3D Pose Estimate of an Object in the Environment

1st Kamalnath Bathirappan
MS. Robotics
Northeastern University
Boston, MA, USA
bathirappan.k@northeastern.edu

2nd Ahilesh Ram
MS. Robotics
Northeastern University
Boston, MA, USA
rajaram.a@northeastern.edu

***Abstract* - This paper presents an implementation of markerless object pose estimation using YOLOv5 and depth sensing capabilities in ROS2. Unlike conventional approaches that rely on markers for pose estimation, our method utilizes deep learning-based object detection combined with depth information to achieve accurate 6D pose estimation of objects. The system leverages the robust object detection capabilities of YOLOv5 with OpenCV's solvePnP for pose calculation, providing a flexible solution that works with both RGB-only and RGB-D camera setups. Our implementation draws inspiration from NVIDIA's FoundationPose, which offers a unified framework for 6D pose estimation, but creates a more accessible solution that doesn't require specialized hardware or Docker implementations. Experimental results with Intel RealSense and Orbbec Astra cameras demonstrate the effectiveness of our approach in real-world scenarios.**

Keywords - Markerless pose estimation, YOLOv5, RGB-D sensing, 6D object pose, ROS2 integration, Depth filtering, Robot perception

I. Introduction

Object detection and pose estimation are fundamental capabilities for robotic systems that need to interact with their environment. Traditional approaches often rely on fiducial markers or predefined patterns attached to objects to simplify the pose estimation task. However, this requirement limits the applicability in many real-world scenarios where objects cannot be modified with markers.

While the find_object_2d package in ROS provides a framework for detecting objects using feature descriptors such as SIFT, SURF, FAST, and BRIEF [1], our approach leverages YOLOv5, a modern deep learning-based object detection system. YOLOv5 offers superior detection performance for common objects and can be easily integrated with depth information from sensors like Intel RealSense or Orbbec Astra cameras to estimate 3D position and orientation.

Our work implements a complete ROS2-based markerless pose estimation system that can determine both position and orientation of objects using only their natural visual features and depth information. This paper discusses the implementation details, challenges encountered, and performance evaluation of our system across both RGB-only and RGB-D camera setups.

II. Related Work

A. *Deep Learning-Based Object Detection*

Modern object detection methods have shifted from traditional feature-based approaches to deep learning models. YOLO (You Only Look Once) represents a family of one-stage object detectors that have achieved state-of-the-art performance in both accuracy and speed [3]. YOLOv5 specifically offers an excellent balance of detection performance and computational efficiency, making it suitable for real-time robotic applications. Unlike feature-based methods, deep learning approaches can better handle variations in appearance, lighting, and viewpoint.

B. *Feature-Based Object Detection*

The find_object_2d package provides a practical framework for feature-based object detection in ROS [5]. It extracts distinctive visual features from template images using algorithms like SIFT, SURF, and FAST, then matches them against features in live camera feeds. For RGB data, it detects objects and estimates their 2D position and orientation in the image plane, publishing this information as ROS messages. This approach works well for objects with strong texture patterns but struggles with plain or reflective surfaces and is sensitive to lighting changes, and works better with flat and plain backgrounds.

The find_object_3d variant extends these capabilities to RGB-D data by incorporating depth information from sensors like the Kinect or RealSense cameras [6]. By combining the 2D feature matches with corresponding depth values, it calculates the full 3D position of detected objects relative to the camera frame. These positions can be broadcast as TF transforms for integration with robotic manipulation tasks. However, the depth-enhanced version still inherits the limitations of feature-based matching and requires depth sensors with good registration between RGB and depth data.

C. *Depth-Enhanced Pose Estimation*

Integrating depth information with visual features significantly improves pose estimation accuracy. Recent approaches combine object detection with depth data from RGB-D cameras to recover the 6D pose of objects [4]. The Dist-YOLO approach extends YOLO's capabilities to include distance estimation, enabling more accurate localization in 3D space [5]. These methods typically extract depth values from the detected region and combine them with 2D detections to solve for the complete pose.

