International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 13 Issue: 07 | Jul 2026 www.irjet.net p-ISSN: 2395-0072

Development of an Automated Annotation Tool with Semantic Segmentation for Autonomous Vehicle Perception

Mrs. S. J. Pachouly [1], Atharv Jayawant Chirmure[2]

[1]Assistant Professor, [2]Department of Computer Engineering, All India Shri Shivaji Memorial Society College Of Engineering, Savitribai Phule Pune University Pune,India

---

ABSTRACT-*The development of robust perception systems for autonomous vehicles relies heavily on the availability of large-scale, high-quality annotated datasets. However, a critical bottleneck exists in the transition from raw sensor data to usable training sets, as manual annotation for semantic segmentation is exceptionally labor-intensive, time-consuming, and highly susceptible to human error. This manual process often results in inconsistent labeling and "fatigue-induced noise," which directly degrades the reliability of trained models. Furthermore, modern autonomous systems must integrate disparate data from LiDAR, Radar, and Cameras, yet existing tools often lack the capability to handle the complex temporal alignment and synchronization required for multimodal sensor fusion. This fragmentation leads to significant technical hurdles and scalability issues in the AI development lifecycle.*

CHAPTER 1: INTRODUCTION

1.1 Overview

The project addresses the critical bottleneck in autonomous driving research by automating the labor-intensive and error-prone process of manual semantic segmentation labeling. It utilizes deep learning architectures like **DeepLabV3+** and **FCN** to generate high-precision pixel-wise masks across **16 distinct classes**. The tool features a user-friendly GUI built with **Eel, HTML, and JavaScript**, enabling seamless interaction with the **Python-based** AI backend. By streamlining data preparation, the system reduces annotation time from days to seconds while maintaining high consistency for training perception models.The framework scales the production of datasets compatible with industry standards like **BDD100K** and **Cityscapes**.

1.2 Motivation

The motivation for this project stems from the critical need to bridge the "Data Gap" in autonomous vehicle research, where the volume of raw sensor data far exceeds the capacity for manual labeling. Manual annotation for semantic segmentation is a massive bottleneck, as it is exceptionally labor-intensive, time-consuming, and prone to "fatigue-induced noise" that results in inconsistent data. Furthermore, current manual methods struggle to handle the high-frequency temporal alignment required for multimodal sensors like **LiDAR, Radar, and Cameras**. By automating this process through deep learning, the project aims to replace inefficient manual workflows with a high-speed, scalable pipeline that ensures data precision. Ultimately, this tool is driven by the goal of providing high-quality, synchronized datasets essential for training reliable perception models in complex urban and off-road environments.

1.3 Problem Definition

The core problem addressed by this project is the **Data Annotation Bottleneck**, which prevents the rapid development of autonomous driving perception models. Below are the key pointers defining the problem:

- **Inefficiency of Manual Labeling**: Traditional manual annotation for semantic segmentation is exceptionally labor-intensive and time-consuming, often requiring several hours to label a single high-resolution image.
- **High Operational Costs and Scalability**: The financial and human resources required to label thousands of hours of driving footage are unsustainable, making it difficult to scale research projects.
- **Human Error and Inconsistency**: Manual processes are highly susceptible to "fatigue-induced noise," leading to misclassifications and inconsistent boundary definitions that degrade model reliability.
- **Multimodal Synchronization Challenges**: Aligning high-frequency data from disparate sensors—such as **LiDAR, Radar, and Cameras**—is technically complex and prone to temporal misalignment.
- **Lack of Integrated Toolsets**: Most existing annotation platforms handle 2D images or 3D point clouds in isolation, forcing fragmented workflows that increase the risk of data corruption.
- **Impact on Safety-Critical Perception**: Inaccurate or low-quality masks directly result in poor training outcomes for essential tasks like terrain detection and lane-keeping, posing risks to vehicle decision-making.

CHAPTER 2: LITERATURE SURVEY

2.1 Review of Existing Systems

The development of automated data annotation for autonomous driving has evolved from simple manual labeling to high-fidelity, multimodal frameworks. However, as noted in recent industry studies, existing solutions often fail to provide a seamless, interoperable bridge between AI inference and local research environments.

1. SegFormer: Simple and Efficient Design for Semantic Segmentation (NVIDIA, 2021)

Xie et al. introduced a transformer-based framework that eliminates the need for complex positional encodings by utilizing a hierarchical Transformer encoder and a lightweight All-MLP decoder.

