TRA235 – Data-driven Product Realization Project report

Safety System Design: Curtain Airbag Requirement Space Generation, Automatic Labeling Model for Window Shape

Group E
Mattia Carlino, Mikael Motin, Lorenzo Paravano, Yusheng Yang

Chalmers University of Technology
Gothenburg, Sweden

ABSTRACT

This project focused on automating the detection of side-window geometries from car images to generate CAD-format outputs through the implementation of a machine learning pipeline. The methodology leveraged advanced deep learning models, specifically U-Net and YOLO, which were trained on a manually labeled dataset. Model performance was evaluated using standardized and widely recognized metrics to ensure a fair comparison. Results indicated that the U-Net model outperformed YOLO, providing smoother and more precise predictions. However, the overall accuracy remained below the threshold for a fully successful implementation, highlighting areas for improvement in edge alignment and scaling precision.

Keywords: Automotive Safety Design, Image Segmentation, CRISP-DM

1. INTRODUCTION

1.1 Background

Side-impact collisions represent a significant safety challenge in automotive design, accounting for a substantial proportion of traffic-related fatalities. Research conducted by the U.S. Insurance Institute for Highway Safety (IIHS) reveals that approximately 60% of fatalities in side-impact collisions result in severe head injuries, underscoring the need for advanced protective systems.

Side curtain airbags are used to mitigate such effects by preventing occupant's head ejection and providing a barrier against external intrusions [1] [2]. According to Autoliv [3], a correct curtain deployment can reduce the risk of life-threatening head injuries in lateral impacts by approximately 50% and the risk of chest injuries by approximately 25%. The curtain airbag is typically packaged in the roof rail and trim and is not visible. It deploys vertically between the occupant's head to protect and absorb energy from the lateral impact.

Such airbags design process proves to be highly intricate [4], as each vehicle model's unique side windows geometry necessitates precise tailoring to accommodate components, deployment areas requirements and safety regulations. The curtain airbag must properly deploy inside a complete vehicle system, interacting with a high number of components (i.e. headliner, ramp brackets, trims, absorbers) and different environment conditions, a process which, may have influence on bag deployment, trim integrity and at the end, on customer protection [4].

Moreover, requirements such as the coverage area, are communicated to airbag manufacturers from car manufacturers (OEMs) during the design process of the airbag, and are influenced by model and segment specific design decisions and regulations [5], involving the positioning of car pillars, roof rail, door glass, occupant seats, and components in front of occupants, such as the dashboard and steering wheel.

This iterative and time-intensive process [6] involves extensive simulations and adjustments, struggling to meet the demands for efficiency, scalability, and precision in modern vehicle manufacturing. Addressing this challenge requires leveraging advancements in data-driven methodologies and artificial intelligence (AI) to automate and early predict surface responses.

Given the potential of such innovative solutions, world's safety supplier leader Autoliv [7] is moving towards utilizing and integrating Machine Learning (ML) and AI [8] in their product development pipelines to automate critical steps in

the design process, including side-window geometry generation and labeling. This allows for enhanced accuracy, reduced costs, and accelerated development timelines.

1.2 Gap Analysis

Manual labeling of window geometries is not only, as previously described, labor-intensive and time consuming, but also prone to potential inconsistencies, leading to inaccuracies in airbag designs. Existing methods for automating this process are limited in their scalability and adaptability across diverse vehicle models and often lack robustness in handling variations in vehicle geometry, failing to integrate seamlessly into industry-standard design workflows.

A gap in existing industry solutions is therefore identifiable in the lack of an efficient, automated model for accurately generating and labeling side-window geometries of cars allowing for reduced time and effort needed for design tasks. Bridging this gap requires not only advancements in machine learning techniques but also a comprehensive approach that aligns these techniques with practical design requirements and industry standards.

This project, offered by the automotive safety supplier Autoliv, is part of a larger pipeline that envisions a generative model that maps requirements to design and response space. The knowledge gained regarding requirement space generation will serve as a foundation for the company to approach the redesign of more complex tasks and processes, such as the prediction of airbag shapes and moving towards fully automated design solutions driven by AI and requirements-driven modeling.

1.3 Problem Formulation

As previously mentioned, the overarching problem addressed by the project is the inefficiency and inaccuracy inherent in the current methodologies used for side-window geometry generation and labeling. In particular, the project seeks to answer the following question:

1. How can object detection convolutional neural networks (CNNs) be utilized to automate the detection and labeling of side-window geometries in side-view car images?

1.4 Aim and purpose

Utilizing the GP22 car styling dataset, with over 1,400 unique car geometries, and ML techniques, the project aims at exploring two CNNs architectures, YOLO11 and U-Net, to identify an optimal model for streamlining the labelling of window positioning and automating the process of creating side-window geometries. Subsequently, it strives to deliver a trained CNN capable of detecting and labelling window geometries of side-view car images, providing the output in a Computer Aided Design (CAD) compatible format.

This automated labelling can then be utilized in CAD applications by Autoliv professionals to create precise geometries for both side windows, supporting the design of tailored airbag systems across different vehicle models. The CNN model is therefore designed to be integrated in a streamlined pipeline that minimizes manual intervention in the mapping of requirements to side-window airbag design and response space.

1.5 Delimitations

The scope of this project is limited to the identification and development of an accurate image segmentation model that incorporates the previously described features. To ensure feasibility within the constraints of the timeline and available resources, certain components have been deliberately excluded, such as a fully functional user interface. Instead, the model is capable of operating locally on a personal computer, maintaining a focus on functionality and efficiency.

Furthermore, the project focuses on automating critical processes for side-window geometries generation but does not extend to the development of a comprehensive, fully automated end-to-end pipeline for airbag design.

