Escuela Superior Politécnica del Litoral

Mobile and Articulated Robots

Final Report

Enhanced 3D Mapping and Autonomous Navigation: Oak-D Camera and Lidar Integration on Jackal Robot with Pan & Tilt

Members:

Martín Sebastián Aguilar Zambrano

André Alberto Aguirre Apolo

Sebastián Alejandro Hidalgo Sánchez

Milton Josué Sánchez Veliz

Group: 10

Date of presentation:

August 31st, 202

Academic term:

2023-2024 PAO 1

Contents

1. Summary

This project simulates a versatile mobile robot with a LiDAR sensor and a pan-tilt camera module, providing a virtual testing ground for optimizing robotic systems in complex environments.

The integration of these components is a significant leap in robotic perception, enabling dynamic field of view adjustments and precise spatial data capture.

The goal is to create a flexible robotic platform for safe and efficient infrastructure inspections like bridges, tunnels, pipelines, and power lines. The LiDAR sensor and camera identify structural issues and facilitate 3D models for planning.

Simulated robots offer cost-effective, risk-free development iterations and realistic scenario exploration.

The literature review highlights advancements like 3D mapping with RGB-D cameras, autonomous navigation, LiDAR sensors, and computer vision in aquaculture.

Methodology covers the Jackal robot's design, its adaptability for research, and key libraries like Gmapping for LIDAR-based mapping. It also discusses the kinematic and dynamic model of the Jackal robot and the WidowX XM430 Pan & Tilt system's capabilities.

Added code snippets illustrate essential equations for point transformation, occupancy grid mapping, point cloud fusion, and SLAM.

The current research emphasizes the Jackal robot's robust navigation and mapping capabilities, thanks to advanced sensors. Recommendations include understanding kinematic constraints, addressing wheel slip, and designing smoother trajectories for optimized robot performance. Achievements include 2D and 3D mapping, obstacle detection, and dynamic obstacle avoidance, showcasing the synergy between sensing technology and intelligent decision-making in robotics.

Keywords: Robotic perception, LiDAR sensor, 3D mapping, infrastructure inspection, simulated robot

2. Introduction

In the era of rapidly advancing technology, robotics has emerged as a transformative field with the potential to revolutionize industries, automate tasks, and enhance human capabilities. One of the key challenges in robotics is developing robots that can seamlessly navigate and interact with complex and dynamic environments. To address this challenge, this project proposal focuses on the simulation of a mobile robot integrated with a pan and tilt mechanism with a LiDAR camera module, offering a virtual platform to test, refine, and optimize robotic systems.

This mentioned integration marks a significant advance in robotic sensing and perception. This combination grants the robot the ability to dynamically adjust its field of view, capturing detailed spatial data and generating accurate representations of its surroundings. Such an innovation holds the promise of elevating robotics to new heights of adaptability and functionality, making it an indispensable tool for an array of applications across multiple domains.

3. Problem statement

3.1.-Specific Problem or Application

The specific problem at hand is the need for a versatile and adaptable robotic platform that can navigate, perceive, and interact effectively within intricate and ever-changing surroundings. In the proposed project, the goal is to improve and make infrastructure inspection much safer for structures such as bridges, tunnels, oil pipelines, and power lines. The LiDAR sensor, along with the camera, can identify damages or anomalies in the structures. Additionally, a mapping of the infrastructures can be created to obtain a 3D model and bring it into software where changes can be made before actual construction takes place.

3.2.-Why is design needed?

The design of a simulated mobile robot equipped with a pan and tilt LiDAR camera module is crucial to overcome the limitations of physical prototyping and real-world testing. While physical robots entail significant costs, time investment, and potential safety risks during development and testing, simulated mobile robots offer a controlled, repeatable, and cost-effective alternative. These benefits are detailed below:

1. **Efficient Development Iterations:** The virtual environment enables rapid design iterations without the constraints of physical manufacturing and assembly. This agility accelerates the development timeline, allowing for quicker refinement of algorithms and behaviors.
2. **Risk-Free Testing:** Simulated environments provide a safe space for testing and experimentation, eliminating the risk of damaging hardware or endangering human operators. This risk-free testing is crucial for fine-tuning algorithms and strategies.
3. **Realistic Scenario Exploration:** The simulated robot's pan and tilt LiDAR camera module captures the intricacies of the physical world, enabling the exploration of complex scenarios, such as navigating cluttered spaces, mapping intricate structures, and adapting to dynamic changes in the environment.

4. Literature review

4.1.- 3D mapping using RGB-D cameras.

An innovative system is currently being made that robustly generates highly accurate maps using an RGB-D camera. What stands out about this approach is its independence from additional sensors or odometry. With the availability of affordable and lightweight RGB-D sensors like the Microsoft Kinect, this method is applicable to small domestic robots such as vacuum cleaners as well as flying robots like quadcopters. Moreover, this system can also be used for free-hand reconstruction of detailed 3D models without movement restrictions. Alongside the system description, a comprehensive experimental evaluation is presented using a publicly available benchmark dataset. The impact of various parameters, such as the choice of feature descriptor, the number of visual features, and validation methods, is analyzed and discussed. The results of the experiments demonstrate the system's robust ability to handle challenging scenarios, such as rapid camera movements and feature-poor environments, while remaining fast enough for real-time operations. Furthermore, this system is available as open-source and has already been widely embraced by the robotics community [1].

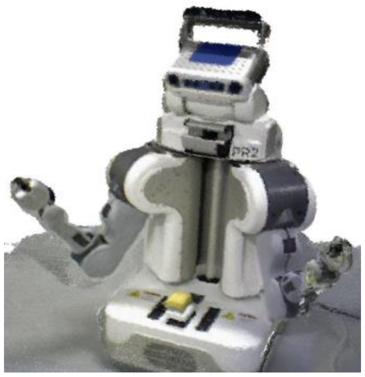


Figure 1. Top: Occupancy voxel map of the PR2 robot. Voxel resolution is 5 mm. Occupied voxels are represented with color for easier viewing. Bottom row: A sample of the RGB input images.

4.2.-Autonomous navigation in mobile robots

Navigating through unstructured environments is a fundamental skill for intelligent beings and is therefore a central focus of several research. Navigation is a complex task that hinges on the development of an internal spatial representation grounded in identifiable landmarks and robust visual processing. This representation must concurrently support continuous self-localization ("I am here") and a representation of the destination ("I am going there"). Recent advancements in Artificial Intelligence (AI) and related technologies offer promising avenues to achieve this. The increasing deployment of robots in the manufacturing industry is a clear trend, and autonomous robots have the potential to streamline a wide range of labor-intensive tasks in factory settings, thus enhancing productivity.

Realizing an autonomous multifunctional robotic platform presents several technical challenges. Some investigations are primarily aimed at addressing the core issue of enabling robots to navigate autonomously within factory environments. The robotic platform should be capable of recognizing markers on the factory floor and autonomously following a designated path from point A to point B while avoiding obstacles. To achieve this, the number of sensors is minimized to reduce the Bill of Materials (BOM) cost and maximize battery life. Cameras (RGB), motor encoders, and a cost-effective IMU for robot localization are used, along with an electric drive system for propulsion. Additionally, it is necessary to employ neural networks to identify markers and paths within the factory environment, Simultaneous Localization and Mapping (SLAM) techniques for robot localization, and a navigation algorithm to guide the robotic platform to its destination [2].

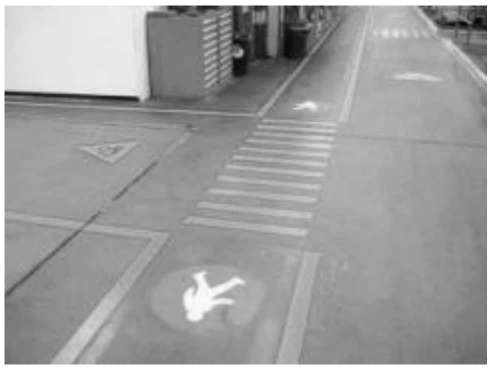


Figure 3. Camera image

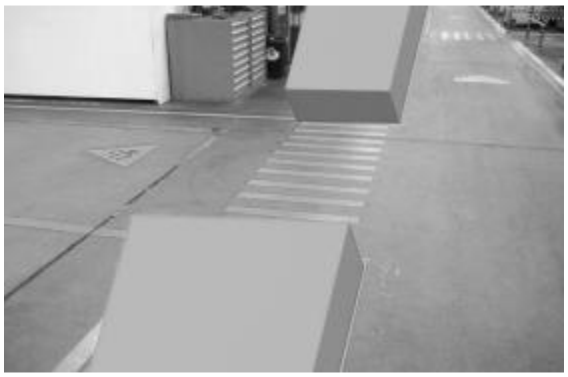


Figure 2. Augmented objects over camera image

4.3.-LIDAR sensors in robotics

In situations where human safety is at risk, and environmental hazards are a concern, the utilization of robots emerges as a viable solution to mitigate these challenges. These robots rely on a multitude of sensors to ascertain a clear path and precisely determine their position. Nonetheless, traditional sensors come with inherent limitations, such as restricted detection ranges, limited spatial resolution, and intricate data processing requirements.

