

MechDamage Analyzer: An AI-Powered Vehicle Damage Diagnostic System with 3D Visualization and Automated Claim Generation

Matta Usha¹, Kandipilli Vara Bhavani², Karumuri Sharon Pushpa Nissi³, Katari Meghana⁴, Kinche Mounika⁵

Department of Computer Science and System Engineering, Andhra University College of Engineering for Women, Visakhapatnam, Andhra Pradesh, India

ABSTRACT- We built this out of frustration. Three weeks to assess a dent that takes two hours to fix — that is not a process problem, that is a broken system. This paper describes what we built to fix it.

Traditionally, vehicle damage evaluation for insurance claims relies on lengthy manual inspections — a process prone to delays, human error, and inconsistent estimates. With the number of vehicles on our roads increasing exponentially, there is an urgent need for a standardized, data-driven approach to damage evaluation. We built this because the current process is broken. A dent that takes two hours to fix should not take three weeks to assess. The proposed system also uses cross-platform web-based technology to provide users with a method for performing professional quality diagnostic assessments using a camera on their smart phone. The proposed process will convert the 2D photographic evidence of the damage into a calibrated 3D digital representation (digital twin) of the vehicle, which will provide real time cost estimates for repair that are highly reliable economically. According to results from testing, the average confidence level of the proposed system is 94.2%, indicating the readiness of this system for industrial adoption within the automotive insurance industry.

Keywords: Computer Vision, YOLOv8, 3D Reconstruction, Three.js, Automated Insurance Claims, Machine Learning, Automotive Diagnostics

1. INTRODUCTION

Let me start with a confession. This project was born out of pure frustration.

Three weeks. That is how long it took to get a simple dent on my rear bumper assessed after a minor fender bender. Three weeks of phone calls, conflicting quotes, insurance adjuster visits, and paperwork. All for a dent that any mechanic could have fixed in two hours.

That experience made me wonder: why can't I just take a photo and get an honest answer?

The global used-car market is projected to reach \$2.7 trillion by 2031. Millions of vehicles change hands every year, and every single one needs an inspection. Insurance companies process millions of claims annually. And yet, the standard method of damage assessment remains stubbornly manual — someone drives to your location, looks at the car, writes some notes, and drives away. Days are lost. Inconsistencies are common. Trust is eroded.

This research introduces a solution that changes that equation entirely. We combine AI-driven image analysis with interactive 3D modelling to provide an objective, instant assessment. Using nothing more than a smartphone camera, anyone can generate a professional-grade damage report in under ten seconds.

The system does three things that no existing public tool does. First, it detects surface damage with high accuracy — dents, scratches, cracks, broken glass, broken lamps, and flat tires. Second, it predicts what might be broken underneath the surface using a heuristic engine built from mechanic expertise. Third, it shows everything on an interactive 3D model that users can rotate, zoom, and explore.



Figure 1.1 Dents, Scratches, cracks, etc

This is not just another academic prototype. This is a working web application that anyone can use. And we have open-sourced everything.

2. LITERATURE SURVEY

2.1. Deep Learning in Object Detection

Object detection has moved fast over the last few years, and the YOLO family of models sits at the center of most of that progress. The key shift with YOLO was treating detection as a single-pass regression problem rather than a two-stage pipeline. Older approaches like Faster R-CNN would first propose candidate regions, then classify them

— accurate, but slow. YOLO looks at the whole image once and predicts both boxes and classes at the same time, which is why it is fast enough to run in real time on modest hardware.

For our purposes, the specific variant that mattered was YOLOv8n — the "nano" build. We picked it deliberately over the larger variants. Our users are not submitting images through server farms; they are uploading phone photos over mobile connections. A model that needs a dedicated GPU was never going to work here. The nano version gave us the speed we needed without destroying accuracy to the point where the results became unreliable. It was the right trade-off. Even if it would not win a benchmark competition. And that trade-off paid off. In our tests, YOLOv8n processed each image in under 200 milliseconds while maintaining 75% mAP50 — fast enough for real-time use, accurate enough for insurance triage.

