E3S Web of Conferences **705**, 02004 (2026) https://doi.org/10.1051/e3sconf/202670502004
AREEV 2026

Low-cost multimodal fusion of 2D LiDAR and RGB camera for accurate object-level perception

Anil M. Kabbur[1], *Vinod Kumar V. Meti*[2*], *Prakash Pawar*[1], *Sumanth B*[3], *Karan G*[3], *and Mohammed Yunus M. G. Khatib*[3]

[1]*Electronics and Communication Department, Indian Institute of Information Technology, Dharwad, India*
[2,3]*Automation and Robotics Department, KLE Technological University, Hubli, India*

Abstract. Sensor fusion between LiDAR and camera modalities has emerged as an effective approach for improving perception in autonomous robotic systems. However, most existing solutions rely on high-cost sensors and computationally intensive algorithms. This paper presents a low-cost LiDAR–camera fusion framework using a YDLiDAR TG30 2D LiDAR and a Logitech C270 RGB camera to achieve real-time object-level perception. The system is developed using ROS 2 Humble and integrates intrinsic and extrinsic calibration to accurately project LiDAR point clouds onto camera images. A custom fusion node performs coordinate transformation and depth-based visualization in real time. Experimental validation conducted in indoor environments demonstrates reliable projection accuracy within near-range distances, even under low-light conditions. The proposed framework offers an affordable, modular, and computationally efficient solution suitable for educational robotics, indoor navigation, and resource-constrained mobile platforms.

Keywords: YDLiDAR, Sensor Fusion, Multimodal, ROS, Object-level Perception

1. Introduction

For any robot or autonomous system to function effectively, it needs a clear understanding of its surroundings. Whether it's navigating a room, avoiding obstacles, or performing tasks in unfamiliar environments, the system must be able to detect and interpret objects around it in real time. This capability, often referred to as object detection, is at the heart of many modern robotic applications. However, object detection systems that rely purely on visual data often struggle in dark or poorly lit environments. This project was inspired by that exact challenge—the goal was to improve a robot's ability to detect objects, even when lighting conditions are far from ideal.

---

*Corresponding author: vinod_meti@kletech.ac.in

This research focuses on the integration of a 2D LiDAR with a camera for real-time projection of LiDAR point clouds onto image frames. Specifically, we utilize the YDLiDAR TG30, a 2D LiDAR capable of 360-degree scans with a range of up to 12 meters, and the Logitech C270 webcam, which captures images at 720p resolution. These hardware components were chosen for their affordability and availability, making the system practical for research and educational use. The entire system is developed and deployed within the Robot Operating System 2 (ROS 2) humble framework on Ubuntu 22.04. ROS 2 provides modularity, real-time communication capabilities, and extensive support for sensor integration and visualization. The software stack includes Python 3.10 for scripting, OpenCV for image processing, CV Bridge for bridging ROS and OpenCV images, and RViz2 for real-time visualization of the fused data. A custom Python node, named fusion_node.py, is responsible for the real-time processing of LiDAR and camera data streams. The calibration process plays a critical role in this system, as it ensures the spatial alignment between the LiDAR and the camera. Extrinsic calibration parameters were obtained using a checkerboard-based method, where both the LiDAR and camera observed a shared calibration target. This calibration data was then used to compute the transformation matrix required to map LiDAR points into the camera's field of view accurately. This research work intentionally avoids complex object detection tasks using deep learning and instead focuses on sensor-level fusion, which provides a more computationally lightweight and efficient system. The goal is to visualize how accurately LiDAR points can be overlaid on a camera image, with distance-based coloring that enhances understanding of spatial depth in a visual context. This type of fusion is particularly useful in scenarios such as warehouse navigation, obstacle detection, and low-cost robotic mapping systems where budget and processing power are limited. Basic block diagram for robust object level perception as shown in Figure 1.

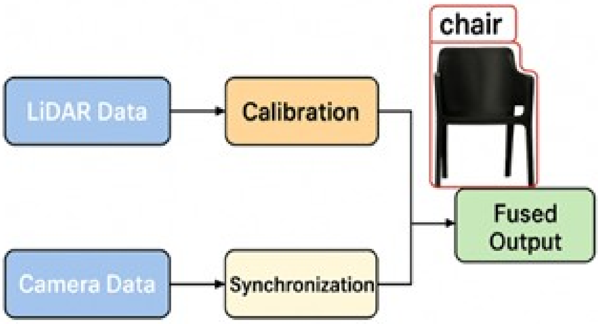


Figure 1. Basic Block Diagram for Robust Object level Perception

1.1. Comparison with Existing LiDAR-Camera Fusion Approaches

Unlike conventional LiDAR–camera fusion systems that rely on high-cost 3D LiDARs and depth cameras, the proposed approach focuses on affordability, computational simplicity, and ease of deployment. Existing methods often utilize sensors such as Velodyne VLP-16, Ouster OS1, or RGB-D cameras, which significantly increase system cost and processing requirements. In contrast, the proposed framework employs a 2D YDLiDAR TG30 and a standard Logitech C270 camera, reducing hardware cost while still achieving reliable object-level perception.

Additionally, many existing fusion frameworks integrate deep learning–based object detection or tightly coupled SLAM pipelines, which demand GPU-intensive computation and complex parameter tuning. The proposed system intentionally avoids such heavy processing and instead emphasizes sensor-level geometric fusion, enabling real-time performance on resource-constrained platforms.