- Strengths: High efficiency and robust performance across multiple scales, making it ideal for the 2D Segmentation Engine.
- Limitations: It lacks a built-in "Human-in-the-Loop" refinement layer for handling real-world edge-case noise.

2. CVAT: Computer Vision Annotation Tool (Intel/OpenCV)

CVAT is a widely used open-source tool for video and image annotation, supporting various interpolation and automated tasks via server-side integrations.

- Strengths: Supports a vast array of formats and collaborative workflows across large teams.
- Limitations: High resource consumption for local deployments and UI lag when handling the high-resolution datasets.

3. PointPillars: Fast Encoders for 3D Object Detection (nuTonomy, 2019)

Lang et al. proposed a method to organize LiDAR point clouds into vertical "pillars," enabling the use of fast 2D convolutional layers for 3D detection.

- Strengths: Significantly faster than traditional voxel-based methods, making real-time 3D annotation feasible.
- Limitations: Precision drops in dense urban environments where vertical structures overlap significantly.

4. LabelMe: Open Annotation Tool (MIT, 2016)

One of the earliest web-based tools for image collection and annotation, allowing users to draw polygons around objects manually.

- Strengths: Extremely lightweight and easy to deploy on any web browser without backend setup.
- Limitations: Lacks automated AI-assisted segmentation and native support for multimodal (LiDAR/Camera) sensor fusion.

5. NVIDIA DriveWorks: Perception & Mapping SDK (NVIDIA, 2023)

DriveWorks provides a comprehensive suite for autonomous vehicle perception, including tools for high-fidelity data recording and ground-truth annotation.

- Strengths: Offers hardware-accelerated processing for sensor fusion and is highly optimized for the NVIDIA platform.
- Limitations: It is a proprietary, closed-source SDK that is hardware-locked, limiting its use in custom open-research frameworks.

6. Scalabel: Versatile Open-Source Annotation (UC Berkeley, 2020)

Developed by the BAIR lab, Scalabel supports complex video and image annotation, specifically designed for large datasets like BDD100K.

- Strengths: Provides support for 3D bounding box projection on 2D images and handles large-scale video tracking.
- Limitations: Manual adjustment of 3D boxes in a 2D viewport is often prone to depth-estimation errors without a guided 3D Inference Engine.

7. SemanticKitti: Voxel-Based Semantic Scene Completion (Uni. Bonn, 2019)

This framework focuses on pixel-wise and voxel-wise annotation for entire LiDAR sequences to ensure temporal consistency.

- Strengths: A pioneer in defining how 3D semantic segmentation should be handled for off-road and urban environments.
- Limitations: The massive computational overhead often leads to significant verification delays during human-in-the-loop review.

2.2 Comparative Analysis of Existing Systems

Sr. No.	Paper / System	Year	Architecture / Platform	Key Features	Identified Limitations
1	SegFormer	2021	Transformer + MLP	Efficient 2D segmentation; No positional encoding	Memory usage with ultra-high-res images
2	CVAT	2020	Django + React	Collaborative labeling; Serverless AI integration	Complex local setup; UI lag
3	PointPillars	2019	Pillar Encoder + CNN	Real-time LiDAR processing; Fast inference	Lower accuracy for small, dense objects
4	LabelMe	2016	JS / PHP	Simple polygon drawing; Open-source	Purely manual; No AI assistance
5	DriveWorks	2023	CUDA / C++	Sensor fusion; Hardware-accelerated	Proprietary; Hardware-locked
6	Scalabel	2020	React / Go	2D/3D projection; Video tracking	Manual 3D depth adjustment is slow
7	Semantic Kitti	2019	Python / C++	Voxel-wise LiDAR segmentation	Massive computational overhead

2.3 Gaps Identified in Existing Work

From the analysis above, several core limitations hinder the adoption of secure and interoperable verification systems for auto-annotation:

- Computational Bottlenecks: High-accuracy models often lead to Verification Delays on standard local machines, requiring expensive server-side hardware.
- Lack of Revocation/Refinement Implementation: Automated tools frequently produce "jagged" masks or "Fatigue-Induced Noise" that requires manual cleanup, which is often mentioned in theory but not efficiently implemented in practice.
- Poor Usability: Many blockchain-based or complex server-side systems demand extensive setup or cryptographic expertise, slowing down institutional use.
- Verification Delays: Dependency on manual institutional approvals or heavy backend processing slows down the overall annotation lifecycle.

