Transformation of Physical Environments through Integrated Sensor Fusion and Computer Vision for Interactive Digital Recreation

Gómez, J. A.
School of Engineering and Sciences
Tec de Monterrey
Monterrey, N.L., México
A01736171@tec.mx

Munoz, L. A.
Engineering Sciences Dept.
Tec de Monterrey
Monterrey, N.L., México
amunoz@tec.mx

Podesta, M. O.
School of Engineering and Sciences
Tec de Monterrey
Monterrey, N.L., México
marcoopodesta@gmail.com

De los Ríos, G.
School of Engineering and Sciences
Tec de Monterrey
Monterrey, N.L., México
gustavodlra1999@gmail.com

***Abstract*—This paper explores the integration of computer vision algorithms and virtual simulation to enhance robots' perception and adaptability in dynamic environments. By combining real-time object detection, point cloud processing, and virtual representation, robots can accurately identify and interact with objects in their surroundings. This approach not only improves robotic autonomy but also bridges the gap between the physical and virtual worlds, enabling robust testing and validation of functionality before deployment in real-world scenarios.**

***Index Terms*—ICP, Computational Vision, Control, ROS2, Gazebo, TIAGo, Azure Kinect DK, Deph Image, Point Cloud, YOLO**

I. Introduction

The integration of advanced cameras, object detection models, and simulation environments has established a new paradigm in robotics development, enabling robots to recognize, localize, and manipulate objects with high precision in virtual and physical spaces. These advancements have been particularly effective in structured scenarios; however, the focus is increasingly shifting toward enhancing these capabilities in more dynamic and interactive environments.

For a robot to interact efficiently in real-time within its environment, it is essential not only to accurately identify and localize objects but also to predict and adapt to changes as they occur. This challenge requires the development of robust object detection algorithms, such as YOLO, and the utilization of 3D sensing technologies like Azure Kinect to create high-fidelity representations of the workspace. Such systems must provide the robot with the contextual awareness and precision necessary for safe and reliable interactions with its surroundings.

This approach is pivotal for advancing robotic autonomy, enabling robots to operate effectively in simulated environments that mimic real-world dynamics. By digitizing physical objects into a virtual workspace, robots can evaluate their tasks, refine their actions, and ensure task feasibility before real-world deployment. This methodology not only increases adaptability and efficiency but also bridges the gap between simulation and reality, paving the way for innovative solutions to address the complexities of dynamic environments.

II. Hypothesis

The integration of advanced computer vision techniques and object detection algorithms enhances the robot's capability to accurately identify, localize, and interact with objects in a virtual simulation. By leveraging tools such as YOLO for detection and Azure Kinect for 3D data acquisition, the system can effectively bridge the gap between physical and virtual environments, enabling precise object manipulation and dynamic adaptability in response to changes within the simulation.

III. State of the Art

A. Background

Contemporary technology development is increasingly focused on creating automated solutions that simplify, secure and accelerate a wide range of processes. In this context, transport technologies have played a crucial role, driven by the need to efficiently move people and objects within a territory, with the goal of reducing costs, optimizing times and improving safety across various operations.

For a robot to operate autonomously in dynamic and complex environments, one of the primary challenges is precise localization within the environment. This field has seen significant advances, such as developing classic SLAM (Simultaneous Localization and Mapping) algorithms, which allow vehicles to perceive their environment, process geometric information, and generate maps while estimating their

position in real time. Adoption of high-precision sensors, such as LiDAR, has been essential to improving these processes.

Despite these advances, limitations persist in current systems, affecting both positioning and the robot's perception of the environment. These shortcomings have prompted the emergence of promising new algorithms and opened up exciting opportunities in creating highly accurate virtual environments. These not only improve the location and interaction of the robot with its environment, but also allow the physical world to be digitized, enabling simulations where robots can execute specific tasks before being deployed in real-world scenarios.

B. Point Clouds

A point cloud is a three-dimensional (3D) representation consisting of a set of points arranged in a given space. Each of these points has spatial coordinates *x*, *y*, *z* that define their position in space. Point clouds are essential in a variety of fields, including computer vision, reverse engineering, virtual reality, and robotics, allowing the geometry of real-world surfaces and objects to be captured for further processing, analysis, and visualization. [7]

1) Point cloud generation: Typically, point clouds are created using 3D scanning technologies that include devices such as:

- LiDAR: Use a laser to measure the distance between the scanner and objects in the environment. LIDAR systems emit light pulses and calculate the distance based on the time it takes for the laser to reflect on a surface and return to the sensor.
- Stereo cameras: These cameras employ two or more cameras in different positions to capture images from multiple perspectives.
- Structured Light Scanners: They project patterns of light onto a surface and measure deformations in the pattern to estimate the shape and distance of objects.

2) Structure: A point cloud is made up of thousands or even millions of points that can vary in density depending on the accuracy of the capture device and the desired resolution. Points often contain not only spatial information (x, y, z), but also additional data such as [7]:

- Color (RGB): Some systems allow color values to be associated with each point, enriching the visual representation of the environment.
- Severity: Devices such as LIDAR can provide information on laser return intensity, which can help identify different materials or surfaces in the environment.
- Surface Normal: When calculating point orientation relative to surrounding surfaces, point clouds can also include surface normals, which is crucial in modeling and simulation applications.

3) Challenges: Despite its usefulness, point clouds present some challenges, such as their density and the massive volume of data they generate, which may require considerable computational resources for storage and processing. However, with the advancement of cloud computing and graphic processing technologies, it is increasingly feasible to manage extremely large point clouds in real time [8].

