignment of the two point clouds, c) cutting, d) de-mounting, and e) repeating the process. Lastly, the raw timbers are assembled, as shown in Figure 5.

Figure 4: Workflow: a) mounting, b) scan, c) cut, d) demounting, and e) machined forked, and f) individual element with joinery geometry.

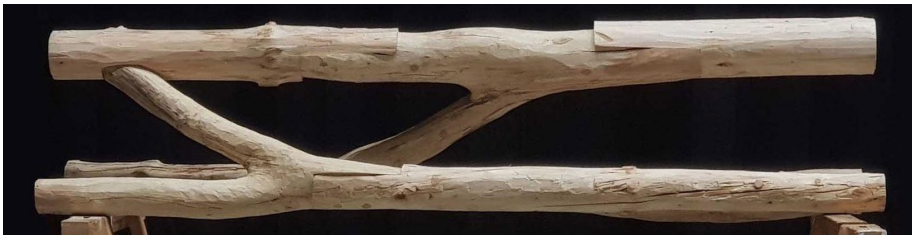

Figure 5: Assembly of the prototype using three tree forks and four straight segments.

3.2. CLUSTERING

Point cloud clustering subdivides a point cloud model into smaller elements. Cockroach implements multiple clustering methods such as color-, normal-, and distance-based clustering. The last method, Euclidean clustering, can segment detached elements belonging to the same scene. This function is particularly useful to identify multiple objects, as shown in Figure 6. The objects in the scene are first isolated by removing the plane via the RANSAC fitting plane. Second, the elements are filtered from noise with statistical outlier removal and the remaining point cloud is segmented. The same procedure is repeated twice per rock once it is turned. Finally, the stones are recomposed (see Figure 6B) and meshed (see Figure 6C). Point cloud normal-based clustering and key points detection allow digital reconstruction of larger objects composed of several complex irregular geometries, as illustrated in the preliminary study for digitizing dry-stone walls (see Figure 7). Automatic point cloud segmentation in a CAD environment can become a performant tool for architects and designers to describe and rapidly obtain 3D models out of any significant quantities of irregular elements or re-used components in construction. By building new digital catalogs from

stocks of re-used elements, designers can be informed from the early project stage with reliable CAD models of unpredictable geometries or non-standard building components. We designed and made available an online catalog with numerous scanned and reconstructed models of mineral construction waste that designers can inspect and download (Broyet et al., 2021). The website has been developed and made publicly available as a proof-of-concept for a novel approach to design with re-used elements and construction cataloging thanks to accessible point cloud processing tools.

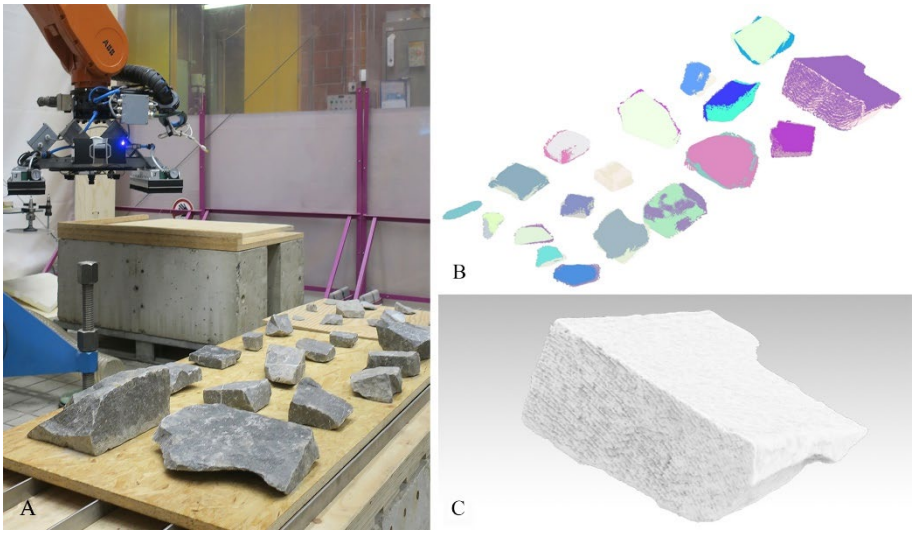

Figure 6: Automated point cloud scanning and post-processing for CAD models of irregular mineral scraps. The process can be applied to any other batch of irregular objects.

