# Project Documentation: Vehicle Damage Detection and Carry Forward Algorithm DAMAGEiD

Gouthamsimha Koppula

January 2024

## 1 Introduction

This document outlines the key aspects of the project undertaken during my time at DAMAGEiD, where we focused on developing our capabilities in car damage assessment. The project primarily involved improving damage detection and implementing a novel Carry Forward Algorithm.

### 1.1 Objectives

**Damage Detection:** Leveraging YOLOv8, a leading object detection model, to identify and classify damages in car images with precision.

**Carry Forward Algorithm:** Introducing an innovative approach through LOFTR Matcher and tomography matrix to intelligently track pre-existing damages, providing a holistic view of a vehicle's condition.

This documentation serves as a comprehensive record of the methodologies applied, the technologies integrated, and the results achieved during the process.

## 2 Technologies Used

### 2.1 YOLOv8 Model for Damage Detection

The YOLOv8 (You Only Look Once) model is employed for efficient real-time object detection, specifically for identifying damages in car images. It divides the image into a grid, predicting bounding boxes and class probabilities for each grid cell, offering both speed and accuracy.

### 2.2 LOFTR Matcher for Carry Forward Algorithm

The LOFTR (LOcal Feature-based TRacking) Matcher is crucial for tracking pre-existing damages across images. Leveraging its ability to match local features like keypoints, LOFTR establishes correspondences between images. It

serves as a key component in the carry forward algorithm, ensuring the continuity of damage information.

### 2.3 Homography Matrix

The Homography Matrix is fundamental in computer vision, used here to map points between images. Paired with the LOFTR Matcher, it estimates a transformation that tracks damages across different viewpoints. The Homography Matrix acts as a bridge, facilitating the mapping of corresponding points and ensuring seamless tracking of damages.

## 3 Data Processing

Data processing plays a pivotal role in the success of the car damage detection and carry forward algorithm. For damage detection, a meticulously annotated dataset was curated using the MakeSense tool. The process involved manually marking various types of damages on car images, categorizing them into distinct classes:

1.Paint Scratch
2.Clear Cut Scratch
3.Broken
4.Dent
5.Dink

I used makesense tool to manually annotate the dataset. This annotated dataset served as the foundation for training the YOLOv8 model, providing it with a diverse range of examples to learn from.

In the context of the carry forward algorithm, images were acquired during a car ride, both in and out the vehicle. Using the MakeSense tool again, damages were marked on these images to establish a reference for tracking pre-existing damages. This comprehensive dataset, comprising images with labeled damages, laid the groundwork for the LOFTR Matcher and Homography Matrix components of the algorithm.

Background removal techniques were used to remove all the background noise and to enhance the results.

The manual curation of the dataset ensures that the algorithm is trained and tested on real-world scenarios, enhancing its accuracy and robustness in detecting and tracking various types of damages on cars.

# 4 Implementation

## 4.1 YOLOv8 Model

The YOLOv8 model was initially implemented using default weights and parameters on our dataset. However, the initial results were suboptimal, failing to accurately detect damages on car images, especially when it came to identifying smaller details.

**Background Removal Experiment:** In an attempt to improve performance, an experiment was conducted by removing the background from the car images before feeding them into the YOLOv8 model. While this approach showed some improvement, the model still struggled to recognize finer details, leading to an incomplete detection of damages.

For background removal, we first tried to use YOLOv8 Segmentation model to remove the background but the huge processing times couldn't make it possible. We ended using 'rembg' library to remove the background

**Image Cropping:** To overcome the limitations identified in the previous experiment, we implemented a new approach. In this method, we cropped the images using a grid system, ensuring that each resulting cropped image was at least 640 pixels in size.

However, the outcomes from this approach did not meet our expectations. Consequently, we refined our strategy by focusing on the damaged areas within the images and cropped them to obtain more accurate representations. By default YOLO bounding box coordinates are positioned at the top left, but with these modifications, the bounding box is consistently centered.

Subsequently, we executed the model once again on this updated set of images to assess its performance under these refined conditions.

## 4.2 LoFTR Matcher Implementation

In the carry forward algorithm, the LOFTR (Local Feature-based Template-free Re-identification) model is utilized for key points matching between pairs of images. This process is essential for establishing correspondences between points in different images, a critical step in tracking pre-existing damages.

**Initialization** The LOFTR matcher is initialized with a specific configuration (default_cfg). This configuration includes parameters that guide the behavior of the model during key point extraction.

**Key Point Detection** Once initialized, the LOFTR matcher is applied to the pair of preprocessed images. The get_matches function orchestrates this process,

identifying key points based on local features. The result includes matched keypoints and their confidence scores.