Experimental results demonstrate that the proposed method achieves accurate point projection within a 1–3 m range in indoor environments, which is sufficient for obstacle detection, navigation, and safety monitoring tasks. While high-end systems provide richer 3D semantics, the proposed approach offers a practical trade-off between performance, cost, and system complexity, making it well suited for educational robotics and small-scale mobile platforms.

Overall, this paper contributes a complete pipeline for 2D LiDAR and camera fusion using open-source tools and accessible hardware as shown is Table 1.The results highlight the feasibility and effectiveness of real-time point cloud projection, and the modular architecture provides a solid foundation for future extensions, including 3D LiDAR integration, semantic segmentation, and dynamic obstacle tracking.

Table 1. Modular Framework for Multi-Modal Data Fusion and Perception

Sl No.	Module Name	Definition / Purpose	Key Components Involved	System Layer
1	Sensing Layer	Acquires raw data from the environment.	YDLiDAR TG30, Logitech C270 USB Camera.	Hardware/ Driver
2	Data Synchronization	Ensures that LiDAR scans and camera images being processed correspond to the same moment in time.	ROS 2 message timestamps.	Software (Node)
3	Calibration	Determines sensor intrinsic parameters and the spatial relationship (extrinsic) between sensors.	Checkerboard pattern, ROS 2 calibration tools, RViz2 for alignment verification.	Software (Setup)
4	Fusion Node	Transforms LiDAR points into the camera frame and projects them onto the 2D image plane.	fusion_node.py, Python 3.10, OpenCV, NumPy.	Software (Core Logic)
5	Visualization	Displays the final fused output of the camera feed with overlaid LiDAR depth points.	RViz2 (for debugging/calibration), OpenCV imshow (for real-time output).	Software (Output)

The selected references represent key developments in LiDAR camera fusion, including semantic fusion techniques, sensor calibration methods, and multi-sensor perception frameworks. In addition to advancements in autonomous navigation, LiDAR camera fusion techniques have significant implications in environmental and sustainability domains. Such systems can be effectively utilized in applications including disaster monitoring, forest surveillance, smart transportation, and infrastructure inspection, aligning with sustainable development goals. By enabling accurate perception and mapping in complex environments, the proposed approach contributes to intelligent systems that support environmental monitoring and resource efficient operations. This alignment strengthens the relevance of the study within the broader scope of environmental and energy related research.

2. Methodology

The core objective of this system is to fuse data from a 2D LiDAR and a camera in order to enable meaningful visualizations and object-level perception. To achieve this, we integrate the YDLiDAR TG30 sensor with the Logitech C270 USB camera, both of which complement each other's capabilities—LiDAR providing accurate depth measurements, while the camera captures rich visual information. The hardware setup includes the YDLiDAR TG30, a 2D LiDAR capable of a 12-meter range and operating at a scan frequency of 10 Hz. It is well-suited for short-range real-time mapping applications. Complementing the LiDAR is the Logitech C270, a UVC-compliant USB webcam that captures 720p resolution video. Together, these devices offer a low-cost yet effective sensing combination for 2D spatial perception with visual context. On the software side, the entire system is built within the ROS 2 Humble framework on Ubuntu 22.04. This combination provides excellent support for modular robotics development and offers long-term support and stability. In this work Python 3.10 is used as the primary programming language due to its flexibility and proficiency of use with ROS 2 nodes. Several key libraries were used to process and visualize the data: OpenCV for image processing, CV Bridge for converting ROS image messages to OpenCV format, and RViz2 for 3D visualization and debugging. Cameras are widely used in robotics for object detection because they are affordable and provide detailed color and texture information. Yet, they have a major weakness: they rely heavily on ambient light. In low-light conditions, such as nighttime or indoor environments with minimal lighting, the quality of images captured by cameras deteriorates. As a result, algorithms that rely on visual features may fail to detect objects accurately, or at all. This limitation makes it difficult to deploy camera-only systems in real-world scenarios where lighting isn't always controllable.

To fill this gap, LiDAR (Light Detection and Ranging) sensors offer a compelling solution. Unlike cameras, LiDAR sensors don't depend on light. Rather, they send out laser pulses and calculate the time interval for the signals to reflect back from nearby surfaces. This gives accurate distance measurements, regardless of how bright or dark the environment is. LiDAR excels at providing reliable spatial information, making it ideal for detecting the presence and position of nearby objects in all lighting conditions. However, while LiDAR offers excellent range data, it lacks visual context. The data is often sparse, especially with 2D LiDARs, and doesn't provide color, shape, or texture details needed to understand what an object is. That's where sensor fusion comes in. By combining data from both a camera and a LiDAR sensor, we can create a more complete and reliable picture of the environment. The idea is simple: use the camera to provide rich visual information, and the LiDAR to supply accurate distance data. Together, they compensate for each other's weaknesses. In this project, we fused data from a Logitech C270 webcam and a YDLiDAR TG30 to enable object detection that works well, even in dark conditions.