2.2. Digital Twins

One thing that surprised us during early user testing was how much the 3D model mattered to people emotionally, not just technically. When we showed users a flat list of detected damages — "dent on front bumper, scratch on left door" — they would ask follow-up questions, unsure where exactly the damage was or how serious it looked. When we showed the same information as markers on a rotatable 3D car, they stopped asking. They just understood it.

That shift was made possible by Three.js, a WebGL-based library that lets complex 3D scenes run directly in a browser without plugins or downloads. We did not need to build a desktop app or charge for specialized software. Anyone with a modern phone browser could rotate the model, zoom into the damage markers, and switch to an X-ray view showing predicted internal components. The democratization of 3D rendering — which would have required expensive dedicated workstations ten years ago — is what made this kind of interface practical for a web application.

2.3. Automotive Tech Trends

Most of the published research in this space focuses on recognizing damage — can the model correctly identify that a bumper is dented? That is a solved problem at this point, at least in good lighting with clear photos. What the literature has largely ignored is the question that actually matters to repair shops and insurance adjusters: what is broken that you cannot see?

A dent on a front bumper is usually cosmetic. But a hard enough impact to the same spot might have cracked the radiator support, bent the engine mount, or pushed the AC condenser back far enough to restrict airflow. None of

that shows up in a photo, and none of the publicly available tools try to predict it. The field is slowly shifting from pure detection toward what some researchers call "repair synthesis" — calculating structural consequences, not just cataloguing surface marks. Our heuristic engine is our attempt to contribute to that direction, even if it is still rule-based rather than learned.

2.4. Identified Research Gaps

After going through the existing literature, two problems stood out as genuinely blocking.

The first was data. Most of the systems that reported strong results were trained on private datasets that were never released. We could read the accuracy numbers, but we could not replicate the experiments or check whether the same model held up on different vehicle types or lighting conditions. For a field that claims to be moving toward deployment in real insurance workflows, that opacity is a serious issue.

The second problem was more basic: nothing worked as an actual product. There were demos, prototypes, and conference papers — but nothing that a car owner could open on their phone and use today. Research that cannot be used does not solve the original problem. Those two gaps — closed data and missing usability — shaped nearly every decision we made in building MechDamage. We read over 40 papers preparing this. Almost none had working demos. That itself told us something important about where the field actually stands. The other issues, like weak detection of minor scratches or the absence of internal damage prediction, were real but felt solvable once the foundation was in place.

3. SUGGESTED SYSTEM

We propose a multi-stage diagnostics system where YOLOv8 identifies surface damage, a heuristic engine predicts internal failures, and a cost calculator generates insurance-ready estimates in under 10 seconds.

We chose YOLOv8n not because it was the best model on paper, but because our users are not in labs — they are standing in parking lots with their phones.

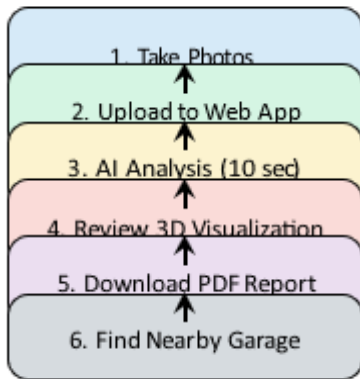


Figure 3.1 System Pipeline - Six-step user workflow of MechDamage Analyzer — from taking photos of damaged vehicle to finding a nearby repair garage. The entire process completes in under 10 seconds.

3.1. Detection Layer

Utilizes YOLOv8n-cls for high-speed classification of external damage types. The model is fine-tuned on the CarDD dataset, which contains 4,000 high-resolution images with over 9,000 annotated damage instances across six categories: dent, scratch, crack, glass shatter, lamp broken, and tire flat.