**Confidence Sorting** The confidence scores associated with each matched keypoint pair are used to sort the matches. Sorting based on confidence prioritizes more reliable matches. The sorted indices are then used to obtain the final set of matched keypoints.

**Importance in Carry Forward Algorithm** The identified key points form the foundation for computing the homography matrix. This matrix facilitates the tracking of pre-existing damages across different images. The accuracy and reliability of the matches directly influence the precision of the subsequent steps in the carry forward algorithm.

### 4.3 Homography Matrix Application

After obtaining matched keypoints using the LOFTR matcher, the next crucial step in the carry forward algorithm involves applying a homography matrix. This matrix transformation is fundamental for tracking pre-existing damages across different images.

Homography Matrix Estimation The homography matrix (H) is computed using the RANSAC algorithm. This robust estimation method helps filter out outliers, providing a more accurate transformation between the matched keypoints in the two images.

Here, mkpts0 and mkpts1 represent the matched keypoints in the source and destination images, respectively. The third parameter, cv2.RANSAC, indicates the robust estimation algorithm used, and the last parameter, 5.0, denotes the threshold for considering a point as an inlier.

**Key Points Transformation** Once the homography matrix is obtained, it is applied to the keypoints from the source image (mkpts0). This transformation results in the corresponding points in the destination image. This step is crucial for establishing spatial correspondence between the keypoints, especially when there are variations in viewpoint or scale between the images.

The transform_points function iterates through the matched keypoints, applying the homography matrix to each point and storing the transformed coordinates.

**Error Calculation** To assess the accuracy of the homography transformation, the algorithm calculates the Euclidean distance between the transformed points and the manually marked points in the destination image. This distance is a measure of how well the homography aligns the matched keypoints. Additionally, the percentage error is calculated based on the actual distance between the points.

**Result Analysis** The transformed points and associated errors provide valuable insights into the effectiveness of the homography transformation. By analyzing the errors, the algorithm can distinguish between accurately tracked points and those that may have deviated during the transformation process.

### 4.4 Other Methods we tried

Before giving the nod to Homography Matrix, we tried to implement point triangulation method to find the equivalent point in another image.

We've also tried another key point matching techniques like SIFT, ORB, Lightglue, BRIEF but LoFTR outperformed every other model in terms of both speed and accuracy.

## 5 YOLOv8 Model Training

Utilized Weight & Biases (W&B) for comprehensive analysis, providing insights into training loss, validation metrics, and the impact of parameter tuning.

1. **Tuned Parameter Configuration for 5 Classes(tuned_param_crop_5_classes)** Fine-tuned model parameters to optimize YOLOv8 for detecting 5 specific classes, aiming for improved performance within this subset.

2. **Training and Testing on All Cropped Image (trn_tst_all_cropped)** Conducted training and testing on a diverse set of cropped images to evaluate the model's generalization across various scenarios.

3. **Configuration for All Cropped Images with 8 Classes(all_crop_8Cls)** Focused training on YOLOv8 to detect 8 classes within all cropped images, expanding the model's scope of detection.

4. **Configuration with Background Consideration for 8 Classes (all_crop_bg_8Cls)** Trained the model to detect 8 classes, considering background details to enhance object-background distinction.