The Logitech C270 is a simple and affordable webcam that captures 720p images at 30 frames per second. It’s commonly used for basic vision tasks in robotics. On the other hand, the YDLiDAR TG30 is a compact 2D LiDAR sensor capable of a full 360-degree scan, with a range of up to 12 meters. It provides distance measurements in a horizontal plane and works reliably at up to 10 Hz. Though not as powerful as more expensive 3D LiDARs, the TG30 is accurate enough for short-range navigation and obstacle detection tasks, and its low cost makes it accessible for hobby and educational projects. A custom-developed ROS 2 node named fusion_node.py serves as the backbone of the data fusion pipeline. This Python script subscribes to both LiDAR scan data and camera image topics. It then uses the extrinsic calibration parameters to transform the LiDAR points into the camera frame and project them onto the 2D image plane. The final result is an annotated camera feed with depth-aware LiDAR points, helping identify where objects exist in the environment based on both depth and vision. The entire setup process—from hardware connection and driver configuration to launching the nodes and verifying the output in RViz2—was carefully documented and executed to ensure replicability and minimal error. With this robust foundation, we proceeded to implement and validate the fusion system in a controlled indoor setting, gradually fine-tuning the parameters to optimize point cloud alignment and projection quality. The proposed system architecture for LiDAR camera sensor fusion is as shown in Figure 2.

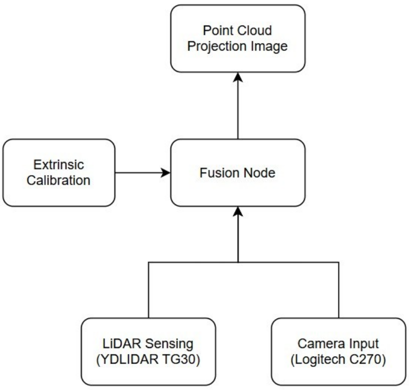


Figure 2. The proposed system architecture for LiDAR camera sensor fusion

The proposed system architecture for LiDAR–camera sensor fusion is designed to facilitate real-time projection of spatially filtered point clouds onto 2D camera images. It consists of a modular integration of hardware sensing elements and a software stack built on the ROS 2 Humble middleware. At the hardware level, the system incorporates a YDLiDAR TG30 sensor and a Logitech C270 USB camera. The LiDAR provides 2D range measurements with a 360° field of view, which are essential for constructing spatial awareness of the surrounding environment. The camera captures RGB images at 720p resolution, serving as the projection canvas for the LiDAR data. These sensors are connected to a processing unit a laptop equipped with an NVIDIA RTX 3050 GPU which is responsible for handling all computation-intensive tasks such as transformation, fusion, and visualization. The software layer is organized into four functional modules. The first module consists of sensor drivers that initialize and publish sensor data to ROS 2 topics. The LiDAR driver, launched via the custom TG.launch.py script, publishes scan data to the /scan topic, while the v4l2_camera node streams raw images to the /image_raw topic. The second module is the calibration layer, which includes both intrinsic calibration of the camera and extrinsic calibration between the LiDAR and camera coordinate frames. This calibration ensures that the spatial relationship between the two sensors is precisely captured, allowing accurate transformation of 3D LiDAR data into the 2D image plane. At the core of the architecture is the custom fusion node, fusion_node.py, implemented in Python. This node subscribes to both /scan and /image_raw, processes the incoming LiDAR data by converting polar coordinates to Cartesian points, and applies the extrinsic transformation matrix obtained from RViz2-based calibration. Using camera intrinsics, the transformed points are projected onto the image plane. An additional range filtering mechanism is employed to display only those LiDAR points falling within a specified depth window, enabling a targeted visualization of objects at certain distances. The processed frames are rendered using OpenCV, offering a live visual fusion of depth data overlaid on RGB frames.

Finally, the system includes a visualization layer comprising RViz2 and OpenCV interfaces. RViz2 is primarily used during the calibration stage to verify frame alignment, while OpenCV handles the final real-time display of the fused image. The architecture is modular and extensible, supporting the addition of object detection algorithms or 3D SLAM functionalities in future iterations. Built entirely using ROS 2 Humble, the architecture ensures real-time communication between nodes, robust synchronization, and compatibility with modern robotic frameworks.

2.1. Setting up the sensors: First let’s make sure that all the drivers of the YDLidar are properly installed if they are not so that the LiDAR works providing the correct results. The YDLiDAR TG30 is a 2D LiDAR with a 360 degrees of field of vision and a 12 meter range, while the camera captures RGB images at a resolution of 720p. Technical specifications are as listed in Table 2.

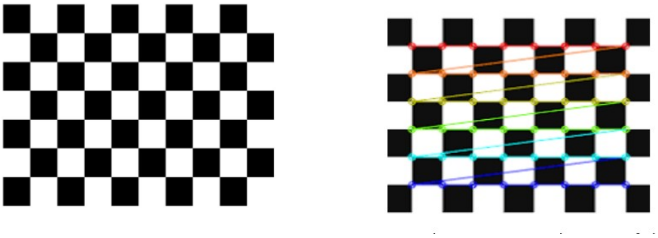
Table 2. Technical, Electrical and Mechanical Characteristics of YDLiDAR TG30

Category	**Specification**
Technical Specifications	
Range	0.05 – 30 m
Scan Angle	360° omnidirectional
Angular Resolution	0.09° – 0.22° per point
Distance Accuracy	±4 cm
Sampling Frequency	Up to 20 kHz
Scan Rotation Speed	5 – 15 Hz
Electrical & Environmental	
Supply Voltage	5.0 – 5.2 V DC
Operating Current	400 – 480 mA (starting ~500 mA)
Interface	UART, 512000 bps
Laser Wavelength	895 – 915 nm (Class I, eye-safe)
Laser Power	~85 mW
Operating temperature	–10 °C to +50 °C
Mechanical Characteristics	
Dimensions,Weight	Ø75.8 mm × 34.7 mm,~214 g
Protection Level	IP65

Now both of these sensors need to be mounted on a rigid. The position of the LiDAR should be in such a way that it's a bit behind and below the camera. Once this placement of the sensors is done, now the intrinsic and extrinsic calibration must be done to make sure that the co-ordinate frames of both the camera and LiDAR are unified. Technical, Electrical and Mechanical Characteristics of YDLiDAR TG30 as listed in Table 2.