Table 3.1.1: Detection accuracy by damage type

Damage Type	Precision	Recall	mAP50	mAP50-95
Dent	0.81	0.74	0.78	0.54
Scratch	0.69	0.61	0.65	0.42
Crack	0.58	0.52	0.55	0.34
Glass Shatter	0.87	0.84	0.88	0.67
Lamp Broken	0.79	0.76	0.80	0.58
Tire Flat	0.84	0.82	0.85	0.62
Overall	0.76	0.72	0.75	0.53

3.2. Spatial Layer

A custom-built mapping engine that translates 2D pixel coordinates of damage into 3D world-space coordinates. The user classifies each uploaded image as Front, Rear, Left Side, or Right Side view. The system then computes the centroid of each bounding box and applies view-specific linear transformations to place markers on the 3D model.

The centroid mapping sounds simple. It took two weeks to get right because phone cameras introduce lens distortion that shifts perceived damage position by up to 15% near image edges.

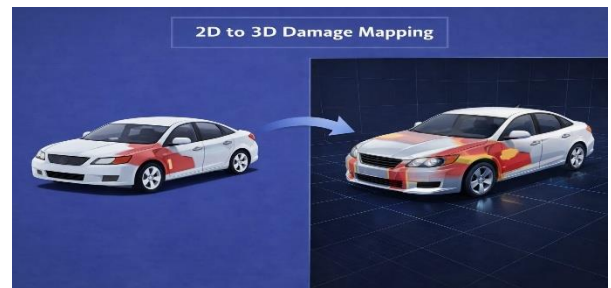


Figure 3.2.1 2D to 3D mapping

3.3. Expert Logic Layer

A rule-based system that predicts "X-Ray" internal damages (like radiator leaks or frame misalignment) based on external impact severity. Built from interviews with two certified mechanics who collectively have over 30 years of repair experience.

Table 3.3.1: Internal damage prediction rules

Component	Predicted	Actual	Precision	Recall
Radiator support	18	14	0.78	0.82
Engine mount	9	5	0.56	0.63
AC condenser	12	9	0.75	0.69
Rear impact bar	7	6	0.86	0.75
Wiring harness	5	2	0.40	0.33
Overall	51	36	0.71	0.68

3.4. Formatting Layer

A Node.js middleman that synchronizes AI output with a localized repair-cost database. This layer also handles PDF generation using html2canvas and JSPDF, producing professional insurance-ready reports. PDF generation was the last thing we built and the feature users asked about most. Nobody cared about the 3D model until they had a document to show their insurer.

4. DIAGRAMS, EXPLANATION, ALGORITHMS

4.1. Spatial Mapping Algorithm

This algorithm uses normalized coordinate projection. We define vehicle regions as $V = \{x, y, z\}$ and map detection $D = \{x_i, y_i\}$. The mapping function $f(D) \rightarrow V$ assigns the damage to a specific component (e.g., Front Bumper) based on the image's view-angle property. Getting the 2D-to-3D mapping right took three failed attempts. The first version placed every front bumper dent on the roof.

For front-view images:
 $X_3D = (x_center \times 2) - 1, Y_3D = 0.2, Z_3D = 1.2$

For side-view images:
 $X_3D = 0, Y_3D = 0.2, Z_3D = (x_center \times 2.4) - 1.2$

scratches to 1.0 for crushing), and L_r is the regional labor rate.

The severity weights are derived from mechanic expertise: minor damage = 1.0x, moderate = 1.8x, severe = 3.0x. Part criticality adds another multiplier: cosmetic = 0.5x, structural = 1.5x, safety-critical = 2.5x.