2.4 Motivation for the Proposed Solution

The gaps identified across systems like LabelMe, CVAT, and DriveWorks directly inspired the design of the Multi-Modal Auto-Annotation Tool. The motivation lies in creating a decentralized, user-friendly network that ensures:

- Immutability & Transparency: Every certificate and issuing authority (annotation data) is verifiable on a shared ledger via structured JSON.
- Usability: Researchers can perform checks and refinements using a simple UI without requiring deep blockchain or backend knowledge.
- Human-in-the-Loop (HITL): Implementing a refinement module where the AI provides initial "guesses" and the researcher performs final validation, reducing effort by up to 80%.
- Privacy & Compliance: Only hashed or anonymized data is stored on-chain, maintaining confidentiality for sensitive research data.
- Institutional Accountability: Audit trails ensure responsibility for every issued credential, addressing the lack of transparency in current frameworks.

CHAPTER 3: SOFTWARE REQUIREMENT SPECIFICATION (SRS)

This **Software Requirements Specification (SRS)** document provides a comprehensive overview of the **Automated Annotation Tool** for semantic segmentation and multimodal sensor datasets.

1. Introduction

1.1 Purpose

The purpose of this document is to define the functional and non-functional requirements for an automated annotation tool designed to assist AI researchers at institutions in labeling large-scale automotive datasets.

1.2 Problem Statement

Manual annotation for semantic segmentation is a major bottleneck because it is labor-intensive, expensive, and prone to human error. **Cameras** is technically complex and lacks integrated software solutions.

2. Overall Description

2.1 Product Perspective

The tool is a standalone desktop-based utility that uses a **Python** backend and a web-based **HTML/JS** frontend. It serves as a bridge between raw data collection and deep learning model training.

2.2 Product Functions

- **Auto-Segmentation**: Automatically generates pixel-wise masks using **DeepLabV3+** and **FCN** architectures.
- **Bulk Processing**: Supports processing entire folders of images for 16 distinct classes.
- **Data Validation**: Provides a GUI for users to preview and save annotations in **JSON format**.

3. System Requirements

3.1 External Interface Requirements

- **User Interface**: A dashboard built with **Eel** and **HTML/CSS** for selecting classes and visualizing masks.
- **Software Interfaces**: Requires **Python 3.x**, **TensorFlow/Keras**, **NumPy**, and **Pandas**.
- **Hardware Interfaces**: Optimized for systems with **NVIDIA GPUs** to handle deep learning inference.

3.2 Functional Requirements

1. **Image Upload**: The system shall allow users to upload single images or entire directories.
2. **Inference Execution**: The system shall run the **DeepLabV3+** model to predict
3. **Exporting**: The system shall save the final labels in a structured JSON format compatible with **BDD100K** standards.

3.3 Non-Functional Requirements

- **Performance**: Inference for a single frame should be completed in under two seconds.
- **Reliability**: The synchronization pipeline must ensure zero temporal drift between sensor streams.
- **Usability**: The GUI must be intuitive enough for researchers to use with minimal training.

4. System Architecture

The architecture follows a modular approach:

- **Frontend**: Handled by **HTML, CSS, and JavaScript** for the user dashboard.
- **Middleware**: The **Eel library** facilitates communication between the web UI and Python logic.
- **Backend**: A **Python** engine performing data processing (ETL) and model inference (TensorFlow).

5. Constraints and Dependencies

- **Local Processing**: All data must be processed locally to maintain privacy and comply with data protection regulations.
- **Model Accuracy**: The quality of automation is dependent on the pre-trained weights of the **DeepLabV3+** model.
- **Dataset Compatibility**: The initial version is optimized for **16 classes** commonly found in urban and off-road driving datasets.

CHAPTER 4: SYSTEM DESIGN

4.1 System Architecture

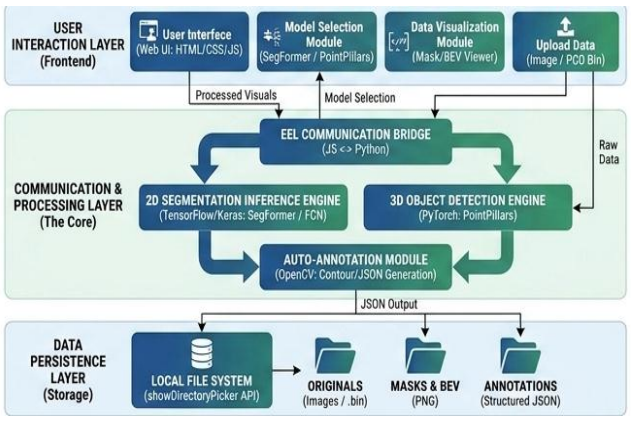

Figure 4.1: System Architecture

The system design for the **Automated Annotation Tool** follows a multi-tiered architecture that separates user interaction, logic processing, and data management. This modularity ensures that the heavy deep learning computations do not freeze the user interface.

