

An Open-source Library for Processing of 3D Data from Indoor Scenes

José María Martínez-Otzeta¹^a, Iñigo Mendiadua²^b, Itsaso Rodríguez-Moreno¹^c,
Igor Rodríguez¹^d and Basilio Sierra¹^e

¹*Department of Computer Science and Artificial Intelligence, University of the Basque Country (UPV/EHU),
Donostia-San Sebastián, Spain*

²*Department of Languages and Computer Systems, University of the Basque Country (UPV/EHU),
Donostia-San Sebastián, Spain*

Keywords: Computer Vision, Pointcloud, 3D Segmentation, Open Source.

Abstract: In recent years affordable 3D data acquisition devices have appeared in the market. Researchers and developers have been able to use them in a much larger scale than ever in a wide range of applications, from robotics to autonomous driving. One of these applications is the processing of 3D indoor scenes, usually in the context of autonomous navigation of mobile robots, but also in building mapping for map reconstruction or assessment of the location of structural elements. In this paper we report on the development of an open source Python package (indoor3d) for processing of 3D data obtained indoors. This package is built on top of the Open3D package, with the aim of making easier to perform common tasks that arise in indoor data processing. It has already been helpful in tasks in two different projects: in one of them was useful in the search of structural elements in a pointcloud obtained by a HoloLens device, and in the other in the location of the handle of a door for a mobile robot navigation application.


1 INTRODUCTION


During the last years we have been witnesses of a big explosion in the number of 3D data acquisition devices around us. Fields like autonomous driving (Arnold et al., 2019) or industrial robotics (Lin, 2020) make use of them extensively, but also small-scale laboratories performing research on computer vision (Guo et al., 2020) or robotic navigation (Zieliński and Markowska-Kaczmar, 2021) can benefit of these sensors. Due to the widespread possibility of acquiring 3D data, need of processing and interpreting has also skyrocketed. In the hardware side, specialized devices like the Velodyne¹ series are intended for high-end applications with tight time response requirements. But other much more affordable devices can also be used when no critical performance is needed, as is the case of the Intel RealSense line (Zabatani et al., 2019). Not only data acquisition


needs to meet high performance standards, data processing is also very demanding. It is almost mandatory to employ GPU units to train machine learning models, and sometimes also for inference.


Open source software for efficient 3D data processing has been available for a long time. One of the reference libraries is the Point Cloud Library (PCL) (Rusu and Cousins, 2011), written in C++. PCL is a library for processing of n-dimensional pointclouds and 3D geometry processing. It is free software, distributed under a BSD license. It is integrated into ROS (Quigley et al., 2009), the Robot Operating System, widely used in the robotics community. Its data structures make extensive use of SSE optimizations available in modern processors. The implementation of most mathematical computations relies in Eigen (Guennebaud et al., 2010), an open source library for linear algebra. PCL also supports OpenMP² and Intel TBB library (Reinders, 2007) for parallelization. Nvidia GPUs are also supported.


More recently, Open3D (Zhou et al., 2018) has been presented as an alternative, written in Python and C++ with front-end in both languages. Its aim is to provide ease of use and an environment where