Currently, there are investigations where an autonomous mobile robot has been meticulously developed, equipped with a Light Detection and Ranging (LiDAR) sensor to effectively navigate around obstacles. The robot's movement is guided by the Braitenberg vehicle strategy, a well-established approach in the field. The entire system, encompassing sensor data collection and control algorithms, is implemented on a single computing board, specifically the Raspberry Pi 3.

The experimental findings substantiate the effectiveness of the LiDAR sensor, showcasing consistent distance measurements that remain unaffected by object color or variations in ambient light intensity. The mobile robot adeptly avoids obstacles of varying sizes and colors, demonstrating its robust autonomous navigation capabilities. Importantly, it can safely traverse within enclosed spaces without any adverse impact on walls or obstacles, highlighting its potential for use in real-world scenarios [3].

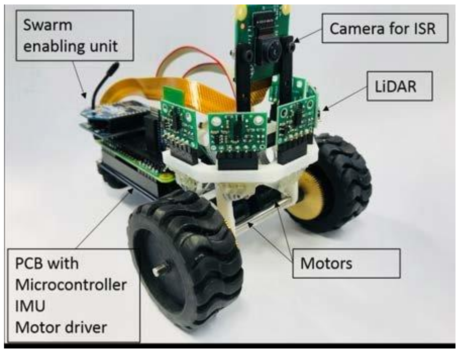


Figure 4. LiDAR sensor used in a Orion Robot

4.4.-Computer vision systems for robotics

Some papers introduce novel, cost-effective solutions for automating various tasks within fish farming operations. Specifically, computer vision technology is leveraged to facilitate non-contact estimation of fish weight. The approach employs synchronized convergent stereo vision systems to achieve 3-D segmentation of fish within tanks and sea cages. To address variations in illumination, a series of preprocessing algorithms is implemented.

In the fish 3-D segmentation process, specific fish characteristics are identified in both images. Once these points are successfully detected and verified in both images, it is necessary to employ stereo vision matching principles to achieve accurate 3-D segmentation of the fish. Estimating fish weight is accomplished using well-established length-weight relationships commonly used in the field of aquaculture [4].

Figure 5. Computer vision based individual fish identification using skin dot pattern.

4.5.-Sensor integration in robots

The position and orientation for mobile robots are determined by information collected from multiple sensors. In different papers, the approach is to integrate the information from odometry with the inertial system using Unscented Kalman Filter (UKF). The UKF is the newest extension of the widely used estimation method, Kalman Filter. The UKF is more accurate and simpler than the extended Kalman filter applied to nonlinear systems [5].

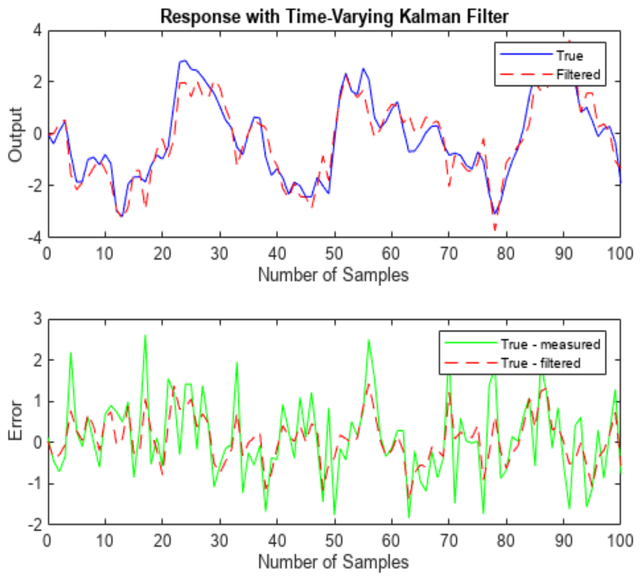


Figure 6. Response with time varying Kalman Filter

5. Methodology

5.1.-Four-Wheeled Jackal Robot

The Jackal is a versatile four-wheeled mobile robot developed by Clearpath Robotics. It's designed to provide researchers and developers with a platform for various robotics applications, including mapping, navigation, and exploration. The four-wheeled design contributes to its stability and mobility in a wide range of environments.

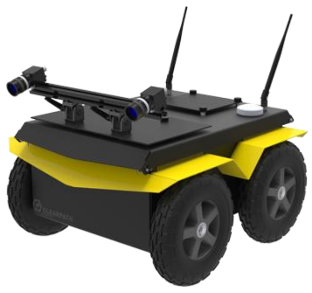


Figure 7. Jackal robotic - isometric view

5.1.1.- Degrees of Freedom (DoF) of the Four-Wheeled Jackal

The four-wheeled Jackal typically has three degrees of freedom, which enables it to navigate and move in various directions effectively. The robot's mobility allows it to move forward, backward, rotate around its vertical axis (yaw), and strafe sideways. These capabilities make it suitable for tasks that require both precision and maneuverability.

5.1.2.- Previous Uses in the Industry

The four-wheeled Jackal robot has been utilized in several industries and research domains:

- **Research and Academia**: It's extensively used in academic and research environments to develop and test algorithms related to mapping, localization, and path planning.
- **Autonomous Exploration:** The robot is commonly employed for autonomous exploration tasks, such as mapping unknown environments and generating 3D models.
- **Industrial Applications:** Due to its rugged design and ability to operate in various terrains, the Jackal can be used for industrial applications like outdoor surveillance, environment monitoring, and inspection.

5.1.3.- Libraries for Robot Perception and navigation

Perception:

- LIDAR Sensor: ROS provides packages like the "laser_geometry" package for converting LIDAR point cloud data into usable formats. The "pointcloud_to_laserscan" node can convert point clouds into laser scans for navigation and mapping.
- Disparity Camera (Stereo Vision): OpenCV can also be used for working with stereo camera data. You can leverage stereo calibration and depth map generation functions to extract depth information from the stereo camera setup.

Navigation:

For jackal navigation, we use Gmapping library, specifically focuses on mapping using laser range data from LIDAR sensors. Here's a brief summary of how the "gmapping" library facilitates robot navigation:

5.1.4.- Gmapping Library for Robot Navigation

- **Localization**: Gmapping starts by estimating the robot's initial position within the environment. This is achieved by comparing the robot's sensor data (LIDAR scans) with the existing map. The particle filter algorithm is often used for localization, allowing the robot to approximate its position based on sensor observations and movement history.
- **Mapping**: Once the robot's initial pose is estimated, it starts moving through the environment. Gmapping simultaneously builds a map of the environment using the LIDAR data. The map is represented as a grid-based occupancy map, where each cell indicates whether the area is occupied or free based on sensor readings.
- **Scan Matching:** Gmapping performs scan matching, a process of aligning the current LIDAR scan with the previous scans in the map. This aligns the new sensor data with the existing map to improve the accuracy of the map and the robot's position estimate.
- **Loop Closure**: As the robot navigates, it may encounter areas it has previously visited. Gmapping detects these situations and attempts to close loops in the map. Loop closure ensures that the map remains consistent and prevents the accumulation of errors over time.
- **Occupancy Grid Update:** The occupancy grid is continuously updated as the robot moves and collects more sensor data. The map evolves over time to reflect changes in the environment.
- **Path Planning and Navigation:** Once the map is built and the robot's position is accurately estimated, the generated map can be used for path planning and obstacle avoidance. The robot can plan a path through the environment using the map to avoid obstacles and reach its destination.

5.1.5.- Jackal Measures and Kinematic equations

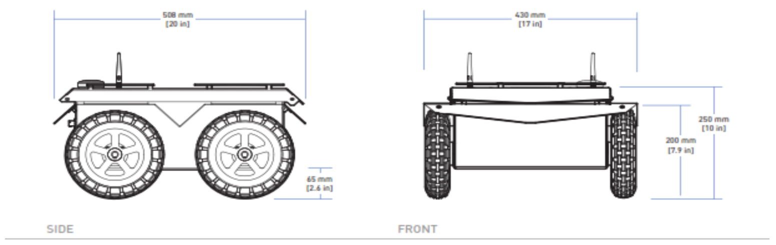
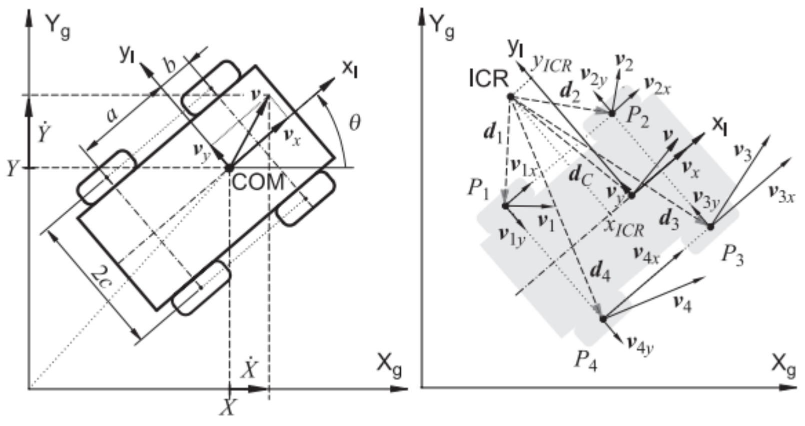


Figure 8. Jackal measures

For the jackal robot, we need to obtain the kinematic and dynamic model for a differential four wheeled robot with rear traction: The formulation assumes that the robot moves on a planar surface. The robot's structure consists of four wheels (two wheels on the right side and another two wheels on the left side). The robot's movement is based on a differential drive mechanism, where the left and right wheels are organized independently. First, assume the robot is moving on two-dimensional plane with inertial coordinate frame (Xg, Yg) as portrayed.