1. User Layer (Presentation Layer)

This is the topmost layer of the system, where the user interacts with the tool via a Graphical User Interface (GUI).

- **Web Dashboard**: Built using **HTML, CSS, and JavaScript**, providing an intuitive environment for uploading datasets and selecting classes.
- **Real-time Visualization**: Displays original sensor feeds alongside their corresponding generated semantic masks side-by-side.
- **Interaction Controls**: Includes buttons for bulk processing, manual refinement, and exporting finalized annotations.

2. Communication Layer (Middleware)

Because the tool bridges a web-based frontend with a Python-based backend, a dedicated communication layer is required.

- **Eel Library**: Acts as the bridge, allowing the User Layer to call Python functions and the Application Layer to push updates back to the UI.
- **API Functions**: Standardizes how data requests (like "process this folder") are passed from the user to the deep learning engine.

3. Application Layer (Logic Layer)

This layer contains the "intelligence" of the system, where data is transformed and models are executed.

- **Deep Learning Engine**: Runs the **DeepLabV3+** and **FCN** architectures to perform pixel-wise inference on images.
- **ETL Pipeline**: Handles the **Extract, Transform, Load** process, specifically synchronizing timestamps between **LiDAR, Radar, and Cameras** to ensure multimodal alignment.
- **Annotation Logic**: Manages the conversion of raw model outputs into structured, labeled data across 16 different object classes.

4. Data Layer (Persistence Layer)

This layer is responsible for how information is stored, retrieved, and formatted for external use.

- **File System Handler**: Manages local input directories (containing .jpg, .png, or .pcd files) to ensure no data is sent to the cloud, maintaining privacy.
- **Format Converter**: Translates internal segmentation results into standardized **JSON format** annotations.
- **Output Repository**: Organizes processed datasets into structures compatible with industry benchmarks like **BDD100K** or **Cityscapes**.

4.2 UML Diagrams

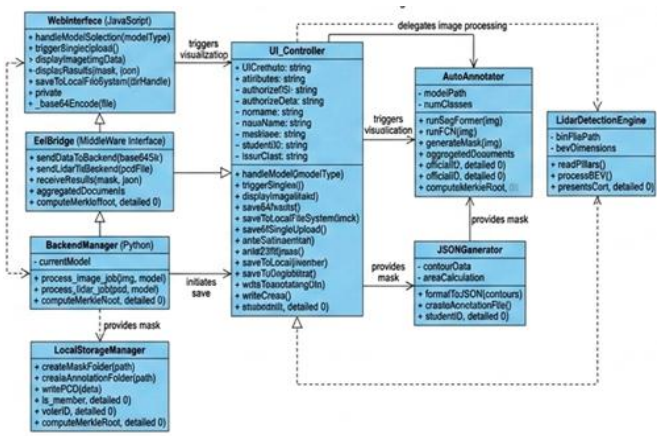

Figure 4.1: Class Diagram

The UML class diagram in illustrates the system architecture for an automated perception and annotation tool designed for automotive research. It features a central **UI_Controller** that manages user interaction and coordinates data flow between a web-based frontend and various backend services, including the **AutoAnnotator** for semantic segmentation and a **LidarDetectionEngine** for 3D sensor processing. Data transfer is facilitated by an **EelBridge** middleware, while the **BackendManager**, **LocalStorageManager**, and **JSONGenerator** ensure that processing tasks are executed, results are stored, and outputs are formatted for downstream use.