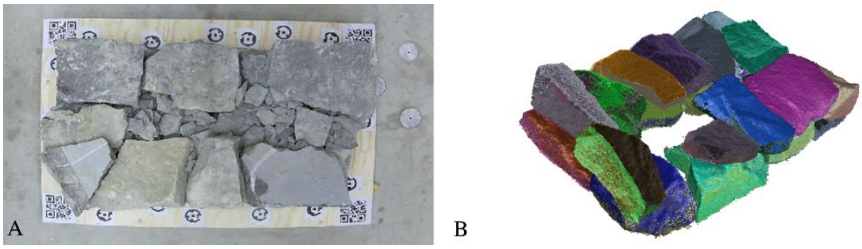

Figure 7: Segmentation of the dry-stone wall for 3D reconstruction and position identification.

4. Conclusions and Future Work

Cockroach showed how it is possible for designers to efficiently model even the most irregular objects from LiDAR scans. Given the rising need for using recycled, non-standard elements, we believe that high-level point cloud processing will be proved to be a valuable asset for sustainable computational design. Accessible advanced point cloud manipulation tools allow every designer to extrapolate meaningful information from the most intricate scan of any building, component, or raw resource. Considering the rise of embedded systems in the AEC practice and the growing presence of spatial sensors in renovation applications, point clouds may well be geometric data with which

architects will need to become increasingly familiar. The drafting process itself can be done using scan data instead of 2D or 3D drawing, whether for architectural heritage projects or real-time manufacturing and assembly processes. From the review of existing point cloud processing tools, it is evident that these tools are still not user-friendly and the only available solutions are stand-alone software applications making them difficult to implement into existing CAD software. While Cockroach was started from concrete research goals for robotic fabrication of irregular timbers and mineral scraps, the developed .NET interface allows this library to be used in several software packages such as Rhinoceros3d, Unity, Revit-Dynamo. In addition, the lack of documentation and maintenance of the open-source repositories makes state-of-the-art methods hardly accessible. The proposed tool attempts to reduce this step through research projects and applications, although they are not the main goal. The focus is given to a structured, well-documented, and maintained open-source repository Cockroach.

Regarding Cockroach's future developments, the main objective is to migrate the current library towards a self-contained, shippable, standalone, and cross-platform version of the software. The software will be composed of a basic visualizer (e.g., OpenGL or Vulkan), a GUI system (e.g., DearImgui), an integrated shell input as the main UI, a macro reader, and a file importer. The goal of developing the standalone Cockroach version is to extend the usability of the software to a broader public of users from different disciplines. This format may allow other high-level users than AEC designers to profit from intuitive but effective point cloud processing tools like Cockroach.

Acknowledgements

The authors would like to acknowledge the financial support of the Ecole Polytechnique Fédérale de Lausanne (EPFL), and in particular the ENAC Faculty with the Cluster Grant which partially contributed to fund the development of Cockroach.

References

Bell, A., Chambers, B., Butler, H & others (2021). *Point Data Abstraction Library* (version 2.3.0). https://doi.org/10.5281/zenodo.4031609.

Broyet, A., Settimi, A., Gamerro, J. & Weinand, Y. (2021). *Eesd-Ibois Scanned Stones Dataset.* Retrieved November 19, 2021 from https://ibois-epfl.github.io/eesd-ibois-scanned-stones-dataset/.

Caron, D., Vulgar, M. & Allen, A. (2020, April 21). *Pclpy: PCL for Python.* Retrieved November 19, 2021 from https://github.com/davidcaron/pclpy/releases/tag/0.12.0.