D. Marker-Based Pose Estimation

Many existing solutions rely on fiducial markers for accurate 6D pose estimation. These approaches attach visual markers to objects and use the known geometry of these markers to compute full pose information [6]. While effective, this requirement limits applicability in scenarios where objects cannot be modified.

E. Learning-Based Pose Estimation

Recent advances in deep learning have enabled more sophisticated pose estimation approaches. NVIDIA's FoundationPose [7] represents a state-of-the-art solution that can estimate 6D poses of novel objects without markers using either CAD models or a small set of reference images. It employs neural implicit representations and transformer-based architectures to achieve high accuracy and generalization capabilities.

III. System Architecture

Our system architecture is designed as a modular pipeline that leverages ROS2 (Robot Operating System 2) to enable flexible, distributed processing for markerless pose estimation. The architecture consists of five key modules that work together to transform raw camera data into 6D pose information for detected objects.

The Image Acquisition Module serves as the entry point for the system, capturing both RGB and depth information from various supported sensors. For RGB-only implementations, standard USB cameras provide color images at resolutions up to 1920×1080. For RGB-D implementations, structured light sensors (Orbbec Dabai DWC) simultaneously capture aligned color and depth frames. The module handles camera interfacing, image acquisition, and preliminary processing such as color conversion and depth registration. Within the ROS2 framework, this module publishes synchronized image topics that conform to the standard sensor_msgs/Image message type, making the data accessible to other nodes in the system.

The Object Detection Module processes the RGB images to identify and localize objects of interest. At its core, this module utilizes YOLOv5, a state-of-the-art deep learning-based object detector that provides real-time performance while maintaining high accuracy. YOLOv5 divides input images into a grid and predicts bounding boxes and class probabilities in a single forward pass, making it significantly faster than two-stage detectors. Our implementation supports both pre-trained COCO dataset classes (80 common objects) and custom-trained models for specific application domains. The module extracts bounding box coordinates, class labels, and confidence scores from the detection results, filtering them based on predefined confidence thresholds and target object classes.

The Depth Processing Module becomes active in RGB-D implementations and is responsible for extracting reliable depth information corresponding to detected objects. This module analyzes the region of interest defined by each object's bounding box in the depth image, applying statistical methods to filter out invalid measurements and noise. The module compensates for common depth sensing issues such as missing values at object boundaries, specular reflections, and measurement errors at extreme distances. By computing robust statistics (mean, median, or percentile-based) on the valid depth values within the region of interest, it provides accurate distance information critical for subsequent pose estimation.

The Pose Estimation Module represents the core computational component of the system, transforming 2D object detections into full 6D poses (position and orientation). For each detected object, this module creates a 3D model based on class-specific dimension priors stored in a database. It then establishes correspondences between the 2D bounding box corners and the 3D object model, using these correspondences to solve the Perspective-n-Point (PnP) problem through OpenCV's solvePnP function. The module employs different strategies based on available sensor data:

- In RGB-only mode, it relies on assumed object dimensions and camera calibration parameters to estimate pose with scale ambiguity.
- In RGB-D mode, it incorporates the measured object distance to resolve scale ambiguity and improve position accuracy. The output includes the full 6D pose represented as a translation vector and rotation vector (later converted to quaternion for ROS2 compatibility).

The ROS2 Integration Module handles the communication aspects of the system, managing the flow of data between modules and providing interfaces for external components. This module configures and manages the ROS2 topics, services, and transform frames required for system operation. It publishes pose estimation results as geometry_msgs/PoseStamped messages on dedicated topics and broadcasts transform frames using the ROS2 TF system, enabling visualization in RViz2 and integration with other ROS2 components such as navigation or manipulation systems. Additionally, it provides debug outputs through visualization topics that overlay detection and pose information on camera images.