(a) Free-body kinematics (b) Wheel velocities relationships

From Fig. 1(a) it is easy to derive kinematic equation of motion using rotation matrix as follows:

$$\dot{\boldsymbol{q}} = \begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ \omega \end{bmatrix}$$

Where q˙ ∈ R 3 is the generalized velocity vector and ω denotes angular velocity of the vehicle.

To complete the kinematic model of Jackal additional velocity constraints should be considered with respect to the inertial frame. According to geometrical situation presented in Fig. (b) one can prove that coordinates of velocities of points P1, P2, ... , P4 where the wheels of the robot touch the plane must satisfy:

$$vL \triangleq \ v_{1x} = v_{2x} \ , vR \triangleq \ v_{3x} = v_{4x},$$

$$vR \triangleq v_{2y} = v_{3y}\ ,\ vB \triangleq\ v_{1y} = v_{4y}$$

where vL, vR denote longitudinal coordinates of left and right wheel velocities, vF and vB, are lateral coordinates of velocities of front and rear wheels, respectively.

Remark: From Fig. 1(b) it is clear that viy is equal to zero for straight motion only (i.e. if ω = 0), otherwise viy ≠ 0 that implies lateral skid that is necessary to change orientation of such vehicle.

Notice that ωL and ωR which denote angular velocities of left and right wheels, respectively, can be regarded as control inputs at kinematic level and can be used to control longitudinal and angular velocity according to the following relationships:

$$v_x = r\frac{w_L + w_R}{2},\ \ w = r\frac{-w_L + w_R}{2c}$$

while r is so called effective radius of wheels, and 2c is a spacing wheel track depicted in Fig. 1(a).

***Remark:** Is very important to note that equations are valid only if longitudinal slip does not appear, otherwise they should be treated as just approximations and to improve accuracy parameters c and r should be identified experimentally.*

5.2.-WidowX XM430 Pan & Tilt

The WidowX XM430 Robot Turret is a fully programmable pan and tilt system designed around the DYNAMIXEL XM-430-W350-T servos from Robotis. The servos offer a high resolution of 4096 positions and user definable PID parameters. Temperature monitoring, positional feedback as well as voltage levels, load and compliance settings are all user accessible as well. The WidowX XM430 Turret comes with the DYNAMIXEL U2D2 for ROS control and the CM9.04c for Serial Firmware control.

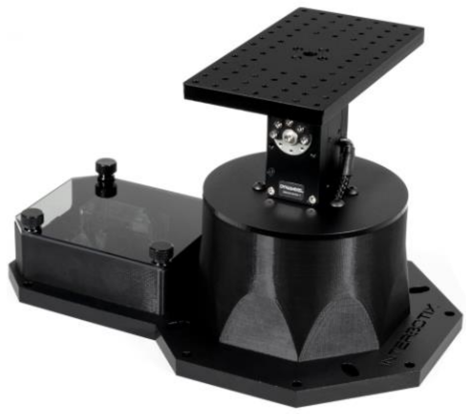


Figure 9: WidowX XM430 Pan & Tilt - Isometric View

5.2.1.- Degrees of Freedom (DoF) of the WidowX XM430 Pan & Tilt

As the WidowX XM430 Pan & Tilt Robot has a Pan & Tilt system shown in its name, it is going to have only 2 degrees of freedom: horizontal rotation and a vertical rotation, represented by pan and tilt respectively. For more details:

- **Pan (Horizontal Rotation):** This axis allows rotation around a vertical axis, allowing the system to sweep from left to right or vice versa. This axis provides one degree of freedom.
- **Tilt (Vertical Rotation):** This axis allows rotation around a horizontal axis, enabling the system to move up and down. This axis also provides one degree of freedom.

These two rotational axes provide the necessary freedom of movement to position the system's camera or sensors in various directions within a 3D space.

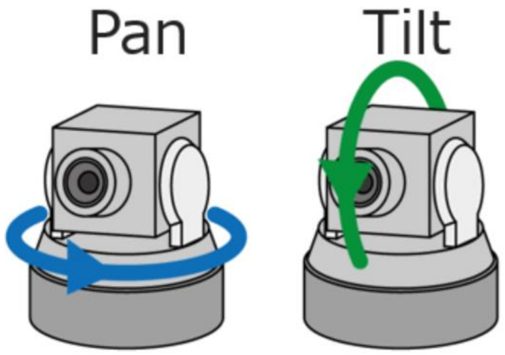


Figure 10: Pan & Tilt movement represented by a camera.

5.2.2.- Previous Uses in the Industry for a Pan & Tilt system

A robot with a Pan & Tilt System has had been utilized for many applications such as:

- **Pan-tilt-zoom cameras:** It can be an essential part of modern surveillance systems. They can direct the attention to suspicious events.
- **Predictive tracking of an object:** This concept revolves around the synergy between the mechanical movement of pan and tilt systems and the predictive algorithms that enable these mechanisms to anticipate an object's future position.

5.2.3.- Libraries and concepts for 3D-Mapping using the Oak-D camera.

Before reviewing some of the libraries that the present project used to do 3D-Mapping, the Oak-D camera will be described as the component who can make the 3D-Mapping possible.

The OAK-D baseboard has three on-board cameras which implement stereo and RGB vision, piped directly into the OAK SoM for depth and AI processing. The data is then output to a host via USB 3.1 Gen1 (Type-C).

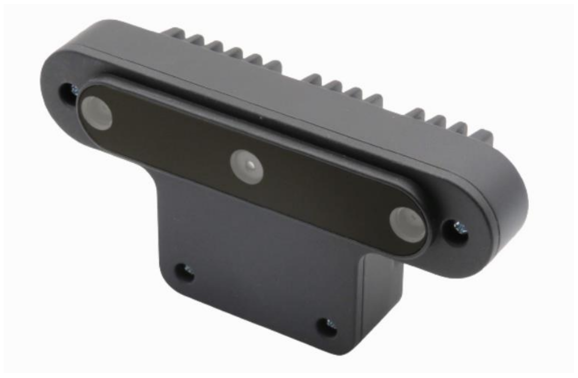


Figure 11: Oak-D Camera

5.2.4.- OctoMap

The OctoMap library for ROS (Robot Operating System) is a powerful and widely used mapping framework that focuses on generating and managing 3D occupancy maps. These maps are constructed by representing the environment as a collection of voxels (3D pixels) and determining whether each voxel is occupied by an obstacle or free space.

Some of the key features and functionalities of the OctoMap library for ROS include:

- **Occupancy Mapping:** OctoMap creates 3D occupancy maps by representing each voxel as a probabilistic value indicating the likelihood of occupancy. This probabilistic representation allows for more nuanced and informative maps compared to binary occupancy grids.
- **Volumetric Representation:** The library builds a volumetric representation of the environment, capturing obstacles and structures in 3D space. This enables robots to navigate through complex environments, avoiding collisions and planning optimal paths.
- **Dynamic Updating:** OctoMap can handle dynamic environments by updating the occupancy probabilities as new sensor data becomes available. This makes it suitable for scenarios where the environment changes over time.
- **Integration with ROS:** OctoMap is specifically designed for seamless integration with the Robot Operating System (ROS). ROS nodes can subscribe to sensor data, process it using the OctoMap library, and publish the resulting maps or other relevant information for navigation and decision-making.
- **Mapping and Localization:** OctoMap is often used for simultaneous mapping and localization (SLAM), where a robot constructs a map of its environment while simultaneously estimating its own position within that map.

5.2.5.- Homogeneous Transformation Matrix:

The Homogeneous Transformation Matrix is used to get a point's position in a 3D space (x, y, z) respect with an established frame [A], that at first was respective with another frame [B].

$$^{A}p = {}^{A}_{B}T\,{}^{B}p$$

Where ^{B}p is the point's position respect with frame B, $^{A}_{B}T$ is the Homogeneous Transformation Matrix from frame B to A and ^{A}p is the point's position respect with frame O.