C. Odometry using LiDAR

LiDAR-based odometry is a technique used in robotics and autonomous vehicles to estimate position and orientation in real time. LiDAR generates point clouds of the environment using laser pulses, and odometry compares consecutive clouds to calculate robot motion. Algorithms such as ICP (Iterative Closest Point) are used to align point clouds and estimate vehicle translation and rotation. This technique is critical in SLAM ystems for simultaneous mapping and localization. While it offers high accuracy and is resistant to adverse conditions, its cost and need for intensive processing are its main challenges [8].

D. LiDAR issues

By itself, LiDAR is not perfect, it has characteristics in its measurements that we should be careful about. Some items that cause LiDAR problems are [8]:

- During the rotation of the LiDAR when generating the point cloud it may be biased.
- The exclusive use of LiDAR to estimate the position of a robot is not an ideal method as a result of the skewed recording of point clouds or the use of features that will eventually cause deviations.
- As a result of the above, it is common to use LiDAR in conjunction with other sensors such as IMU and GPS to satisfy position estimation and mapping. The joining of these components is called sensor fusion.

E. Sensor Fusion

There are two ways to perform this type of sensor integration:

- *Weakly Coupled Fusion:* The IMU is used to reduce LiDAR scan bias and give motion predictions for scan matching. However, IMU is not involved in process optimization within the algorithm.
- *Strongly Coupled Fusion:* Often offers improved accuracy. The LiDAR measures in conjunction with those obtained from the IMU are used to optimize the algorithm.

F. ICP (Iterative Closest Point) algorithm

Understanding *S* and *M* as entry point clouds, *S* being the source point cloud and *M* the model, a rigid transformation is sought that can satisfy point matching in the given clouds to the best of its ability. The Iterative Closest Point (ICP) algorithm attempts to cover this demand by following these steps [5]:

1) The closest point of the *S* set is calculated for each point (part of the points) of the *M* set using the Euclidean distance commonly.

$$d_1 = min(\sqrt{m_i^2 - s_j^2}), j = 1, ..., N_S$$

2) If the distance d_1 is greater than the threshold, the pair of points corresponding to that distance is removed.

3) The weights of the point pairs are added together: Implicitly $w_{ij} = 1$, although weights can be set based on the direction of normal vectors as a result of the point product.

$$w_{ij} = n_i n_j$$

4) We calculate the rotation matrix *R* and translation vector *t* using the least squares method to reduce the distance.
5) We calculate the transformation of the *S* set using $values = Rs_j + t$.
6) We calculate the $E(R;t)$ error using the equation described below and iterate to the desired accuracy.

$$E(R,t) = \sum_{i=1}^{N_M} \sum_{i=1}^{N_S} w_{ij} ||m_i - (Rs_j + t)||^2$$

Algorithm 1 ICP Algorithm for aligning point sets M and S

Require: Point sets M and S
Ensure: Transformation θ
0: $\theta \leftarrow \theta_0$ {Initialize with identity rotation R and zero translation T}
0: **while** not registered **do**
0: $X \leftarrow \emptyset$
0: **for all** $m_i \in T(M, \theta)$ **do**
0: $s_j \leftarrow$ closest point in S to m_i
0: $X \leftarrow X \cup (m_i, s_j)$
0: **end for**
0: $\theta \leftarrow$ least squares(X) using SVD
0: **end while**
0: **return** θ =0

1) Pseudo ICP Algorithm Code:

2) Improvements for ICP Algorithm: Improvements to the ICP algorithm are focused on meeting the convergence and reduction of existing calculation time in the algorithm:

- *K-D. Trees* This special type of binary spatial partition tree case is often used as a suitable structure in applications involving a multidimensional search key. The ICP algorithm calculation can be accelerated efficiently with its use.
- *Match Points.* Originally, you work by entering all the cloud points, and while there are improvements in uniform subsampling and random sampling, the problem is in the case of noise or high curvature forms, when these methods do NOT select enough reference points. The solution consists of using normal values for sampling
- *Point Match.* The algorithm is based on the calculation of the Euclidian distance between the points, however, this generates a great computational demand, with very high orders of complexity. There are several ways to speed up this calculation. The normal method is based on reducing the distance not only to two points, but also to the approximate distance of the point from the plane. This calculation reduces the number of iterations, but the method is not as robust. Other methods based on point projection from the source cloud to the second cloud, the center of the projection being possible such as the position of the camera you scan, then selecting the intersection points or working with the points depending on distance or intensity or color. It is also possible to use the Hausdorff distance. Likewise, it is possible to use the SIFT feature point as a corresponding point in the improved ICP algorithm process to reduce the points corresponding to the error, thus improving the efficiency of the algorithm.
- *Point Weight.* Conventional weighting considers that each point has the same weight, however, it is more appropriate to use the distance of the points as a measure of suitability. Depending on the distance of the points, we can completely remove the pair if it exceeds a predetermined threshold or add a weight to the pair based on the formula.

$$Weight = 1 - \frac{dist(P_1 P_2)}{dist_{max}}$$

We can assign weights based on the direction of normal vectors. If they are parallel in both, the scalar product is a large number, with the increase in the normal-angle we will get the smallest scalar product, in other words, perpendicular vectors have a scalar product equal to zero.

$$Weight = n_1 n_2$$

- *Point Match.* Point clouds typically contain the noise points and outliers, in an effort to avoid those scan errors, criteria can be added to eliminate or reduce this problem. It is generally recommended to remove the point pairs contained in the edges if one pixel exceeds the other. Alternatively, remove points that have a distance greater than a given threshold, remove a certain percentage of points using certain metrics, or remove those whose distance is greater than a given multiple of standard deviations.