1.6 Significance of the Project

The proposed model aims to address the critical bottleneck in the design of automotive safety systems described above. By introducing a scalable and efficient approach to generating and labeling side-window geometries, this work is responsible for several impactful outcomes.

Firstly, the automation of manual tasks reduces the time and effort required for the generation of side-window geometries, accelerating design iterations and evaluations, and ensuring consistent and precise labeling, directly enhancing the effectiveness of side curtain airbags and contributing to overall vehicle safety.

Moreover, the outputs produced by this project are compatible with CAD formats, enabling seamless integration into existing design workflows and ensuring practical applicability in industrial contexts. The focus on AI enhanced design processes aligns with broader trends in predictive and data-driven airbag design development [9][10], advancing innovation in automotive safety system development.

2. FRAME OF REFERENCE

2.1 CRISP-DM

CRISP-DM (Cross Industry Standard Process for Data Mining) [14] is a widely recognized methodology that provides a structured approach to data mining and machine learning projects. It consists of six sequential and iterative phases: business understanding (defining project objectives and requirements), data understanding (collecting and exploring data), data preparation (create the final dataset), modeling (selecting and applying machine learning models), evaluation (assessing the model's performance), and deployment (implementing the model in a real-world environment).

2.2 Dataset

The dataset used in this project is the GP22 dataset [11], which contains a total of 1,480 side-profile images of cars. It includes both production vehicles and concept cars starting from the 1960s. The dataset provides comprehensive classifications based on car segments, brands, and vehicle categories, serving as essential criteria for identifying and evaluating vehicle types and sizes.
In addition, the dataset contains a set of labels for the design features of the cars. These labels were later used to apply various transformations to the images. The design features of the cars consist of six key components: body, bodyside, cabin, daylight opening (DLO), wheel, and tire.
All these design features are represented as labels in the form of coordinates stored within .txt files in the dataset.

2.3 Data Preparation

Data preparation [15] is a critical phase in any machine learning project, involving several steps to ensure a high-quality and consistent dataset for effective model training. The process begins with data exploration, using visualization techniques to convert raw data into meaningful insights. This is followed by data labeling, where images are manually annotated to highlight the geometry of interest, such as window areas. For this project, the LabelMe tool was used, which allows for precise annotations and saves them in JSON format. Data preprocessing addresses quality [16] and consistency by removing irrelevant or redundant information, mitigating noise, and handling outliers or missing data. Within this, data cleaning ensures the dataset's reliability by correcting errors, removing outliers, and aligning annotations. Data transformation standardizes the dataset, applying techniques like normalization and attribute construction to enhance comparability and model performance. Finally, data augmentation generates additional data by applying transformations to existing images, improving model generalization and preventing overfitting, which is crucial for handling diverse and unseen data in tasks like image segmentation.

2.4 Dataset splitting

The final step before proceeding with model development is splitting the dataset into training, validation, and test sets. It is generally considered best practice [17] to allocate 80% of the images to the training set, with 10% each assigned to the validation and test sets. This distribution ensures that there are sufficient images for training while reserving adequate data for evaluating and validating the model’s performance.

2.5 Image Segmentation

Image segmentation [18] is one of the most active research areas in computer vision, as it enables the partitioning of visual data into distinct and meaningful regions that correspond to objects or features of interest. It has found widespread applicability in various domains, including autonomous vehicles (for obstacle recognition), biomedical research (for tumor detection or organ segmentation), visual surveillance, and industrial inspection (for automated quality control). By assigning labels to individual pixels based on morphological or semantic criteria, image segmentation can be viewed as an evolution of both image classification and object detection. Segmentation tasks can

be divided into semantic segmentation, where images are divided into groups of pixels sharing a common semantic label, and instance segmentation, where the goal is to partition the image into individual object instances.
Traditional approaches to image segmentation include thresholding, edge detection, and clustering methods such as k-means clustering. More recently, techniques leveraging deep neural networks, including Convolutional Neural Networks (CNNs), encoder-decoder models, Recurrent Neural Networks (RNNs), and Long Short-Term Memory (LSTM) networks, have demonstrated significant improvements in accuracy and robustness, thus expanding the scope and reliability of segmentation-based applications. Due to relevance constraints, this discussion will focus solely on the encoder-decoder architecture of the U-Net model and the YOLOv11 model, excluding coverage of other architectures.

2.6 U-Net Model

The U-Net architecture [12] is a powerful convolutional neural network (CNN) designed for image segmentation tasks, particularly in biomedical image processing. Its distinctive U-shaped structure consists of two main components, a contracting path and an expansive path.
The contracting path captures context by progressively reducing the spatial dimensions of the input image while increasing the number of feature channels. The expansive path aims to achieve precise localization by restoring the spatial dimensions of the image. A key feature of the U-Net model is the use of skip connections, which link the encoder and decoder layers to preserve fine-grained details.

2.7 YOLOv11

YOLOv11 is a deep learning model architecture designed for real-time object detection tasks. It is the latest version of the Ultralytics YOLO series of real-time object detectors, building upon previous iterations with significant improvements in architecture and training methods. In this project, is utilize YOLOv11m-seg, a model pretrained on the COCO dataset. The YOLOv11 [13] architecture consists of three main components: the backbone, neck, and head. The backbone serves as the primary feature extractor. The neck component acts as an intermediate processing stage to aggregate and enhance feature representations. Lastly, the head functions as the prediction mechanism.

2.8 Evaluation metrics

To evaluate the performance of the models, different evaluation criteria were used. The following section describes these criteria.

