Article **ECORFAN Journal-Spain**
June 2022 Vol.9 No.16 28-33

User interface design and object segmentation applied to Autominy platform

Diseño de interfaz de usuario y segmentación de objetos implementado en la plataforma Autominy

LÓPEZ-PÉREZ, Manuel Aarón†*´, SANDOVAL-GIO, Jesús´, IX-ANDRADE, Freddy Antonio´ and MOLINA-CÉSPEDES, Julio´´

´Tecnológico Nacional de México, Instituto Tecnológico de Mérida, Departamento de Ingeniería Eléctrica y Electrónica (DIEE), México.
´´Universidad Politécnica de Yucatán, México.

ID 1st Author: *Manuel Aarón, López-Pérez* / **ORC ID:** 0000-0001-5659-1848, **arXiv ID Author:** Aaronlpez_, **CVU CONACYT ID:** 1082426

ID 1st Co-author: *Jesús, Sandoval-Gío* / **ORC ID:** 0000-0001-5847-3669, **Researcher ID Thomson:** V-1930-2018, **arXiv ID Author:** jesus.sandoval, **CVU CONACYT ID:** 297308

ID 2nd Co-author: *Freddy Antonio, Ix-Andrade* / **ORC ID:** 0000-0003-2420-4879, **arXiv ID:** freddy.ix, **CVU CONACYT ID:** 470127

ID 3rd Co-author: *Julio César, Molina-Céspedes* / **ORC ID:** 0000-0002-6705-5976, **arXiv ID Author:** Julio_molina**, CVU CONACYT ID:** 597664

DOI: 10.35429/EJS.2022.16.9.28.33 Received January 25, 2022; Accepted June 30, 2022

Abstract

This work proposes to design and implement a user interface to the Autominy platform, which is used for teaching robotics at the Universidad Politécnica de Yucatán. In addition, implementing an object segmentation algorithm improves the robot's environment perception. For the development of the user interface, a framework called KivyMD based on the Python language was used. As far as the segmentation code is concerned, The Point Cloud Library (PCL) is a library which facilitates the management of a large amount of point cloud processing. These are theoretically supported by OcTree sample reduction and by finding the nearest neighbor using K-d Trees. Both techniques are written to create a ROS (Robotic Operating System) Node to improve communication between the actuators of the Autominy robot. In addition to providing an application with which the mobile robot can be manually controlled, a different method for obstacle perception is proposed for autonomous or manual navigation.

Framework, Autonomous, Reduction, Segmentation, Perception, Processing, Actuators, Application

Resumen

Este trabajo propone diseñar e implementar una interfaz de usuario a la plataforma Autominy, la cual es usada para la enseñanza de robótica en la Universidad Politécnica de Yucatán. Así como implementar un algoritmo de segmentación de objetos para mejorar la percepción del entorno del robot. Para el desarrollo de la interfaz de usuario se utilizó un framework llamado KivyMD basado en el lenguaje Python. En cuanto al código de segmentación, se recurrió al apoyo de "The Point Cloud Library (PCL)", el cual es una librería que facilita el procesamiento de las nubes de puntos. Estas se sostienen teóricamente por la reducción de muestras por OcTree y por encontrar el punto más cercano mediante los K-d Trees. Ambas técnicas están escritas para que se cree un Nodo ROS que facilite la comunicación entre los actuadores del robot Autominy. Además de entregar un aplicativo con el cual se puede controlar de manera manual el robot móvil, se propone un método diferente para la percepción de obstáculos para una navegación autónoma o manual.

Framework, Autónomo, Reducción, Segmentación, Percepción, Procesamiento, Actuadores, Aplicación

Citation: LÓPEZ-PÉREZ, Manuel Aarón, SANDOVAL-GIO, Jesús, IX-ANDRADE, Freddy Antonio and MOLINA-CÉSPEDES, Julio. User interface design and object segmentation applied to Autominy platform. ECORFAN Journal-Spain. 2022. 9-16:28-33.

* Author's Correspondence (E-mail: MG20080759@merida.tecnm.mx)
† Researcher contributing first author.

Introduction

The Autominy platform is an autonomous mobile device that uses an Intel camera and a 2D Lidar sensor for object detection. The main framework for controlling all sensors is ROS (Robot Operating System) which runs in Windows and Linux. ROS can handle the programming, control, and simulation using python and C++ languages. However due to the complexity of Autominy's control executing commands typed in a terminal, it is necessary to create a user interface to enhance the experience using this platform.

This work is transcendental because; the Universidad Politecnica de Yucatán did not previously have a practical way to control this mobile robot except for command lines, which could represent certain complications. In other words, the enhancement of this; serves as an optimization for teaching subjects like control or basic robotics.