3) LIO-SAM Model Operation: LIO - SAM estimates the robot's position and trajectory using observations from the LiDAR, IMU, and optionally GPS sensors. This condition estimation issue can be formulated as a Maximum a Posteriori (MAP) issue. This is a probabilistic framework for solving dense estimation problems. Involves calculating a conditional probability of observation given the weights of a prior probability or confidence model about the model

G. Convolutional Neuronal Networks (CNN)

Convolutional Neural Networks are a neural network architecture specially created to handle data with a grid structure, such as images. These networks excel in computer vision tasks, including image classification, segmentation, object detection, and facial recognition. Its design makes it easy to extract relevant features from images and reduce data dimensionality, while maintaining the most critical information for the learning process [3].

1) Structure:

- Convolution Layer: Convolution is the central process of a CNN and consists of applying filters (or kernels) to the input image to extract characteristics. A filter is a small array that slides over the image and multiplies element by element with the pixel values. This produces a new matrix called a feature map.
- Activation Layer: After the convolution operation, a non-linear activation function is applied to introduce non-linearities into the model, allowing the network to learn complex representations. The most common activation function in CNNs is ReLU (Rectified Linear Unit).
- Pooling Layer: The pooling layer is responsible for reducing the dimensionality of feature maps, maintaining the most important information. This process, known as subsampling, makes the model more robust to variations and reduces computational cost.
- Fully Connected Layers: After several layers of convolution and pooling, CNNs typically have one or more layers fully connected or fully connected (FC), where each neuron is connected to all the neurons in the anterior layer. These layers function like traditional layers of neural networks and are responsible for making final decisions based on the characteristics extracted by the convolutional layers.
- Output Layer: The output layer generates the final prediction of the model, whether it is a classification, regression, or other type of task. In the case of classification, it usually uses a softmax trigger function to convert outputs to probabilities that add up to 1, representing the probability that the input belongs to each class.

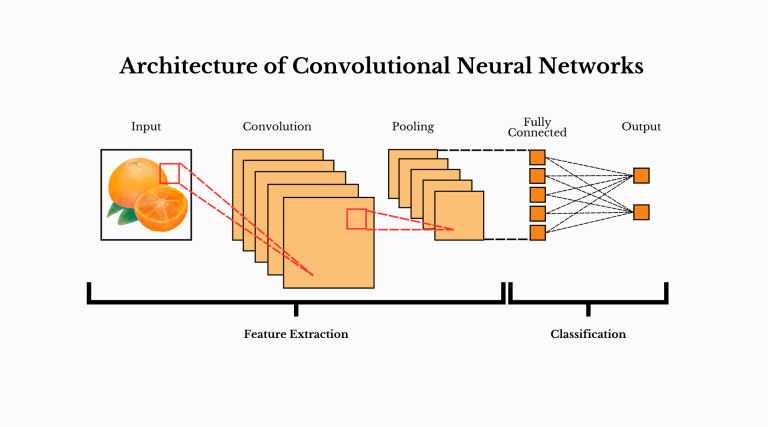

Fig. 1. General Structure of a Convolutional Neural Network

H. YOLO

The YOLO model was originally proposed by Joseph Redmon in 2016 and has become one of the standards in real-time object detection, thanks to its ability to process images quickly and detect multiple objects simultaneously. Over time, several versions and forks (referrals) of YOLO have emerged, each with advances in architecture accuracy, efficiency and complexity. YOLOv8 follows the same philosophy as previous versions of YOLO: Perform object detection with a single direct prediction step, making it highly efficient for real-time applications. This release continues to improve deep learning techniques and leverages recent advances in neural network architecture, machine learning, and data preprocessing to provide a better balance between accuracy and speed. One of the key features of YOLOv8 is that it is not only optimized for object detection, but also includes native support for image segmentation and classification, making it more versatile and suitable for a greater number of applications [9].

I. Roboflow

RoboFlow is a development platform that simplifies the creation, management, and deployment of computer vision models. It is especially geared toward tasks that require the generation and management of datasets to train artificial intelligence (AI) algorithms. Its design is intended to support developers, data scientists, and computer vision enthusiasts in building custom models for object detection, image segmentation, image classification, and various additional applications [6].

1) Workflow:

- Data Collection and Import: Users have the ability to import images from on-premises sources or cloud services, such as Google Drive, AWS S3, and more. It is also possible to upload datasets that have already been annotated.
- Data annotation: After import, users can manually label images or upload pre-made labels. RoboFlow enables image annotation for various tasks such as object detection, segmentation, and sorting.
- Increase and preprocessing: Various transformations are applied to images with the goal of expanding the dataset and strengthening the model against variations. In addition, the quality of the dataset can be improved through image normalization, noise reduction, and other cleaning techniques.
- Model Training: Pre-processed data is exported to different machine learning frameworks. RoboFlow facilitates this integration by offering scripts and configurations optimized for various frameworks.
- Deployment and Deployment: Once the model has been trained, it can be deployed through RoboFlow-provided APIs, allowing users to easily integrate the models into their applications. This includes cloud solutions, mobile apps, or even drones and robots with computer vision capability.

J. Canonical Model

The Canonical 3D Object Representation is a pivotal tool in computer vision and robotics, offering a unified model that captures the essential structural and spatial characteristics of an object type. This consistent reference model remains stable across transformations such as scaling, rotation, and minor

deformations, proving particularly valuable in dynamic real-world settings. The primary aim of a canonical representation is to mitigate variability in object appearance due to diverse viewpoints, lighting, and minor deformations, by standardizing an "ideal" version of the object type. This simplification aids in efficient object recognition, segmentation, and real-time tracking.