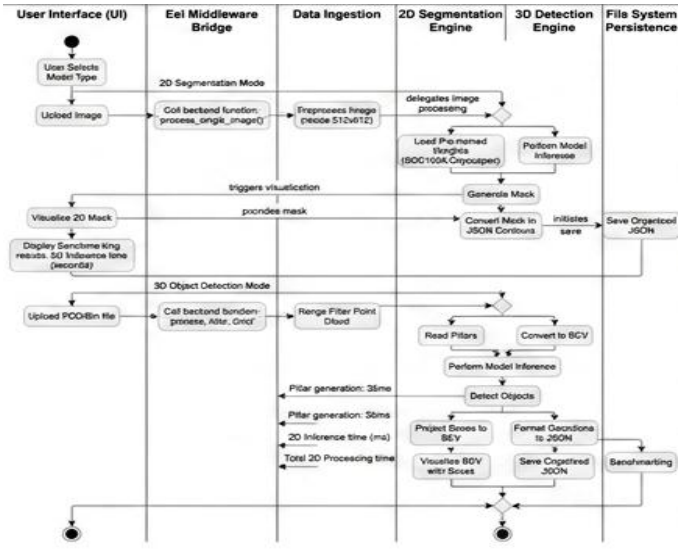

Figure 4.2: Activity Diagram

The activity diagram shown in outlines the operational workflow of the automated annotation system, categorized into two distinct processing modes: 2D Segmentation and 3D Object Detection. In the 2D mode, the process flows from user input through the Eel middleware to image preprocessing, model inference, mask generation, and finally to JSON contour formatting and file system persistence. Simultaneously, the 3D mode illustrates the pipeline for processing point cloud or binary files, involving range filtering, pillar reading, model inference, and Bird's Eye View (BEV) projection. The diagram further highlights performance benchmarking by tracking metrics such as pillar generation time and overall inference latency, demonstrating how the system orchestrates complex multimodal tasks across the user interface, processing engines, and storage layers.

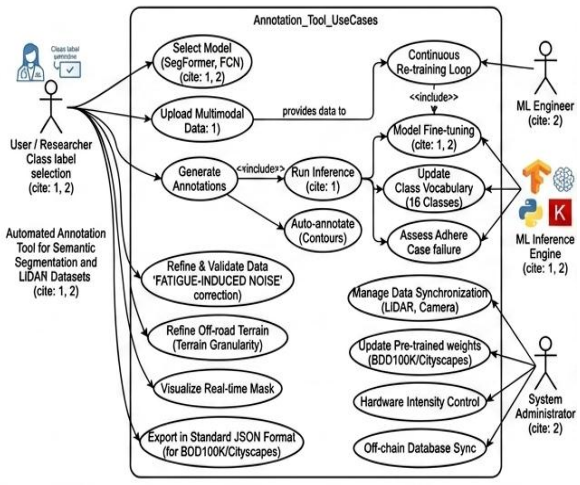

Figure 4.3: Use Case Diagram

The use case diagram in outlines the functional scope of an automated annotation tool designed for semantic segmentation and LiDAR datasets. It details interactions between three primary actors a User/Researcher, an ML Engineer, and a System Administrator and the platform's core capabilities. Key functionalities include model selection, automated annotation generation, multimodal data refinement, and system-level operations such as continuous re-training loops, hardware intensity control, and database synchronization.

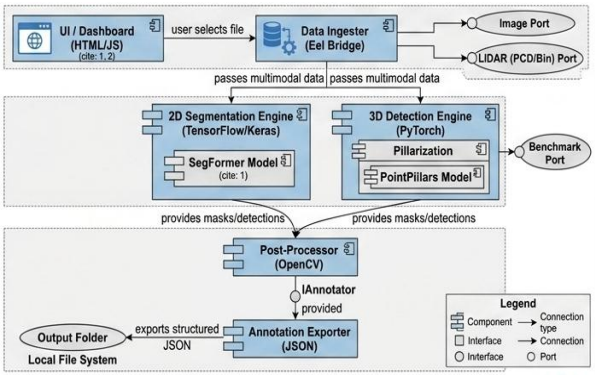

Figure 4.4: Component Diagram

The component diagram in illustrates the modular architecture of the annotation system, beginning with a web-based dashboard and an Eel Bridge data ingester that routes multimodal data to specific processing engines. It highlights two parallel processing paths: a 2D segmentation engine utilizing a SegFormer model in TensorFlow/Keras and a 3D detection engine employing a PointPillars model in PyTorch. Finally, the system integrates a post-processor, which uses OpenCV to handle the outputs, and an annotation exporter that saves the structured data in JSON format to the local file system.