In addition, within other problems to solve, perception represents an important challenge. (BORGES- MONSREAL et al., 2021.) used the Intel D435 Camera to apply artificial vision methods for achieving autonomous navigation without obstacles. Therefore, it is proposed a different process for identifying objects using the point cloud data from the Depth Camera; this idea is based on Euclidean cluster extraction theory which could facilitate object detection. The content of this paper is divided into theoretical foundation which explains the basics of the robot, as well as the software used to develop, and data architecture fundamentals for implementing the clustering algorithms. Then the methodology expounds how the codes work. Next the results section shows in a visual way, the clusters of the objects in the robot's peripheral perception. And finally, the conclusions and the future work of this line of research.

Theoretical Foundation

Autominy

Autominy is a research platform developed at Free University of Berlin, which emulates in a scale of 1:10 an autonomous car. (Alomari, K. et al. 2020.) It's equipped with sensors that provide information as to the angular position of the steering wheel, and an encoder to determine the position of the engine. In addition, it has a processing unit supplied by Intel, like the Realsense D435 Camera. For controlling the actuators, an Arduino Nano board is used. (Autominy, n.d.)

Figure 1 Autominy Platform
Source: (Alomari, K. et al. 2020.)

ROS

ROS is an operating system created for facing control problems, since there are many types of robots which have different hardware and functionalities.

Specifically, ROS is a set of software libraries that makes use of specific tools for the control, visualization and simulation. Making this using not only for academic purposes, but also in industry, since this is open source.

The fundamental concepts for implementing this software are **nodes, topics** and **services**. Nodes are the processes in which the computing is implemented. As ROS is designed to be modular; the system is composed of several nodes. These nodes are connected through the messages exchange. This message data structure is strictly written as an integer, float, or boolean.

A node sends a message by publishing through a specific topic, which could be a simple string as "Odometry" or "Map". There may be multiple publishers and subscribers for one single topic, and a single node could publish or subscribe to many topics. (Quigley et al., 2009)

Planar Segmentation Methods

Discrete point processing has always been represented as one of the most challenging tasks for the perception of the environment. The massive points acquired from a depth camera, make the process a high-memory and time consuming task. Down sampling allows processing algorithms to not have to analyze each point of the data, which means less resource consumption. Planar segmentation methods are examples which use only a few points of the total point cloud and maintain exactly the same result as if each point were analyzed.(Yamaguchi,et al. 1984)

ISSN: 2444-3204

Octree Method consists in a 2n x 2n x 2n array, and it is formed by repeatedly dividing itself in octants and sub octants, until obtaining sections with only one constant value, otherwise another ramification is required. As it can be observed in Figure 2; this process is usually represented as a tree with eight ramifications, each node corresponds to an array, while the eight children node is related to the octants. Explained in other words, the implementation of this method is effective for saving time and processing power of the point cloud. (Wen et al., 2019).

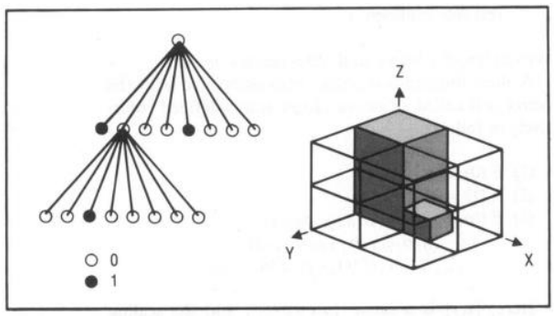


Figure 2 Octree-based region growing for point cloud segmentation
Source: (Yamaguchi, et al., 1984)

A KD-tree is defined as an algorithm to store a finite set of points from a space of k-dimensions. This structure separates the data space into multiple sub spaces and each node is divided into two subspaces; if the contained data is less than the lower limit, this node won't divide its data space again.(Kraus, Piotr et al. 2008) As shown in Figure 3. all divisions will be done in a specific dimension, using a hyper-plane which is perpendicular to the corresponding axis. At the root, all the secondary elements will divide in function of the first dimension, namely, if the first coordinate of the first dimension is less than the root, this will be in the left sub-tree, otherwise it'll be at the right sub-tree. Each level downside at the tree, splits into the next dimension until all the coordinates are used up.

