Bagzel: A Bazel Extension for Reproducible Dataset Builds from ROS 1 and ROS 2 Bags

Leon Pohl [1], Lukas Beer [1], George Sebastian [1], and Mirko Maehlisch [1]

[1] *Institute for Autonomous Driving, University of the Bundeswehr Munich, Germany*

9 February 2026

Summary

Robotic systems such as autonomous vehicles, mobile manipulators, and service robots continuously record large amounts of multimodal sensor data. In the Robot Operating System (ROS) ecosystem (Macenski et al., 2022; Quigley et al., 2009), these recordings are typically stored as bag files, i.e., ROS 1 rosbag files (ROS Community, 2025) and ROS 2 rosbag2 files (ROS 2 Community, 2025). These bags act as system recorders for robotic platforms, capturing time-synchronized streams from cameras, LiDAR, radar, GNSS/IMU, and internal states. While bags are ideal for debugging, replay, and sharing, they are not directly suitable for use as training datasets, which require structured and reproducible representations. **Bagzel** is an open-source software that leverages the Bazel (Bazel Team, 2025) build system to transform collections of bag files into structured, standardized, and reproducible datasets, including exports in formats such as nuScenes (Caesar et al., 2020). Within Bagzel, raw bags and derived data products (e.g., frames, trajectories, maps, labels, nuScenes-format datasets) are expressed as Bazel targets. By reusing Bazel's artifact-based, incremental, and deterministic build model, Bagzel enables scalable and maintainable data pipelines. Furthermore, it introduces the concept of large-file hashing and cluster integration, enabling efficient handling of large raw bag files and compute resources. Bagzel is publicly available at https://github.com/UniBwTAS/bagzel.

Statement of Need

Robotics research and development increasingly relies on supervised and self-supervised learning methods that consume large, diverse datasets. In ROS systems, such data is almost always first recorded as bag files. However, there is

a substantial gap between these raw bag files and the curated datasets required for training and evaluating models. In practice, teams must extract task-specific subsets, maintain preprocessing pipelines, track provenance, and rerun expensive preprocessing after changes.

Bagzel addresses this gap by treating dataset construction from bags as a build process. Bagzel is a Bazel extension consisting of a set of Starlark rules plus supporting Python and C++ libraries. This lets users declare raw bags, preprocessing steps, and derived datasets (e.g., frames, trajectories, maps, labels, and nuScenes-format exports) as Bazel targets. It then reuses Bazel's precise dependency tracking, incremental and parallel execution, and hermetic builds across local and cluster environments. This enables reproducible dataset generation by unifying raw data and code builds, supports scalable incremental retraining as new data becomes available, and integrates naturally with continuous integration and continuous deployment (CI/CD) workflows.

Bagzel targets robotics and machine learning (ML) practitioners working with ROS-based autonomy stacks in both academic and industrial settings, providing an open-source framework for scalable and reproducible dataset construction from bags.

State of the Field

General ML workflow frameworks such as TensorFlow Extended (TFX) (Baylor et al., 2017), Kubeflow Pipelines (Kubeflow Community, 2025), MLflow (Chen et al., 2020), and DVC (DVC Team, 2025) provide abstractions for defining data processing graphs, tracking experiments, and integrating with CI/CD systems. While widely adopted, these frameworks are not tightly integrated with the ROS ecosystem and do not natively leverage Bazel's build and dependency tracking model.

Within the ROS ecosystem, existing tools for extracting datasets from bag files are typically tailored to specific target formats or application scenarios. The work most closely related to Bagzel is rosbag2nuscenes (Chrosniak & Behl, 2023), which converts ROS 2 bag data into the nuScenes dataset format. Bagzel builds on this line of work by leveraging Bazel's incremental build model and explicit dependency tracking to support reproducible and scalable dataset construction.

Software Design

The full pipeline is structured into three stages: (i) training and architecture definition, (ii) dataset preparation and structuring, and (iii) training execution. Bagzel fills the gap in the dataset preparation stage. The overall concept is illustrated in Figure 1.