# 6 YOLOv8 Damage Detection Metrics & Detections

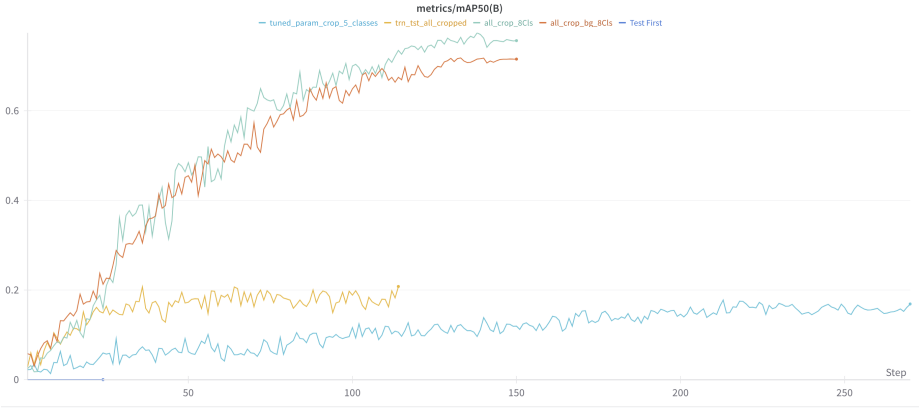


Figure 1: MAP50 for all runs

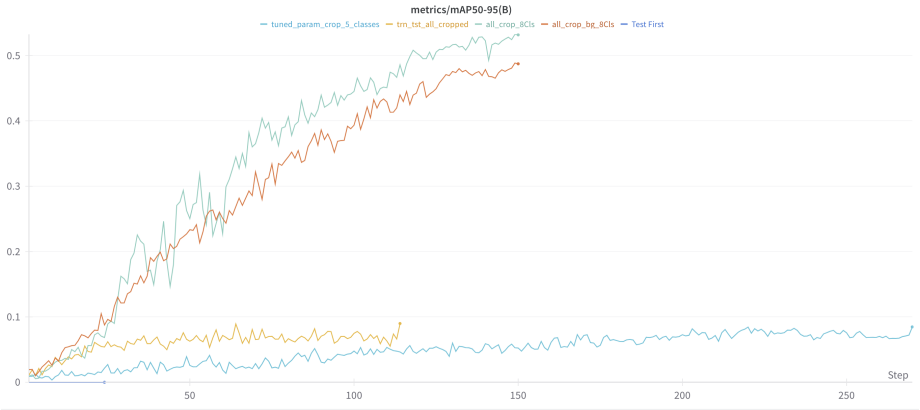


Figure 2: MAP50-95 for all runs

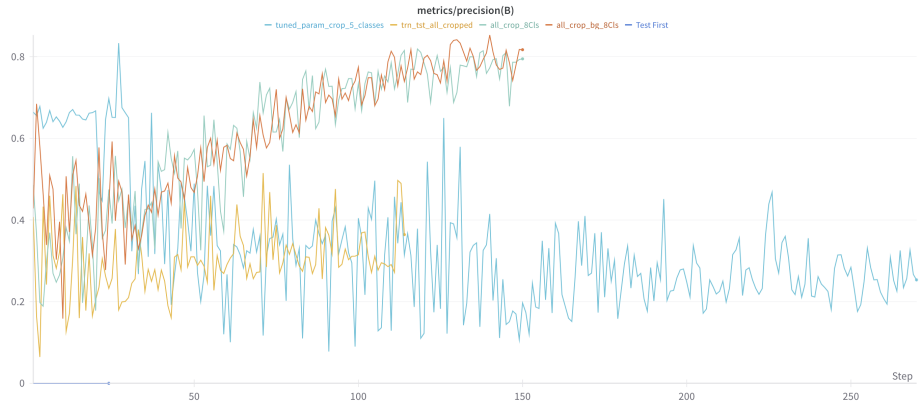


Figure 3: Precision for all runs

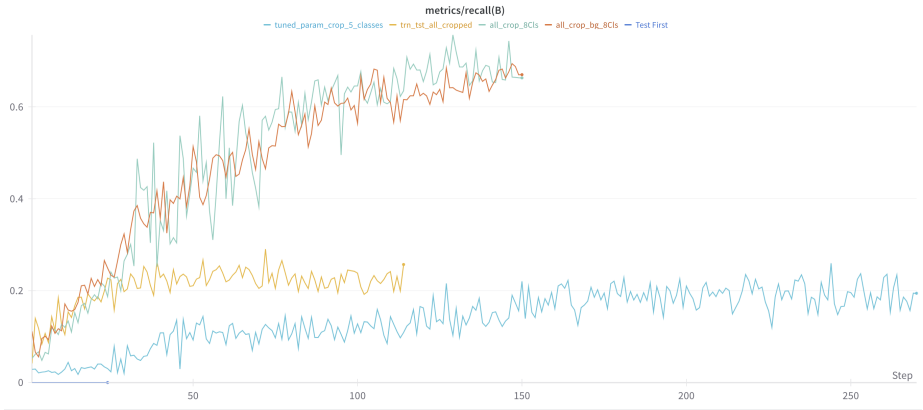


Figure 4: Recall for all runs


Figure 5: Damage Detections


Figure 6: Damage Detections


Figure 7: Damage Detections

# 7 CarryForward Matches

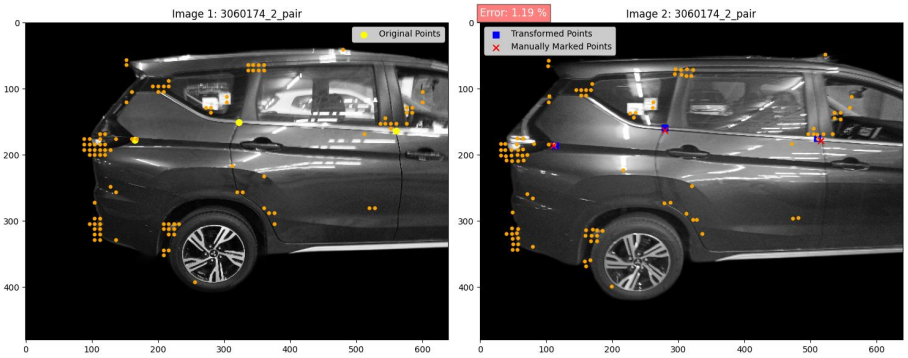


Figure 8: Equivalent Points

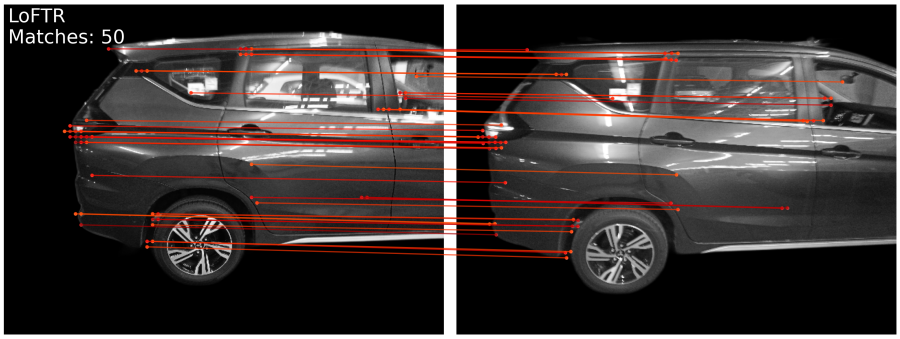


Figure 9: Matching Points

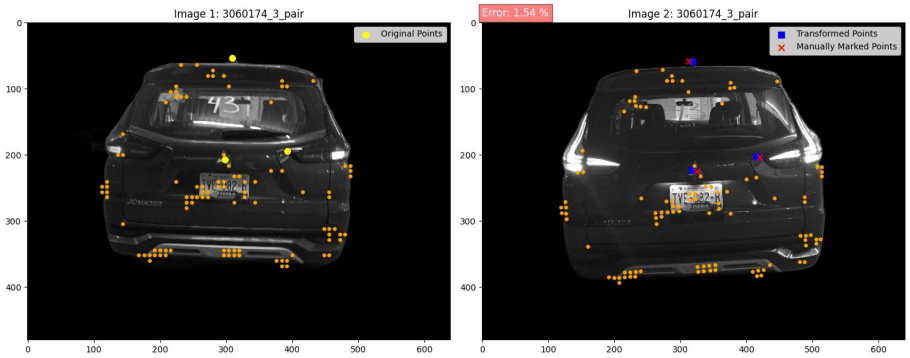


Figure 10: Equivalent Points

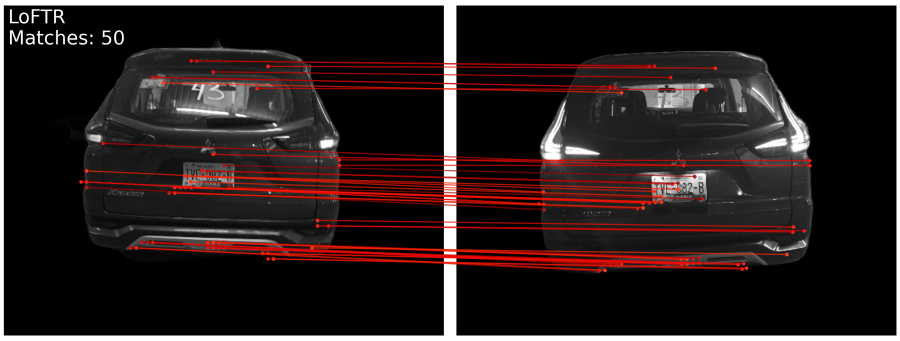


Figure 11: Matching Points

# 8 Conclusion

In conclusion, the DAMAGEiD project focused on improving car damage assessment through advanced techniques like YOLOv8 for damage detection and a novel Carry Forward Algorithm using LOFTR Matcher and Homography Matrix. Despite challenges, the project demonstrated promising outcomes in both damage detection and tracking pre-existing damages. Ongoing refinement and user feedback integration will be key for enhancing the system's effectiveness in real-world scenarios

# References


Sun, Jiaming, Shen, Zehong, Wang, Yuang, Bao, Hujun, & Zhou, Xiaowei. (2021). *LoFTR: DetectorFree Local Feature Matching with Transformers.* In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).* IEEE/CVF.

Jocher, G., Chaurasia, A., Qiu, J. (2023). *YOLO by Ultralytics* (Version 8.0.0). Retrieved from `https://github.com/ultralytics/ultralytics`. Software. AGPL-3.0 License.