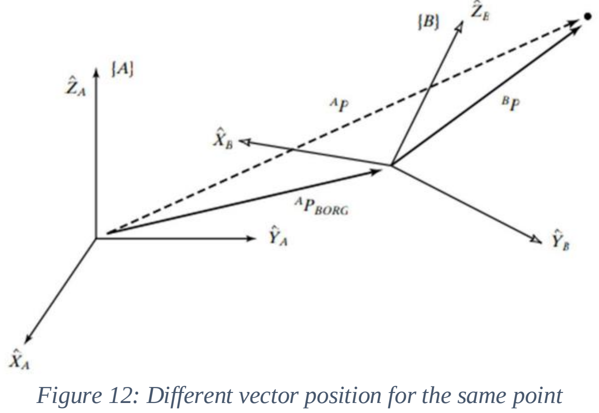


Figure 12: Different vector position for the same point

The structure of a Homogeneous Transformation Matrix is:

$$\left[\begin{array}{ccc|c} \cos\theta_n & -\sin\theta_n\cos\alpha_n & \sin\theta_n\sin\alpha_n & r_n\cos\theta_n \\ \sin\theta_n & \cos\theta_n\cos\alpha_n & -\cos\theta_n\sin\alpha_n & r_n\sin\theta_n \\ 0 & \sin\alpha_n & \cos\alpha_n & d_n \\ \hline 0 & 0 & 0 & 1 \end{array}\right] = \left[\begin{array}{ccc|c} & R & & T \\ \hline 0 & 0 & 0 & 1 \end{array}\right]$$

Figure 13: Homogeneous Transformation Matrix structure

Where T and R are the translational and rotational vector respectively from one frame to another.

5.2.6.- Occupancy Grid Mapping Equation:

Occupancy grid mapping represents the occupancy of space as a grid of cells. The occupancy value can be binary, indicating whether the cell is occupied or not.

$$p(m \mid z, x) = p(m \mid z', x) * p(z \mid x, m) / p(z' \mid x)$$

Where:
p(m | z, x) = Probability of occupancy given sensor measurements and pose
p(m | z', x) = Previous occupancy probability
p(z | x, m) = Probability of measurements given pose and map
p(z' | x) = Probability of measurements given previous pose

5.2.7.- Point Cloud Fusion Equation:

When fusing multiple point clouds, you can combine them using a weighted averaging scheme:

$$Fused\_Point = (w1 * Point\_Cloud\_1) + (w2 * Point\_Cloud\_2) + \ldots + (wn * Point\_Cloud\_n)$$

Where:
Fused_Point = Resulting fused point cloud
Point_Cloud_1, Point_Cloud_2, ..., Point_Cloud_n = Individual point clouds
w1, w2, ..., wn = Weights assigned to each point cloud (sum of weights is 1)

5.2.8.- SLAM Equation (Simplified):

SLAM involves estimating the map `m` and the robot's path `x` simultaneously:

$$p(m, x \mid z) = p(x \mid z, m) * p(m)$$

Where:
p(m, x | z) = Joint probability of map and path given sensor measurements
p(x | z, m) = Probability of path given measurements and map (posterior)

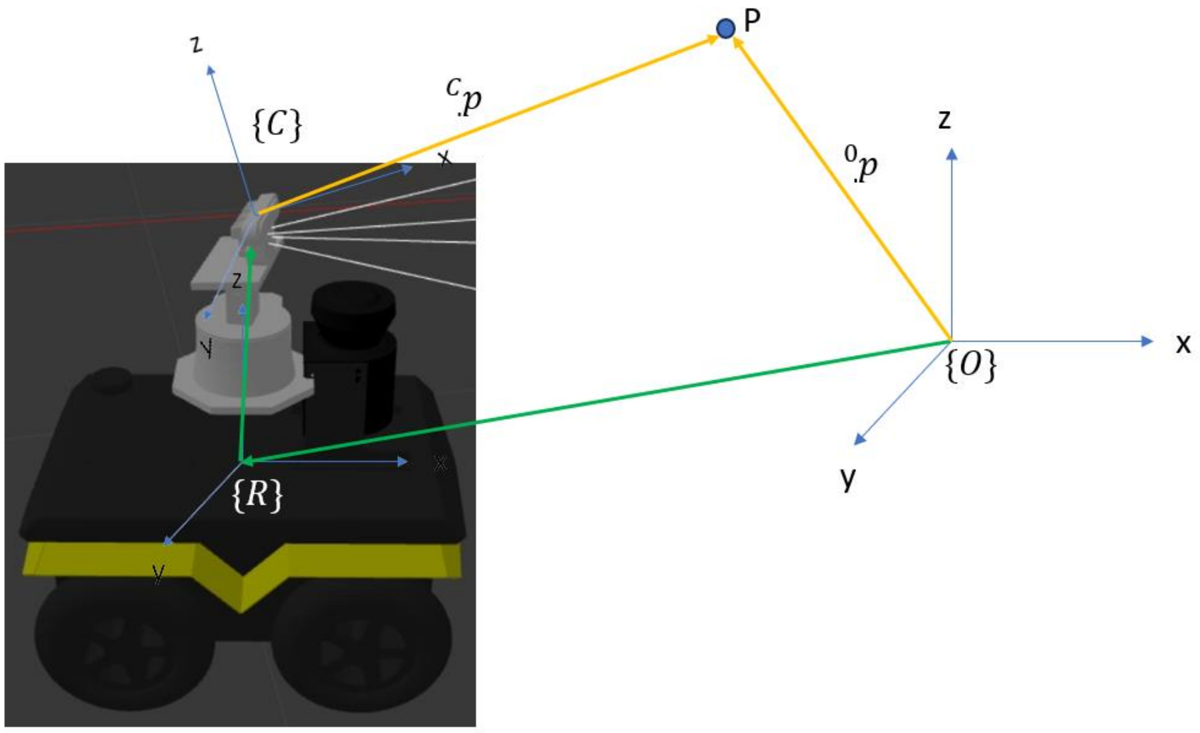
p(m) = Prior probability of the map

Overall, the OctoMap library in ROS serves as a fundamental tool for robotic perception, environment modeling, and navigation in 3D space. Its ability to create accurate and probabilistic 3D occupancy maps makes it an asset for a wide range of robotic applications, from autonomous vehicles to industrial robots.

Octomap in the present proyect takes disparity maps by Gazebo Plugins, and also the odometry to model a three-dimensional space.

5.2.9.- Pan & Tilt Control

The Oak-D camera has always to look at a desired point. As the Oak-D camera depends only on the Pan & Tilt system, the objective is to find a delta pan angle and a delta tilt angle to determinate the control of the joints.
To do this, some homogeneous transformation matrixes were used. Let's first see the representation of all the vector and references frames presented in the problem:



Let:

${}^{R}_{O}\boldsymbol{T} = \begin{bmatrix} {}^{R}_{O}\boldsymbol{R} & {}^{R}_{O}\boldsymbol{p} \\ \boldsymbol{0}' & \boldsymbol{1} \end{bmatrix}$, *the homogeneous transformation matrix from {O} to {R}*

${}^{O}_{R}\boldsymbol{T} = \begin{bmatrix} ({}^{R}_{O}\boldsymbol{R})' & -({}^{R}_{O}\boldsymbol{R})'\,{}^{R}_{O}\boldsymbol{p} \\ \boldsymbol{0}' & \boldsymbol{1} \end{bmatrix}$, *the homogeneous transformation matrix from {O} to {R}*

${}^{C}_{R}\boldsymbol{T}' = \begin{bmatrix} {}^{C}_{R}\boldsymbol{Rz} & {}^{C}_{R}\boldsymbol{p} \\ \boldsymbol{0}' & \boldsymbol{1} \end{bmatrix}$, the homogeneous transformation matrix from {R} to {C} but only considering pan angle (1st joint rotation).