^a <https://orcid.org/0000-0001-5015-1315>

^b <https://orcid.org/0000-0003-2519-4094>

^c <https://orcid.org/0000-0001-8471-9765>

^d <https://orcid.org/0000-0002-1432-102X>

^e <https://orcid.org/0000-0001-8062-9332>

¹<https://velodynelidar.com/>

²<http://openmp.org>

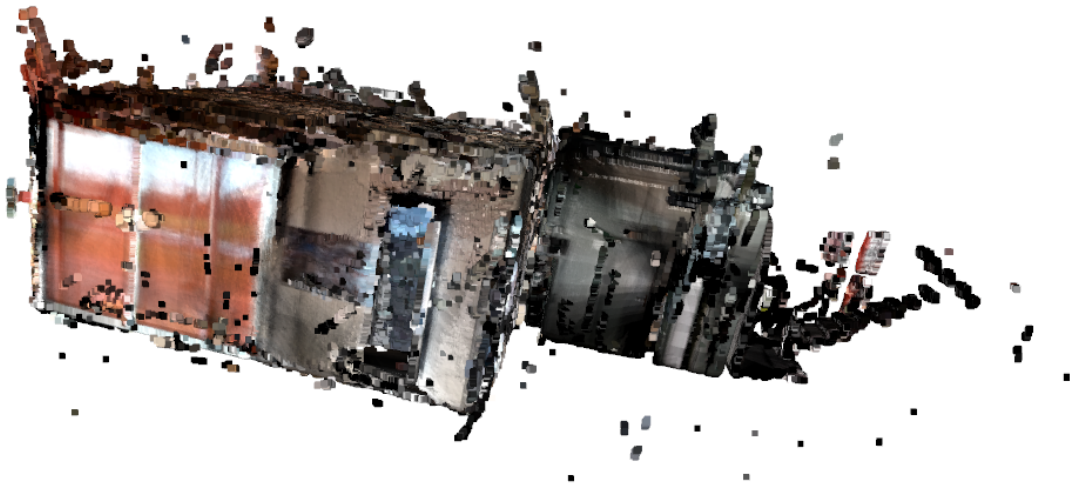


Figure 1: Pointcloud in the LANTEGI4.0 project.

rapid prototyping is possible, in addition to the development of full-fledged applications. The Open3D backend is implemented in C++11 and set up for OpenMP parallelization. It is open source and released under the MIT license. It is possible to work with pointclouds, meshes or RGB-D images, and input/output algorithms, sampling, data conversion and visualization is available for each of them. Other algorithms such as ICP registration, normal estimation, ICP registration (Besl and McKay, 1992), and volumetric integration (Curless and Levoy, 1996) have also been implemented.

In this paper we present an open source package written on top of Open3D and focused in the tasks that are usually found when working with indoors 3D data. We also present the application of the functions of this package in two different projects. The first one is LANTEGI4.0, a project in which one of the tasks was to find structural elements in a pointcloud representing a big section of a building; the second one is BotaRobota, a project in which one of the tasks was to locate the position and orientation of a door handle.

The paper is organized as follows: after the introduction, a section called Library Structure gives an outline of the arrangement of the library functions in different modules; then two Usage Cases are shown, demonstrating the usefulness of the library in two different projects; the paper concludes with a Conclusion section followed by the bibliography.

2 PACKAGE STRUCTURE

The package *indoor3d* is built on top of Open3D. Open3D provides data structures for three kinds of representations: pointclouds, meshes, and RGB-D

images. In our package we will only deal with pointclouds. The contribution over Open3D is the integration of implicit knowledge about the room geometry. For example, Open3D finds planes in the pointcloud using RANSAC (Fischler and Bolles, 1981), and our package finds sets of mutually parallel or perpendicular planes from their results. Likewise, Open3D provides clustering functions, and our package tells whether the clusters are close to the limits of the room (ceiling, floor and walls) or not.

The package is structured in five different modules:

- vector
- plane
- pointcloud
- findroom
- clusteringroom

The modules *vector* and *plane* define functions that perform geometric operations on vectors and planes, respectively. In the *pointcloud* module there are functions that take pointclouds as parameters or that return pointclouds. The module *findroom* is the place for code that is useful when looking for rooms in a pointcloud of a building. And clustering and segmentation utilities belong to the *clusteringroom* module.

The functions in the modules are commented with Python docstrings in such a way that it is possible to generate automatically documentation in HTML or PDF format using Sphinx³.

The main idea is that indoor applications which make use of pointclouds may share some common

³<https://www.sphinx-doc.org/>



(a) Original pointcloud.

(b) Pointcloud with its pseudo-negative in green.

Figure 2: Example of the pseudo-negative of a pointcloud.

tasks. For example, it is usual to try to locate a door, a wall, the floor, or estimate the distance from some object to the wall. These tasks usually imply to be able to find planes, compute angles and distances, check if floor and ceiling are parallel to each other, find the point of intersection of two walls, etc. This package is intended to help the researcher or developer with the more menial tasks.

Vector Module. In this module, functions that work with vectors are defined. For example, it is possible to convert any iterable into a Python numpy array, which is the default representation of vectors in this module. There are functions to get the unit vector of an arbitrary vector, to compute the angle between vectors or to find the point that is located at some distance of a given point in a given direction.

Plane Module. The functions that deal with planes are defined here. A specialized plane class is constructed from the four parameters of the plane equation $Ax + By + Cz + D = 0$. The objects of this class can return the normal of the plane.

In this module we find utilities to compute the angle between planes, between the plane and the axes, between the plane and the plane perpendicular to one of the canonical axes, to check if two planes are perpendicular or parallel up to some given tolerance, the point that lies in the intersection of three planes, distances between planes and points, etc.