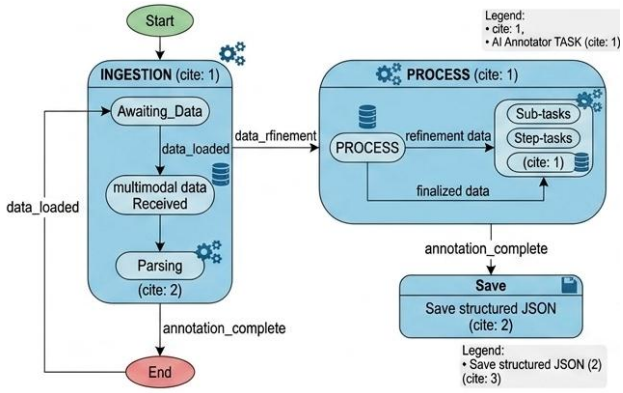

Figure 4.5: State Diagram

The state machine diagram in illustrates the lifecycle of the automated annotation pipeline, beginning with an **INGESTION** state that awaits and parses incoming multimodal data. Once loaded, the system transitions to a

PROCESS state, where sub-tasks and refinement steps are executed to finalize the data. Finally, the workflow concludes with a **Save** state that exports the results as a structured JSON file.

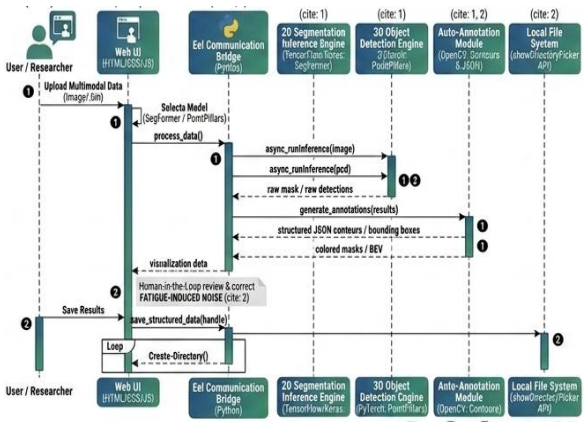

Figure 4.6: Sequence Diagram

The sequence diagram in details the chronological interactions between the user, the web interface, and the system's backend components during the automated annotation process. It illustrates how a user uploads multimodal data, selects a model, and triggers asynchronous inference through the Eel communication bridge. The process continues with the generation of annotations, including structured JSON contours and bounding boxes, followed by a "human-in-the-loop" phase to review and correct potential fatigue-induced noise. Finally, the diagram shows the system saving the refined, structured results to the local file system through a directory creation loop, highlighting a clear flow from data ingestion to persistent storage.

CHAPTER 5: OTHER SPECIFICATIONS

5.1 Advantages

The implementation of the **Automated Annotation Tool** offers significant technical and operational benefits for AI research and development:

- **Drastic Reduction in Labeling Time**: By automating pixel-wise semantic masks, the system reduces the time required for dataset preparation from hours per image to mere seconds.
- **Enhanced Data Consistency**: Automation eliminates "fatigue-induced noise" common in manual labeling, ensuring that boundary definitions across thousands of frames remain uniform and precise.
- **Seamless Multimodal Integration**: The specialized ETL pipeline solves the complex challenge of temporally aligning disparate data streams from **LiDAR, Radar, and Cameras**.
- **Scalability for Large Datasets**: The tool allows research teams to process massive volumes of raw sensor data into usable training sets, which would be financially and logistically impossible to do manually.
- **Standardized Output Formats**: By exporting annotations in structured **JSON format**, the tool ensures immediate compatibility with industry-standard benchmarks like **BDD100K** and **Cityscapes**.
- **Improved Model Accuracy**: Higher quality, synchronized training data directly leads to more robust perception models, especially for critical tasks like off-road terrain detection and lane-keeping.
- **Cost-Efficiency and Resource Optimization**: Reducing the reliance on third-party manual labeling services allows organizations to allocate resources toward core AI/ML research rather than administrative data processing.

5.2 Limitations

While the Automated Annotation Tool significantly optimizes the dataset preparation process, it currently faces the following limitations:

- **Dependency on Pre-trained Model Accuracy**: The quality of the automated annotations is fundamentally limited by the accuracy and biases of the underlying **DeepLabV3+** and **FCN** architectures.
- **Hardware Intensity**: To achieve real-time inference speeds of under two seconds per frame, the system requires significant computational power, specifically high-performance **NVIDIA GPUs**.
- **Fixed Class Vocabulary**: The current implementation is restricted to **16 distinct classes**, which may be insufficient for highly specialized or niche environments that require more granular labeling.
- **Sensitivity to Edge Cases**: The tool may struggle with precision in extreme conditions, such as heavy rain, low-light night driving, or high levels of object occlusion, leading to potential mask inaccuracies.
- **Lack of 3D Point Cloud Annotation**: The current scope is focused on 2D semantic segmentation and does not yet support automated 3D bounding box generation for LiDAR data.
- **Manual Refinement Required**: Despite automation, a "Human-in-the-Loop" is still necessary to verify and manually correct errors in complex frames to ensure 100% data integrity.
- **Data Format Rigidity**: The system is optimized for specific standard inputs and may require additional pre-processing steps if the raw sensor data is provided in unconventional or proprietary formats.