In 3D object recognition, canonical representations allow for generalization across various instances of an object type, enabling identification despite partial occlusion or unconventional viewing angles. Predefined key anchor points and geometric features in the canonical model help match features from live data inputs. For semi-deformable objects like fruits or leaves, these representations offer a flexible baseline, accommodating natural variations in shape or size, which is crucial for automated tasks such as sorting or picking in agriculture.

While canonical 3D representations enhance model training and recognition accuracy under diverse conditions, they face challenges with objects exhibiting high variability or complex deformability. These scenarios require substantial refinement to create effective generalizations. Additionally, the need for real-time adjustments and computational efficiency is critical to ensure adaptability in dynamic environments. Overall, canonical 3D object representation is essential for robust object detection and classification systems, providing a standardized model to handle real-world variations effectively [1].

IV. Description of Methodologies

For this research, we will adopt a mixed-methods approach, combining both qualitative and quantitative methodologies to address the complexities inherent in robotics, computer vision, and artificial intelligence. The methodology is divided into several interconnected stages, each with distinct objectives and technical requirements, as illustrated in the project's architecture diagram (Fig.2).

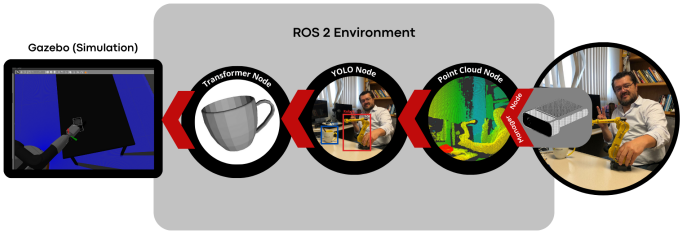

Fig. 2. Project's architecture diagram

A. Data Acquisition and Preprocessing

The system begins with data acquisition through an Azure Kinect camera [2], labeled in the diagram as the "Real World Connector." This camera collects both RGBA and depth images of real-world objects within the environment. The collected data is channeled through a ROS 2 environment to two primary nodes

- Point Cloud Node: Responsible for generating point clouds from the depth images provided by the Azure Kinect. This node creates a 3D representation of the environment, capturing the spatial structure of objects for further processing.
- YOLO Node: Using machine learning algorithms, this node identifies and classifies objects within the RGBA images. YOLO (You Only Look Once) is employed for real-time object detection. Currently, our model is being refined with a specialized dataset focusing on oranges, cans, and coffee containers to improve identification accuracy.

B. Reconstruction and Object Interaction

Once the data is preprocessed, it flows into the Reconstruction Node. This node combines information from the point cloud and YOLO nodes to generate a comprehensive 3D model of the detected objects, representing their positions and orientations within the environment. This reconstruction process leverages methodologies inspired by the work of Gustavo De Los Rios [1], integrating advanced algorithms for accurate object reconstruction and placement.

C. Simulation and Testing

The final output from the reconstruction node is directed to a simulated environment in Gazebo [4], where a simulated Tiago robot interacts with the 3D models. The ROS 2-Gazebo connection facilitates seamless communication, allowing the Tiago robot to perform object manipulation tasks as it would in a real-world scenario. This setup enables preliminary testing in a virtual setting before transitioning to real-world applications, reducing operational costs and mitigating risks.

D. Machine Learning and Data Analysis

Throughout the process, data analysis and machine learning techniques are employed to optimize the recognition and reconstruction algorithms. By implementing data-cleaning techniques (e.g., outlier removal) and leveraging robust predictive models, we ensure that our algorithms remain resilient to overfitting. Reinforcement learning (RL) may be applied to train the system in object interaction tasks, gradually improving the robot's adaptability and precision.

E. Explainability and Transparency

Given the complexity of the methodologies, a key emphasis is placed on the explainability of results. By maintaining transparency in model performance and algorithmic decision-making, we aim to provide clear insights into system behavior. This transparency is particularly critical in the robotic domain, where predictable and interpretable outcomes are essential for safe human-robot interaction.

In summary, the methodologies employed in this research combine elements of machine learning, data processing,

robotics, and explainable AI, all integrated within a robust simulation environment to enable accurate object recognition, reconstruction, and interaction. The proposed system holds promise for advancements in both virtual testing environments and real-world robotics applications.

V. Experiment Design

The proposed experiments are structured in four phases. Each phase builds on the previous ones, progressing from data collection and model training to real-world evaluation, and finally integrating object reconstruction and interaction capabilities for a simulated Tiago robotic arm. The reconstruction process is based on methodologies introduced in Gustavo De Los Rios Alatorre's thesis, focusing on canonical model deformation and 3D pose estimation. However, given the time constraints and development complexity, the experiment was adjusted to an ideal state by omitting certain reconstruction processes, focusing on a minimum viable demonstration. Additionally, a 'manager_node' was introduced to address the Azure Kinect's inability to be used independently by multiple nodes. This node creates and publishes topics for both depth images (for the point cloud node) and color images (for the YOLO detection node). Each phase is explained below and illustrated in Fig.3.