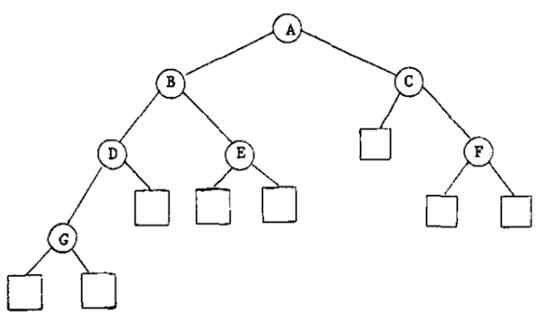


Figure 3 Nearest neighbor search by using Partial KD-tree method
Source: (Kraus, Piotr et al. 2008)

Methodology and development

Preparations

First, it is necessary to establish a link between the roscore form Autominy and the roscore from the main computer. To do this, the main computer environment should have the next variables *ROS_MASTER_URI* and *ROS_IP* with IP from Autominy as the master, and the computer as slave. This configuration is represented in Figure 4.

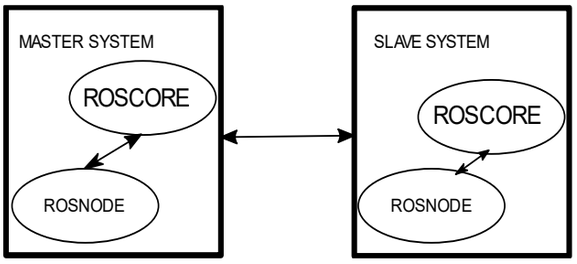


Figure 4 ROS System connection
Source: (Own Source)

User Interface Development

KivyMD is a framework for app development using Python, so the integration with ROS commands is an ideal option. There are two main files, the former is the python file which contains the ROS commands and the logic output when pressing buttons and setting dials. The latter is a ".kv" file with the design of digital buttons and sliders, also containing the properties of each component. Both files are linked through a main class in the python file, also this principal file has all the topics needed to control the Autominy. The figure 5 shows the libraries imported from Autominy's workspace and KivyMD. These libraries will be utilized later, but for now it is important to emphasize the design of the user interface.

```
#! /usr/bin/env python3.8

from kivy.uix.widget import Widget
import rospy

from kivymd.app import MDApp
from kivy.uix.screenmanager import Screen
from kivy.lang import Builder
from kivy.core.window import Window
#from kivy.uix.boxlayout import BoxLayout
#from kivy.uix.button import Button
from autominy_msgs.msg import SpeedPWMCommand
from autominy_msgs.msg import NormalizedSpeedCommand
from autominy_msgs.msg import NormalizedSteeringCommand
from autominy_msgs.msg import SteeringPWMCommand
from os import system

from std_msgs.msg import Int8
from std_msgs.msg import Bool
from std_msgs.msg import String as Str
```

Figure 5 Libraries imported from KivyMD
Source: (Own Source)

ISSN: 2444-3204

The control interface is simple; it has two sliders which set the speed and steering nodes values and publish information to actuators of the robot. Arrows are implemented as buttons of direction that change the values automatically, as well as a stop button that kills all processes and messages that are running in that moment. Equally important, visualization in real time is necessary, so that the Realsense Camera topics are streamed via web, for this an external package was installed, which essentially sets up a webserver on the robot that allows to select and view image streams from a web browser. Because of the web based nature anyone on the same network will have the ability to monitor cameras.

Clustering algorithm Development

The code starts creating a ROS package using C++ as a principal language, when compiling this package, a ROS node is created which will induce a process using three-dimensional planar segmentation methods, in this way, Point Cloud Library (PCL) makes the data processing implementation easier. (Rusu and Cousins, 2011).

Once the node is running, a subscriber and publisher is described, the former for establishing the input data, the latter for printing processed data. The algorithm waits to read the input data from the depth camera and once it's done, immediately makes the conversion from ROS to PCL data type, due to the PCL libraries being specifically made to work with certain types of data. Now that the data are homogeneous, p (x,y,z) point cloud coordinates are subjected to a downsampling process to enhance time processing and memory consumption.

Then, a Kd-tree object is created to search the nearest point from a reference and cluster it. It is important to mention that for this process, an index vector was created, containing information of the current index. Each index of every detected cluster is conserved. (e.g., cluster *index [0]* is composed of all the indexes from the first cluster in the point cloud). Furthermore, a clustering tolerance was set by empirical syntonization; this is because there is not a standard perception hardware and the precision of sensing 3d coordinates of each sensor may be diverse. Let's say if the tolerance is overly lowercase, a single object could be segmented in many clusters; on the other hand, if it's too meaningful many objects could belong to a single cluster. Now the clusters are stored in the index vector, and through an iteration, new clusters are invoked and added to the main point cloud.

Finally, after all the clusters are processed from the main point cloud, every single segmentation is converted back from PCL to ROS data type and published via ROS protocol for visualization or even setting a trajectory avoiding those blobs.