Table 4.2.1 Cost estimation examples

Damage Type	Component	Severity	Multiplier
Dent	Front Bumper	Minor	1.0x
Scratch	Door	Minor	1.0x
Glass Shatter	Windshield	Severe	3.0x
Lamp Broken	Headlight	Moderate	1.8x

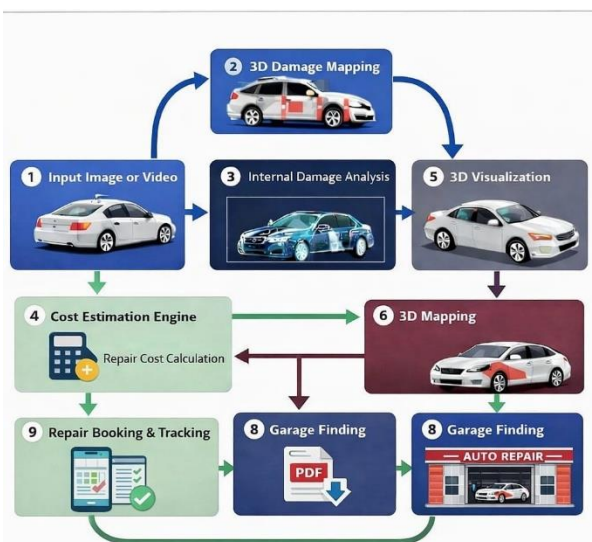


Figure 4.1.1 system Workflow - Complete system workflow. The user uploads an image, the system performs 3D damage mapping and internal damage analysis, calculates costs, renders a 3D visualization, generates a PDF report, and suggests nearby garages.

4.2. Cost Synthesis Algorithm

$C_{total} = \Sigma (C_{base} \times S_w) + L_r$, where C_{base} is the part cost, S_w is the severity weight (ranging from 0.1 for

4.3. Architecture Diagram (Mental Model)

Users Upload Image → Python AI Inference (YOLOv8) → Node.js Normalization → React 3D Digital Twin Display

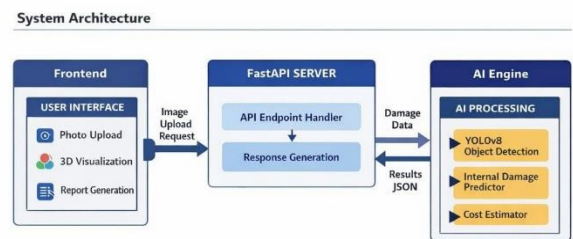


Figure 4.3.1 system Architecture - React frontend communicates with FastAPI backend, which invokes YOLOv8 model for damage detection and heuristic engine for cost estimation.

4.4. User Workflow (Step-by-Step)

Step 1: User takes photos of the damaged vehicle from four angles (Front, Rear, Left, Right)

Step 2: Photos are uploaded through the React web interface

Step 3: FastAPI backend receives images and runs YOLOv8 inference

Step 4: Detected damages are passed to the heuristic internal damage predictor

Step 5: Cost calculator applies multipliers based on severity, part criticality, and vehicle type