The system workflow follows a logical sequence starting with data acquisition and ending with pose publication:

1. The Image Acquisition Module captures synchronized RGB and depth frames, publishing them to corresponding ROS2 topics.
2. The Object Detection Module subscribes to RGB image topics, applies YOLOv5 detection, and extracts object information.
3. For each detected object of interest, the system:
    - In RGB-D mode: The Depth Processing Module extracts and filters depth information from the corresponding region in the depth image.
    - The Pose Estimation Module computes the 6D pose using appropriate strategies based on available data.
4. The ROS2 Integration Module publishes the pose information and broadcasts transform frames.

5. External systems such as RViz2 or robot controllers consume the published pose data for visualization or further processing.

This modular architecture provides several advantages, including the ability to substitute different implementations for specific modules (e.g., different object detectors), distributed processing across multiple compute nodes, and flexible deployment options ranging from embedded systems to high-performance computing environments.

IV. Implementation

A. Initial Feature-based Object Detection (RGB & RGBD)

Our implementation begins with feature-based object detection using the find_object_2d package to establish a baseline approach. We found that feature-based detection performs exceptionally well in controlled environments with flat, plain backgrounds. The high contrast between the object and background creates distinct feature points that are easily matched. However, performance degrades significantly with cluttered backgrounds, varying lighting conditions, or when detecting objects with minimal texture. This limitation led us to explore deep learning alternatives that offer more robust detection across diverse environments.

The feature-based approach provides a foundation for understanding the challenges of markerless pose estimation, but its practical applications are limited by its sensitivity to environmental conditions. For our final implementation, we transition to a more robust deep learning-based detection system while incorporating lessons learned from the feature-based approach, particularly regarding the importance of distinctive visual features for accurate pose estimation.

Implementation of find_object_2D with RGB [12] & with RGBD [13] can be seen in the reference section.

B. YOLOv5 Integration

We utilize YOLOv5 as the foundation for our object detection pipeline. The model is loaded using PyTorch Hub, allowing easy deployment without complicated setup procedures. YOLOv5 provides bounding box coordinates, class labels, and confidence scores for detected objects.

For custom objects not in the COCO dataset, we developed a training pipeline that includes data collection, annotation, and model training. Our custom dataset included objects such as blue box, coin, tape, earphone, and earphone capsule, which were labeled using the LabelImg application.

Implementation of YOLO object detection with RBG camera with [15] & without ROS2 [14] & with RGBD [16] can be seen in the reference section.

C. Camera Calibration and Depth Integration

Camera calibration is essential for accurate pose estimation. We used OpenCV's calibration routines with a checkerboard pattern to obtain intrinsic camera parameters (camera matrix K and distortion coefficients) for both RGB-only and RGB-D cameras.

For RGB-D cameras, we process depth images to extract accurate distance measurements. The depth processing module handles:

- Region-of-interest extraction from the detected bounding box
- Filtering of invalid depth measurements (zeros or NaNs)
- Statistical aggregation to obtain robust depth values

D. Markerless Pose Estimation

The core of our system is the markerless pose estimation algorithm, which uses object class information to make assumptions about 3D structure. For each detected object, we define a cuboid model with class-specific dimensions:

These dimensions are used to create a 3D model of the object, which is then aligned with the detected 2D bounding box using OpenCV's solvePnP function:

E. ROS2 Integration

The system is fully integrated into the ROS2 ecosystem, providing standard interfaces for data exchange and visualization. We implement:

1. ROS2 Node Structure: Both RGB-only (PoseEstimatorNode) and RGB-D (PoseEstimatorRGBDNode) implementations follow ROS2 node design patterns.
2. Topic Publishers:
   - o Pose information is published as geometry_msgs/PoseStamped messages.
   - o Visualization images with overlaid axes are published as sensor_msgs/Image.
3. TF Broadcasting: Object poses are broadcast as TF transforms to enable visualization and spatial relationships.
4. RViz2 Visualization: Custom RViz2 configuration files (camera_config_rgb.rviz and camera_config_rgbd.rviz) are provided for easy visualization of the system's output.