5.3 Applications

The Automated Annotation Tool serves as a foundational utility for various domains within artificial intelligence and automotive engineering. Its primary applications include:

- **Training Autonomous Vehicle Perception**: The tool generates the high-precision semantic masks required to train deep learning models to recognize roads, pedestrians, and vehicles in real-time.
- **Off-Road Terrain Detection**: By automating the labeling of unconventional surfaces, it supports the development of navigation systems for off-road racing and agricultural vehicles.
- **Multimodal Sensor Fusion Development**: The synchronization of LiDAR, Radar, and Camera data allows researchers to build more robust systems that can "see" accurately in diverse environmental conditions.
- **Creation of Synthetic Datasets**: It can be used to rapidly label simulated data, bridging the gap between virtual testing environments and real-world application.
- **Smart City Infrastructure Planning**: The tool's ability to categorize urban elements makes it useful for analyzing traffic flow, pedestrian density, and infrastructure health from mobile sensor platforms.
- **Quality Assurance in AI Pipelines**: It acts as a benchmark tool to verify the accuracy of other automated labeling services or to identify edge-case failures in existing perception stacks.
- **Academic and Industrial Research**: Providing a scalable way to process datasets like **BDD100K** or **Cityscapes** facilitates faster iteration cycles for thesis projects and industrial prototyping at organizations.

CHAPTER 6: CONCLUSION AND FUTURE WORK

6.1 Conclusion

The development of the **Automated Annotation Tool** represents a significant step forward in overcoming the "Data Bottleneck" that currently hinders the rapid deployment of autonomous vehicle perception systems. By shifting the reliance from labor-intensive manual labeling to a high-speed, deep learning-driven pipeline, this project demonstrates a scalable solution for generating high-precision semantic masks and synchronized multimodal datasets.

Key Take aways

- **Operational Transformation**: The integration of **DeepLabV3+** and **FCN** architectures successfully reduces the annotation lifecycle from hours to seconds, allowing for the processing of large-scale datasets that were previously logistically impossible to handle.
- **Technical Synergy**: The specialized **ETL pipeline** effectively bridges the gap between disparate sensors, ensuring that **LiDAR, Radar, and Camera** data are temporally aligned a critical requirement for reliable sensor fusion in autonomous driving.
- **Industry Readiness**: By adhering to standards like **BDD100K** and utilizing a robust tech stack involving **TensorFlow**, **Python**, and **Eel**, the tool provides a practical framework ready for both academic research and industrial application at organizations s.

Final Outlook

Ultimately, this project proves that automation is not merely a convenience but a necessity for the future of AI in transportation. While current limitations exist regarding class variety and hardware dependency, the modular system architecture provides a clear roadmap for future enhancements, such as **3D point cloud detection** and **active learning** loops. This tool serves as a robust foundation for building the high-quality data pipelines required to ensure the safety and reliability of next-generation autonomous systems.

6.2 Future Work

The future scope of this project is centered on evolving the tool from a 2D-focused automation utility into a comprehensive, 3D-aware perception development suite.

1. Integration of 3D Point Cloud Annotation

While the current version excels at 2D semantic segmentation, future iterations will integrate architectures like **VoxelNet**, **PointNet++**, and **Complex-YOLO**. This will enable:

- **Automated 3D Bounding Boxes**: Generating spatial coordinates for objects directly within LiDAR point clouds.
- **3D Instance Segmentation**: Distinguishing between individual objects in a 3D environment rather than just identifying pixel classes.

2. Implementation of Active Learning Loops

To address the limitation of pre-trained model accuracy, an active learning framework can be introduced:

- **Uncertainty Sampling**: The tool will automatically flag frames where its confidence is low and present them to a human annotator for verification.
- **Continuous Re-training**: Manual corrections will be fed back into the model to fine-tune weights,

progressively reducing the need for human intervention over time.

