SAARTHI: Smart Auto Assessment and Roadside Technical Help Interface

Chirayu Sanghvi(✉) and Alina Vereshchaka

Department of Computer Science and Engineering,
State University of New York at Buffalo, Buffalo, USA
{chirayus,avereshc}@buffalo.edu

Abstract. Emergency vehicle accidents pose significant challenges to operational efficiency and financial stability within emergency services, impacting organizations and communities. These incidents result in substantial repair costs, prolonged vehicle downtime, and potential legal liabilities, straining crucial public safety resources. Additionally, issues like inflated repair costs, inefficient roadside assistance, and lengthy insurance processes compound these challenges. The Smart Auto Assessment and Roadside Technical Help Interface (SAARTHI) provides a comprehensive end-to-end solution, integrating with emergency service protocols to leverage computer vision technologies for damage assessment, consistent repair cost estimation, immediate repair coordination, towing services, and faster insurance processes. By addressing these issues, SAARTHI enhances the efficiency and reliability of emergency response systems, ensuring continuous service coverage and improved operational readiness. This socially beneficial solution strengthens emergency services and promotes community safety and well-being. SAARTHI demonstration and source code are available at sites.google.com/view/saarthi-home.

Keywords: Emergency vehicles · Roadside Assistance · Convolutional Neural Networks · Object detection · Instance Segmentation · Salient object detection · End-to-end framework · Non-Maximum Suppression

1 Introduction

Emergency vehicle accidents, particularly those involving ambulances and fire trucks, pose operational and financial challenges. Immediate consequences include vehicle downtime, increased operational costs, and potential legal liabilities. The total number of ambulance crashes, including minor "fender benders," has been estimated at 6,500 per year [1]. Fire truck crashes occur at a rate of approximately 30,000 per year, having potentially dire consequences for the vehicle occupants and the community if the fire truck was traveling to provide emergency service [2]. Each year, there are approximately 300 fatalities in the U.S. that occur during police pursuits [3]. These crashes often occur at high speeds, at night, and on local roads. The economic impact of vehicle crashes is substantial, with the government paying an estimated $35 billion annually [4].

R. Thomson et al. (Eds.): SBP-BRiMS 2024, LNCS 14972, pp. 35–46, 2024.
https://doi.org/10.1007/978-3-031-72241-7_4

These accidents lead to issues such as inflated repair costs, delays in vehicle recovery due to inefficient roadside assistance, and prolonged insurance processes that extend vehicle downtime. These challenges strain emergency services financially and impede their ability to provide timely and effective responses.

In this paper, we introduce the Smart Auto Assessment and Roadside Technical Help Interface (SAARTHI), an end-to-end framework that addresses these challenges. SAARTHI leverages advanced artificial intelligence (AI) technologies, including object detection, instance segmentation, and salient object detection, to improve the way emergency vehicle damages are assessed and repairs are managed. This platform assesses damage from user-uploaded images, classifies the damages into six categories-dent, scratch, crack, lamp broken, glass shatter, and tire flat, and further distinguishes them as major and minor damages, also providing immediate repair cost estimates. While current processes for dealing with emergency vehicle damages are generally functional, SAARTHI aims to enhance existing protocols by offering rapid damage assessment tools, supporting decision-making for fleet managers and maintenance departments.

One of the issues in emergency vehicle operations is the variability and inconsistency in repair estimates for similar damages. There are inflated repair costs due to the lack of standardized pricing. For example, two similar ambulance accidents might result in vastly different repair quotes from different service providers, causing potential overpayment. Implementing a standardized repair cost estimation system can aid government agencies in planning their budgets for upcoming years by reducing overpayment for repairs.

Another challenge is the efficiency of roadside assistance, which is crucial for keeping emergency vehicles operational. Delays in returning a fire truck to service after an accident can leave the fire department short of essential resources, impacting their ability to respond effectively to emergencies.

To address these issues, we introduce SAARTHI, a comprehensive framework that manages the entire lifecycle of an emergency vehicle from accident to return to service. This end-to-end AI-powered framework can be directly implemented to expedite vehicle damage management. Key features of SAARTHI include:

- Real-time damage assessment. We implemented Resnet-based [5] mask-RCNN [6] model utilizing MMDetection [7], and U-net to perform real-time damage assessments from user-uploaded images.
- Detailed reporting and documentation. The system generates detailed reports and documentation of the damage, including images, statistical graphs, estimated repair costs, and other critical details.
- Repair cost estimation. By integrating Non-Maximum Suppression [8,9] and considering the base prices of different types of damages along with their impact factors, we provide accurate repair cost estimations.
- Immediate assistance via chatbot. SAARTHI includes a chatbot feature that provides immediate assistance to users.