Once, the position point from the camera frame is obtained, delta pan angle ($\boldsymbol{\delta\alpha}$) and delta tilt angle ($\boldsymbol{\delta\beta}$) can be calculated by the following process:

$$ {}^{C}\boldsymbol{p} = {}^{C}_{O}\boldsymbol{T}{}^{O}\boldsymbol{p} = {}^{C}_{R}\boldsymbol{T}'{}^{R}_{O}\boldsymbol{T}{}^{O}\boldsymbol{p} = \begin{bmatrix} {}^{C}_{R}\boldsymbol{Rz} & {}^{C}_{R}\boldsymbol{p} \\ \boldsymbol{0}' & \boldsymbol{1} \end{bmatrix} \begin{bmatrix} ({}^{R}_{O}\boldsymbol{R})' & -({}^{R}_{O}\boldsymbol{R})'{}^{R}_{O}\boldsymbol{p} \\ \boldsymbol{0}' & \boldsymbol{1} \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \\ \boldsymbol{z} \end{bmatrix} $$

The tilt is not used so far, because the first main goal is to get the delta pan angle ($\boldsymbol{\delta\alpha}$) and then the delta tilt angle ($\boldsymbol{\delta\beta}$).
Let:
$\boldsymbol{\alpha_p(t)}$ and $\boldsymbol{\beta_p(t)}$ be the angles of pan and tilt respective with the jackal frame, where $\boldsymbol{\alpha_p(t)}$ is used for the rotational matrix ${}^{C}_{R}\boldsymbol{Rz}$ (z because is a rotation respect with the vertical axils).

$$ \boldsymbol{\delta\alpha = tan^{-1}}\left(\frac{{}^{C}\boldsymbol{Py}}{{}^{C}\boldsymbol{Px}}\right) $$

and:

$$ \boldsymbol{\delta\beta = tan^{-1}}\left(\frac{{}^{C}\boldsymbol{Pz}}{\sqrt{({}^{C}\boldsymbol{Py})^2 + ({}^{C}\boldsymbol{Px})^2}}\right) $$

It's good to mention that the delta pan and tilt angle present should be a little increment or decrease because the frames are changing through time, that's why it's recommended *to multiply it by a proportion that lets the control to change those increase and decrease* ***quickly.***

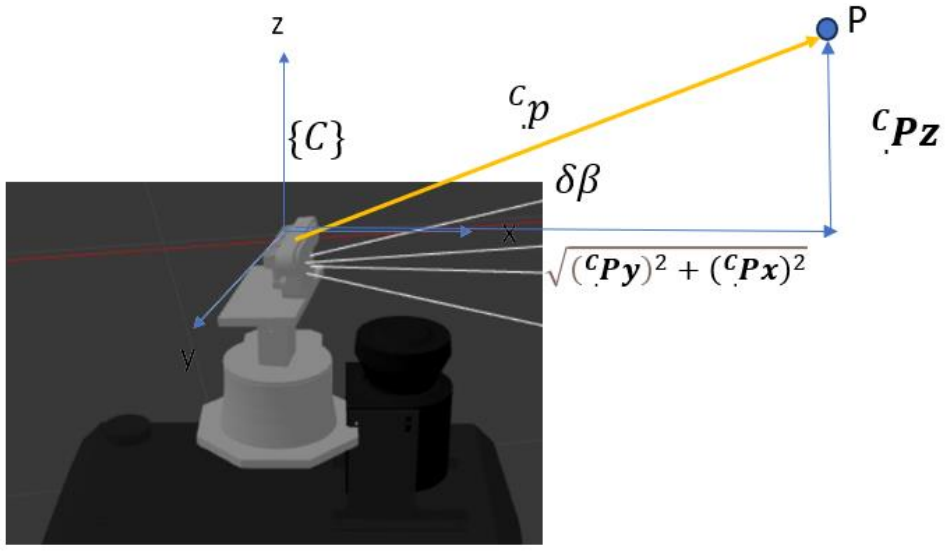
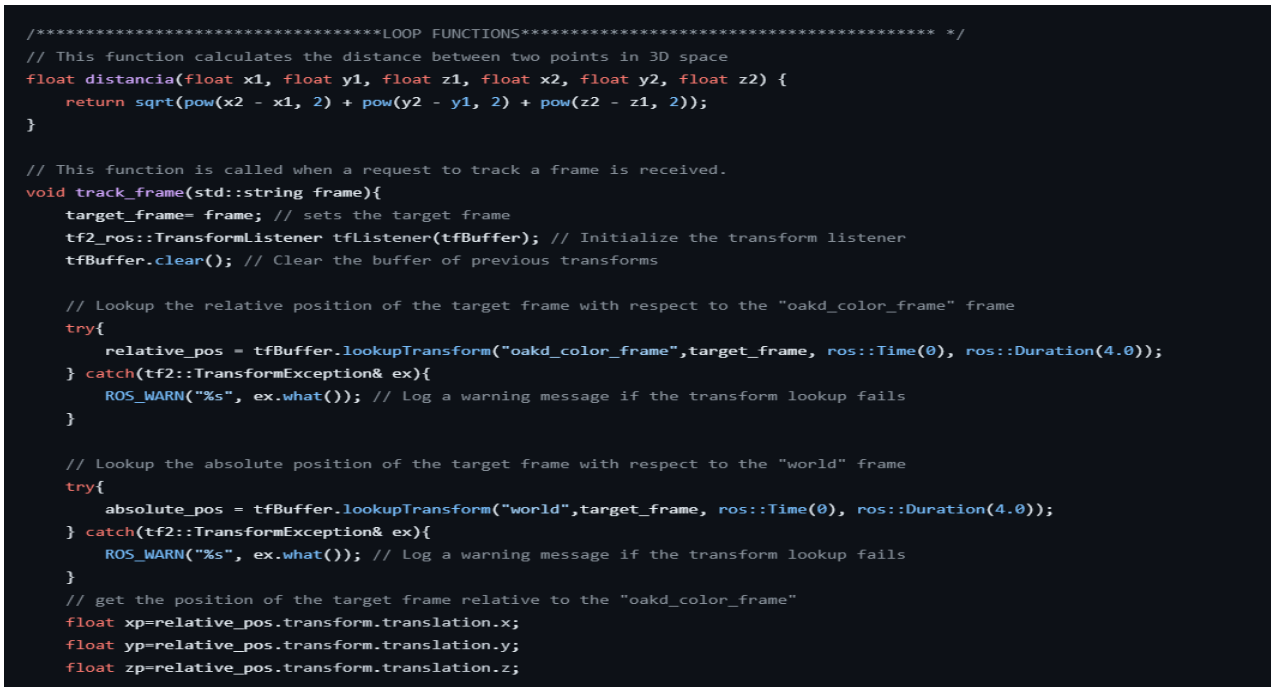
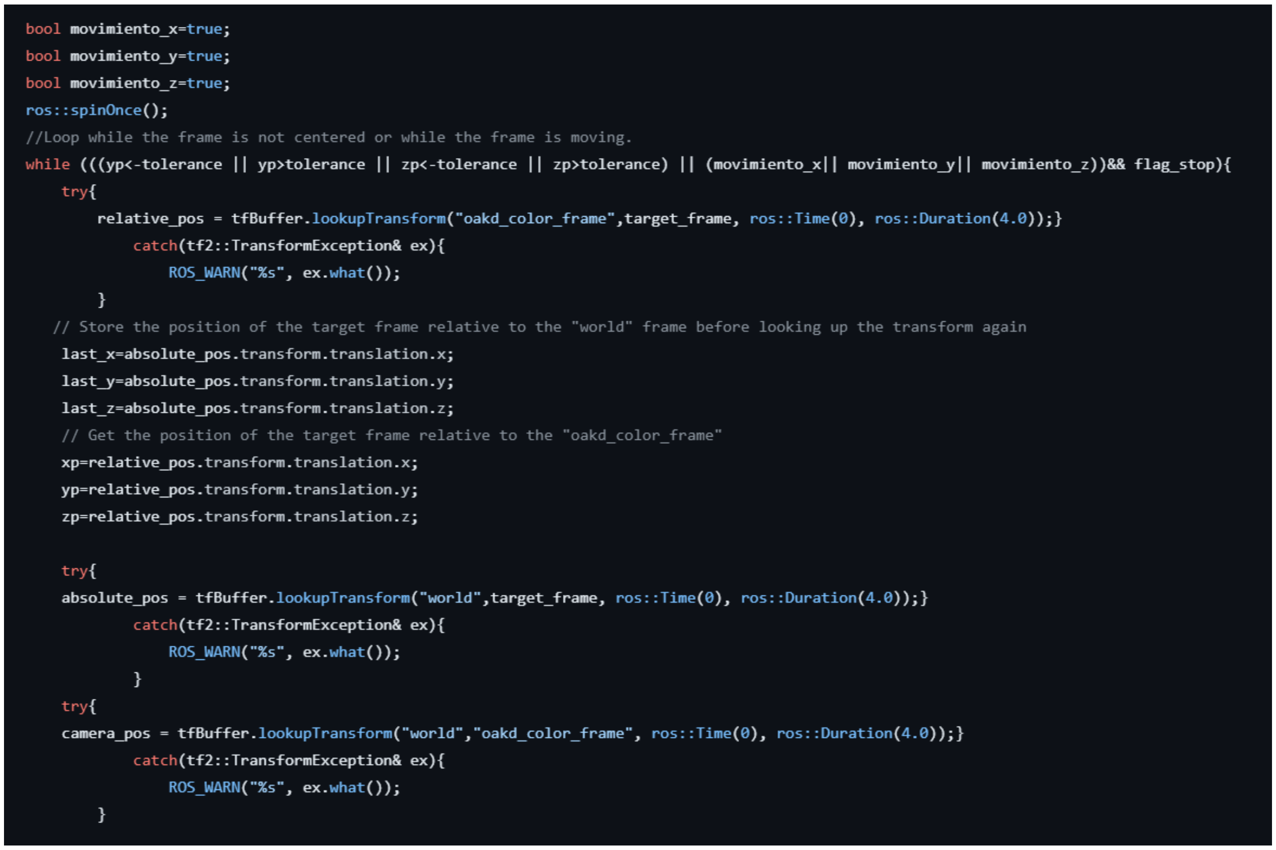

z
P
{C}
${}^{C}p$
${}^{C}\boldsymbol{Pz}$
$\delta\beta$
X
$\sqrt{({}^{C}\boldsymbol{Py})^2 + ({}^{C}\boldsymbol{Px})^2}$
Y