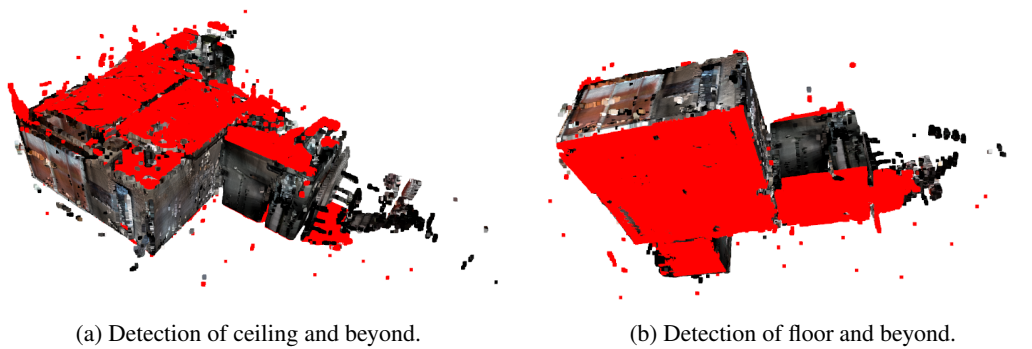
Pointcloud Module. This module contains the functions that compute distances between points and planes to pointclouds, as well as those which check if a point is inside the convex hull of a pointcloud. It is also possible to partition a pointcloud in three subsets according to a plane: those closer to the plane than a given tolerance, those at one side of the plane and those the another side. This is useful when a plane that we presume is a door has been detected, and we want to know which points are in the door, which in

front of the door and which behind the door. The same procedure, but with five subsets, can be applied taking a pointcloud and two parallel planes. This is useful, for example, for segmenting the points in a room according to the planes defining the ceiling and the floor. An example of the kind of pointcloud expected is shown in Figure 1.

It is also possible to subtract a pointcloud from another pointcloud. In this case, if pointcloud B is subtracted from pointcloud A, all the points in pointcloud A that are closer to pointcloud B than some threshold are removed. It is also possible to subtract a convex hull from a pointcloud. If a point is inside (could be parameterized to be outside instead) the convex hull, it is removed.

There are also functions for performing a uniform 3D sampling inside a convex hull by rejection sampling. This allows the definition of another function that finds the pseudo-negative of a pointcloud. Given a pointcloud A, first its convex hull is computed. Then, another pointcloud B is created from a uniform random sampling in the convex hull. And finally, pointcloud A is subtracted from pointcloud B. With this procedure it is possible to find door frames, as shown in Figure 2.

Findroom Module. This module is in charge of finding structural elements that could belong to a room. For example, it is possible to find a collection of N planes in the pointcloud, where N is given by the user, and search for sets of six planes that might delimit a room. The procedure looks for three pairs of parallel planes that are perpendicular between them. These planes would be the pairs (ceiling - floor, wall left - wall right, wall front - wall -rear), all of them composed by two parallel planes. The condition of parallelism is also possible to be relaxed by some tolerance, to account for noise in the data. If it is known that some axis of the frame of reference of the pointcloud is congruent with the vertical orientation in the real world, this information can be given to the function to improve the search for planes.



(a) Detection of ceiling and beyond.

(b) Detection of floor and beyond.

Figure 3: Example of the pseudo-negative of a pointcloud.

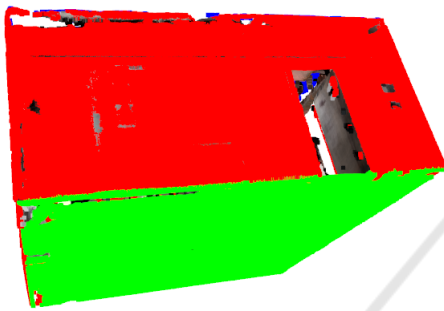


Figure 4: Room found by the pointcloud processing procedure.

There are also functions for computing the eight points that delimit the vertices of the room, and for extracting the points that are inside and outside the room, or very close to the boundaries.

Clusteringroom Module. In this module there are functions meant to be used after a room has been detected and the planes of floor, ceiling and walls have been computed. There are clustering functions that take into account the distance to the ceiling, floor and walls and mark some clusters as door or lamp candidates. It is also possible to get a graphical depiction of the clusters.