The rest of the paper is structured as follows: Sect. 2 covers related work. Section 3 presents our approach and algorithms for damage detection and repair cost estimation. Section 4 details the SAARTHI framework. Section 5 provides experimental results, and Sect. 6 concludes the paper.

2 Background

Advanced AI technologies, such as object detection [10] and instance segmentation [11], offer robust solutions for assessing vehicle damage [12]. There is abundant research that has utilized Convolutional Neural Networks (CNNs) for detecting damages in vehicles [13,14]. CNNs have shown great promise in accurately identifying various types of vehicle damage. However, one of the significant challenges in this field has been the accurate detection of overlapping damages. Overlapping damages complicate the task of isolating individual damages, making it difficult to provide precise repair cost estimates.

In the SAARTHI framework, after implementing the initial damage assessment using object detection and instance segmentation, we apply the Non-Maximum Suppression (NMS) technique [15] to eliminate the overlapping detection of damages. NMS is crucial for refining the results of damage detection by ensuring that only the most significant damages are considered. By focusing on the most significant damage, we can more accurately identify the estimated repair cost. Detecting vehicle damage, particularly with irregular shapes and flexible boundaries, poses significant challenges. Scratches and cracks often have similar contours and colors, leading to misclassification [16]. To address this, we use Salient Object Detection (SOD) methods, which refine boundaries and segment objects with irregular shapes. SOD locates all salient objects without classifying them, ensuring accurate assessment of dents, scratches, and cracks by focusing on the location and extent of the damage.

3 Methodology

3.1 Damage Detection And Segmentation For Emergency Vehicle

The aim of car damage assessment is to accurately detect, classify, and contour damages on vehicles, aligning with the objectives of instance segmentation and object detection. We implement a Mask R-CNN [6] model with a ResNet-50 [5] backbone using the MMDetection [7,17] toolbox (Fig. 1). The model is initially pre-trained on the COCO dataset [18] and fine-tuned on the CarDD dataset [12], which consists of approximately 4000 images, manually annotated with bounding boxes and masks to improve the performance on vehicles assessments.

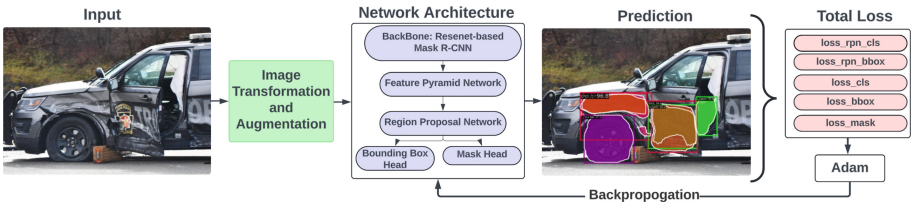


Fig. 1. Network architecture of the Mask R-CNN model with a ResNet-50 backbone, used for damage detection and segmentation in emergency vehicles

To enhance the training process, we implement a custom hook that dynamically modifies the data augmentation pipeline, incorporating steps to increase model robustness. Images are loaded with their annotations (bounding boxes and masks), randomly resized with scales from 0.1 to 2.0. Then randomly cropped, flipped to handle orientation variations, and padded to 640×640 pixels. This augmentation strategy introduces diverse transformations, improving the model's generalization.

3.2 Non-Maximum Suppression for Bounding Box Filtering

To refine our object detection results and prepare data for repair cost estimation, we apply the Non-Maximum Suppression (NMS) [8,9] algorithm. NMS reduces the number of overlapping bounding boxes by retaining only the most relevant ones, ensuring each detected object is represented by a single bounding box, thus enhancing the accuracy of subsequent cost estimation.

The NMS algorithm computes the Intersection over Union (IoU) between bounding boxes to measure overlap. It selects the box with the highest confidence score and suppresses all other boxes with an IoU greater than a specified threshold. This process is repeated until only the most significant bounding boxes remain (Algorithm 1). Figure 2 shows the NMS results on a sample input.