Figure 14: delta tilt angle geometry calculation

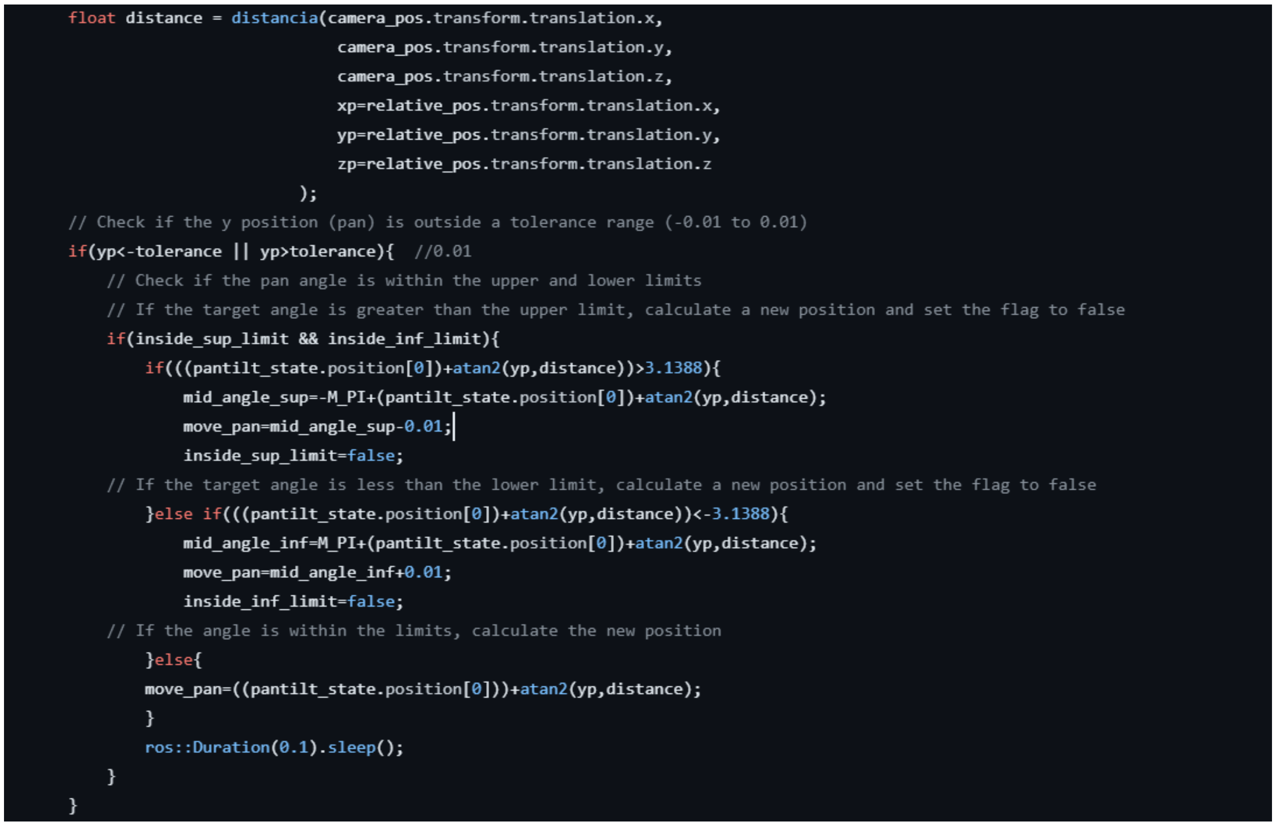
5.2.10.- *Pan & Tilt Control loop as a ROS node*



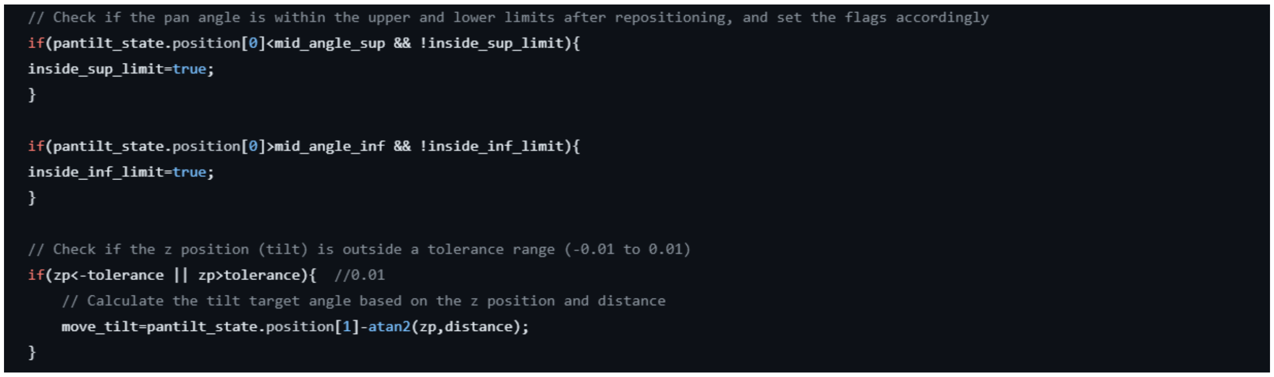
```
/************************************LOOP FUNCTIONS******************************************* */
// This function calculates the distance between two points in 3D space
float distancia(float x1, float y1, float z1, float x2, float y2, float z2) {
    return sqrt(pow(x2 - x1, 2) + pow(y2 - y1, 2) + pow(z2 - z1, 2));
}

// This function is called when a request to track a frame is received.
void track_frame(std::string frame){
    target_frame= frame; // sets the target frame
    tf2_ros::TransformListener tfListener(tfBuffer); // Initialize the transform listener
    tfBuffer.clear(); // Clear the buffer of previous transforms

    // Lookup the relative position of the target frame with respect to the "oakd_color_frame" frame
    try{
        relative_pos = tfBuffer.lookupTransform("oakd_color_frame",target_frame, ros::Time(0), ros::Duration(4.0));
    } catch(tf2::TransformException& ex){
        ROS_WARN("%s", ex.what()); // Log a warning message if the transform lookup fails
    }

    // Lookup the absolute position of the target frame with respect to the "world" frame
    try{
        absolute_pos = tfBuffer.lookupTransform("world",target_frame, ros::Time(0), ros::Duration(4.0));
    } catch(tf2::TransformException& ex){
        ROS_WARN("%s", ex.what()); // Log a warning message if the transform lookup fails
    }
    // get the position of the target frame relative to the "oakd_color_frame"
    float xp=relative_pos.transform.translation.x;
    float yp=relative_pos.transform.translation.y;
    float zp=relative_pos.transform.translation.z;
```



```
bool movimiento_x=true;
bool movimiento_y=true;
bool movimiento_z=true;
ros::spinOnce();
//Loop while the frame is not centered or while the frame is moving.
while (((yp<-tolerance || yp>tolerance || zp<-tolerance || zp>tolerance) || (movimiento_x|| movimiento_y|| movimiento_z))&& flag_stop){
    try{
        relative_pos = tfBuffer.lookupTransform("oakd_color_frame",target_frame, ros::Time(0), ros::Duration(4.0));}
            catch(tf2::TransformException& ex){
                ROS_WARN("%s", ex.what());
        }
   // Store the position of the target frame relative to the "world" frame before looking up the transform again
    last_x=absolute_pos.transform.translation.x;
    last_y=absolute_pos.transform.translation.y;
    last_z=absolute_pos.transform.translation.z;
    // Get the position of the target frame relative to the "oakd_color_frame"
    xp=relative_pos.transform.translation.x;
    yp=relative_pos.transform.translation.y;
    zp=relative_pos.transform.translation.z;

    try{
    absolute_pos = tfBuffer.lookupTransform("world",target_frame, ros::Time(0), ros::Duration(4.0));}
            catch(tf2::TransformException& ex){
                ROS_WARN("%s", ex.what());
            }
    try{
    camera_pos = tfBuffer.lookupTransform("world","oakd_color_frame", ros::Time(0), ros::Duration(4.0));}
            catch(tf2::TransformException& ex){
                ROS_WARN("%s", ex.what());
        }
```



```
    float distance = distancia(camera_pos.transform.translation.x,
                               camera_pos.transform.translation.y,
                               camera_pos.transform.translation.z,
                               xp=relative_pos.transform.translation.x,
                               yp=relative_pos.transform.translation.y,
                               zp=relative_pos.transform.translation.z
                             );
    // Check if the y position (pan) is outside a tolerance range (-0.01 to 0.01)
    if(yp<-tolerance || yp>tolerance){  //0.01
        // Check if the pan angle is within the upper and lower limits
        // If the target angle is greater than the upper limit, calculate a new position and set the flag to false
        if(inside_sup_limit && inside_inf_limit){
            if(((pantilt_state.position[0])+atan2(yp,distance))>3.1388){
                mid_angle_sup=-M_PI+(pantilt_state.position[0])+atan2(yp,distance);
                move_pan=mid_angle_sup-0.01;
                inside_sup_limit=false;
        // If the target angle is less than the lower limit, calculate a new position and set the flag to false
            }else if(((pantilt_state.position[0])+atan2(yp,distance))<-3.1388){
                mid_angle_inf=M_PI+(pantilt_state.position[0])+atan2(yp,distance);
                move_pan=mid_angle_inf+0.01;
                inside_inf_limit=false;
        // If the angle is within the limits, calculate the new position
            }else{
            move_pan=((pantilt_state.position[0]))+atan2(yp,distance);
            }
            ros::Duration(0.1).sleep();
        }
    }
```



```
// Check if the pan angle is within the upper and lower limits after repositioning, and set the flags accordingly
if(pantilt_state.position[0]<mid_angle_sup && !inside_sup_limit){
inside_sup_limit=true;
}

if(pantilt_state.position[0]>mid_angle_inf && !inside_inf_limit){
inside_inf_limit=true;
}

// Check if the z position (tilt) is outside a tolerance range (-0.01 to 0.01)
if(zp<-tolerance || zp>tolerance){  //0.01
   // Calculate the tilt target angle based on the z position and distance
   move_tilt=pantilt_state.position[1]-atan2(zp,distance);
}
```

