Point cloud processing and computer vision for object detection in dense environments

Master's Final Project

REPORT

This is a temporary cover

David Pérez Ruiz

Master's degree in Automatic Control and Robotics

Project director:

Alexandre Perera Lluna

June 2021

Barcelona, Spain

A mi familia.

Declaration of Honor

I declare that,

the work in this Master Thesis is completely my own work, no part of this Master Thesis is taken from other people's work without giving them credit, all references have been clearly cited.

I understand that an infringement of this declaration leaves me subject to the foreseen disciplinary actions by The Universitat Politècnica de Catalunya - BarcelonaTECH.

David Pérez Ruiz — June 14^{th}, 2021

Student Name — *Signature* — *Date*

Title of the Thesis : Point cloud processing and computer vision for object detection in dense environments

Contents

List of Figures

Figure 1: Red lion logo

Hello, here is some text without ae meaning. This... ee

This is the abstract.

1 Introduction

The aim of this project is to test different algorithms with the purpose of detecting objects in dense environments. Note that by detecting I mean discerning if a certain object is present in an environment, and if it is, find its position. The type of dense environment that will be used for this project will be a simulation of a supermarket shelf, where objects are close together in relatively confined spaces. To that effect, a testbed with fake food and house furniture will be built. The environment will be captured with a depth camera, able to output both a 3D point cloud and a 2D image. Hence, the approach is twofold: computer vision algorithms for the 2D image and point cloud processing-related algorithms for the point cloud data.

The aim of this work is to provide a comprehensive study of the most prominent approaches one may take to tackle the task of detecting objects in a scene. That is, instead of going too much in depth into one single research topic the aim of this work is to provide many working solutions using both well established and state of the art techniques. Hence, this work is focused in providing the widest array of solutions possible under the time constraints of this thesis, and so, the focus is not set into providing a production-ready system.

2 Methodology and tooling

Here goes methodology and tooling intro.

2.1 Assumptions

Asunciones básicas que se toman en este proyecto.

1. No assumptions can be done about the camera's position and movement. The camera may be still facing the testbench or slightly moving.

2. There will always be some light shed on the testbench, but its intensity may vary.

2.2 The testbench

Explicar con texto qué testbench se va a usar, porqué se ha escogido ese, lo que motiva la disposición de los elementos y, sobretodo, fotos.

2.3 Data acquisition

Decir primero que la cámara a usar será 3D porque así tienes nubes de puntos e imágenes 3D de una. Hablar de los tipos de cámaras que hay y en qué tecnologías se basan. Presentar comparativa de todas ellas y decir cuál se escoje. Hablar del driver que usa para adquirir imágenes. Luego hablar de cómo se guardan los point clouds, su formato PCD, y porqué de dónde viene este formato. Lo mismo con los diferentes formatos de imagen (que sea breve).

2.4 Frameworks and languages

This section is designed to introduce the languages and frameworks that will be used all throughout the project and provide some reasoning on why they are used, as well as to provide some background

in the array of options out there and why ones are chosen above others. Since most of the concepts of this work are yet to be introduced in the upcoming sections I will try to gently introduce and reference the frameworks this section will describe as they go.

This work is based on a dual approach to the concept of computer vision[1]: 2D digital image processing (in the form of bitmaps) and 3D point cloud processing. Hence, it makes sense to assess the capabilities of the most important libraries that tackle those tasks.

Most prominent 2D image processing libraries There are many image processing libraries. However, only the most notable ones are listed here. The newest machine/deep learning libraries, although used extensively for computer vision-related tasks, are not included in this list but rather given a dedicated chapter later on.

- OpenCV: the Open Source Computer Vision library was started in 1999 by an Intel Research member called Gary Bradski, with the aim of making computer vision universally available. Its developing team has changed with time, but receives periodic improvements from the computer vision community. So much so that it incorporates some Deep Learning algorithms. OpenCV was built from the start with optimization and real-time applications in mind, making its algorithms as fast they can go. On top of that, it uses hardware acceleration[2] when it can. Its interfaces are C++ (the language in which it is written), Python, Java and MATLAB, which is very useful when incorporating OpenCV into existing code bases not necessarily written in C++. The fact that the interface supports higher-level languages like Python (via Python bindings created using the Python/C API[3]) makes writing programs fast and enhances productivity by being able to use widely popular packages like Numpy, Matplotlib, and others.

  OpenCV is the go-to library for image processing (so much so that, at the time of writing, their webpage states 18 million downloads), so the newest algorithms in the literature tend to be implemented first there, either by the same author or people who understood them. This and the fact that it is a 20+ year old library makes it incredibly robust and proof tested.

  The library comes with an Apache 2 license, making it free to distribute, modify and use even for commercial applications (arguably this is one of the driving forces behind its popularity). Patented algorithms implemented in OpenCV, however, are not actually free to use and may require royalties or some kind of fees to the patent holders (like the SIFT algorithm, explained in this project).

- Scipy: the Scipy package [1] was published in 2001 with the purpose of being a whole ecosystem of open-source tools for mathematics, science and engineering. It being an ecosystem provides a core library on top of supporting other packages like SymPy, NumPy, Matplotlib, and others. In particular, Scipy's core module `ndimage` supports several image processing related tasks. It has been superseded by Scikit-image.