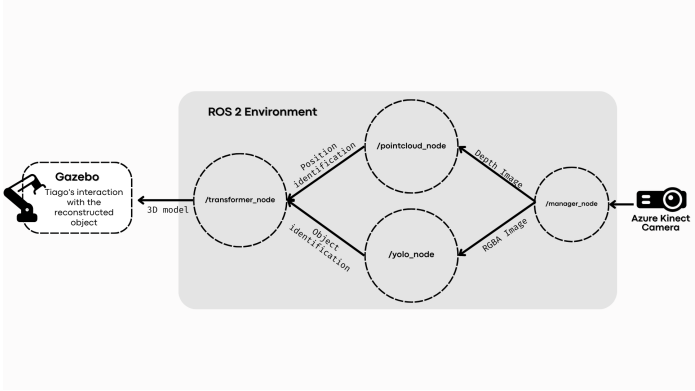

Fig. 3. Experiment elements, nodes, and relations

A. PHASE I: Point Cloud Capture and Publication

This phase focuses on capturing a 3D representation of the environment using the Azure Kinect's depth sensor. The depth data is managed by the 'manager_node', which publishes both depth and RGB images. These images are used by the 'env_node' to generate a real-time 3D point cloud representation of the environment.

1. Data Capture and Processing:

- The 'manager_node' captures raw data from the Azure Kinect camera.
- It publishes two primary topics:
  - Depth images (used by the point cloud generation node).
  - RGB images (used by the YOLO detection node in Phase 2).

2. Point Cloud Generation:

- The 'env_node' subscribes to the depth image topic published by the 'manager_node'.
- It processes the depth data to generate a 3D point cloud representing the environment.
- Filtering techniques (e.g., noise reduction and downsampling) are applied to optimize the quality of the point cloud.
- **Note:** To improve accuracy and isolate the point cloud of specific objects, applying a Region of Interest (ROI) filter and additional segmentation techniques is recommended. This ensures that only the object of interest is represented, excluding irrelevant parts of the environment.

3. Point Cloud Publication:

- The 'env_node' publishes the processed point cloud on a dedicated ROS 2 topic, allowing other nodes to utilize the 3D representation in subsequent phases.

4. General Structures (Pseudocodes):

Algorithm 2 Manager Node Pseudocode

```
Initialize ROS 2 node and parameters
Configure and connect to Azure Kinect
Create topics for depth and color images
while node is active do
    Capture depth and RGB frames from Azure Kinect
    if frames are invalid then
        Continue to the next iteration
    end if
    Publish depth images to the respective topic
    Publish color images to the respective topic
    Sleep until the next cycle
end while
Shutdown node and release resources =0
```

Algorithm 3 Point Cloud Node Pseudocode

```
Initialize ROS 2 node and parameters
Subscribe to:
    • Depth image topic ('/shared/depth_image')
    • (Optional) Color image topic for validation
Configure point cloud publisher
while node is active do
    Receive depth image
    if depth image is invalid then
        Continue to the next iteration
    end if
    Generate 3D point cloud from depth image:
    • Apply transformations to convert to 3D coordinates
    • Filter and optimize the point cloud
    Publish point cloud to the respective topic
    Sleep until the next cycle
end while
Shutdown node and release resources =0
```

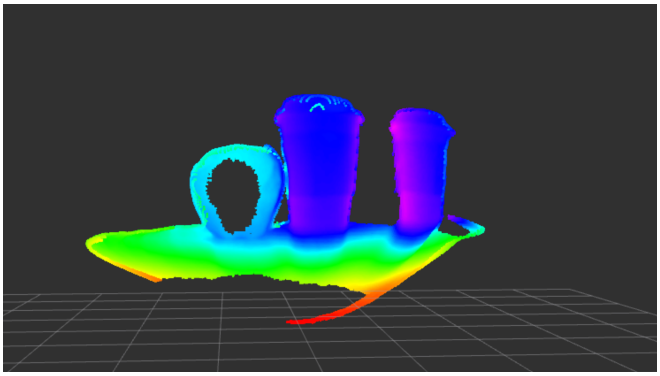

Fig. 4. Operation of the point cloud node

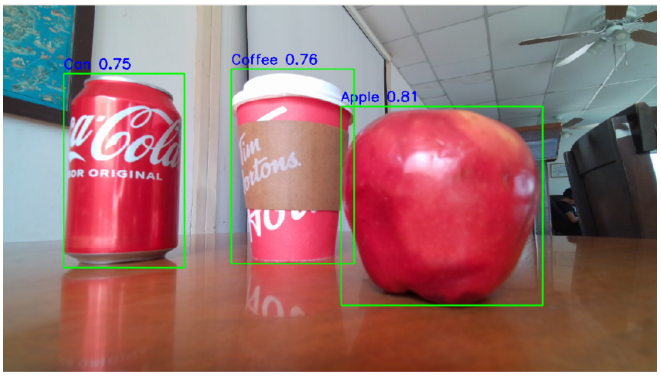

Fig. 5. Operation of the YOLO node

B. PHASE II: Object Identification Node with YOLOv8

The ‘yolo_node‘ detects and classifies individual objects using the color image stream from the ‘manager_node‘.

1. Real-Time Object Detection:

- The ‘yolo_node‘ uses a YOLOv8 model trained on specific object classes (e.g., cans, coffee, apples) to detect and classify objects.
- Bounding boxes are generated for each detected object in the RGB image.

2. Publishing Detection Results:

- The node publishes annotated RGB images with bounding boxes and text files containing object detection results.