V. Experiments and Results

A. Dataset and Training

For our experimentation, we created a custom dataset consisting of five object classes: blue box, coin, tape, earphone, and earphone capsule. We collected multiple images of each object in different positions and angles to ensure robust detection. These images were manually annotated using the LabelImg application to create bounding box coordinates in YOLO format.

The training process involved:

1. Data collection with standard webcams and RGB-D cameras (Orbbec Dabai DWC)

2. Manual annotation of objects with bounding boxes
3. Dataset splitting into training (80%) and validation (20%) sets
4. Training YOLOv5 models with different configurations

Training was performed with the standard YOLOv5 training pipeline, monitoring key metrics including box loss, object loss, classification loss, precision, recall, and mAP (mean Average Precision). Training results demonstrating the convergence of loss functions and improvement in detection metrics over training epochs.

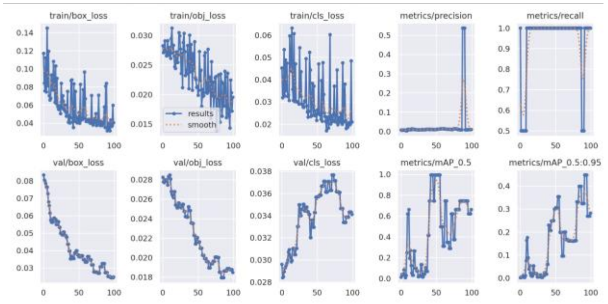


Fig1. Training Loss and Results

B. Model Evaluation

The trained model's performance was evaluated using standard object detection metrics. The recall-confidence, precision-recall, and precision-confidence curves for the trained model, are shown with different performance characteristics for each object class. The blue box class achieved the highest precision (0.995) while the coin class showed lower precision (0.497), but together they achieved an overall mAP@0.5 of 0.746.

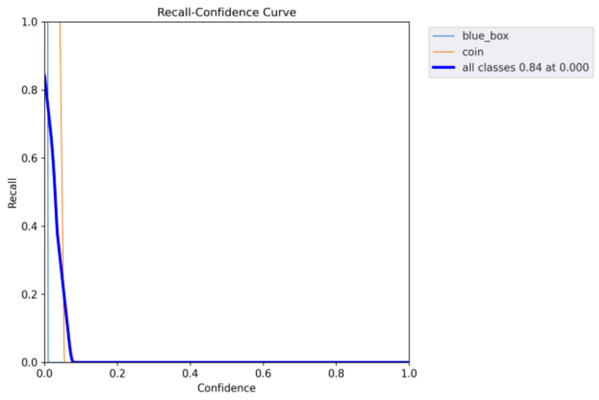


Fig2. Model Evaluation Results

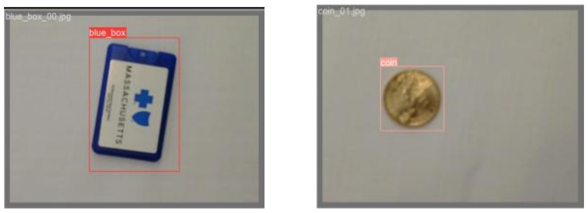


Fig3. Prediction Results of Model

As we focused more on pose estimation and not training the model, we used a limited amount of data for training, and further used existing models like YOLO v5 to save time on the training part. This also affects the efficiency in dynamic environments due to the lack of training data for custom objects.