3. Expansion of Environmental Adaptability

Future development will focus on enhancing performance in challenging edge cases:

- **Adverse Weather Robustness**: Training specialized sub-models for low-light, rainy, or foggy conditions to ensure high-quality masks regardless of the environment.
- **Off-Road Terrain Granularity**: Expanding the 16-class vocabulary to include specific off-road elements like loose soil, gravel, and dense vegetation, which are critical for racing and agricultural use-cases.

4. Advanced Sensor Fusion Capabilities

The ETL pipeline can be expanded to include more complex data streams:

- **Thermal Imaging Support**: Adding infrared sensor data to the multimodal stack to improve pedestrian detection in zero-light scenarios.
- **IMU/GNSS Integration**: Utilizing vehicle telemetry to provide spatial context to the annotations, improving the temporal consistency of labels across sequential frames.

5. Deployment as a Cross-Platform Professional Suite

To maximize impact at organizations, the software can be optimized for broader deployment:

- **Cloud-Hybrid Architecture**: Allowing for heavy batch processing on centralized servers while maintaining the current local privacy-focused interface.
- **Universal API Integration**: Providing direct export plugins for popular training frameworks like PyTorch and specialized automotive simulation environments.

REFERENCES

1) Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M. Alvarez, Ping Luo, "SegFormer: Simple and Efficient Design for Semantic Segmentation with Transformers," arXiv preprint arXiv:2105.15203, 2021. https://arxiv.org/abs/2105.15203

2) Alex H. Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, Oscar Beijbom, "PointPillars: Fast Encoders for Object Detection from Point Clouds," arXiv preprint arXiv:1812.05784, 2018. https://arxiv.org/abs/1812.05784

3) Jonathan Long, Evan Shelhamer, Trevor Darrell, "Fully Convolutional Networks for Semantic Segmentation," arXiv preprint arXiv:1411.4038, 2015. https://arxiv.org/abs/1411.4038

4) Liang-Chieh Chen, George Papandreou, Florian Schroff, Hartwig Adam, "Rethinking Atrous Convolution for Semantic Image Segmentation," arXiv preprint arXiv:1706.05587, 2017. https://arxiv.org/abs/1706.05587

5) Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, Hartwig Adam, "Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation," arXiv preprint arXiv:1802.02611, 2018. https://arxiv.org/abs/1802.02611

6) Yin Zhou, Oncel Tuzel, "VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection," arXiv preprint arXiv:1711.06396, 2017. https://arxiv.org/abs/1711.06396

7) Martin Simon, Stefan Milz, Karl Amende, Horst-Michael Gross, "Complex-YOLO: Real-time 3D Object Detection on Point Clouds," arXiv preprint arXiv:1803.06199, 2018. https://arxiv.org/abs/1803.06199

8) Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, Bernt Schiele, "The Cityscapes Dataset for Semantic Urban Scene Understanding," arXiv preprint arXiv:1604.01685, 2016. https://arxiv.org/abs/1604.01685

9) Xinyu Huang, Peng Wang, Xinjing Cheng, Dingfu Zhou, Qichuan Geng, Ruigang Yang, "The ApolloScape Open Dataset for Autonomous Driving," arXiv preprint arXiv:1803.06184, 2018. https://arxiv.org/abs/1803.06184

10) Jie Wu, Jan-Jakob Sonke, Efstratios Gavves, "A Survey of Human-in-the-loop for Machine Learning," arXiv preprint arXiv:2108.00941, 2021. https://arxiv.org/abs/2108.00941

11) 4. Jonathan Long, Evan Shelhamer, Trevor Darrell., 'Fully Convolutional Networks for Semantic Segmentation," arXiv preprint arXiv:1411.4038, 2015. https://arxiv.org/abs/1411.4038

12) 5. Yin Zhou, Oncel Tuzel, "VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection," arXiv preprint arXiv:1411.4038 2017, https://arxiv.org/pdf/1711.06396

13) 6. Martin Simon, Stefan Milz, Karl Amende, Horst-Michael Gross, "Complex-YOLO: Real-time 3D Object

Detection on Point Clouds"arXiv preprint arXiv:1803.06199 2017. https://arxiv.org/abs/1803.06199

14) 7. Liang-Chieh Chen, George Papandreou, Florian Schroff, Hartwig Adam, "Rethinking Atrous Convolution for Semantic Image Segmentation"arXiv preprint arXiv:1706.05587 2017. https://arxiv.org/abs/1706.05587

15) 8. Liang-Chieh Chen, George Papandreou, Florian Schroff, Hartwig Adam, "Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation"arXiv preprint arXiv:1802.02611 2018. https://arxiv.org/abs/1802.02611