- Scikit-image: the package scikit-image [2] is an open source image processing library that derived from Scipy, with the intention to focus on image processing related tasks. It was first released in 2009 and has since undergone a long way of continuous improvements from the

[1] Taking the meaning of *computer vision* in the broadest sense possible, that is, making machines sense the environment, irrespective of how the environment is recorded.

[2] Like SIMD instruction sets.

[3] `https://docs.python.org/3/c-api/index.html`

community. It does not come with as wide an array of solutions as OpenCV does, but instead provides (1) high quality open and free pythonic code that has been peer reviewed prior to its inclusion in the package; (2) a focus on education; (3) a sufficient range of state of the art algorithms; and (4) efficient code that runs reasonably fast (some sections of the code are written in Cython or use Numba) and uses the de facto standards of Python in the science realm (that is, NumPy and Matplotlib).

3D point cloud processing libraries There are fewer options regarding the processing of point clouds. OpenCV, although known for its 2D image processing capabilities, has some 3D processing features. But the go-to library is actually the Point Cloud Library (PCL) [3], launched in 2011. Just like OpenCV, it is implemented in C++ with efficiency and speed in mind[4], and includes almost all the latest and greatest algorithms from the literature, either implemented by their own authors or others persons in the community. Despite its popularity, it does not provide an interface other than the C++ API in which it was coded.

The framework of choice for the 2D approach (Section 3) is OpenCV, used with the Python bindings. Combining the completeness and robustness of OpenCV with the boost in productivity Python itself as a language offers (compared to C++) and the array of packages that can be used alongside is key to this work. The framework of choice for the 3D approach (Section 4) is PCL, again, for its completeness and robustness. Sadly, there are no official Python bindings, which means that all work will need to be carried out with its C++ API and no interaction with the Python interpreter is possible (unless I built custom bindings, which is not feasible due to time constraints). Code from the 3D approach should be self contained, and any interaction using Python would need to be done by firstly writing the data to disk and then reading it again from the other end (from the Python's end that is).

Deep learning frameworks Section 3.6 of this work explores the use of Deep Learning for approaching the object detection issue. Hence, it is pertinent to provide a full comparison of the competitive deep learning frameworks available. It has to be noted that deep learning is a type of approach that works for both 2D images and 3D point cloud data.

- TensorFlow:
- PyTorch:
- MXnet:

Essential libraries Python is one of the most popular [**?**] languages for two main reasons: first, the fact that it is a dynamic[5] high level language that allows for high productivity; and second because of its ecosystem. Ecosystem meaning the set of quality libraries that surround the language itself and make it ideal for a wide array of tasks. Those libraries are contributed freely by the community, and have become so ubiquitous that are almost considered *de facto* standards, particularly in the scientific computing area. The ones used for this project are listed below:

1. NumPy:

[4]SIMD instruction set for CPU computing and CUDA for GPU computing.
[5]Meaning it executes in runtime tasks that other languages would do in compile time.

2. Matplotlib:

3. Numba: TODO:mencionar la sección exacta en la que se usa!

And last but certainly not least, the Python Standard Library, which, as the name suggests, comes with the Python distribution. It offers a wide range of facilities to everyday hassles programmers may encounter, and does so in an OS-independent API manner. Such facilities include I/O functionality, networking, parsing, database management, email handling, and a long etcetera. Full list is provided in the official documentation[6].

3 2D approach to object detection

2D approach introduction goes here. Example citation [4].

3.1 Dataset creation and tooling

3.2 The Hough Transform

3.3 Scale Invariant Feature Transform (SIFT)

3.4 Speeded-Up Robust Features (SURF)

3.5 Oriented FAST and Rotated BRIEF (ORB)

3.6 The world of Neural Networks

3.6.1 Retina Net for object detection

3.6.2 Fully Convolutional Networks (FCN) for object segmentation

3.6.3 TODO: A few notes on optimized model inference technologies

4 3D approach to object detection

3D approach introduction goes here.

4.1 Filtering

TODO: Decir que filtering es como dataset creation del caso 2D

[6] `https://docs.python.org/3/library/`

4.2 Registration

4.3 Sample Consensus (SAC) segmentation

4.4 Euclidean Cluster segmentation

4.5 Region Growing segmentation

4.6 Local descriptor matching

4.6.1 Scale Invariant Feature Transform (SIFT)

4.6.2 SHOT descriptor

4.6.3 PFH descriptor

4.6.4 FPFH descriptor

4.7 Global descriptor matching

4.7.1 GSHOT descriptor

4.8 Correspondence Grouping

4.9 Implicit Shape Model

5 Combining the 2D and 3D approaches

Combining the 2D and 3D approaches goes hereee.

6 Future Work

Future Work goes hereee.

7 Conclusions

Conclusions go here.

References

[1] E. Jones, T. Oliphant, and P. Peterson, "Scipy: Open source scientific tools for python," 01 2001.

[2] S. van der Walt, J. L. Schönberger, J. Nunez-Iglesias, F. Boulogne, J. D. Warner, N. Yager, E. Gouillart, and T. Yu, "scikit-image: image processing in python," *PeerJ*, vol. 2, p. e453, Jun 2014.

[3] R. B. Rusu and S. Cousins, "3d is here: Point cloud library (pcl)," in *2011 IEEE International Conference on Robotics and Automation*, pp. 1–4, 2011.

[4] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. The MIT Press, 2016.