Evaluation Criteria	Impact Area	Explanation
Intersection over Union (IoU) [19]	Geometry Accuracy	It measures the intersected area between the predicted region and the ground-truth region
Precision [19]	Geometry Accuracy	It is the proportion of correctly predicted target pixels
Recall [19]	Geometry Accuracy	It is the proportion of true target pixels correctly identified
F1 score [19]	Geometry Accuracy	Balance between precision and recall
Dice [19]	Geometry Accuracy	It measures the similarity between two sets of data
Mean Absolute Error (MAE) [20]	Dimension Precision	It measures the average absolute difference between the predicted values and the ground truth
Hausdorff Distance [21]	Boundary Adherence	It represents the greatest distance from a point in the predicted region to the nearest point in the ground-truth region

3. METHODS

3.1 Overview of CRISP-DM and Adaptations

The project workflow was structured around an enhanced version of the CRISP-DM methodology as proposed by Bokrantz et al [14]. This adaptation builds upon the original CRISP-DM framework by introducing an additional Operations and Maintenance phase. This phase addresses the long-term sustainability of AI solutions, mitigating risks such as data and concept drift that can degrade model performance over time.

Moreover, the integration highlights a more dynamic and iterative workflow, encouraging agility between the CRISP-DM phases rather than adhering to the original linear and sequential process. A key aspect of this enhanced methodology is the clear definition of roles (data engineer, data scientist, and domain expert) which are deemed essential for successful implementation of AI projects and proved important in the project lifecycle. To plan and structure the integration of CRISP-DM into our workflow, we created the following diagram. This helped us identify the skills required to address the tasks necessary for implementing our solution effectively.

Fig. 1: CRISP-DM Task-Skill Diagram
CRISP-DM Task-Skill Bundle.pdf

3.2 Project Organization

The project team was composed of the three core roles: data engineer, data scientist, domain expert. Additionally, we introduced the fourth role: CRISP-DM expert. Each role contributed unique expertise to the iterative development process. Since no team-member had any specific experience in the domain, the project supervisor, along with the representative from Autoliv, acted as domain experts, providing critical insights into the requirements for a successful project outcome. The data scientists led the modeling efforts, including the implementation and evaluation of used instance segmentation models, the data engineer role focused on facilitating collection of data, and finally the CRISP-DM expert, made sure the team adhered to the CRISP-DM methodology, maintaining a structured workflow.

The team established a GitHub repository to synchronize and keep track of the code throughout the project. Each team member worked on their own branch, which allowed for parallel development without conflicts and new features were continuously pushed to the main branch.

3.3 Business Understanding

Due to lacking the domain expertise role, at the end of each week, the team held progress meetings along with the supervisor and Autoliv representative, during which results from the iterative CRISP-DM process were reviewed and tasks for the upcoming week were defined. Regular feedback from the supervisor, coupled with input from Autoliv, was crucial in refining the project's objectives and methodology. This iterative approach encouraged constant feedback and adjustments across all project phases, thus establishing a dynamic workflow that responded to evolving challenges.

3.4 Data Understanding

The project initially began with an exploration of the GP22 dataset [11] provided by Autoliv. During this phase, the team assessed data quality, identified potential gaps, and determined the need for preprocessing. In this context, we

visualized the design feature labels associated with the images to assess their accuracy and consistency. Although we were working with an existing dataset, our first step was to ensure that every image in the dataset had an associated label. Once this was verified, we created a simple Python script to draw bounding boxes on the images based on the labels provided in the dataset's .txt files. By displaying the original images alongside their corresponding annotated masks, we ensured that the annotations were correctly aligned after transformations. Data visualization not only served to validate the integrity of the dataset but also provided useful insights into the dataset's structure, enabling us to better comprehend the available data and to select the right data preprocessing steps.

While the dataset was clean and well-labeled, the images of the cars were taken from different distances, resulting in some cars being larger than others, even if cars were comparatively smaller in reality. Additionally, cars varied significantly in orientation.

3.5 Data Preparation

Preparing high-quality training data was a critical step in the project. The GP22 dataset was already labeled; however, these labels could not be used for our primary objective. For this reason, we proceeded with annotating the images. This process involved labeling the side-view car windows in the images using the LabelMe tool, as recommended by the supervisor and Autoliv. To ensure consistency and accuracy, the team established a detailed annotation guideline. The guideline specified naming conventions for labels, the appropriate components within the labeling tool to use, general instructions for drawing labels, and strategies for handling edge cases, such as labeling convertibles where intuitive decisions might pose challenges.

Initially, a random selection of images was made and each team member labeled 50 images using polygons that outlined the windows, resulting in a dataset of 200 labeled images. However, during weekly review meetings with the supervisor and the company representative, it became apparent that the initial annotations did not meet the project domain criteria. The lack of domain-specific knowledge within the team had led to inconsistencies and inaccuracies in the labeling process. In response to the feedback, the annotation guidelines were refined to incorporate a deeper understanding of the domain. Key updates included clearer instructions emphasizing that lines should only outline the fragile parts of the windows (i.e. the glass), ensuring that windows were properly separated by their supporting frames, differentiating window style choices from actuality, and requiring annotations to be as detailed as possible. These improvements, informed by domain expertise, significantly enhanced the quality of the training data.

3.6 Data Preprocessing

Following the discovery of key insights during the data understanding phase, we implemented a series of preprocessing steps to optimize the data for effective training. These initial steps included data cleaning, transformation, and augmentation. As we iteratively refined our approach, we continued to introduce new preprocessing steps in response to revisits to the business understanding, modeling, and evaluation phases.

During data cleaning, the dataset was inspected for inconsistencies, missing annotations, and potential outliers, such as images of convertible cars. Although some images of convertibles were identified, we decided to retain them in the dataset.