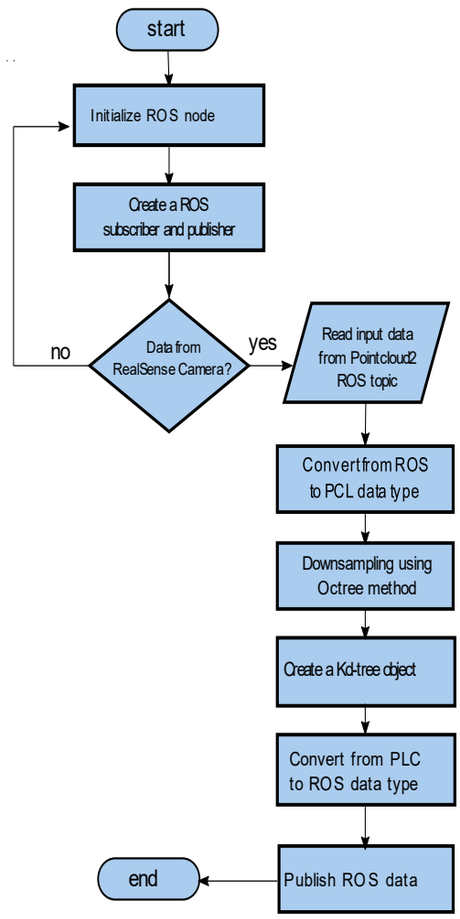


Figure 6 Clustering Pseudocode *(Own Source)*

Results

The User Interface designed is capable of sending ROS commands via ROS topics initialized in the master's roscore. This implies a bidirectional communication between the two computers. Consequently, Autominy's data image can be streamed and supervised by the user. Figure 7 highlights the GUI and the information from the camera.

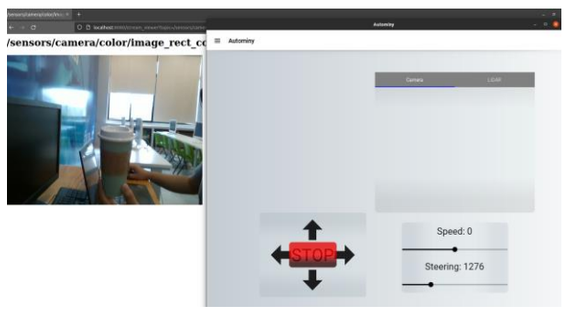


Figure 7 User Interface
Source: (Own Source)

ISSN: 2444-3204

A slider of speed and steering were created, and until the user releases up the dial the data string command is stored in the memory, then when the user presses the forward button the command string is sent.

Likewise, a condition is implemented, this is to prevent a malfunction or decontrol. Only the data string can be sent by pressing the forward button if the speed of the slider is greater than zero. Conversely if the speed value is less than zero, the command will only be sent by pressing the backwards button.

In regards to the steering commands, the center of the slider represents a zero degree direction. and the same speed logic is applied.

In any event, if the stop button is pressed, it will kill all the processes that are running. This application is running from a python file in a computer, but easily can be converted into a portable device.

Equally Important, the clustering algorithm results are also interesting. As mentioned before, information from the depth camera was read and processed; passing through a downsampling method and finally a clusterization technique. An experiment with different parameters took place, changing the leaf size of the voxel grid downsampling filter.

The first test was set by a leaf size of 1 cm in relation to the depth camera measurement. The cluster tolerance was set at approximately 1 cm. As shown in Figure 8 the clustered point clouds are overlapped in the ROS camera topic, this is because of better visualization of the results. Clearly, a perceptive delay is present, which is due to the latency of sending the point cloud to the computer and sending back to the Autominy's processor. When the program was compiled in the Autominy's CPU, the struggle was significantly decreased.

It is important to mention the fact of setting the cluster tolerance approximately at 1 cm, almost the objects are perfectly separated into important clusters. But if this element changes, the resolution of wrapping objects can considerably enhance. It is important to mention the cluster colors were set randomly and do not have any relation to the distance of the elements.