2.2. Camera Intrinsic Calibration:

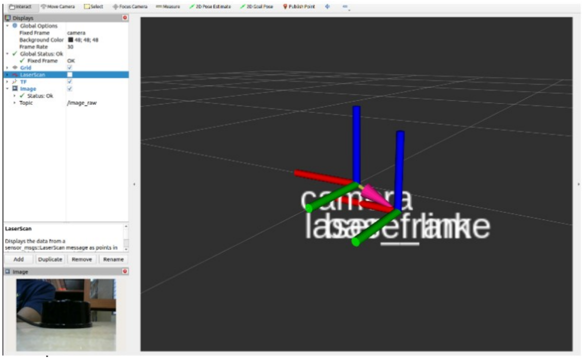
Camera intrinsic calibration was performed using a checkerboard-based method to estimate focal length, optical center, and lens distortion coefficients. As shown in Figures 3(a) and 3(b).



(a) (b)

Figure 3. (a) and (b) depict the chessboard pattern techniques employable for calibration.

Multiple images of the calibration pattern were captured at varying orientations and distances. The calibration parameters were computed using a ROS 2-compatible calibration tool and stored in a YAML configuration file, which was later loaded during runtime to correct image distortion and support accurate point projection.



Global Options
Fixed Frame camera
Background Color 48; 48; 48
Frame Rate 30
Global Status: Ok
Fixed Frame OK
Grid ☑
LaserScan ☐
TF ☑
Image ☑
Status: Ok
Topic /image_raw

LaserScan

Displays the data from a sensor_msgs::LaserScan message as points in

Figure 4. Projection coordinates onto image plane

Figure 4 shows that the coordinates required for the calibration. The coordinates which represent the camera is represented as camera as per the physical setup.

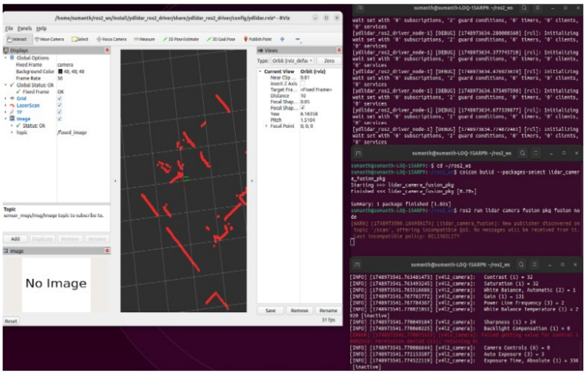
2.3 Extrinsic Calibration between Sensors: To unify the LiDAR frame to the camera frame, the extrinsic parameter is really important.

$$T = \begin{bmatrix} 1 & 0 & 0.12 \\ 0 & 1 & 0.00 \\ 0 & 0 & 0.05 \end{bmatrix} \qquad \text{………….……………... (1)}$$

Extrinsic calibration between the LiDAR and camera was achieved by defining a static transformation based on the physical mounting arrangement. Initial estimates of translation and rotation were obtained through visual alignment in RViz2, followed by manual refinement to minimize projection error. The final transformation matrix defines the relative pose of the LiDAR frame with respect to the camera frame and is applied to transform LiDAR points into the camera coordinate system before projection.

2.4 Data Fusion Algorithm:

The core of the fusion system is a custom ROS2 node developed in Python. This node synchronizes incoming data streams from both sensors and performs geometric transformations to project LiDAR points onto the camera's image plane. As shown in Figure 5.



Global Options
Fixed Frame camera
Background Color 48; 48; 48
Frame Rate 30
Global Status: Ok
Fixed Frame OK
Grid ✓
LaserScan ✓
TF ✓
Image ✓
Status: Ok
Topic /fused_image

Topic
sensor_msgs/msg/Image topic to subscribe to.