Cignoni, P., Corsini, M. & Ranzuglia, G. (2008). *ERCIM Newy. MeshLab: an Open-Source 3D Mesh Processing System.* Retrieved November 19, 2021 from http://vcg.isti.cnr.it/Publications/2008/CCR08.

Ervine, L. & Girot C. (2014). *Point Cloud Components: Tools for the Representation of Large Scale Landscape Architectural Projects.* Retrieved November 30, 2021 from https://archive.arch.ethz.ch/dla2014/talk_pdfs/DLA_2014_4_Lin.pdf.

Girardeau-Montat D., (2020). *CloudCompare.* Retrieved November 19, 2021 from http://www.danielgm.net/cc/release/.

Mellado, N. (2018). *OpenGR: a 3D Global Registration Library.* Retrieved November 19, 2021 from https://github.com/STORM-IRIT/OpenGR.

Mollica Z. (2016). *Tree Fork Truss: An Architecture of Inherent Forms.* Retrieved November 30, 2021 from https://zacharymolli.ca/assets/download/Mollica-DMThesis-Tree-Fork-Truss-an-Architecture-of-Inherent-Forms.pdf.

Newnham, C. & Gwyllim, J. (2016). *Tarsier* (version 1.0.1.0). Retrieved November 30, 2021 from https://bitbucket.org/camnewnham/tarsier/src/master/.

Pomerleau, F., Colas, F., Siegwart, R. & Magnenat, S. (2013). Autonomous Robots. *Comparing ICP Variants on Real-World Data Sets*, 34(3) (pp. 133.148). https://doi.org/10.1007/s10514-013-9327-2.

Rusu, R. B. & Cousins, S. (2011, May 13). 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation*, (pp. 1-4). https://doi.org/10.1109/ICRA.2011.5980567.

Sharp, N. & others (2019b). *Polyscope*. Retrieved from November 29, 2021 from www.polyscope.run.

Sharp, N., Keenan, C. & others (2019a). *Geometry-central.* Retrieved November 30, 2021 from www.geometry-central.net.

Tamke, M., Edvardsen, D.F., Beetz, J., Evers, H.L., Krijnen, T., Hecher, M., Zwierzycki, M., Panitz, M., Wessel, R., Vock, R., Ochmann, S. & Gadiraju, U (2016). *DURAARK Evaluation Report D7.4, FP7 – ICT – Digital Preservation*, Grant agreement No.: 600908. Retrieved November 30, 2021 from https://www.researchgate.net/publication/340267140_Parametric_architectural_design_with_point-clouds_Volvox.

The CGAL Project (2021). *CGAL.* In CGAL User and Reference Manual (version 5.3). Retrieved November 19, 2021 from https://doc.cgal.org/5.3/Manual/packages.html.

Vestartas P. (2021). *Design-to-Fabrication Workflow for Raw-Sawn-Timber using Joinery Solver.* Thesis. EPFL. http://dx.doi.org/10.5075/epfl-thesis-8928

Vestartas, P., Settimi, A. (2020). *Cockroach: A Plug-in for Point Cloud Post-Processing and Meshing in Rhino Environment,* EPFL ENAC ICC IBOIS. Retrieved November 19, 2021 from https://github.com/ibois-epfl/Cockroach.

Zampogiannis, K., Fermuller, C. & Aloimonos, Y. (2018). Cilantro: A Lean, Versatile, and Efficient Library for Point Cloud Data Processing. *Proceedings of the 26th ACM international conference on Multimedia (pp. 1364-1367).* https://doi.org/10.1145/3240508.3243655.

Zhou, Q., Parl, J. & Koltun, V. (2018). Open3D: A Modern Library for 3D Data Processing. *ArXiv, abs/1801.09847.* https://arxiv.org/abs/1801.09847.

Zwierzycki, M., Evers, H. L. & Tamke, M. (2016). Parametric Architectural Design with Point-clouds: Volvox. In *Complexity & Simplicity: Proceedings of the 34th eCAADe Conference* (Vol. 2, pp. 673-682).