C. RGB vs. RGB-D Performance Comparison

We conducted comparative experiments between RGB-only and RGB-D approaches to pose estimation:

1. RGB-Only Implementation:
   - Relies solely on 2D bounding box coordinates and assumed object dimensions
   - Higher frame rates (15-20 Hz on standard laptop hardware)
   - Less accurate in position estimation, particularly in depth (z-axis)
   - More sensitive to background complexity and lighting conditions

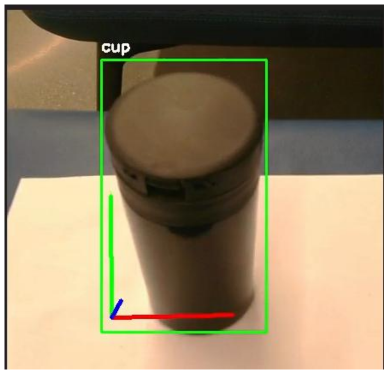


Fig4. YOLO Pose RGB

2. RGB-D Implementation:
   - Incorporates depth information from RGB-D cameras (Orbbec Astra)
   - Lower frame rates (8-12 Hz) due to additional depth processing
   - Significantly better position accuracy, especially in depth estimation
   - More robust to lighting variations
   - Challenges with reflective objects causing invalid depth readings

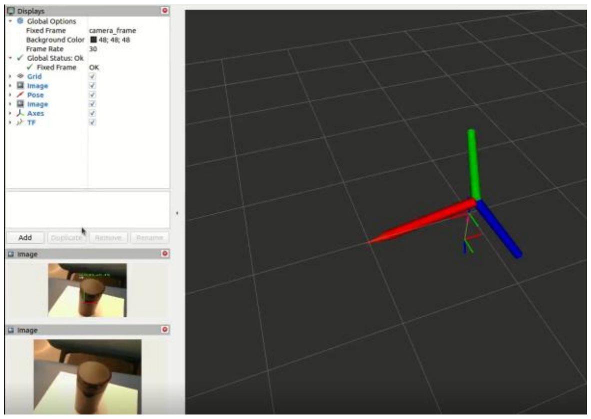
The RGB-D approach demonstrated superior pose estimation accuracy, particularly for depth estimation. However, we encountered challenges with depth image quality and object boundary detection. To address this, we implemented a method that extracts the average depth value from the valid pixels within the object's bounding box, improving robustness against outliers and invalid readings.

D. *ROS2 Integration and Visualization*

Both implementations were successfully integrated into ROS2, with the following components:

1. Camera Publisher Node: Captures and publishes RGB and depth images to ROS2 topics.
2. Pose Estimator Nodes: Subscribe to image topics, perform detection and pose estimation, and publish results.
3. TF Frames: Broadcast transform frames for each detected object.
4. RViz2 Visualization: Custom configuration files for visualizing the results.

The figures show examples of successful object detections and pose estimations, with the system correctly identifying objects and visualizing their 3D poses using coordinate axes. The RViz2 visualization demonstrates the system's ability to track multiple objects simultaneously and display their spatial relationships.



```
Global Options
  Fixed Frame         camera_frame
  Background Color    48; 48; 48
  Frame Rate          30
Global Status: Ok
  Fixed Frame         OK
Grid                  ✓
Image                 ✓
Pose                  ✓
Image                 ✓
Axes                  ✓
TF                    ✓
```

Fig5. RViz Pose Visualization

E. *Limitations and Challenges*

Our experimentation revealed several limitations and challenges:

1. Background Complexity: The RGB-only approach performed best with plain backgrounds; complex backgrounds degraded detection accuracy.
2. Viewing Angle Restrictions: Large changes in object viewing angles affected recognition and pose estimation accuracy, particularly for objects with less distinctive features.
3. Depth Sensor Limitations: Issues with depth readings for reflective or transparent objects, and at object boundaries where depth discontinuities occur.
4. Dataset Size: Limited training data affected model generalization capabilities.
5. Assumption of Fixed Dimensions: The predefined object dimensions limited accuracy for objects with significant size variations.