Type: Orbit (rviz_defa Zero
Current View Orbit (rviz)
Near Clip ... 0.01
Invert Z Axis ☐
Target Fra... <Fixed Frame>
Distance 10
Focal Shap... 0.05
Focal Shap... ✓
Yaw 6.16358
Pitch 1.5104
Focal Point 0; 0; 0

```
wait set with '0' subscriptions, '2' guard conditions, '0' timers, '0' clients,
'0' services
[ydlidar_ros2_driver_node-1] [DEBUG] [1748973634.280008560] [rcl]: Initializing
wait set with '0' subscriptions, '2' guard conditions, '0' timers, '0' clients,
'0' services
[ydlidar_ros2_driver_node-1] [DEBUG] [1748973634.377745710] [rcl]: Initializing
wait set with '0' subscriptions, '2' guard conditions, '0' timers, '0' clients,
'0' services
[ydlidar_ros2_driver_node-1] [DEBUG] [1748973634.476923619] [rcl]: Initializing
wait set with '0' subscriptions, '2' guard conditions, '0' timers, '0' clients,
'0' services
[ydlidar_ros2_driver_node-1] [DEBUG] [1748973634.575497598] [rcl]: Initializing
wait set with '0' subscriptions, '2' guard conditions, '0' timers, '0' clients,
'0' services
[ydlidar_ros2_driver_node-1] [DEBUG] [1748973634.677339877] [rcl]: Initializing
wait set with '0' subscriptions, '2' guard conditions, '0' timers, '0' clients,
'0' services
[ydlidar_ros2_driver_node-1] [DEBUG] [1748973634.774872461] [rcl]: Initializing
wait set with '0' subscriptions, '2' guard conditions, '0' timers, '0' clients,
'0' services
```

```
sumanth@sumanth-LOQ-15ARP9:~$ cd ~/ros2_ws
sumanth@sumanth-LOQ-15ARP9:~/ros2_ws$ colcon build --packages-select lidar_camer
a_fusion_pkg
Starting >>> lidar_camera_fusion_pkg
Finished <<< lidar_camera_fusion_pkg [0.78s]

Summary: 1 package finished [1.02s]
sumanth@sumanth-LOQ-15ARP9:~/ros2_ws$ ros2 run lidar_camera_fusion_pkg fusion_no
de
[WARN] [1748973580.184950174] [lidar_camera_fusion]: New publisher discovered on
 topic '/scan', offering incompatible QoS. No messages will be received from it.
 Last incompatible policy: RELIABILITY
```

```
[INFO] [1748973541.763481473] [v4l2_camera]:   Contrast (1) = 32
[INFO] [1748973541.763493245] [v4l2_camera]:   Saturation (1) = 32
[INFO] [1748973541.765516686] [v4l2_camera]:   White Balance, Automatic (2) = 1
[INFO] [1748973541.767763772] [v4l2_camera]:   Gain (1) = 131
[INFO] [1748973541.767784367] [v4l2_camera]:   Power Line Frequency (3) = 2
[INFO] [1748973541.770021851] [v4l2_camera]:   White Balance Temperature (1) = 2
920 [inactive]
[INFO] [1748973541.770049184] [v4l2_camera]:   Sharpness (1) = 24
[INFO] [1748973541.770060225] [v4l2_camera]:   Backlight Compensation (1) = 0
[ERROR] [1748973541.770075123] [v4l2_camera]: Failed getting value for control 1
0092545: Permission denied (13); returning 0!
[INFO] [1748973541.770086644] [v4l2_camera]:   Camera Controls (6) = 0
[INFO] [1748973541.772153507] [v4l2_camera]:   Auto Exposure (3) = 3
[INFO] [1748973541.774522119] [v4l2_camera]:   Exposure Time, Absolute (1) = 336
 [inactive]
```

Figure 5. Projection coordinates onto image plane

The algorithm follows these steps:

 # Pseudo Code for algorithm

Algorithm LiDAR_Camera_Fusion

Input: LiDAR_scans (raw 2D LiDAR data), Camera_frames (RGB images), Camera_intrinsics (fx, fy, cx, cy), Extrinsic_matrix (R, T)

Output: Fused_Image (RGB image with depth projection)

```
1. Data Acquisition

    1.1 Initialize LiDAR (YDLiDAR TG30) and Camera (Logitech C270).
    1.2 While system is running:

            - Capture LiDAR_scan from TG30
            - Capture RGB_frame from C270
- Ensure both data are time-synchronized
```

```
2. Point Conversion

   2.1 For each angle θ and distance d in LiDAR_scan:

       - Compute Cartesian coordinates:
             $x_L = d_i \cos(\theta_i)$                (1)
             $y_L = d_i \sin(\theta_i)$                (2)
             $z_L = 0$   # 2D LiDAR (planar assumption)
             - Store point $P_L = [x_L, y_L, z_L, 1]$         (3)

3. Coordinate Transformation

   3.1 For each LiDAR point P_L:

       - Transform into Camera coordinate frame:
             $P_C = T_{CL} \cdot P_L$              (4)
             - Extract $(X_C, Y_C, Z_C)$ from P_C

   3.2 Discard points where $Z_C \le 0$ (behind camera)

4. Projection onto Image Plane

   4.1 For each transformed point $(X_C, Y_C, Z_C)$:

       - Compute pixel coordinates:
             $u = (f_x X_C / Z_C) + c_x$           (5)
             $v = (f_y Y_C / Z_C) + c_y$           (6)
       - If u,v inside image boundaries:
             - Overlay depth/color marker on RGB_frame
             - Optionally encode LiDAR depth as a colormap

4.2 Return the fused image:

   Fused_Image = RGB_frame with LiDAR projections
   End Algorithm
```

1. Data Acquisition

The node subscribes to:

- /scan : Contains polar range data from the LiDAR.
- /image_raw : Provides raw RGB frames from the camera.
- Both sensors are initialized, and synchronized frames are collected. Synchronization ensures the chair (or any object) is in the same position in both LiDAR and camera data.