Data transformations were applied to standardize the dataset and focus the model on relevant features. One key transformation involved flipping all images horizontally to ensure uniform orientation. To automate this, a ResNet18-based classifier was trained on a manually labeled subset to identify the direction each vehicle faced (left or right). Images predicted to face right were flipped horizontally, aligning all vehicles to face left, which was the chosen standard for the project. This ensured consistency across the dataset, which was crucial for effective training.

Another transformation addressed background noise, which was identified during the modeling phase as a significant factor reducing recall. Using the predefined feature labels in the GP22 dataset, the images were cropped to include only the relevant parts of the car. This removed unnecessary background elements, centered the vehicles in the frame, and reduced distractions for the model.

During a review with the supervisor, it was identified that the varying sizes of vehicles in the dataset posed a significant issue. For example, smaller cars occasionally appeared larger than SUVs due to differences in image scales. To address this, a scaling step was introduced to standardize the relative sizes of the vehicles. Leveraging domain knowledge from the supervisor, the car's front-wheel rim was selected as a reference point for scaling, as rims were consistent in size and already annotated as a class feature in the GP22 dataset.

Using these annotations, we automated the process of identifying the front-wheel rim. Since all cars in the dataset were oriented in the same direction after preprocessing, the front-wheel rim was determined as the bounding box with the smallest x-coordinate. A scaling factor was then calculated based on the area of the front-wheel rim bounding box, comparing it to a reference car.

$$Scaling\ factor\ = \sqrt{\frac{rim\ area\ (reference\ car)}{rim\ area\ (current\ car)}} \quad (1)$$

A script was developed to apply this scaling factor to the dimensions of each image, using an arbitrary reference car as the baseline. The height and width of each image were multiplied by the calculated scaling factor, ensuring proportional adjustments. Finally, all images were cropped to the original size of 1024x1024 pixels, preserving uniformed image sizes across the dataset while maintaining the relative proportions of the vehicles.

Further, different data augmentation techniques were applied to increase the size of the dataset, such as horizontal flipping and random brightness and contrast adjustments. These transformations help the model become more robust to lighting conditions, perspectives, and small positional variations in the input images.

A custom augmentation pipeline was implemented using Albumentations [22], which applied several transformations to both the images and their corresponding masks, where the masks represent the annotated areas in binary format, with white pixels indicating the object of interest and black pixels representing the background.

The augmentations included random brightness and contrast adjustments, Gaussian blur to simulate image noise and enhance the model's robustness, and hue-saturation-value (HSV) shifts to simulate varying lighting conditions and improve the model's ability to generalize. Additionally, a resizing step ensured that all augmented images and masks were resized to 1024x1024 pixels to maintain uniformity for the model input. This augmentation approach aimed to create a more varied dataset, helping the model perform better.

3.7 Modelling

The modeling phase focused on developing instance segmentation models capable of accurately identifying side-view car windows. This phase involved selecting appropriate architectures, training the models, and iteratively refining them based on evaluated results.

During the first iteration of the project, the U-Net model was selected due to its effectiveness in pixel-level segmentation tasks, which suited our use case segmenting detailed areas of car windows. Due to limited data, a ResNet-34 backbone, pretrained on ImageNet, was used as the encoder to leverage transfer learning, reducing the training time and providing more accurate results. After preprocessing the images and converting the dataset labels into a U-Net format using custom scripts, we divided the dataset into train, test and validation sets. The U-Net model was subsequently trained for 10 epochs on the initial set of 200 labeled entries. The results were challenging and prompted a revisit to the earlier phases where new preprocessing methods were implemented.

In parallel, we worked on implementing a second model to compare with the U-Net model. For this, the YOLOv11 model was selected due to its object detection capabilities, its ability to perform effectively with smaller datasets, and its efficiency in segmentation tasks at a lower computational cost. It was a more sustainable choice compared to U-Net, with significantly faster training times. This advantage was important, as it allowed us to evaluate the impact of different preprocessing steps at a faster pace. Training the YOLO model followed a similar approach to training the U-Net model. The preprocessed dataset was divided into train, test, and validation sets and the labels were converted to YOLO format using an existing library called 'labelme2yolo' developed by Wang Xin under the MIT license [23].

Initial training for both models revealed performance discrepancies, particularly with recall rates when evaluating, prompting a revisit to the data preparation phase. At first glance, the YOLO model was better at identifying car windows but was lacking detail in its segmentation. Key adjustments included further refining the data transformation pipeline, addressing issues like inconsistent vehicle sizes and background noise. Additionally, more images were labeled accumulating to 800 in total.

3.8 Evaluation

Evaluating the models was an important part in verifying model improvements. The models were evaluated both visually and using evaluation metrics. Visually, we looked for detailed segmentation and high-definition masks. The goal was to assess how accurately the models could delineate the boundaries of car windows, ensuring that the segmentation masks were not only close to the true boundaries but also captured the finer details of the window contours. This visual inspection was crucial in identifying any potential misclassifications, particularly in regions with reflections, background noise or when front and back windows were not clearly separated by a supportive frame.

In addition to visual inspection, we used quantitative evaluation metrics to objectively measure the performance of both models. Using Albumentations, we prepared a test set that underwent transformations, including horizontal flipping, random brightness/contrast adjustments to mimic how the models would perform on realistic images. We computed

the following metrics across a test dataset: IoU scores, Dice scores, Precision scores, Recall scores, F1 scores, Hausdorff distances, and MAE scores. The metrics were calculated for each test image in the dataset by comparing the predicted mask with the ground truth mask.

To identify the best and worst predictions, we computed the IoU score for each test image. The best prediction was selected as the image with the highest IoU, and the worst prediction was the image with the lowest IoU. These images were displayed alongside their corresponding ground truth and predicted masks which gave us a better understanding of outliers in the dataset.

