BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Excel@FIT 2026
http://excel.fit.vutbr.cz

Scene Annotation Using Augmented Reality and AI

Kirill Kurakov*

Abstract

This work presents a mobile AR application that enables users to annotate 3D objects by simply walking around them with a smartphone. The system combines real-time object detection and instance segmentation on a lightweight server, isolates target objects within the AR point cloud across multiple viewpoints, monitors scan quality through coverage and stability metrics, and exports the annotated objects as PLY files uploaded to a server. The result is an accessible, equipment-free pipeline for 3D scene annotation suitable for robotics, AR training data collection, and spatial computing.

*xkurak03@vutbr.cz, *Faculty of Information Technology, Brno University of Technology*

1. Introduction

Annotated 3D scene data is a fundamental requirement for training models in autonomous navigation, robotic manipulation, and augmented reality. Collecting such data today demands depth cameras, LiDAR rigs, or manual CAD-level labeling — all expensive and non-scalable. A smartphone, by contrast, already contains an RGB camera and an AR depth estimation framework, and this work exploits that to build an annotation pipeline with no extra hardware.

More concretely, given a real-world scene viewed through a smartphone camera, the goal is to detect and identify objects of interest, precisely segment each object across multiple viewpoints, reconstruct a labeled 3D point cloud per object, and export the result in a standard format with quality guarantees.

Each of these steps has existing solutions, but none of them fit the mobile, real-time setting. NeRF-based reconstruction [1] produces high-quality geometry but requires minutes of offline processing and dense capture. Depth-sensor pipelines (e.g., Azure Kinect [2]) achieve real-time results but require dedicated hardware. Pure 2D annotation tools (CVAT [3], LabelMe [4]) do not recover 3D structure. There is a clear gap for lightweight, real-time, RGB-only 3D annotation on mobile.

The proposed solution fills this gap with a mobile AR pipeline. A mobile AR application sends camera frames to a remote server that runs an AI segmentation model. The returned mask is projected into 3D space onto the scene's point cloud, and a brush-based interface lets the user refine the selection directly on the device. A scan quality monitor enforces sufficient coverage and point density before the session can be finalized.

2. System Pipeline and Architecture

The Scene Annotation application is a two-component system consisting of a mobile client and a remote server that communicate asynchronously. The client captures the live camera view, tracks the device's position in space, and reconstructs a sparse 3D representation of the scene, while the server hosts two neural models: an object detector that identifies candidate objects in the live feed, and a segmentation model that isolates the confirmed target from its surroundings. This separation keeps the mobile side lightweight, allows the models to run on more powerful hardware without draining the device's battery or thermal budget, and ensures that the user interface remains responsive while segmentation requests are processed in the background Figure 1.

2.1 Object Detection

In the detection phase, the client on demand transmits downscaled frames to the server, where YOLO [5] returns 2D bounding boxes with class labels and confidence scores. Detections are filtered by class and confidence, and the remaining boxes are back-projected into world space using the corresponding pose and the sparse point cloud, yielding a 3D position for each candidate. A marker Figure 6 is rendered at this position in AR, allowing the user to visually verify and

tap the intended target to start a segmentation session Figure 1.

2.2 Object Segmentation

Once the target is confirmed, the client enters the segmentation loop. For each captured frame, the image is uploaded to the server together with a 2D prompt derived from the tapped marker's projection, and FastSAM [6] returns a binary mask of the target object. The mask is then back-projected onto the ARKit point cloud: for every point whose projection into the current frame falls inside the mask, a per-point confidence counter is incremented. Points consistently labeled as foreground across multiple viewpoints accumulate higher confidence, while transient false positives are suppressed. This multi-view aggregation is robust to individual mask errors and produces a clean separation between the target object and the surrounding scene Figure 2.

2.3 Scan Quality Monitoring

A scan quality monitor runs alongside the segmentation loop to prevent premature finalization of incomplete scans. Three metrics are evaluated continuously: coverage, computed by dividing the horizontal plane around the object's centroid into eight sectors and tracking which sectors have contributed at least one confirmed frame, and point density, computed as the number of foreground points per unit volume of the object's bounding region. Third metric, stability, measured as the displacement of the estimated object center between successive updates, which indicates whether the segmentation has converged on a consistent 3D location rather than still drifting as new frames arrive.

2.4 Export and Upload

On successful completion, the accumulated foreground points are exported as a point cloud in the PLY format. The file is written locally on the device and then uploaded to the server, where it is stored together with the session's metadata: the set of photographs captured during the segmentation session, the detected class, and the quality metrics. This produces a complete, reproducible record of each annotation session and makes the resulting data immediately available for downstream use without a separate transfer step. An example of the resulting outputs is shown in Figure 4.

3. Selected Features

3.1 Visual Scanning Hints

To guide the user through complete coverage of the target object, on-screen hints are rendered during the segmentation loop, shown as the visual hints step in the segmentation pipeline Figure 2. The hints reflect the current state of the coverage monitor: the horizontal plane is divided into eight sectors relative to the object centroid, and each uncovered sector is indicated as a directional prompt in AR space. As the user walks around the object, sectors are marked as covered and their corresponding hints disappear. This feedback loop turns structured scanning into a simple visual task and noticeably improves the completeness of the resulting annotations compared with unguided capture.

3.2 Manual Segmentation Refinement

For cases in which FastSAM produces inaccurate boundaries — thin structures, reflective surfaces, or ambiguous foreground/background transitions — the application provides a polygon-brush tool for manual correction, shown as the refinement step in the segmentation pipeline Figure 2. The user draws polygons directly on the live AR view, and the enclosed region is applied as an additive or subtractive correction to the accumulated point-level confidence. Corrections persist across frames through the same back-projection mechanism used for automatic masks, so a single polygon has a consistent 3D effect regardless of the viewpoint from which it was drawn Figure 7.

4. Conclusions

This work has presented a mobile AR system for real-time 3D annotation of everyday objects, showing that a standard smartphone is sufficient hardware when the software guides the user through capture and enforces scan completeness automatically. The resulting pipeline produces annotated point clouds that are immediately usable for robotics, AR training, and spatial computing, without requiring depth sensors, offline processing, or manual labeling. Future work will focus on improving performance on reflective and partially occluded objects and evaluating the annotations as training data for downstream 3D perception models.

Acknowledgements

I would like to thank my supervisor doc. Ing. Vítězslav Beran, Ph.D., for guiding me throughout this work and for providing incredibly helpful advice along the way.

References

[1] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. In *European Con-*

ference on Computer Vision (ECCV), pages 405–421, 2020. https://www.matthewtancik.com/nerf.

[2] Microsoft Corporation. Azure Kinect DK documentation. online documentation, 2024. https://learn.microsoft.com/en-us/azure/kinect-dk/.

[3] CVAT.ai Corporation. CVAT: Computer vision annotation tool. open-source software, 2024. https://www.cvat.ai/.

[4] Bryan C. Russell, Antonio Torralba, Kevin P. Murphy, and William T. Freeman. LabelMe: A database and web-based tool for image annotation. *International Journal of Computer Vision*, 77(1–3):157–173, 2008. DOI: 10.1007/s11263-007-0090-8.

[5] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016. https://pjreddie.com/darknet/yolo/.

[6] Xu Zhao, Wenchao Ding, Yongqi An, Yinglong Du, Tao Yu, Min Li, Ming Tang, and Jinqiao Wang. Fast segment anything. arXiv preprint arXiv:2306.12156, 2023. https://arxiv.org/abs/2306.12156.