Mathematical representations are;
The LiDAR provides range measurements di, at angle θi:
L={(di,θi)|i=1,2,…,N},
The camera captures an RGB image:
I={I(u,v)|u=1…W,v=1…H}
where W and H denote image width and height.
A timestamp ‘t’ ensures synchronization:

2. Point Conversion

Each 2D polar LiDAR reading is converted into 3D Cartesian coordinates assuming a planar surface (z = 0). The points are represented in homogeneous coordinates for matrix operations. TG30 LiDAR provides distances for angles. These are converted into Cartesian coordinates in the LiDAR frame. As shown in (1)–(3), the LiDAR polar measurements are converted into Cartesian coordinates

Thus each LiDAR point in homogeneous coordinates:

$$P_{L,i} = \begin{bmatrix} x_{L,i} \\ y_{L,i} \\ z_{L,i} \\ 1 \end{bmatrix} \qquad \ldots\ldots\ldots\ldots\ldots\ldots..(2)$$

3. Coordinate Transformation

The 3D LiDAR points are transformed into the camera coordinate system using the predefined extrinsic transformation matrix. This maps the spatial information captured by the LiDAR to the perspective of the camera. Each LiDAR point is transformed into the camera’s coordinate system using the extrinsic calibration matrix (rotation + translation).

Using the extrinsic calibration matrix:

$$T_{CL} = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix}_{4\times4} \qquad \ldots\ldots\ldots\ldots\ldots\ldots.(3)$$

where;
R∈R3×3: rotation matrix
T∈R3×1: translation vector
The transformation is: PC,i=TCL⋅PL,i

$$\begin{bmatrix} X_{C,i} \\ Y_{C,i} \\ Z_{C,i} \\ 1 \end{bmatrix} = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_{L,i} \\ y_{L,i} \\ z_{L,i} \\ 1 \end{bmatrix} \qquad \ldots\ldots\ldots\ldots.(4)$$

4. Projection onto Image Plane

3D points projection onto the 2D image plane by using the camera’s intrinsic matrix $[K]$ K. the transformed 3D points are projected onto the 2D image plane. The projection equations follow the pinhole camera model. Camera intrinsics (fx, fy, cx, cy) are used to project 3D LiDAR points into 2D pixel coordinates on the RGB image. This allows overlaying LiDAR points directly on the video frame for object-level visualization.

Using the camera intrinsic matrix:

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(5)$$

The projection is:

$$\begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \frac{1}{Z_{C,i}} \cdot K \cdot \begin{bmatrix} X_{C,i} \\ Y_{C,i} \\ Z_{C,i} \end{bmatrix} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(6)$$

Thus pixel coordinates:

$$u_i = \frac{f_x X_{C,i}}{Z_{C,i}} + c_x, \qquad v_i = \frac{f_y Y_{C,i}}{Z_{C,i}} + c_y$$

Only valid if ZC,i>0 and (ui,vi) lies within the image bounds.

5. Final output:

The fused image is defined as:

$$\mathcal{F}(u,v) = \mathcal{I}(u,v) \ \cup \ \{(u_i, v_i, d_i)\} \quad \ldots\ldots\ldots\ldots\ldots..(7)$$

where the RGB image is enriched with LiDAR-projected depth points.

3. Results and Discussions

This section outlines the results obtained from the sensor fusion of a 2D YDLiDAR TG30 and a Logitech C270 720p USB camera employing ROS 2 Humble on Ubuntu 22.04 for Object-Level accurate perception. This study presented a practical and cost-effective approach to LiDAR–camera sensor fusion aimed at enhancing the perception capabilities of low-cost mobile robots. Conventional systems that rely on expensive infrared cameras or high resolution 3D LiDAR. Our work demonstrated that meaningful and reliable environmental awareness can be achieved using a 2D LiDAR (YDLiDAR TG30) and a standard USB camera (Logitech C270), integrated through a carefully designed fusion framework. The extrinsic calibration process ensured precise alignment between modalities, enabling the generation of accurate point cloud projections onto camera images. Figure 7 illustrates the successful projection of point clouds onto an object, confirming the system's ability to detect and represent objects at varying distances.

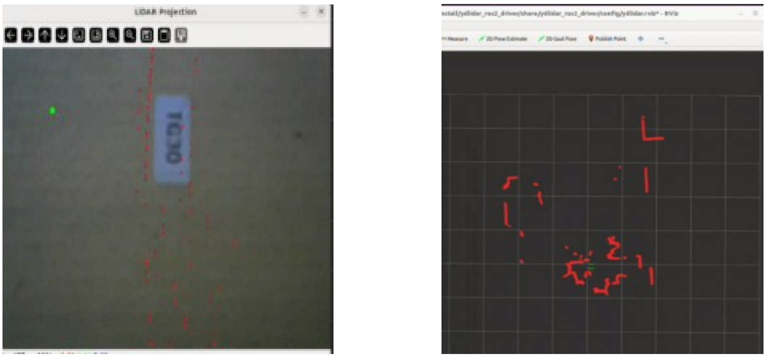
Our research focus is on to achieve the affordable robotic platforms with advanced perception even in environments with limited lighting conditions. By leveraging widely available hardware and open-source software, we created a real-time fusion system that can be readily deployed without the prohibitive costs typically associated with advanced sensor suites. This makes the system highly suitable for small research laboratories, educational robotics projects, and practical applications such as indoor navigation, warehouse management, safety monitoring, and low-light exploration tasks.

The key contributions of this work can be summarized as follows:

Design of a low-cost, real-time LiDAR–camera fusion framework that bridges the gap between affordability and performance.