Despite these challenges, our system demonstrated effective performance in controlled environments and established a foundation for future improvements in markerless pose estimation.

VI. FUTURE WORKS

Our future work will focus on enhancing both the technical capabilities and practical applications of our markerless pose estimation system. Key technical improvements include expanding the training dataset with diverse conditions, implementing instance segmentation for more precise object boundaries, adding temporal filtering to reduce jitter, developing specialized algorithms for challenging materials, and optimizing performance for embedded platforms. We also plan to explore advanced techniques like learning-based keypoint detection, multi-view fusion, uncertainty quantification, and self-supervised learning approaches.

Beyond technical enhancements, we aim to validate our system in real-world robotic applications. By integrating with manipulation tasks such as pick-and-place operations, we can evaluate pose accuracy and reliability in practical scenarios. This closed-loop validation will provide valuable insights for refining our algorithms and addressing application-specific challenges. Through these combined efforts, we intend to make markerless pose estimation more accessible, robust, and reliable across a wider range of robotic applications.

CONCLUSION

This paper presented a markerless object pose estimation system built on YOLOv5 and ROS2 with both RGB-only and RGB-D camera support. By combining deep learning-based object detection with geometric pose solving techniques, our approach enables 6D pose estimation without requiring fiducial markers on objects.

Our experiments with custom-trained models on specific objects (blue box, coin, tape, earphone, and earphone capsule) demonstrated the flexibility of the approach, with the capability to achieve mAP@0.5 of 0.746 for custom object detection. The integration with depth sensing provides improved accuracy for real-world robotics applications, though it comes with additional processing requirements.

The implemented system provides a practical solution for robotic perception tasks where object modification is not feasible. While it does not achieve the same level of performance as specialized solutions like NVIDIA's FoundationPose, it offers a more accessible implementation that can run on standard hardware without requiring specialized GPU acceleration or containerization. The modular ROS2 architecture allows easy integration with other robotics components and visualization tools.

This project was completed within a short timeframe as a quick prototype to understand markerless pose estimation principles. Despite the time constraints, we successfully implemented and tested a complete pipeline from object detection to pose estimation and ROS2 integration. The combination of deep learning, computer vision techniques, and ROS2 integration demonstrates the potential for practical, accessible markerless pose estimation in robotic applications.

References

[1] M. Labbé, "Find-Object," http://introlab.github.io/find-object, 2011.

[2] ROS Wiki, "find_object_2d," http://wiki.ros.org/find_object_2d, 2023.

[3] Ultralytics, "YOLO: Real-Time Object Detection," https://docs.ultralytics.com/, 2023.

[4] L. Joseph, "ROS Robotics Projects," Packt Publishing, 2017.

[5] H. Zheng et al., "Dist-YOLO: Fast Object Detection with Distance Estimation," Applied Sciences, 2022.

[6] M. Fiala, "ARTag, a fiducial marker system using digital techniques," in Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005.

[7] B. Wen, W. Yang, J. Kautz, and S. Birchfield, "FoundationPose: Unified 6D Pose Estimation and Tracking of Novel Objects," https://nvlabs.github.io/FoundationPose/, 2023.

[8] OpenCV, "solvePnP Documentation," https://docs.opencv.org/4.x/d9/d0c/group__calib3d.html#ga549c2075fac14829ff4a58bc931c033d, 2023.

[9] Human Signal, "LabelImg," https://github.com/HumanSignal/labelImg, 2023.

[10] Orbbec, "OrbbecSDK ROS2," https://github.com/orbbec/OrbbecSDK_ROS2, 2023.

[11] NVIDIA, "FoundationPose," https://github.com/NVlabs/FoundationPose, 2023.

[12] find_object_2d_rgb.mp4

[13] find_object_2d_rgbd.mp4

[14] yolo_depth_rgb.mp4

[15] yolo_depth_rgb_ros2.mp4

[16] yolo_depth_rgbd_ros2.mp4