3.9 Deployment

After multiple iterations of going back and forth between the business understanding, data preparation, modelling and evaluation phases, we arrived at the deployment phase. The deployment phase involved preparing the solutions developed during the modeling and evaluation phases for practical application. For this project, the goal was to deliver pre-trained models in a format that could be utilized effectively by stakeholders. To achieve this, a GitHub repository was chosen as the primary delivery platform. This repository was structured to include well-documented Jupyter notebooks detailing the complete implementation of the developed models, including scripts for data preprocessing, training, and evaluation.

Additionally, as the models were designed to generate masks for further use, a script was implemented to convert these outputs into a CAD-compatible format (.DXF). This approach was selected to ensure compatibility with existing workflows in the target domain.

3.10 Operation and Maintenance

The project was conducted within the constraints of a single academic term, which limited the scope of operational and maintenance activities. While the primary objective was to develop and deliver instance segmentation models for side-view car windows, the short project duration precluded extensive testing and deployment in a live production environment. As a result, certain aspects, such as long-term performance monitoring and scalability, were identified as areas requiring future attention.

4. RESULTS

4.1 Preprocessing Steps

800 images were labeled in total and split into train, test and validation sets with a ratio of 0.7, 0.2, and 0.1. Final preprocessing steps included: removing the background, flipping and scaling the cars, and finally, performing various augmentation techniques as demonstrated below.

Fig. 2: Original Image

Fig. 3: Cropped Image

Fig 4: Flipped Image

Fig 5: Augmented Image (Brightness)

Below is an example of the scaling process. In the original images, the car on the left appears noticeably larger than the one on the right, even though they are relatively the same size in reality. After scaling, the masks appear to have relatively the same size, reflecting the corrected proportions.

Fig. 6: Original Car Image Without Scalling

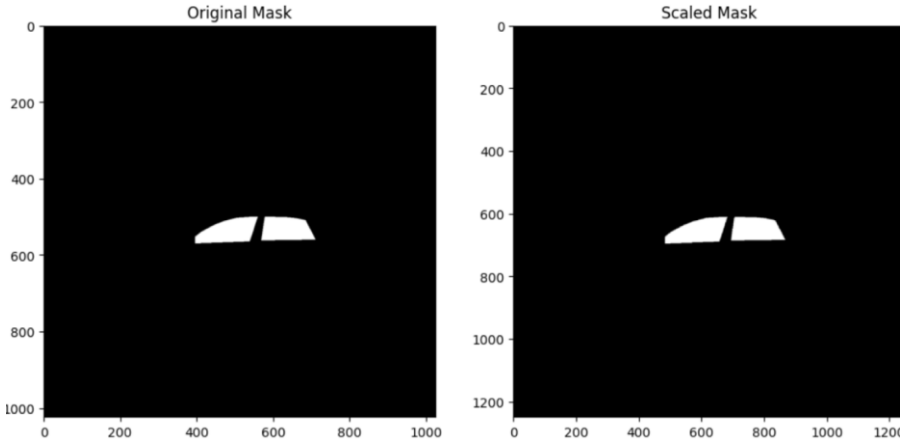

Fig. 7: Car mask after scalling

4.2 Predictions

Both models were trained on the same preprocessed dataset. The U-Net model was trained for 10 epochs, while the YOLO model was trained for 50 epochs. Both models showed significant reductions in training loss, indicating that the models were effectively generalizing to the data. The results demonstrate the models' ability to accurately recognize the windows and predict their geometries.

Below (Fig. 8 and Fig. 9) is an example of the visualization of predictions made by the final models. As observed, the edges of the YOLO model's prediction appear more fragmented and less refined, while the U-Net model's prediction is smoother and better aligned with the edges. Both models demonstrate the capability to detect basic window frames, although with minor deviations.