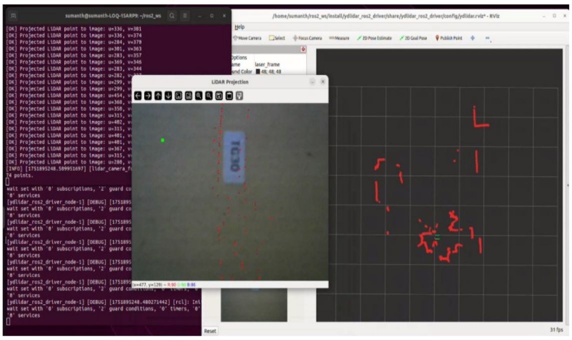
- Successful validation through point cloud projection experiments, demonstrating accurate detection and visualization of object of interest at a particular distance.
- Practical deployment potential in domains such as autonomous indoor navigation, low-light surveillance, and safety-critical monitoring where traditional solutions are either costly or impractical.
- Contribution to the understanding of sensor-level fusion, offering a foundation for future advancements in real-time robotics applications.

Looking ahead, this work paves the way for several promising research directions. These include expanding the framework to incorporate 3D LiDAR and depth cameras for richer spatial understanding in dynamic or outdoor environments. The main objective was to project LiDAR point clouds onto the image frame captured by the camera and visualize them with respect to their distance from the sensor setup. The system architecture included real-time data acquisition from both sensors, synchronization of their data streams, transformation of LiDAR points into the camera coordinate frame employing extrinsic calibration, and visualization in RViz2. The LiDAR was operated employing the `ydlidar_ros2_driver` package, and the fusion node was implemented in the `lidar_camera_fusion_pkg` package under the `fusion_node.py` file. It forms a robust foundation for further experimentation in real-time robotic applications.



(a) (b)

Figure 6. (a) LiDAR Projection and (b) Point-clouds Visualization.



```
[OK] Projected LiDAR point to image: u=336, v=381
[OK] Projected LiDAR point to image: u=336, v=374
[OK] Projected LiDAR point to image: u=284, v=379
[OK] Projected LiDAR point to image: u=301, v=363
[OK] Projected LiDAR point to image: u=283, v=357
[OK] Projected LiDAR point to image: u=369, v=346
[OK] Projected LiDAR point to image: u=283, v=344
[OK] Projected LiDAR point to image: u=282, v
[OK] Projected LiDAR point to image: u=299, v
[OK] Projected LiDAR point to image: u=299, v
[OK] Projected LiDAR point to image: u=454, v
[OK] Projected LiDAR point to image: u=368, v
[OK] Projected LiDAR point to image: u=358, v
[OK] Projected LiDAR point to image: u=315, v
[OK] Projected LiDAR point to image: u=402, v
[OK] Projected LiDAR point to image: u=315, v
[OK] Projected LiDAR point to image: u=401, v
[OK] Projected LiDAR point to image: u=401, v
[OK] Projected LiDAR point to image: u=367, v
[OK] Projected LiDAR point to image: u=315, v
[OK] Projected LiDAR point to image: u=286, v
[INFO] [1751895248.509951697] [lidar_camera_f
74 points.

wait set with '0' subscriptions, '2' guard co
'0' services
[ydlidar_ros2_driver_node-1] [DEBUG] [1751895
wait set with '0' subscriptions, '2' guard co
'0' services
[ydlidar_ros2_driver_node-1] [DEBUG] [1751895
wait set with '0' subscriptions, '2' guard co
'0' services
[ydlidar_ros2_driver_node-1] [DEBUG] [1751895
wait set with '0' subscriptions, '2' guard co
'0' services
[ydlidar_ros2_driver_node-1] [DEBUG] [1751895
wait set with '0' subscriptions, '2' guard co
'0' services
[ydlidar_ros2_driver_node-1] [DEBUG] [1751895
wait set with '0' subscriptions, '2' guard co
'0' services
[ydlidar_ros2_driver_node-1] [DEBUG] [1751895248.480271442] [rcl]: Ini
wait set with '0' subscriptions, '2' guard conditions, '0' timers, '0'
'0' services
```

Figure 7. Successful Projection of point cloud onto an object.

The results obtained show that the projection of LiDAR points onto the camera frame was successful for points within 1 to 3 meters. This was validated visually in RViz2 and in the OpenCV-based projection window as in figure 8 and figure 9 subsequently. Red points were rendered on detected surfaces within the camera image corresponding to LiDAR distance readings. Errors were primarily noticed in calibration drift if the camera or LiDAR was moved after calibration. The accuracy of projection relied heavily on the proper extrinsic calibration between the LiDAR and the camera. The system was found to be suitable for near-range perception tasks such as obstacle detection and object localization. Although the object-specific projection was initially planned employing YOLOv5, it was later dropped to focus solely on accurate point projection by distance. This decision simplified the system, reduced processing overhead, and maintained the real-time capability of the sensor fusion pipeline.

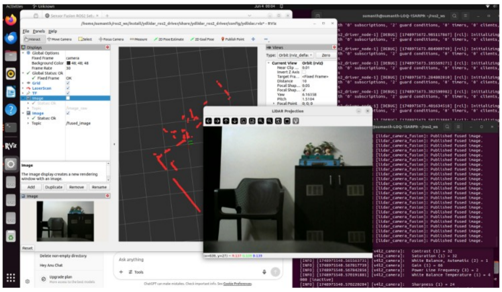


Figure 8. Indoor Environment for specific object-level perception

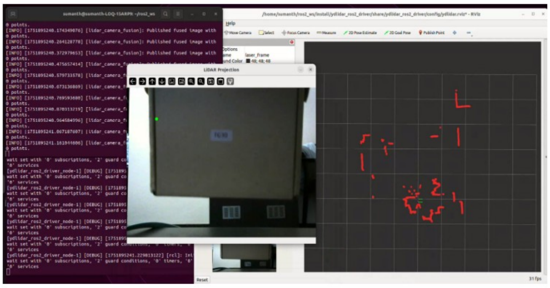


Figure 9. Overall view of program execution, LiDAR Projection and Point clouds Visualization.