This code snippet is responsible for controlling the pan (along the x and y axes) and tilt (along the z axis) movement of the camera based on the position of the tracked object.

- It checks if the y-position (pan) of the tracked object is outside a predefined tolerance range (-0.01 to 0.01). If so, it means that the object is in a position that requires adjustment.

- Within this conditional block, it verifies if the pan position is within the established upper and lower limits (inside_sup_limit and inside_inf_limit are true). If so, it performs the following checks:
    - If the target angle (calculated as the current position plus the angle resulting from $atan2(yp, distance)$)
      is greater than the upper limit (3.1388), it calculates a new pan position (move_pan) and sets the inside_sup_limit flag to false.

    - If the target angle is less than the lower limit (-3.1388), it calculates a new pan position and sets the inside_inf_limit flag to false.
    - If the target angle is within the limits, it calculates the new pan position.

- It checks if the pan position after relocation is within the upper and lower limits and updates the inside_sup_limit and inside_inf_limit flags accordingly.

- It checks if the z-position (tilt) of the tracked object is outside the tolerance range. If so, it calculates the tilt angle (move_tilt) based on the z position and the object's distance.

6. Results

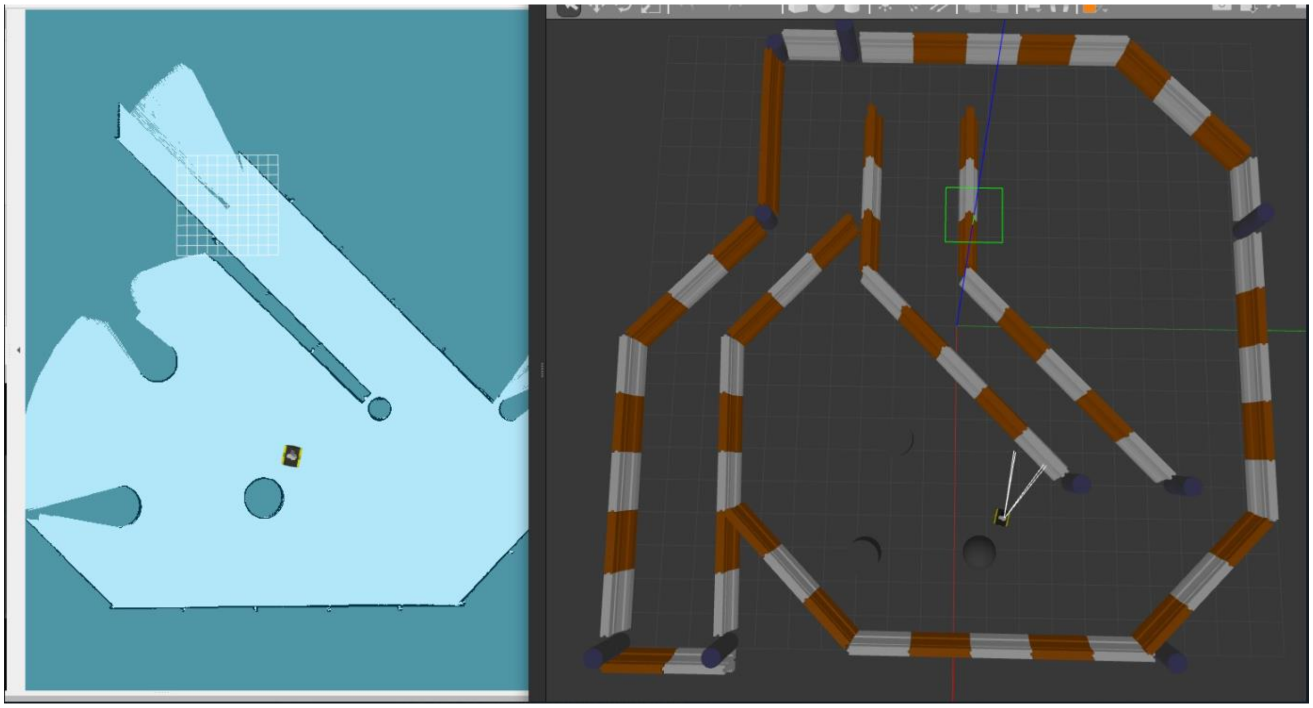


Figure 15 2D map generated with LiDAR sensor

Utilizing the capabilities of the LiDAR sensor, we successfully conducted a comprehensive 2D mapping of the surrounding environment. This achievement has paved the way for the seamless implementation of autonomous navigation for the Jackal robot. By harnessing the precise and rapid data acquisition capabilities of the LiDAR sensor, we were able to construct an accurate representation of the environment in the form of a two-dimensional map.

This cutting-edge mapping process involved the LiDAR sensor emitting laser pulses and precisely measuring the time taken for these pulses to bounce back after hitting various surfaces in the environment. As a result, we obtained an extensive collection of data points that correspond to the distances and angles at which these surfaces were encountered. The culmination of these data points facilitated the creation of a highly detailed and intricate map that vividly illustrates the physical layout of the surroundings.