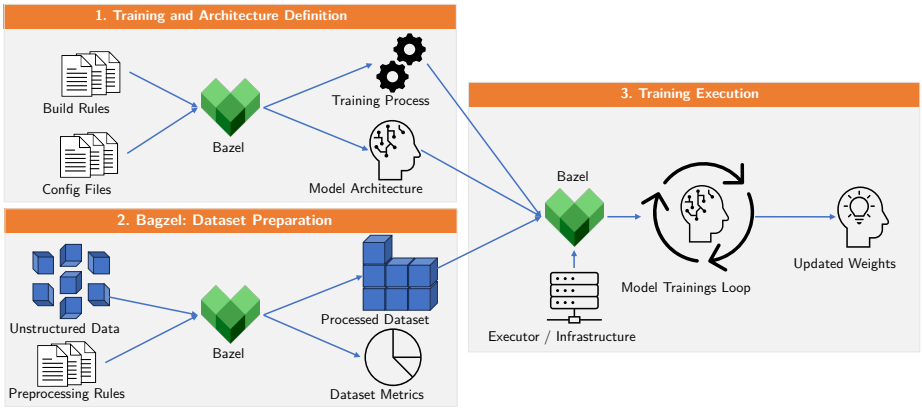


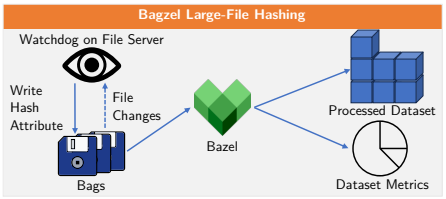
Figure 1: Concept of treating model training as a Bazel build. The workflow is structured into three stages: (i) model architecture and training configuration expressed as Bazel targets, (ii) dataset preparation from bags via Bagzel rules, and (iii) training execution as Bazel actions. In this formulation, both datasets and model artifacts become build products, so changes to any input dependency automatically trigger incremental rebuilding of only the affected downstream artifacts.

Large-File Hashing

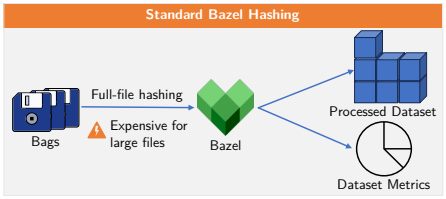
A practical challenge in Bazel is change detection for very large inputs. While Bazel normally hashes file contents byte-for-byte, robotics bag files may exceed 100 GB, making repeated hashing expensive in incremental builds. Therefore, Bagzel *optionally* supports an alternative digest mechanism for large inputs. A watchdog service records a digest for each file as an extended attribute (e.g., `user.bagzel_hash`) whenever a bag file is created or modified (Figure 2). Bazel can be configured to use this attribute as the file's digest via `--unix_digest_hash_attribute_name=user.bagzel_hash`, avoiding full file rehashing. Correctness relies on the attribute being updated on content changes. If the attribute is unavailable or missing, Bazel automatically falls back to its default hashing behavior.

Cluster Integration with Bazel

Figure 3 illustrates an integration concept between Bazel and a Slurm compute cluster for large-scale training. All inputs to a training run, including source code, configuration files, and processed datasets, are modeled as Bazel dependencies. This allows Bazel to hash inputs, reuse cached model artifacts when inputs are unchanged, and trigger new training actions when they are not. Training is implemented as a thin wrapper that submits Slurm jobs and exposes model checkpoints and metrics as Bazel outputs. Therefore, launching a cluster training run reduces to invoking a standard Bazel build command. The same wrapper



(a) Bagzel large-file hashing with extended attributes.



(b) Standard Bazel full-file hashing.

Figure 2: Comparison of hashing strategies for bag inputs in Bagzel. (a) Bagzel large-file hashing uses a file server watchdog and extended file attributes to cache content hashes and avoid rereading unchanged bag files. (b) Standard Bazel full-file hashing recomputes hashes over entire files and becomes expensive for large bag files.

can also be executed directly via Slurm during development, enabling existing training pipelines to be incrementally upgraded to benefit from Bazel's caching, reproducibility, and artifact management.

Dataset Construction with Bagzel

Bagzel supports two dataset builds: (i) extraction of vision datasets from ROS 1 bags, including image streams, annotations, and maps (Figure 4); and (ii) export of synchronized multi-sensor logs from ROS 1 and ROS 2 bags to the nuScenes data format. Building the corresponding Bazel targets produces nuScenes-format datasets, from which synchronized camera views, LiDAR overlays, and bird's-eye-view (BEV) representations can be rendered, as shown in Figure 5.