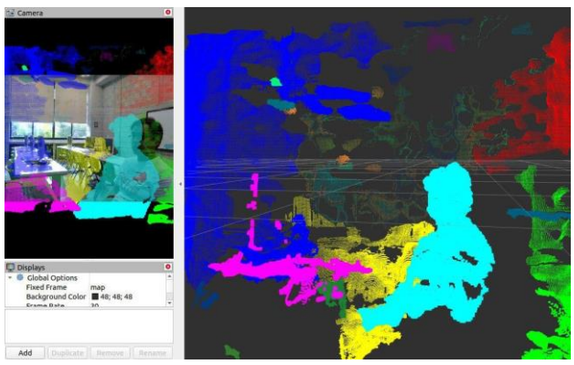
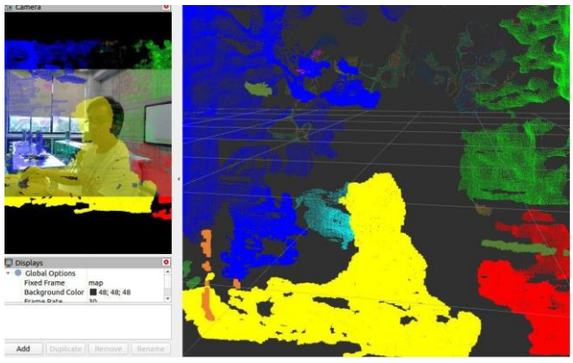
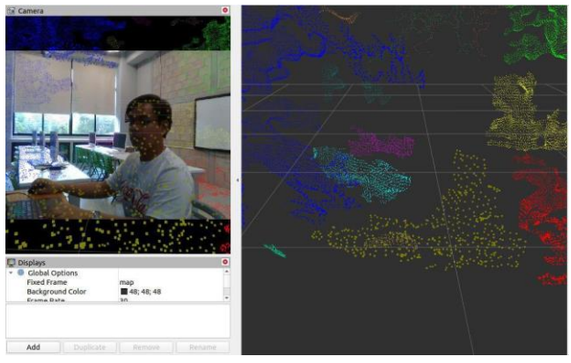


Figure 8 Segmented point cloud
Source: Own elaboration

In the next Figure 9 a better representation of the downsampling can be shown. In this experiment a leaf size of 0.06 cm and 1 cm was applied.



(a) Segmented point cloud with a 1 cm of leaf size



(b) Segmented point cloud with a 0.06 cm of leaf size

Figure 9 Segmented point cloud
Source: Own elaboration

Acknowledgments

To the Universidad Politécnica de Yucatán for providing the platform and the installations in which this work could take place.

Conclusions

To summarize, a graphical user interface was created using a framework called KivyMD, in which one ROS command is sent via ROS nodes. The user can watch in real time what the Autominy's camera is streamed via web server.

ISSN: 2444-3204

In addition, a clustering algorithm was implemented to enhance the perception of the robot, and to implement a different method instead of the camera. The Point Cloud Library was used for this work and implemented to ROS. Techniques of Downsampling and Kd-Tree were executed.

Future Directions

For future work, this clustering method can concur in spatial recognition of these clusters, and complete diverse challenges such as the parking or navigation with objects in movement. Also a certain intelligence can be provided to the Autominy, this if the point clouds segmented are also passing through a neural network which can be identified and classified, for example pedestrians.

Funding by

This work has been funded by CONACYT [005817 - MAESTRÍA EN INGENIERÍA, 2022]

References

Alomari, K., Mendoza, R. C., Sundermann, S., Goehring, D., & Rojas, R. (2020). Fuzzy Logic-based Adaptive Cruise Control for Autonomous Model Car. In ROBOVIS (pp. 121-130).DOI: 10.5220/0010175101210130

Autominy. (n.d.). Autominy core. https://autominy.github.io/AutoMiny/docs/autominy- core/ (accessed: 05.04.2022**)**

BORGES-MONSREAL, G., GARCIA-LOPEZ, R., & MOLINA-CESPEDES, J. (2021) Autonomous Navigation of an Autominy based on the" Sliding Window Technique" and 2D detection. DOI: 10.35429/S.2021.1.3.7.16

Bentley, J. L. (1975). Multidimensional binary search trees used for associative searching. Communications of the ACM, 18(9), 509-517. DOI: 10.1145/361002.361007

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. ICRA workshop on open source software, 3(3.2), 5.

Rusu, R. B., & Cousins, S. (2011). 3D is here: Point Cloud Library (PCL). IEEE International Conference on Robotics and Automation (ICRA). DOI: 10.1109/ICRA.2011.5980567

Wen, L., He, L., & Gao, Z. (2019). Research on 3d point cloud de-distortion algorithm and its application on euclidean clustering. IEEE Access,7,86041–86053. DOI:10.1109/ACCESS.2019.2926424

Yamaguchi, K., Kunii, T., Fujimura, K., & Toriya, H. (1984). Octree-related data structures and algorithms. IEEE Computer Graphics and Applications, 4(01), 53-59. DOI: 10.1109/MCG.1984.275901.

ISSN: 2444-3204