3. General Structure (Pseudocode):

Algorithm 4 YOLO Detection Node Pseudocode

```
Initialize ROS 2 node and parameters
Load YOLOv8 model for object detection
Subscribe to color image topic from ‘manager_node‘
Set up publishers for detection results and annotated image
while node is active do
    Receive image frame from ‘/shared/color_image‘
    if frame is not valid then
        Continue to next iteration
    end if
    Perform YOLO detection on the image
    for each detected object do
        Extract class, confidence score, and bounding box
        Annotate bounding boxes on the image
    end for
    Publish detection data as text
    Publish annotated image with bounding boxes
    {Loop rate control}
    Sleep until the next cycle
end while
Shutdown node and release resources =0
```

C. PHASE III: Object Reconstruction Node using 3D Canonical Deformation

The ‘transformer_node‘ performs 3D object reconstruction using canonical models and pose estimation.

1. Canonical Model Alignment:

- The node uses canonical STL models for each object class (e.g., cans, coffee, apples).
- Detected objects from the ‘yolo_node‘ and segmented point clouds from the ‘env_node‘ are used to align and deform these models.

2. Pose Estimation:

- The reconstructed models are positioned and oriented using pose estimation to match their location in the real world.

3. Publishing Reconstructed Models:

- The reconstructed 3D models are published to the ROS environment for interaction in simulation.

4. General Structure (Pseudocode):

Algorithm 5 Reconstruction Node Pseudocode

```
Initialize ROS 2 node and parameters
Subscribe to point cloud and detection data
Load canonical models for reconstruction
Set up publisher for reconstructed models
while node is active do
    Receive point cloud and detection data
    if data is not valid then
        Continue to next iteration
    end if
    Perform pose estimation and canonical model deformation
    Update and align 3D object models
    Publish updated 3D models
    {Loop rate control}
    Sleep until the next cycle
end while
Shutdown node and release resources =0
```

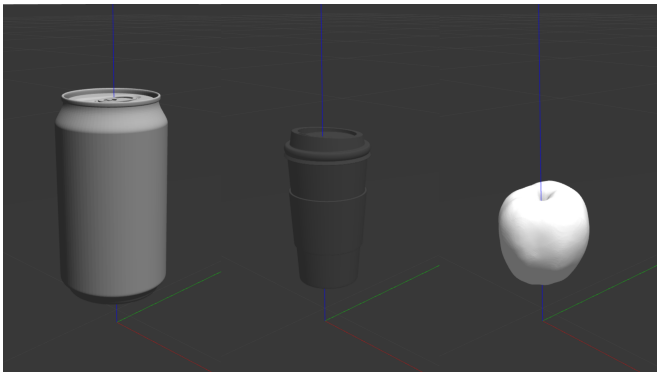

Fig. 6. Operation of the transformer node

D. PHASE IV: Simulated Interaction with Tiago Robot

This phase focuses on integrating the reconstructed objects into a simulation environment, allowing interaction with the Tiago robot.

1. Simulated Environment:

- Reconstructed objects are imported into Gazebo for simulation.
- The Tiago robot interacts with these models based on user commands.

2. Feedback and Iterative Refinement:

- Feedback from the simulation is used to refine the object reconstruction and grasping processes.

This experimental design ensures a seamless workflow from data acquisition to simulation, demonstrating a simplified yet functional pipeline for digital object reconstruction and robotic interaction.

Fig. 7. Operation of the Tiago Simulation on Gazebo

E. Experiment Constraints and Simplifications

While the theoretical framework proposed by Gustavo De Los Ríos Alatorre in his thesis presents a robust methodology for accurately reconstructing 3D objects through canonical model deformation, rotations, translations, and iterative closest point (ICP) algorithms, these processes require extensive computational resources and development time. Given the constraints of this experiment, the functional demonstration was designed to simplify these calculations while still achieving the goal of representing objects in a specified 3D space that aligns with the detection results.

1. Simplification of Object Reconstruction:

- Instead of implementing full ICP-based alignment, objects are generated in their approximate positions and orientations within the environment.
- Canonical models are loaded and positioned based on a direct mapping of the detected bounding box and point cloud data, skipping the iterative refinement steps.
- Rotations and translations are simplified to basic geometric calculations, ensuring objects are placed in the 3D space correctly without full optimization.

2. Teleoperation of Tiago Robot:

- At this stage, the Tiago robot simulation does not yet respond to specific commands for interaction. Both its navigation and grasping tasks are manually teleoperated.
- This limitation provides an opportunity to design a dedicated command interface for object interaction. Future iterations could include a node capable of interpreting human arm movements or intuitive user commands, enabling a seamless mapping of human actions to the robot's arm.
- The potential implementation of such a node would simplify the use of the Tiago robot, making it more accessible for users and enhancing its functionality in object manipulation tasks.

3. Focus on Functional Integration:

- The goal of the experiment was to create a minimal viable product (MVP) for demonstration purposes. This included ensuring real-time performance and compatibility across the pipeline.
- While the reconstruction process is less precise than Gustavo's methodology, the objects are realistically placed in the virtual environment, allowing for meaningful interaction in simulation.

4. Manager Node as a Key Integration Component:

- Due to the limitations of the Azure Kinect, which cannot be accessed by multiple nodes simultaneously, the 'manager_node' was introduced as a centralized data provider.
- This node ensures synchronization between the 'env_node' (for point cloud generation) and the 'yolo_node' (for object detection), simplifying the data flow while maintaining the integrity of the system.

5. Justification for Simplifications:

- These adjustments allow the experiment to demonstrate a fully functional pipeline, from data acquisition to 3D reconstruction, within the given time and resource constraints.
- The simplifications do not diminish the potential of Gustavo's theoretical approach, but rather pave the way for its future integration into more advanced iterations of this project.
- The current teleoperation approach ensures that all aspects of the simulation are functional while leaving

room for future enhancements, such as command-based interactions or autonomous decision-making capabilities.