Algorithm 1 Non-Maximum Suppression (NMS)

```
function NonMaximumSuppression(boxes, scores, IoUThreshold)
    boxes ← {b_1, ..., b_N}                          ▷ List of detection boxes
    scores ← {s_1, ..., s_N}                         ▷ Corresponding detection scores
                          ▷ IoUThreshold: Maximum allowable overlap between bounding boxes
    idxs ← np.argsort(scores)[::-1]
    selected_idxs ← []
    while idxs.size > 0 do
        current_idx ← idxs[0]
        selected_idxs.append(current_idx)
        if idxs.size == 1 then
            break
        end if
        rest_idxs ← idxs[1:]
        rest_boxes ← boxes[rest_idxs]
        ious ← [ComputeIoU(boxes[current_idx], rest_boxes[i]) for i in range(len(rest_idxs))]
        idxs ← rest_idxs[ious < IoUThreshold]
    end while
    return selected_idxs
end function
```

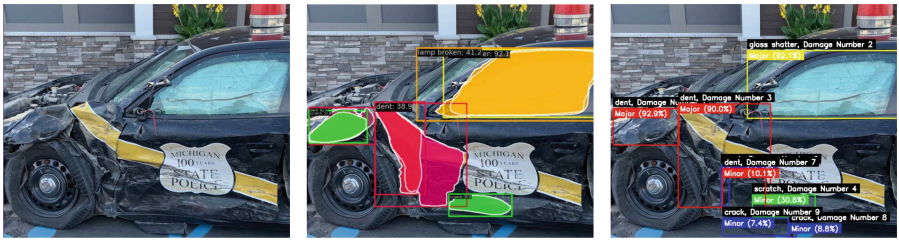
For bounding boxes boxA and boxB defined by corners (x_1, y_1, x_2, y_2):

$$\textbf{Intersection Area} = \max(0, \min(x_2^A, x_2^B) - \max(x_1^A, x_1^B)) \times \max(0, \min(y_2^A, y_2^B) - \max(y_1^A, y_1^B))$$

$$\textbf{Area of boxA} = (x_2^A - x_1^A) \times (y_2^A - y_1^A) \qquad \textbf{Area of boxB} = (x_2^B - x_1^B) \times (y_2^B - y_1^B)$$

$$\textbf{IoU} = \frac{\text{Intersection Area}}{\text{Area of boxA} + \text{Area of boxB} - \text{Intersection Area}} \tag{1}$$

– Area of Intersection is the overlapping area between two bounding boxes
– Area of Union is the total area covered by the two bounding boxes.



(a) Original image (b) Damage detection (c) NMS result

Fig. 2. Non-Maximum Suppression (NMS) results applied on a police vehicle

3.3 Estimating Repair Cost Using Detected Damages

To estimate repair costs for car damages, we employ an algorithm that processes detected bounding boxes, labels, and confidence scores. The methodology involves two primary steps: NMS (Sect. 3.2) and cost calculation.

NMS filters redundant bounding boxes to ensure each damage type is uniquely represented. Subsequently, the repair cost is computed based on predefined base costs for each damage type. This base cost is adjusted by a severity factor, which is derived from the area of the bounding box, the associated confidence score, and a normalization factor to adjust severity. The final repair cost is the sum of these adjusted costs. Repair cost is calculated by:

$$\text{Cost} = \text{Base Cost} \times \left(1 + \frac{(\text{Area} \times \text{Score})}{\text{Normalization Factor}}\right) \tag{2}$$

This approach ensures precise and reflective repair cost estimation based on the severity of detected damages, facilitating accurate repair assessments.

3.4 Salient Object Detection For Emergency Vehicle Damages

Salient Object Detection (SOD) enhances the detection of car damages by refining the boundaries of irregular and slender shapes. SOD focuses on locating all salient objects in an image, highlighting the most noticeable and severe areas of damage, and reducing noise by filtering out less relevant parts. This is useful in complex scenes where it is more important to identify key areas of damage than to classify them.

We apply a modified U-Net [19] model to refine the boundaries of detected damages. Using the CarDD [20] dataset, images are resized to 256×256 pixels and augmented with random resizing and flipping. The U-Net model was trained on an NVIDIA Tesla T4 GPU with a batch size of 32 for 250 epochs, using an Adam optimizer with a learning rate of 0.001 and weight decay of 1e-5. The loss function is Binary Cross-Entropy with Logits Loss (BCEWithLogitsLoss).