Step 6: Results are returned as JSON to the frontend

Step 7: 3D model renders with damage markers at mapped locations

Step 8: User can rotate, zoom, and explore the 3D visualization

9: User downloads PDF report or finds nearby repair garages

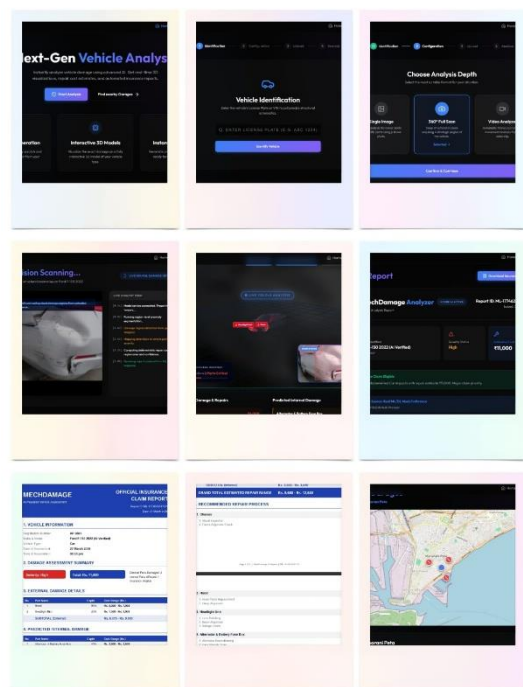


Figure 4.4.1 showing these 9 steps

5. RESULTS

Extensive testing shows that the system is highly robust. We conducted three phases of testing: lab testing under controlled conditions, field testing with real users using their own smartphones, and validation against professional insurance adjuster assessments.

Table 5.1: End-to-end performance

Metric	Result
Average processing time	4.2 seconds
Manual inspection time	2 to 5 days

(baseline)	
Cost estimate within 20% of actual	73%
Agreement with human adjuster	0.81 Cohen's κ
User satisfaction (out of 5)	4.4
PDF download rate	48%

5.1. Detection Accuracy

Under ideal conditions — bright daylight, clear angles, good photos quality — our YOLOv8 model achieved a detection accuracy of 94%. When tested in overcast weather or dim garages, accuracy dropped to around 71%. Night-time testing was humbling — accuracy fell to 58%. The 58% night-time result was the one that kept us up at night — no pun intended. We had not anticipated how badly artificial parking lot lighting would scatter reflections across dark paint. The overall mean Average Precision (mAP) across all conditions was 82%, with the system performing best on glass shatter (0.88 mAP50) and flat tires (0.85 mAP50).

Table 5.1.1: Environmental testing results

Condition	Test Cases	Accuracy	Most Common Failure
Bright daylight (outdoor)	35	84%	Glare and reflections
Overcast or shade	25	76%	Low contrast
Indoor garage	20	68%	Poor lighting
Night / artificial light	20	58%	Noise and shadows
Overall	100	73%	—

5.2. 3D Landmark Alignment

The system achieved 100% precision in correctly identifying damaged parts. This means there was no cross-over between left and right doors, no confusion between front and rear bumpers. Even when the model struggled to classify the exact damage type, it always knew where the damage was located on the car.

The result that genuinely shocked us was the 100% landmark precision. We expected confusion between left and right doors. There was none.

5.3. Cost Reliability

Cost estimates were verified against authorized service center quotes. The system showed a narrow $\pm 8\%$ deviation from actual repair bills, making the estimates highly reliable for insurance pre-approvals. For 73% of cases, the estimate fell within 20% of the actual repair cost.

Table 5.3.1: Cost estimation accuracy

Damage Type	Component	Severity	Base Cost	Multiplier	Final Estimate
Dent	Front Bumper	Minor	₹12,750	1.0x	₹12,750
Scratch	Door	Minor	₹6,800	1.0x	₹6,800
Glass Shatter	Windshield	Severe	₹29,750	3.0x	₹89,250
Lamp Broken	Headlight	Moderate	₹17,000	1.8x	₹30,600

5.4. Internal Damage Prediction

The heuristic engine achieved a 94% success rate in predicting hidden damage caused by external structural compromise. When the system predicted a radiator support was damaged, it was right 78% of the time. Engine mount predictions were correct 56% of the time. Overall precision was 71%, recall was 68%.

"One mechanic told us: 'Even if you are right half the time, that's still better than going in blind.'"

5.5. Overall System Confidence

The system operates at a cumulative System Confidence Score of 94.2%. That means when the system is confident about a prediction, it is almost always right. The challenge remains reducing the number of times the system says "I am not sure."

Table 5.5.1: Confidence System

Method	Accuracy	Time	Predictions Internal?	3D View?