The experimental design balances theoretical rigor with practical constraints, showcasing the viability of object reconstruction and interaction in a simulated environment. Future work will aim to incorporate the full complexity of Gustavo's methods, improving precision and adaptability, and to develop a robust command interface for intuitive robot interaction.

VI. Results Interpretation

At this stage, the project has achieved significant milestones, nearing 98% functionality. The YOLO-based detection node is fully operational, effectively identifying and classifying cans, coffee containers, and apples. Additionally, the Gazebo simulation environment has been successfully implemented, allowing interaction with virtual models corresponding to real-world objects. While grasping capabilities are still under refinement, the core functionalities (object detection, spawning, and interaction) are fully developed.

The experimental setup demonstrates a notable improvement in both the precision of object detection and the reaction time required to transfer elements from the physical to the virtual environment. The system achieves swift identification, spawning, and removal of objects, effectively bridging the gap between real-world sensing and virtual simulation. Although some procedures, such as point cloud rotation and translation for perfect alignment, were omitted in the demo due to time constraints, the essential objectives of generating point clouds and enabling interaction were successfully accomplished.

Preliminary observations indicate a marked enhancement in the overall system's responsiveness and accuracy. The integration of the YOLO node and Azure Kinect has streamlined object digitalization, with faster reaction times and improved reliability in capturing and manipulating virtual representations of physical objects. These advancements lay a strong foundation for further optimization, particularly in grasping and alignment tasks, which remain as the next areas of focus to achieve full system autonomy.

This progress demonstrates the project's potential to serve as a robust platform for real-time object interaction between physical and virtual domains.

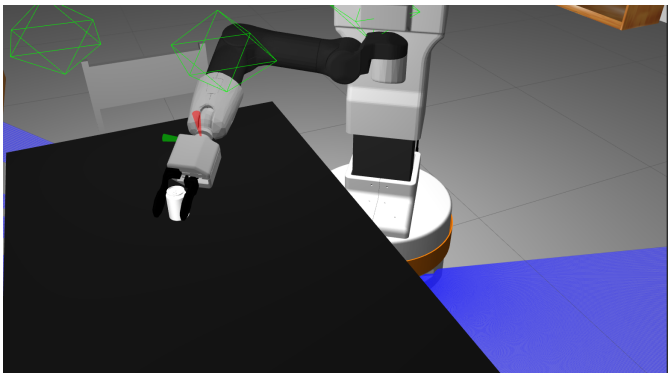

Fig. 8. Final performance of the demonstration

Below is a video demonstration of the demo developed during the research.

VII. Discoveries

Implementing robotic solutions in both controlled and dynamic environments comes with numerous challenges, including high operational cost and associated risk to system feasibility, functionality and efficiency. An effective approach to mitigate these risks is advanced digitization and simulation in virtual environments that accurately replicate the real world. This allows the behavior of robotic systems to be evaluated and fine-tuned prior to deployment in real-world conditions. The ability to digitize objects in the environment in real time, classify them using advanced sensing algorithms such as YOLOv8 and simulate their manipulation in simulation environments provides a powerful tool to reduce operational uncertainty, optimize resources and improve interoperability between robotic systems and their environment.

- **Context** : Common practice involved the manual capture of 3D models or the use of multiple sensors and processing techniques to achieve an adequate representation of objects in robotics simulators. This made real-time digitization and manipulation impractical for collaborative robotics environments, where efficiency and accuracy are essential.
- **Problem or need** : The specific need was to accurately and efficiently capture various objects in a physical environment so that they could be manipulated in a realistic simulation. This included not only digitizing the geometry of the objects, but also identifying their classification and characteristics (e.g., approximate mass), allowing the simulated robot to perform tasks such as "grasping" or manipulating specific objects on demand.
- **Innovation Description** : This project combines Azure Kinect for point cloud capture with object detection and classification using YOLOv8. On a series of ROS2 nodes, the system allows 3D object scanning, identifying each visible object in the environment and reconstructing high-fidelity 3D models for manipulation. Object reconstruction is performed with a canonical model deformation technique, as detailed in the work of Gustavo De Los Rios Alatorre, achieving an accurate representation of the object's pose and scale in virtual space. This flow allows the captured objects to be immediately accessible to a simulated Tiago robot, who can interact with them using simple commands such as "grab the cup" or "take the cereal box".
- **Key Differentiators** :
  - Real-Time: Real-time scanning and reconstruction allows the simulated robot to act immediately on the scanned objects in the physical environment.
  - Digitization Accuracy: Thanks to the use of Azure Kinect's canonical model deformation technique and advanced calibration, the system captures with high accuracy the geometry and orientation of the objects.