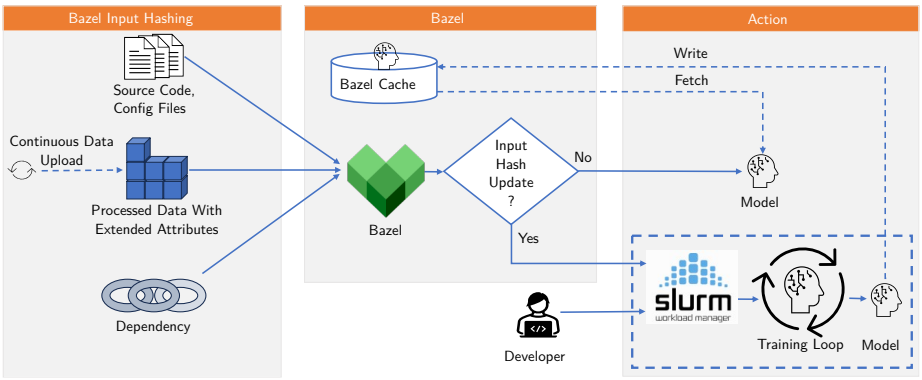


Figure 3: Conceptual Bazel–Slurm integration for training workloads. Bazel determines whether retraining is required based on hashed inputs and, when necessary, submits Slurm jobs that produce model artifacts and metrics as build outputs.

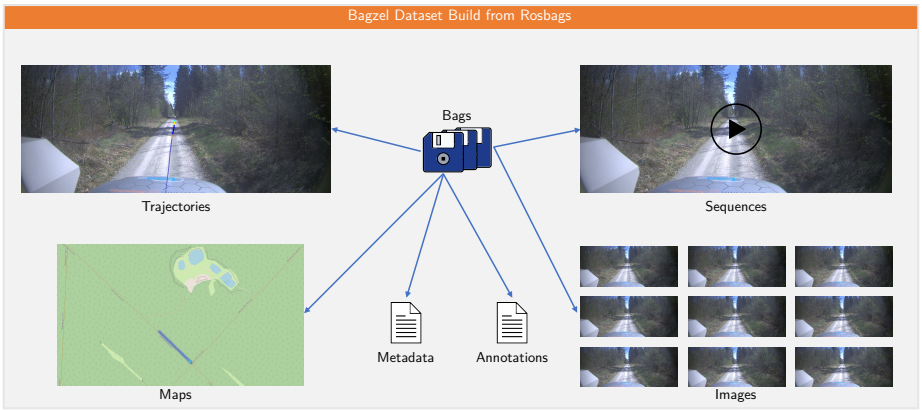
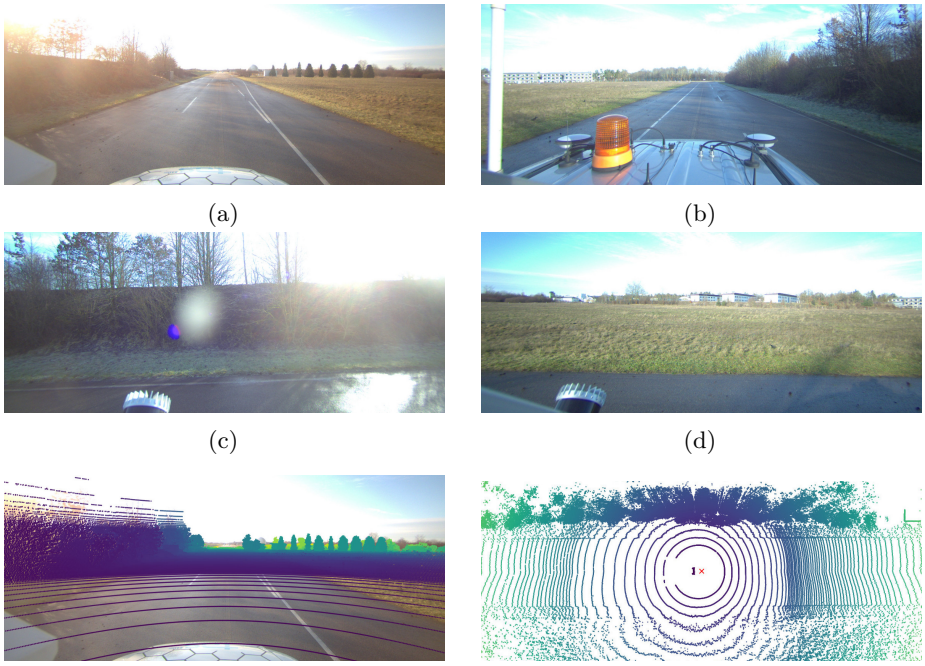


Figure 4: Visual dataset generated with Bagzel from ROS 1 bag files. Each bag is treated as a Bazel build target, and a single build invocation produces structured artifacts such as image sequences, trajectories, maps, metadata, and annotations.



(a) (b) (c) (d) (e) (f)

Figure 5: NuScenes-format dataset generated with Bagzel from bag files. The figure shows representative renderings derived from the dataset, including (a–d) front, rear, left, and right camera views; (e) LiDAR returns overlaid in image space; and (f) a bird's-eye-view (BEV) LiDAR projection.