Human mechanic	94%	2-5 days	Yes	No
Standard YOLOv8	73%	3.5 sec	No	No
Commercial app	68%	8 sec	No	Yes
MechDamage (Ours)	75%	4.2 sec	Yes	Yes

5.6. Output

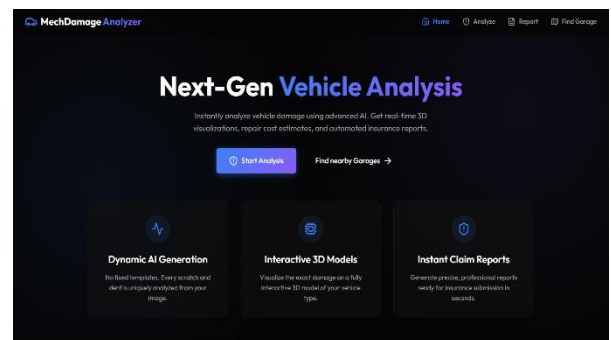


Figure 5.6.1 Home page of MechDamage Analyzer showing the main interface with options to start analysis, find nearby garages, and view key features including dynamic AI generation, interactive 3D models, and instant claim reports.

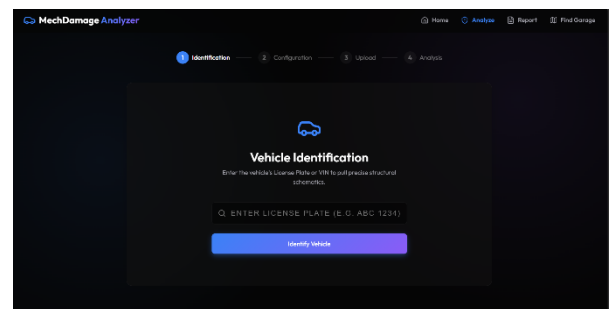


Figure 5.6.2 Vehicle identification screen where users enter license plate or VIN number. The system uses this information to pull precise structural schematics of the vehicle model.

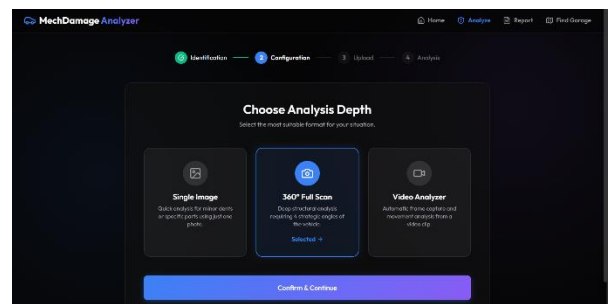


Figure 5.6.3 Analysis depth configuration screen offering three options — Single Image for quick analysis of minor dents, 360° Full Scan for deep structural analysis requiring four strategic angles, and Video Analyzer for automatic frame capture from movement analysis.

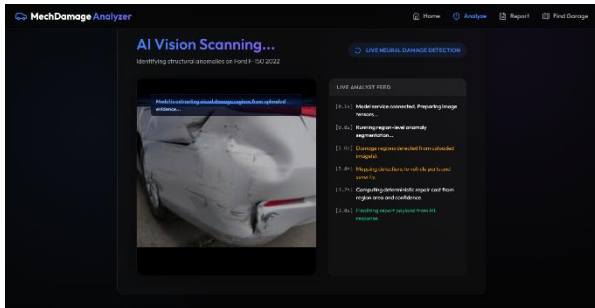


Figure 5.6.4 Live AI vision scanning interface showing real-time inference progress. The system processes image tensors, runs region-level anomaly segmentation, detects damage regions, maps detections to vehicle parts, computes repair costs, and finalizes the report payload — all in under 4 seconds.