3 USAGE CASES

The development of this package started during the LANTEGI4.0 project, funded by the Basque Government, and where the goal is to make advances in the development of a system that would be able to create a 3D model of the true state of the building of a factory from the construction planes and the data collected by an autonomous robotic system.

One of the tasks was to detect structural elements from data collected by a sensor. These data repre-

sented a significant part of the building, bigger than a single room. In Figure 1 the input pointcloud is shown. The results after applying the ceiling and floor detection are shown in Figure 3.

After finding all the planes that delimit the room, they can be extracted as shown in Figure 4. The walls are painted in red, the ceiling in blue and the floor in green.

The clustering functions take into account information about the planes that delimit the room, and this is used to detect clusters that are not likely to be structural elements because are too far away from the walls and the ceiling. In Figure 5 can be seen the detected clusters, while in Figure 6 the clusters which are unlikely to be structural elements are shown.

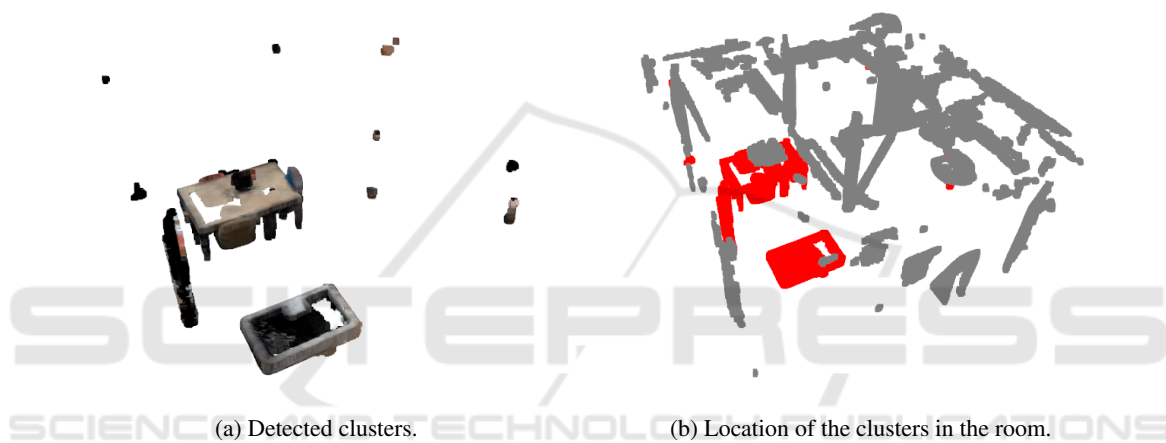
The package developed for the LANTEGI4.0 project has been also used in another project called BotaRobota. BotaRobota is a project funded by the Basque Government with the aim of developing a cleaning robot able to detect and open doors without human intervention. One of the tasks was to detect the door handle in a pointcloud so the robot could be able to open the door. To detect the door handle the first step was to detect the door using the standard plane search procedure, and then the functions of segmenting the pointcloud with respect to a plane, and the functions for computing distances between planes and pointclouds were useful to locate the handle. For the orientation of the handle, other functions that find planes parallel to another to some distance were useful in order to make it easier to find a minimal bounding box parallel to the door. In Figure 7 can be seen the results of this procedure. In short, the functions developed in the LANTEGI4.0 project were useful in making easier to tackle the BotaRobota project and we want to make this code available for anyone who could find it useful.



(a) Set of clusters detected inside a room.

(b) Set of clusters painted in different colors.

Figure 5: Clustering of a room.



(a) Detected clusters.

(b) Location of the clusters in the room.

Figure 6: Clusters labeled as unlikely to be structural elements.