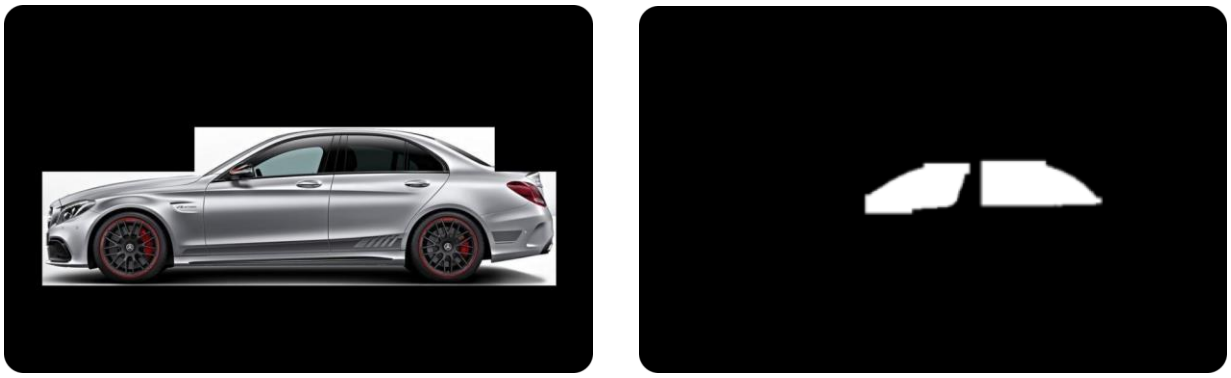

Fig. 8: Prediction of a car using YOLO

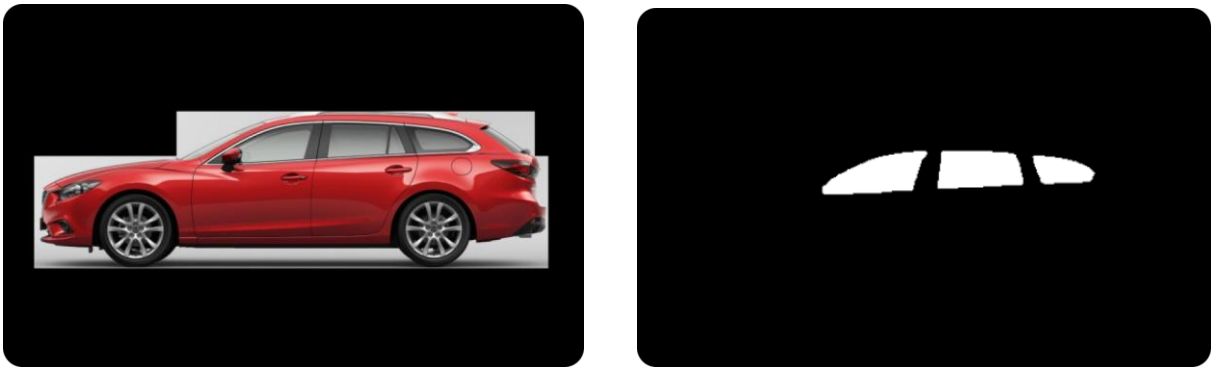

Fig. 9: Prediction of a car using U-net

Since the dataset is diverse and well-represented, including various types of cars such as home cars, luxury cars, and trucks, as well as a wide range of backgrounds, the models are expected to generalize well to new, unseen images, providing a side car is clearly visible in the image. The team took the initiative to test the model by capturing several side-view car images and running them through the pipeline.

Fig. 10: YOLO Model Prediction on Random Image

4.3 Delivery for the Supervisors

The final, polished repository will be delivered to the supervisors, where they will continue working on and maintaining the project. Notably, both output of the final model is mask, which is a binary image/array, that highlights the region of interest (ROI) predicted by the model. Further, the masks were converted into a CAD compatible format (.DXF) (Fig.11).

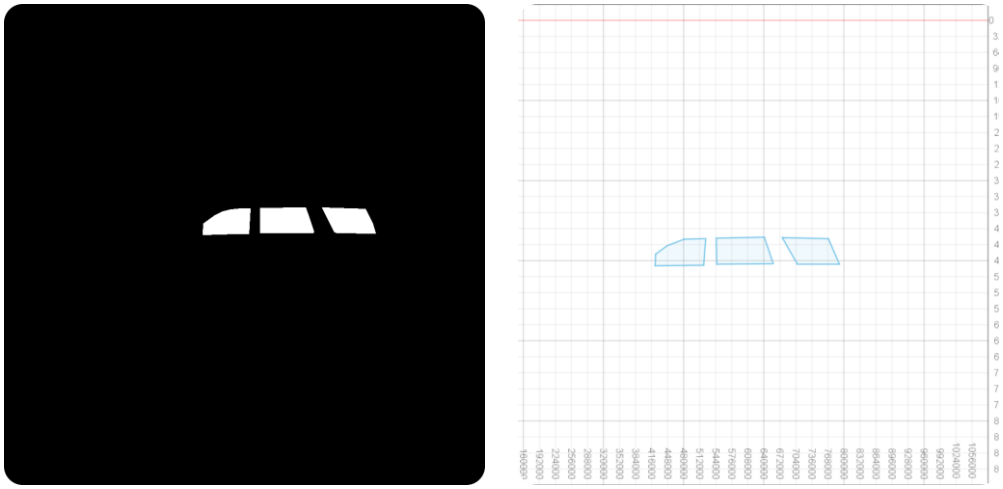

Fig. 11: Predicted mask to CAD format

4.4 Model Performance and Comparison

The same evaluation metrics were implemented to ensure a fair and consistent comparison of model performance. Each model was tested on the same set of 100 images, and different standards are calculated to evaluate the model performance. The results are as follows (Fig.12):

Evaluation Criteria	Explanation	U-NET	YOLO
Intersection over Union (IoU)	Overlap between predicted and ground truth regions	**0.74**	0.73
Dice Coefficient	Measures the similarity between two sets of data	**0.84**	0.82
Precision	Proportion of correctly predicted target pixels.	**0.96**	0.88
Recall	Proportion of true target pixels correctly identified	0.75	**0.78**
F1 Score	Balance between precision and recall	**0.84**	0.82
Mean Absolute Error (MAE)	Measure of how far off the predictions are from the ground truth	**0.0062**	0.0065

Hausdorff Distance	Largest boundary distance between prediction and ground truth	**28.2**	40.02

Fig. 12: Evaluation Metrics

5. DISCUSSION

5.1 Comments on Preprocessing

The scaling process, while functional, was not entirely accurate when overlaying the images, as the bounding boxes for the wheels were not perfectly aligned around the wheels. This misalignment is further exaggerated when scaling the images based on these bounding boxes. At present, we do not have a quantitative method to estimate the margin of error. Instead, the errors are observed visually. Based on these observations, the margin appears to be within 10% of the image width, which, while not ideal, is not entirely unacceptable. Future work could benefit from implementing a more robust preprocessing pipeline, possibly incorporating automated methods to refine bounding box accuracy and minimize scaling errors. These improvements would help ensure that the preprocessing stage does not inadvertently affect the integrity of downstream results

5.2 U-Net vs YOLO

As shown in Fig. 12, the U-Net model demonstrated slightly better performance compared to the YOLO model, which aligns with earlier observations of the predictions. This outcome is expected, as the U-Net model is more complex and resource-intensive than the YOLO model. Visually however, the U-Net model outperformed the YOLO model as the segmentation is more detailed, capturing finer details. Although the YOLO model aided the iterative development, these differences underscore the importance of selecting models based on project-specific requirements.