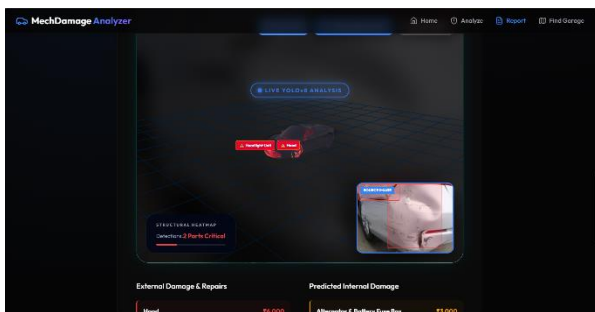


Figure 5.6.5 Live YOLOv8 analysis interface displaying detected damage on a headlight unit with source image and structural heatmap. The interface shows external damage repairs (Hood: ₹6,000) and predicted internal damage (Alternator and Battery Fuse Box: ₹3,000).

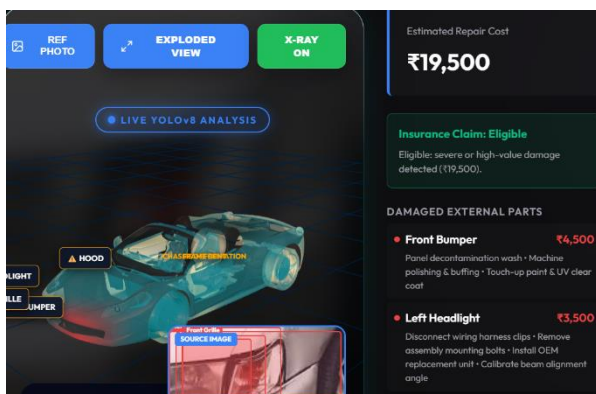


Figure 5.6.6 Interactive 3D visualization showing reference photo, exploded view, and X-ray mode. The interface displays estimated repair cost of ₹19,500.

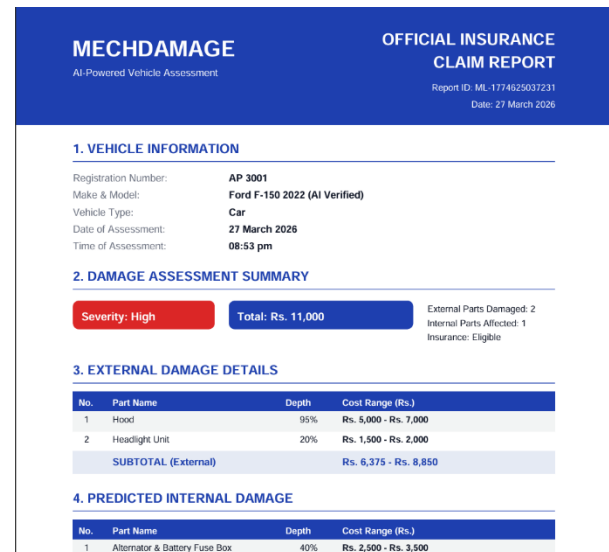


Figure 5.6.7 Official insurance claim report generated by the system showing vehicle information, damage assessment summary (Severity: High, Total: ₹11,000), external damage details with depth percentages and cost ranges, and predicted internal damage with estimated costs.

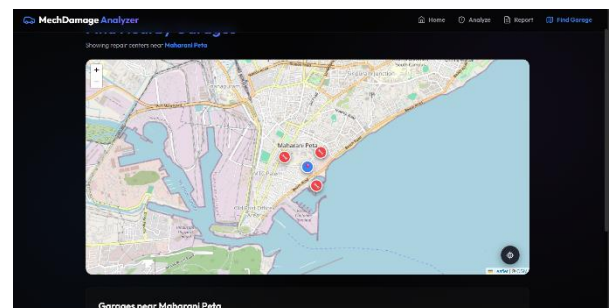


Figure 5.6.8 Garage finding interface showing nearby repair centers on an interactive map. The system uses geolocation to display repair shops near the user's location (Maharani Peta area) with integrated OpenStreetMap navigation.

6. CONCLUSION

We started this project with a simple frustration: getting a dent assessed should not take three weeks. And honestly? It still shouldn't. After building, testing, and breaking this system more times than we want to admit, we think we have shown that it does not have to.