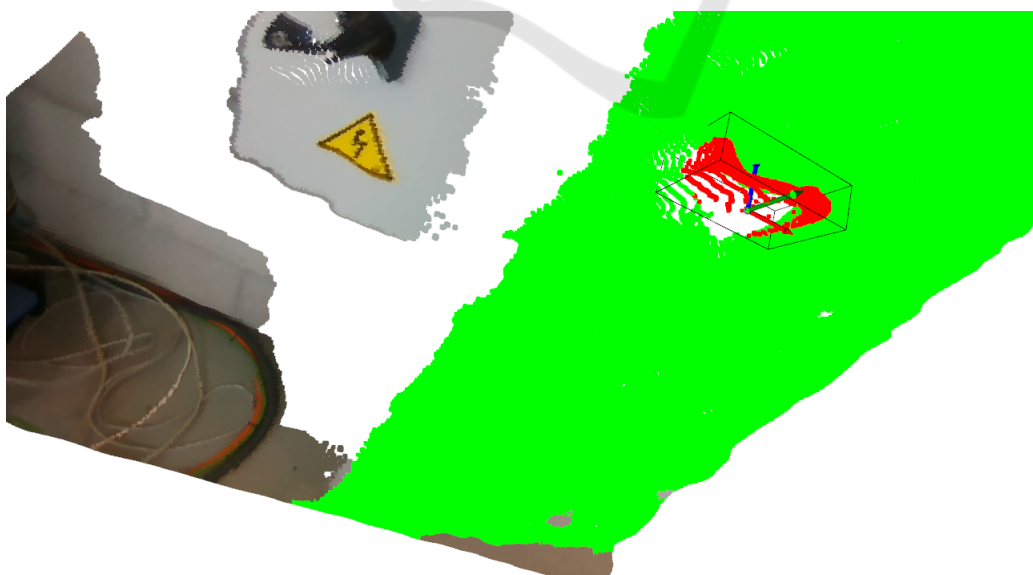


Figure 7: Door handle position and orientation.

4 CONCLUSION

In this paper we have presented a 3D data processing library focused on indoor scenes. We have described its main functionalities and showed its application on two real projects. As further work, we plan to improve the documentation and refactor and clean the code, with the aim of uploading the library to the PyPI repository, the reference place for Python packages. In the meantime, the code is available in GitHub⁴.

ACKNOWLEDGEMENTS

This work has been partially funded by the Basque Government, Spain, grant number IT900-16, ELKARTEK project (LANTEGI4.0 KK-2020/00072) and Euskampus Resilience Covid19 (BotaRobota EUSK20/04), and the Spanish Ministry of Science (MCIU), the State Research Agency (AEI), the European Regional Development Fund (FEDER), grant number RTI2018-093337-B-I00 (MCIU/AEI/FEDER, UE) and the Spanish Ministry of Science, Innovation and Universities (FPU18/04737 predoctoral grant). We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp GPU used for this research.

REFERENCES

- Arnold, E., Al-Jarrah, O. Y., Dianati, M., Fallah, S., Oxtoby, D., and Mouzakitis, A. (2019). A survey on 3d object detection methods for autonomous driving applications. *IEEE Transactions on Intelligent Transportation Systems*, 20(10):3782–3795.
- Besl, P. J. and McKay, N. D. (1992). Method for registration of 3-d shapes. In *Sensor fusion IV: control paradigms and data structures*, volume 1611, pages 586–606. International Society for Optics and Photonics.
- Curlless, B. and Levoy, M. (1996). A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 303–312.
- Fischler, M. A. and Bolles, R. C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395.
- Guennebaud, G., Jacob, B., et al. (2010). Eigen. *URL: <http://eigen.tuxfamily.org>*, 3.
- Guo, Y., Wang, H., Hu, Q., Liu, H., Liu, L., and Benamoun, M. (2020). Deep learning for 3D point clouds: A survey. *IEEE transactions on pattern analysis and machine intelligence*.
- Lin, H. (2020). Robotic manipulation based on 3D vision: A survey. In *Proceedings of the 2020 International Conference on Pattern Recognition and Intelligent Systems*, pages 1–5.
- Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., Ng, A. Y., et al. (2009). ROS: an open-source Robot Operating System. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, Japan.
- Reinders, J. (2007). *Intel threading building blocks: outfitting C++ for multi-core processor parallelism*. "O'Reilly Media, Inc."
- Rusu, R. B. and Cousins, S. (2011). 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China.
- Zabatani, A., Surazhsky, V., Sperling, E., Moshe, S. B., Menashe, O., Silver, D. H., Karni, Z., Bronstein, A. M., Bronstein, M. M., and Kimmel, R. (2019). Intel® RealSense™ SR300 Coded Light Depth Camera. *IEEE transactions on pattern analysis and machine intelligence*, 42(10):2333–2345.
- Zhou, Q.-Y., Park, J., and Koltun, V. (2018). Open3D: A modern library for 3D data processing. *arXiv:1801.09847*.
- Zieliński, P. and Markowska-Kaczmar, U. (2021). 3D robotic navigation using a vision-based deep reinforcement learning model. *Applied Soft Computing*, page 107602.

⁴<https://github.com/rsait/indoor3d>