5.3 Model Performance

From the result section, we can see that the model performance is not yet optimal and fell short of industry standards, as a successful model would typically reach accuracy levels exceeding 90%. Despite this both models demonstrated a solid performance in recognizing general window shapes. However, their ability to detect edges of windows proved challenging, especially in complex scenarios. For example, primary challenges pointed to reflections on the windows, as this would create misleading spots, or in cases where the car windows were black, it became difficult to distinguish them from a black background or car body. These subtle and diverse variations are highly dynamic and manifest differently across images, making it challenging for the model to capture and learn these features effectively, as with the following example.

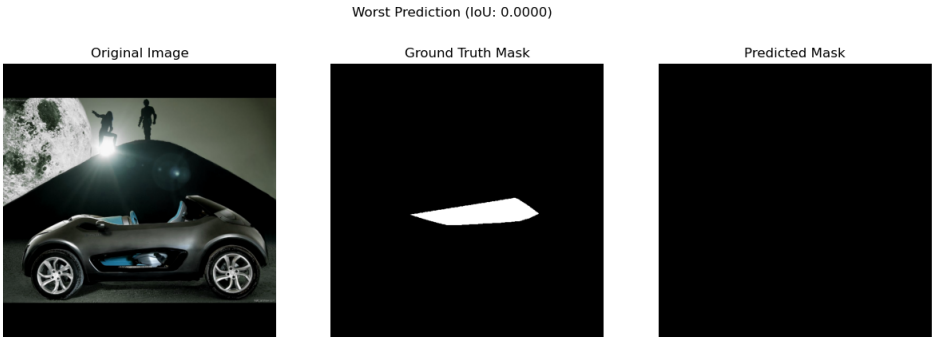

Fig. 13: Example of a Complete Failure of Detection

Another challenge was the requirement to be able to detect the geometry of the front and rear windows even when parts of the window were obstructed or shielded by a rearview mirror. In such cases, it was expected that the models could infer the actual window regardless of obstructions. During the labelling process, windows were manually labeled as passing through obstructions like mirrors, emphasizing the fragile glass parts as in the example below. However, the models struggled to learn these inferred features as there are no obvious features to learn from these obstructions.

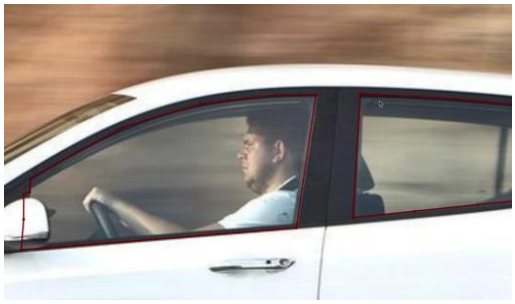

Fig. 14: Example of Labelled Rear Window with Rearview Mirror

Our initial approach in addressing these challenges was to increase the size of the training dataset. However, much to our surprise, increasing the number of training images did not result in significant performance improvements. It turned out that the models quickly learned the basic features of a window and demonstrated an ability to recognize windows during early training with a small dataset. As a reflection, removing non-representative outliers, such as Fig. 13, could improve learning outcomes by reducing noise in the dataset.

5.4 Implications for Industrial Approach

Despite the limitations, the models achieved a foundational level of performance that demonstrates the feasibility of automated side-window geometry detection. This project provides a proof-of-concept for integrating machine learning into automotive safety design workflows, potentially reducing manual labor and accelerating design iterations.

However, for industrial adoption, further refinements are necessary. These include improving edge alignment precision, working with correct scaling of the vehicles, and enhancing generalization across diverse vehicle types. Collaboration with domain experts to refine training datasets and incorporate domain-specific knowledge will be critical to achieving these goals as a future process.

6. CONCLUSIONS

The project demonstrated how object detection CNNs, specifically U-Net and YOLOv11, can be utilized to automate the detection and labeling of side-window geometries in side-view car images. Both models achieved foundational levels of performance, with U-Net outperforming YOLO in terms of segmentation accuracy and edge precision, making it more suitable for the project's requirements.

The final pipeline delivered to AutoLiv stakeholders is a valuable starting point for engineers; however, further improvements are needed for it to function as a stand-alone tool and effectively replace the manual process of designing window geometries. A critical area for future development is the integration of actual vehicle measurements. This could be achieved by implementing a model that can identify the type of car in the input image. By training this model on a dataset that includes car model names, the system could automatically classify vehicles. Next, it would be necessary to retrieve the corresponding real-world measurements. This could be done by querying a data source containing this information or, ideally, using a dataset that already includes the relevant dimensions. These improvements would significantly improve the accuracy and practicality of the pipeline, bringing it closer to full automation. However, if Autoliv wants to experiment with GAN solutions in their pipeline, one would need to further build algorithms based on real-world data that could generate realistic car window measurements and shapes.

With the short project time limit, the operation and maintenance phase of the enhanced CRISP-DM methodology was not explored. Although the result proved limiting in a direct replacement of the manual process. Future work should place an eye on this phase to ensure long-term sustainability, address issues like data drift, and refine the solution for industrial-scale deployment. Suggestions for this include establishing a monitoring framework to track the models' performance over time, where periodically retraining the models with updated datasets that include new variations will prevent the model from becoming biased. Further, a feedback loop where misclassified or poorly segmented examples from production are flagged for review and incorporated into future training sets. These advancements would address current limitations and make CNN-based solutions a viable, standalone tool for automating side-window geometry detection and labeling in industrial applications.

7. REFERENCES

[1] Evans, N. and Leigh, M., "FMVSS 226 Ejection Mitigation: A Review," SAE Technical Paper 2013-01-0469, 2013, https://doi.org/10.4271/2013-01-0469.