4. Conclusion

This work demonstrated the effectiveness of a low-cost multi-modal sensor fusion framework combining a 2D LiDAR (YDLiDAR TG30) and a standard RGB camera (Logitech C270) to achieve precise perception of particular object for mobile robotic applications. We successfully integrated sensing modalities by leveraging ROS 2 Humble on Ubuntu 22.04. This work helps to overcome the built in limitations to using either sensor independently. The process of extrinsic calibration played a significant role in establish precise geometric alignment between depth measurements of LiDAR and imagery from camera. This enabled the generation of reliable point cloud projections onto RGB frames and it helped significantly improving object detection, localization, and scene understanding. Successful projection of point clouds onto real-world objects at various distances, validate the robustness and accuracy of the proposed approach. The indoor experimental result gave satisfied results in this matter. High resolution 3D LiDAR are very expensive and not suitable for immediate beginner working in this domain. Our approach will help to create low cost and effective practical alternative without compromising the integrity of object-level perception. It can be achieved with relatively affordable when supported by a well-designed fusion pipeline. Overall, this research underscores the potential of low-cost sensor fusion systems to broaden the accessibility of autonomous perception technologies, particularly for educational platforms, small-scale mobile robots, and deployments on resource-constrained devices.

5. Extended Observations and Future Work

Throughout multiple calibration and fusion tests, consistency and stability were observed in the transformation matrix when aligning LiDAR point clouds with the camera image plane. This indicates a reliable calibration process and validates the mathematical modeling involved. Variations in lighting conditions and surface reflectivity did impact the quality of visual features, which may influence the accuracy of point cloud overlays. Further improvements can be made by integrating adaptive filtering or point cloud registration techniques. This work investigates real-time synchronization strategies, including hardware based triggering and software level timestamp compensation, to achieve precise temporal alignment in LiDAR camera fusion systems. By combining LiDAR derived geometric depth with semantically rich visual information, the proposed framework enables robust multi-modal perception for complex and dynamic environments. Such capabilities support critical applications including disaster monitoring, forest surveillance, intelligent transportation, and infrastructure inspection. Consequently, the proposed approach aligns with energy efficient and sustainable development goals by enabling scalable, resource-aware environmental monitoring and decision-making systems.

6. References

[1] X. Yuan, Y. Liu, T. Xiong, W. Zeng, and C. Wang, “Semantic fusion algorithm of 2D LiDAR and camera based on contour and inverse projection,” Sensors, vol. 25, no. 8, p. 2526, 2025.

[2] H. Alai and R. Rajamani, “Low-cost camera and 2-D LiDAR fusion for target vehicle corner detection and tracking: Applications to micromobility devices,” Mechanical Systems and Signal Processing, vol. 206, p. 110891, 2024.

[3] Y. Yu, Z. Liu, and L. Chen, “Enhanced SLAM based on 2D LiDAR and RGB-D camera fusion for mobile robots navigation,” Int. J. Vehicle Systems Modelling and Testing, vol. 20, pp. 101–120, 2025.

[4] C. Dai, F. Xu, and Y. Lin, “Enhanced object detection in autonomous vehicles through LiDAR camera sensor fusion,” World Electric Vehicle Journal, vol. 15, p. 2250, 2024.

[5] Y. Liu, X. Zhang, and J. Li, “Design of a mobile 3D imaging system based on 2D LiDAR and calibration with Levenberg–Marquardt optimization algorithm,” Frontiers in Physics, vol. 10, 2022.

[6] L. Schichler, K. Festl, and S. Solmaz, “Robust multi-sensor fusion for localization in hazardous environments using thermal, LiDAR, and GNSS data,” Sensors, vol. 25, p. 2032, 2025.

[7] Z. Fan et al., “LiDAR, IMU, and camera fusion for simultaneous localization and mapping: A systematic review,” Cognitive Systems Research, 2025.

[8] G. Pandey, J. McBride, S. Savarese, and R. M. Eustice, “Automatic extrinsic calibration of vision and lidar by maximizing mutual information,” Journal of Field Robotics, vol. 32, no. 5, pp. 696–722, 2015.

[9] J. Zhang and S. Singh, “Visual–lidar odometry and mapping: Low-drift, robust, and fast,” IEEE Int. Conf. Robotics and Automation (ICRA), 2015.

[10] C. Cadena et al., “Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age,” IEEE Transactions on Robotics, vol. 32, no. 6, pp. 1309–1332, 2016.

[11] W. Maddern et al., “1 Year, 1000 km: The Oxford RobotCar dataset,” Int. J. Robotics Research, vol. 36, no. 1, pp. 3–15, 2017.

[12] A. Geiger, P. Lenz, and R. Urtasun, “Are we ready for autonomous driving? The KITTI vision benchmark suite,” Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR), 2012, pp. 3354–3361.

[13] H. Huang et al., “Multi-sensor fusion for autonomous driving: A review,” IEEE Transactions on Intelligent Transportation Systems, vol. 23, no. 6, pp. 1–15, 2022.