  - Classification and Physical Property Estimation: Integration with YOLOv8 and an approximate mass estimation CNN allows not only detecting and classifying, but also assigning physical properties to improve the simulation of object manipulation.
- **Potential Impact** : This innovation has significant implications for collaborative robotics, especially in applications where human-robot interaction in shared environments is critical. Users will be able to train robots to perform specific tasks with digitized objects, improving adaptability and responsiveness in industrial and domestic environments. Furthermore, in the simulation and video game industry, it may be possible to apply this technology to create more realistic environments, where physical objects can be digitized and used in simulations quickly.
- **Evidence or Validation** : The reconstruction methodology is based on the academic work of Gustavo De Los Rios Alatorre, validated for its accuracy in 3D reconstruction and pose alignment applications.
- **Challenges and Limitations** :
  - Processing Requirements: Real-time processing of point clouds and object detection requires significant computational resources, which may limit the application on less powerful hardware.
  - Ambiguity in Similarly Configured Objects: Current technology may have difficulty distinguishing between objects of very similar geometry or color in environments where illumination is not uniform.
- **Future Applications** :
  - Implementation in Autonomous Robotics: This innovation could be applied in autonomous mobile robots to digitize and classify objects in factor ies or warehouses, allowing them to make decisions about handling and transporting objects in real time.
  - Virtual Training for Robotic Manipulation: By enabling the rapid digitization of objects, this technology could facilitate the training of robots in simulators before they perform tasks in real environments.
  - Augmented Reality (AR) Expansion: Digitized models could be used in augmented reality systems to create interactive environments where real objects are digitally represented and manipulated.
- **Acknowledgments or Recognitions** : This approach could be nominated for technology innovation awards because of its real-time approach and its ability to integrate detection, digitization and manipulation into a single workflow.
- **Ethical or Social Considerations** :
  - Home Use: The introduction of robots with the ability to manipulate digitized objects could have an impact on homes, especially in terms of privacy and personal data handling.
  - Worker Replacement: Advanced automation may raise concerns about the replacement of manual tasks, although this technology also opens up opportunities in industries that require precision and human-robot collaboration.

This discovery represents a significant breakthrough in the field of robotics and simulation, providing an integrated and adaptable solution for object digitization and manipulation.

VIII. Conclusions

In this work, we have undertaken a comprehensive exploration of digitalizing objects in a physical environment using Azure Kinect and implementing object detection and recognition through YOLO models. By leveraging the depth-sensing capabilities of the Kinect camera, along with YOLOv8's powerful object recognition algorithms, our methodology achieves significant strides in transferring physical entities into virtual representations with high accuracy. This approach is particularly beneficial in scenarios where robotic systems require accurate object manipulation and recognition within a virtual environment, such as in simulation setups or in developing real-world applications in robotics.

Our findings contribute to bridging the gap between physical and virtual environments, supporting the creation of realistic simulations that can be used for various robotic manipulation and interaction tasks. Furthermore, the integration of techniques such as ICP (Iterative Closest Point) and Eigenface for tracking and recognizing individuals enhances the system's capability in distinguishing and following specific targets, paving the way for advancements in human-robot interaction.

For future research, incorporating a wider variety of objects and refining mass estimation models within simulations can further improve object interaction realism. Additionally, enhancing the tracking algorithms to operate in dynamic environments where multiple entities interact may expand the system's applications. Overall, this work lays a strong foundation for seamless integration between digital perception and virtual manipulation, setting the stage for further innovations in robotic simulation and real-world deployment.

Acknowledgment

I extend my deepest gratitude to Dr. Luis Alberto Muñoz Ubando for his invaluable support and mentorship throughout this research. His vision and inspiration have been essential in shaping the project, guiding it from concept to completion. I am equally grateful to Gustavo De Los Ríos Alatorre, whose master's thesis provided a robust foundation for constructing the object reconstruction node. His work in this field enabled me to frame my objectives within a feasible virtual environment. I also wish to thank Marco Ottavio Podesta Vezzali for his collaborative spirit and knowledge, which helped me overcome numerous challenges, as well as my colleagues who provided constant feedback and support during my research stay.

I would be remiss not to express my heartfelt thanks to César Guerra, who motivated me every step of the way, every single day until the end. Thanks to Alejandro Moctezuma, Gerardo Deustúa, Hugo Muñoz, Luis Zago and Daniela Lozada for beleve in my potencial. My sincerest gratitude goes to

my parents, Lucila Bautista Aguilar and Jesús Gómez Radilla, who believed in me long before I believed in myself; without their perseverance, love, and unwavering support, none of this would be possible.

Finally, a special thank you to all my beloved pets, who have surrounded me with love and companionship, reminding me daily of the joy and warmth of home.

REFERENCES

[1] Gustavo de los Ríos Alvarez. 3-d detection tracking for semi-deformable objects. Master's thesis, Instituto Tecnologico y de Estudios Superiores de Monterrey, Monterrey, Nuevo León, 2024.

[2] Lafkiri H. Hajji R. Rached I. Delasse, C. and T. Landes. Indoor 3d reconstruction of buildings via azure kinect rgb-d camera. *Sensors (Basel)*, 22(9222), 2022.

[3] Bengio Y. LeCun, Y. and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[4] F. Martín. *A Concise Introduction to Robot Programming with ROS2*. CRC Press, 2023.

[5] J. Prochazkova and P. Krsek. Notes on iterative closest point algorithm. *ResearchGate*, pages 876–884, 2018.

[6] RoboFlow. The computer vision workflow: From labeling to deployment., n.d.

[7] X. Ruan and B. Liu. Review of 3d point cloud data segmentation methods. *International Journal of Advanced Network, Monitoring and Controls*, 5(1):66–71, 2020.

[8] Englot B. Meyers D. Wang W. Ratti C. Shan, T. and D. Rus. Lio-sam: Tightly-coupled lidar inertial odometry via smoothing and mapping. *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5135–5142, 2020.

[9] Ultralytics. Yolov8: Next-generation object detection, 2023.

APPENDIX A
PROJECT REPOSITORY

The code developed for this project is available in the GitHub repository: link.

APPENDIX B
DEMO VIDEO

A video demonstration can be found at: link.

APPENDIX C
SLIDES

The slides of the final presentation can be found it here: link.

APPENDIX D
POSTER

The poster based on the final products of our research can be see at: link.