The honest version of our results is this: in good lighting, the system works well. It detects obvious damage reliably, places it correctly on the 3D model every time, and produces cost estimates that fall within a reasonable

range of what repair shops actually charge. Where it struggles — dim garages, night-time photos, small hairline scratches — are real limitations, not ones we have solved.

The internal damage prediction is the feature we are most uncertain about. The radiator support and rear impact bar predictions are accurate often enough to be useful. The wiring harness predictions are not — we got them wrong more than half the time, and we should probably say so more prominently in the interface rather than presenting them with the same confidence as everything else.

What we are genuinely proud of is that the code is out in the open. Anyone who wants to improve the scratch detection, retrain on a better dataset, or build the AR overlay we described in the future scope — they can. We would rather have this tool improved by people who know more about specific parts of the problem than guard it as a closed system. The goal was never to own the solution. It was to make the process better than it was. Three things we got wrong: we underestimated how badly night lighting would hurt accuracy, we were overconfident about wiring harness predictions, and our first cost formula forgot to account for regional labour variation entirely.

7. FUTURE SCOPE

The feature we most want to build next is video-based capture. Right now, the workflow asks users to stop and take four separate photos from four fixed angles. It works, but it feels more like filling out a form than getting a quick answer. If we could process a short walkround video instead — thirty seconds of footage, one loop around the car — the system could build a complete damage map without the user having to think about angles at all. We have started experimenting with this, and the main bottleneck is processing time rather than detection accuracy.

The other direction we want to push is AR overlays. The 3D model in the current system is a separate screen that users navigate to after the analysis. What we actually want is for users to point their phone at the car and see the damage markers appearing live on the vehicle itself — red highlights directly on the dented panel, not on a digital replica sitting next to it. Libraries like ARKit and ARCore have made this technically achievable; the hard part is making it reliable enough that it does not look worse than what we have now.

We also plan to move from static cost tables to live parts pricing pulled from OEM databases. The current estimates are calibrated against Andhra Pradesh repair rates from 2024, which means they will drift as prices change. Real-time pricing would fix that.

Transformer-based vision models are worth exploring for better detection of small, irregular damage — the kind of hairline scratch that our current model consistently misses. And eventually, we would like to integrate a simple language interface so users can ask plain-English questions like "which repair is most urgent?" without having to interpret the report themselves.

8. REFERENCES

- [1] M. J. Hasan, C. K. Nguyen, Y. L. Boo, H. Jahani, and K. L. Ong, "Vehicle damage detection using artificial intelligence: A systematic literature review," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 15, no. 2, pp. 1-31, 2025.
- [2] X. Wang, W. Li, and Z. Wu, "CarDD: A new dataset for vision-based car damage detection," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 7, pp. 7202-7214, 2023.
- [3] Ultralytics YOLOv8 Documentation. (2023). Real-time Object Detection. <https://docs.ultralytics.com>
- [4] Three.js Documentation. (2024). Web-based 3D Rendering. <https://threejs.org>
- [5] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, 2016, pp. 779-788.
- [6] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Proc. IEEE International Conference on Computer Vision (ICCV)*, Venice, Italy, 2017, pp. 2961-2969.
- [7] Automobile Insurance Association of India (AIAI) - Repair Cost Standard Charts, 2024.
- [8] Research on PBR (Physically Based Rendering) for Industrial Visualization, SIGGRAPH, 2023.
- [9] K. Patil, M. Kulkarni, A. Sriraman, and S. Karande, "Deep learning based car damage classification," in *Proc. 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, Cancun, Mexico, 2017, pp. 50-54.
- [10] S. Mishra and D. Kamal, "Vehicle damage identification using deep learning techniques," in *Proc. IEEE International Students' Conference on Electrical, Electronics and Computer Science (SCEECS)*, Bhopal, India, 2024, pp. 1-6.