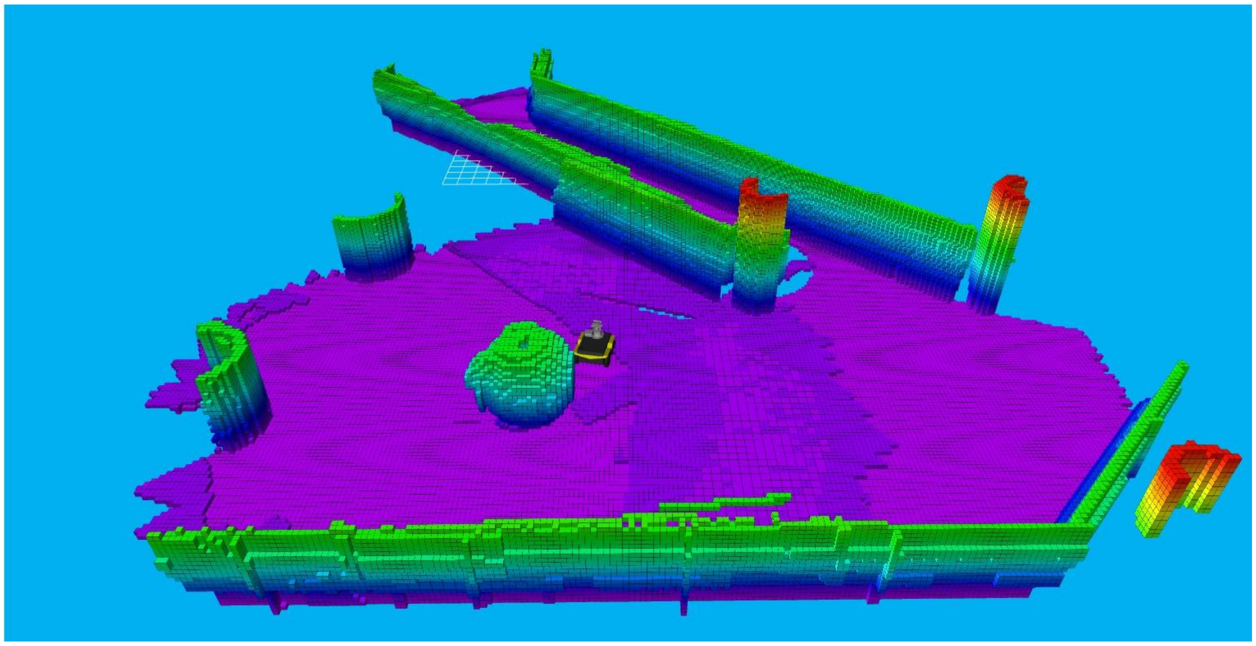
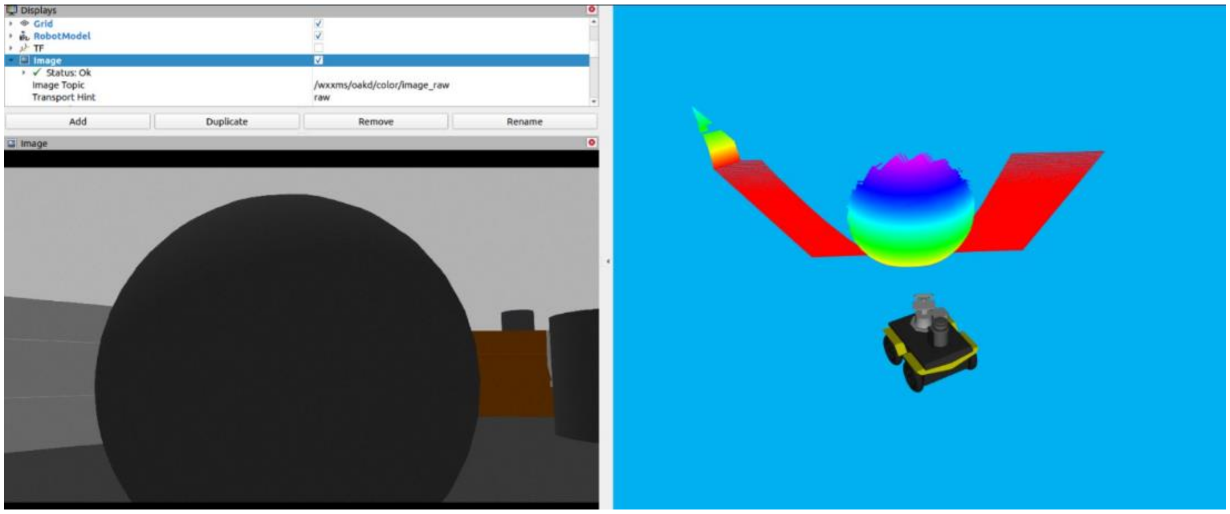


Figure 16 3D occupancy grid map

In a remarkable stride forward, we've harnessed the exceptional capabilities of the Oak-D camera equipped with Pan & Tilt functionality to achieve a groundbreaking feat: the creation of a comprehensive 3D map of the environment, complete with meticulous detailing of all obstacles present. This advanced mapping endeavor paints a vivid and faithful representation of the dynamic surroundings in which the Jackal robot operates, bringing us closer to an unparalleled level of understanding and interaction with our environment.

The Oak-D camera, a technological marvel, forms the heart of this achievement. Its cutting-edge capabilities enable it to capture the environment in full three-dimensional glory. By combining depth-sensing technology with its Pan & Tilt capabilities, the camera scans the surroundings from every conceivable angle, resulting in a rich and highly accurate representation of the environment's topography. This newfound ability to capture not only the layout but also the height and depth variations of objects and surfaces introduces an unprecedented level of detail and realism to our maps.

One of the most significant aspects of this accomplishment is the nuanced depiction of obstacles. The Oak-D camera, with its depth perception capabilities, accurately delineates the obstacles' positions, sizes, and shapes. As a result, our 3D map not only guides the Jackal's navigation but also equips us with valuable insights into potential challenges and interaction points within the environment.



```
Status: Ok
Image Topic        /wxxms/oakd/color/image_raw
Transport Hint     raw
```

Figure 17 Camera POV and 3D point cloud generated by disparity

Platforms such as Rviz were used to see in real time the information on the topics that was issued by the robot. In the same way, it can be observed how the Oak-D camera allows the identification of the obstacle and notifies the robot to take an alternative route in case it is in the middle of the destination it is intended to reach.

We employed sophisticated platforms such as Rviz to facilitate real-time monitoring of the data streams generated by the robotic system. This instrumental capability allowed us to obtain instantaneous visual feedback regarding the robot's interactions with its surroundings. By serving as an interactive window into the robot's operational environment, Rviz enabled us to closely track its maneuvers and responses as they unfolded.

Furthermore, the integration of the Oak-D camera system introduced a significant enhancement to our robotic endeavors. This advanced camera system demonstrated its proficiency by adeptly identifying obstacles within the robot's intended path. As the robot embarked on its trajectory towards the designated destination, the Oak-D camera assumed the role of a vigilant sentinel, meticulously scanning its environment. Whenever an obstacle was detected, the camera promptly relayed this crucial information to the robot's decision-making mechanism.

The pivotal moment arrived when the robot received the obstacle alert. Armed with insights from the Oak-D camera, the robot exhibited its adaptable nature by autonomously recalibrating its course of action. This proclivity for dynamic response was particularly evident when the robot encountered an obstacle en route. Rather than succumbing to impediments, the robot adeptly orchestrated an alternative pathway to bypass the obstacle while diligently maintaining its course towards the predefined goal. This seamless adjustment showcased not only the robot's autonomous capabilities but also highlighted the synergy between state-of-the-art sensing technology and intelligent decision-making processes.

7. Conclusions

- **Robust Navigation:** The Jackal robot demonstrates robust mobile navigation capabilities in a variety of environments. Its four-wheeled design, coupled with advanced sensors like LIDAR and cameras, enables it to navigate through complex terrains and obstacles with relative ease. This highlights the importance of equipping robots with advanced perception systems for successful autonomous navigation.
- **Mapping Accuracy:** By utilizing sensors such as LIDAR, the Jackal robot can create detailed maps of its surroundings. This is crucial for tasks such as environmental exploration, surveillance, and disaster response. Accurate mapping forms the foundation for effective decision-making and planning in various applications.
- **Real-World Applications:** The mobile navigation and mapping capabilities of the Jackal robot have practical applications in industries such as agriculture, mining, search and rescue, and warehouse automation. These capabilities streamline operations, reduce human intervention, and enhance safety in challenging or remote environments.

8. Recommendations

- **Understand Kinematic Constraints:** Gain a thorough understanding of the kinematic model of the robot, considering factors such as wheelbase, wheel diameter, and wheel positions. This knowledge will guide your design choices and help you implement accurate motion control algorithms.
- **Traction and Wheel Slip:** Be aware of potential wheel slip, especially when the robot is navigating on surfaces with low traction. Incorporate strategies to mitigate wheel slip, such as adjusting control inputs or adapting the robot's behavior based on feedback from sensors.
- **Smoother Trajectories:** Design control algorithms that generate smooth trajectories to prevent abrupt changes in velocity or direction. Sudden changes can lead to instability or discomfort in real-life applications.

9. Anexes

https://github.com/Martinelli219/proyectoROS

http://wiki.ros.org/Robots/Jackal

https://docs.clearpathrobotics.com/docs/robots/outdoor_robots/jackal/troubleshooting_jackal
(measurements)

https://www.trossenrobotics.com/widowx-x-series-robot-turret.aspx

https://www.penlink.se/application-notes/application-note/pan-tilt-application-note/

https://docs.luxonis.com/projects/hardware/en/latest/pages/BW1098OAK/

https://www.mdpi.com/1424-8220/15/5/9681/htm

10. References

[1] F. Endres, J. Hess, J. Sturm, D. Cremers and W. Burgard, "3D Mapping with an RGB-D Camera," IEEE transactions, 2014.

[2] S. Harapanahalli, N. O Mahony, G. Velasco Hernandez, S. Campbell, D. Riordan and J. Walsh, "Autonomous Navigation of mobile robots in factory environment," in *29th International Conference on Flexible Automation and Intelligent Manufacturing (FAIM2019)*, Limerick, 2019.

[3] D. Hutabarat, M. Rivai, D. Purwanto and H. Hutomo, "Lidar-based Obstacle Avoidance for the Autonomous Mobile Robot," in *12th Inernational Conference on Information & Communication Technology and System (ICTS),* Surabaya, 2019.

[4] J. R. Martinez-de Dios, C. Serna and A. Ollero, "Computer vision and robotics techniques in fish farms," Cambridge University Press, 2003.

[5] F. Azizi and N. Houshangi, "Sensor integration for mobile robot," in *2003 IEEE International Conference on Systems, Man and Cybernetics. Conference Theme - System Security and Assurance*, Washington DC, 2003.