Research Impact Statement

Bagzel is used at the Institute for Autonomous Driving, University of the Bundeswehr Munich, to generate reproducible datasets from large ROS 1 and ROS 2 bag collections for autonomous driving research. It supports ongoing perception and localization projects where frequent dataset regeneration and retraining are required. The software has been presented at BazelCon 2025 (Pohl et al., 2025a) and ROSCon DE & FR 2025 (Pohl et al., 2025b) and is released as open source with public documentation, enabling adoption by other research groups.

AI Usage Disclosure

Generative AI tools were used in a limited capacity for code assistance and language editing. All generated content was reviewed, tested, and validated by the authors, who take full responsibility for the software and the manuscript.

Acknowledgements

The authors acknowledge support from the Federal Office of Bundeswehr Equipment, Information Technology, and In-Service Support (BAAINBw). This work was also supported by dtec.bw, the Digitalization and Technology Research Center of the Bundeswehr, under project MORE. Dtec.bw is funded by the European Union through the NextGenerationEU program. We thank our colleagues at the Institute for Autonomous Driving, University of the Bundeswehr Munich, for feedback and for providing vehicles, data, and recording infrastructure. We also thank the ROS and Bazel open-source communities.

References

Baylor, D., Breck, E., Cheng, H.-T., Fiedel, N., Foo, C. Y., Haque, Z., Haykal, S., Ispir, M., Jain, V., Koc, L., Koo, C. Y., Lew, L., Mewald, C., Modi, A. N., Polyzotis, N., Ramesh, S., Roy, S., Whang, S. E., Wicke, M., … Zinkevich, M. (2017). TFX: A TensorFlow-Based Production-Scale Machine Learning Platform. *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1387–1395.

Bazel Team. (2025). *Bazel: A fast, scalable, multi-language build system*. https://bazel.build/.

Caesar, H., Bankiti, V., Lang, A. H., Vora, S., Liong, V. E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., & Beijbom, O. (2020). nuScenes: A Multimodal Dataset for Autonomous Driving. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 11621–11631.

Chen, A., Chow, A., Davidson, A., DCunha, A., Ghodsi, A., Hong, S. A., Konwinski, A., Mewald, C., Murching, S., Nykodym, T., Ogilvie, P., Parkhe, M., Singh, A., Xie, F., Zaharia, M., Zang, R., Zheng, J., & Zumar, C. (2020). Developments in MLflow: A System to Accelerate the Machine Learning Lifecycle. *Proceedings of the International Workshop on Data Management for End-to-End Machine Learning (DEEM)*.

Chrosniak, J., & Behl, M. (2023). *ROSBag2nuScenes: Share the bags, spread the joy – autonomous vehicle ROS dataset deployment*. Presentation at ROSCon 2023. https://roscon.ros.org/2023/talks/ROSBag2NuScenes_Share_the_Bags_Spread_the_Joy_-_Autonomous_Vehicle_ROS_Datasets_Deploy.pdf

DVC Team. (2025). *DVC: Data Version Control*. https://dvc.org/.

Kubeflow Community. (2025). *Kubeflow Pipelines: Portable, Scalable Machine Learning Workflows*. https://www.kubeflow.org/docs/components/pipelines/.

Macenski, S., Foote, T., Gerkey, B., Lalancette, C., & Woodall, W. (2022). Robot operating system 2: Design, architecture, and uses in the wild. *Science Robotics*, *7*(66), eabm6074. https://doi.org/10.1126/scirobotics.abm6074

Pohl, L., Beer, L., Sebastian, G., & Maehlisch, M. (2025a). *Bazel Beyond Code: Scalable AI Data Pipelines for Autonomous Systems*. Presented at BazelCon 2025. https://sched.co/2AFgF

Pohl, L., Beer, L., Sebastian, G., & Maehlisch, M. (2025b). *Processing ROSbags at Scale: Reproducible Data Workflows for Robotics*. Presented at RosCon DE & FR 2025. https://roscon.ros.org/de/2025/img/slides/S2_4___Pohl__Processing_ROSbags_at_Scale_Reproducible_Data_Workflows_for_Robotics.pdf

Quigley, M., Gerkey, B., Conley, K., Faust, J., Foote, T., Leibs, J., Berger, E., Wheeler, R., & Ng, A. Y. (2009). ROS: an open-source Robot Operating System. *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA) Workshop on Open Source Robotics*.

ROS 2 Community. (2025). *rosbag2: Record and playback ROS 2 data*. https://github.com/ros2/rosbag2.

ROS Community. (2025). *Rosbag: Record and playback ROS topics*. https://wiki.ros.org/rosbag.