[2] Kaufman R, Fraade-Blanar L, Lipira A, Friedrich J, Bulger E (2017) Severe soft tissue injuries of the upper extremity in motor vehicle crashes involving partial ejection: the protective role of side curtain airbags. 144-152. https://doi.org/10.1016/j.aap.2017.02.027

[3] Autoliv (2023) Annual and Sustainability Report 2023. Retrieved on:10/01/2025 at https://www.autoliv.com/sites/autoliv/files/2024-02/Autoliv_Annual_Sustainability_Report_10K_2023.pdf

[4] Guerra, Claudio & Leomanni, Nicola. (2019). Curtain Airbag design for Robust Deployment. International Journal of Research in Advent Technology. 7. 4-9. https://doi.org/10.32622/ijrat.77201903

[5] Braver, E. R., Kyrychenko, S. Y., & Ferguson, S. A. (2005). Driver Mortality in Frontal Crashes: Comparison of Newer and Older Airbag Designs. Traffic Injury Prevention, 6(1), 24–30. https://doi.org/10.1080/15389580590903140

[6] Arjomandi Rad, M., Cenanovic, M., Salomonsson, K., (2023). Image regression-based digital qualification for simulation-driven design processes, case study on curtain airbag. Journal of Engineering Design 34, 1–22. https://doi.org/10.1080/09544828.2022.2164440

[7] Banker, S. (2022, October 7). *Autoliv's supply chain risk management journey*. Forbes. Retrieved January 17, 2025, from https://www.forbes.com/sites/stevebanker/2022/10/07/autolivs-supply-chain-risk-management-journey/

[8] Arjomandi Rad, Mohammad & Cenanovic, Mirza & Salomonsson, Kent. (2023). Image regression-based digital qualification for simulation-driven design processes, case study on curtain airbag. Journal of Engineering Design. 34. 1-22. 10.1080/09544828.2022.2164440.

[9] Franco, B., Alves Ribeiro, J. M., & Sánchez-Arce, I. d. J. (2023). Development of an Airbag Geometry Specific for Autonomous Vehicles. *Eng*, *4*(4), 2553-2570. https://doi.org/10.3390/eng4040146

[10] Valenzuela Del Río JE, Lancashire R, Chatrath K, Ritmeijer P, Arvanitis E, Mirabella L. Machine-Learning-Accelerated Simulations for the Design of Airbag Constrained by Obstacles at Rest. Stapp Car Crash J. 2024 Jun;67:1-13. https://doi.org/10.4271/2023-22-0001

[11] Gyunpyo Lee and Taesu Kim and Hyeon-Jeong Suk (2022). GP22: A Car Styling Dataset for Automotive Designers. *arXiv*, **2207.01760.** https://arxiv.org/abs/2207.01760

[12] Olaf Ronneberger and Philipp Fischer and Thomas Brox (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. *arXiv*, **1505.04597**. https://arxiv.org/abs/1505.04597

[13] Rahima Khanam and Muhammad Hussain (2024). YOLOv11: An Overview of the Key Architectural Enhancements. *arXiv*, **2410.17725**. https://arxiv.org/abs/2410.17725

[14] Bokrantz, J., Subramaniyan, M., & Skoogh, A. (2023). Realising the promises of artificial intelligence in manufacturing by enhancing CRISP-DM. *Production Planning & Control, 35*(16), **2234–2254**. https://doi.org/10.1080/09537287.2023.2234882

[15] Han, J., Kamber, M., & Pei, J. (2011). *Data mining concepts and techniques third edition. The Morgan Kaufmann Series in Data Management Systems Chapter 3: Data preprocessing***, 83-123.**

[16] Cai , L., & Zhu, Y. (2015). The challenges of data quality and data quality assessment in the big data era. *Data science journal*, 14, **2-2**. https://doi.org/10.5334/dsj-2015-002

[17] V. R. Joseph (2022), Optimal ratio for data splitting. *Statistical Analysis and Data Mining: An ASA Data Science Journal*, **531–538**. http://dx.doi.org/10.1002/sam.11583

[18] Minaee, Shervin and Boykov, Yuri and Porikli, Fatih and Plaza, Antonio and Kehtarnavaz, Nasser and Terzopoulos, Demetri (2022), Image Segmentation Using Deep Learning: A Survey. *arXiv*, **3523-3542**. https://arxiv.org/abs/2001.05566

[19] Wang, Z., Wang, E. & Zhu, Y (2020). Image segmentation evaluation: a survey of methods. *Artif Intell Rev 53*, **5637–5674**. https://doi.org/10.1007/s10462-020-09830-9

[20] Syed Zaini, Syed Zakwan and Sofia, Nur Najihah and Marzuki, Mohd and Abdullah, Mohd Firdaus and Ahmad, Khairul Azman and Isa, Iza Sazanita and Sulaiman, Siti Noraini (2019). Image Quality Assessment for Image Segmentation Algorithms: Qualitative and Quantitative Analyses, *2019 9th IEEE International Conference on Control System, Computing and Engineering (ICCSCE)*, 66-71. http://dx.doi.org/10.1109/ICCSCE47578.2019.9068561

[21] Karimi, Davood and Salcudean, Septimiu E (2020), Reducing the Hausdorff Distance in Medical Image Segmentation With Convolutional Neural Networks. *IEEE Transactions on Medical Imaging 39*, **499-513**. https://arxiv.org/abs/1904.10030

[22] Buslaev, Alexander and Iglovikov, Vladimir I. and Khvedchenya, Eugene and Parinov, Alex and Druzhinin, Mikhail and Kalinin, Alexandr A. (2020), Albumentations: Fast and Flexible Image Augmentations. *Information 11*, 125. https://www.mdpi.com/2078-2489/11/2/125

[23] Wang Xin (2024), Labelme2YOLO https://pypi.org/project/labelme2yolo/