4 SAARTHI Framework

The SAARTHI framework (Fig. 3) integrates advanced AI techniques for vehicle damage assessment and repair coordination, streamlining the process from accident to repair completion. In this context, "users" refers to emergency vehicle drivers or fleet managers, depending on the organization's protocol.

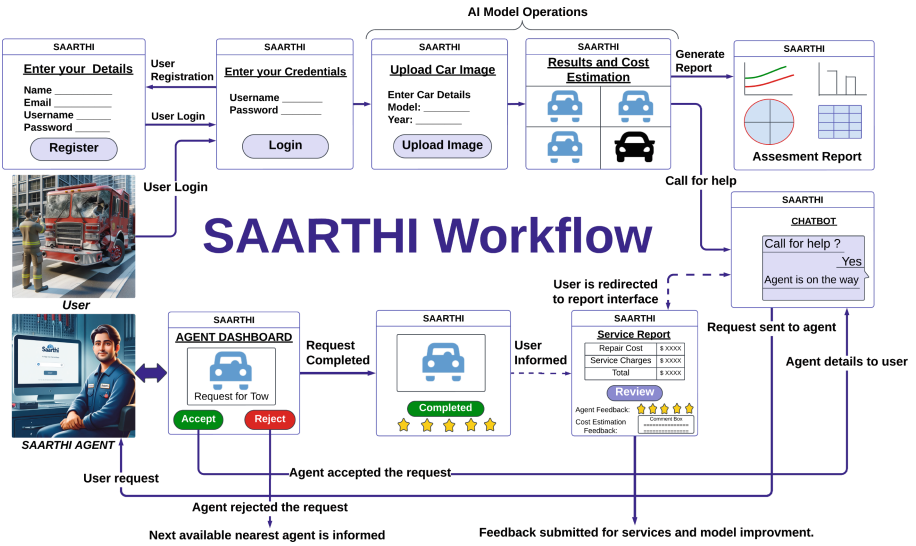


Fig. 3. SAARTHI Workflow. Please refer to Sects. 4.1–4.6 for more details

4.1 User Registration and Login

Users register on the SAARTHI platform. Once logged in, they can view their damage assessment history and past help requests on a personalized dashboard. Here they can also start a new assessment by uploading an image of the vehicle.

4.2 Image Upload and AI Operations

Users can upload an image of their damaged emergency vehicle along with optional details such as the car model and year. This image serves as the input for AI model operations, which include the following steps:

1. **Instance Segmentation and Object Detection** (Sect. 3.1): The uploaded image undergoes instance segmentation and object detection to identify and classify different types of damage on the vehicle, including dent, scratch, crack, lamp broken, glass shatter, and tire flat.

2. **Non-Maximum Suppression (NMS)** (Sect. 3.2): NMS is applied to eliminate redundant bounding boxes, ensuring each type of damage is represented only once.
3. **Cost Estimation** (Sect. 3.3): The system estimates the repair cost based on the identified damages, using predefined base costs adjusted by the area and confidence score of each detected damage.

The assessment allows users to view the original image, the damage-detected image, the image after applying NMS, and the result of SOD. The assessment report includes statistical graphs, providing a comprehensive understanding of the damages. The report also includes a table of estimated repair costs with labels for individual damages added in the total estimated repair cost.

4.3 Request Handling and Agent Coordination

Users can initiate a chatbot conversation for further assistance. Developed using JavaScript and the BotUI library [21], the chatbot offers options such as creating and downloading a PDF of the assessment report and connecting with the nearest SAARTHI agent based on the users location. Users can request a tow or on-the-spot repairs. The system uses integrated Google Maps API [22] to fetch the users location and create a request on the SAARTHI agent dashboard.

4.4 SAARTHI Agent

SAARTHI agents are emergency vehicle repair experts who have their own accounts on the SAARTHI portal. They are available 24/7 to provide assistance at any time of the day. Agents are registered on the portal by an administrator after verifying their identity and credentials. During registration, the agents details including shop location (latitude and longitude), city, phone number and personal informations are collected. Each agent is assigned a unique ID to facilitate tracking and coordination.

4.5 Nearest Agent Selection

When a user requests assistance, the system identifies the nearest available agent based on distance as follows:

1. **Calculate Distance**: Distance is calculated by Haversine formula [23,24]:

$$d = 2r \cdot \arctan 2\left(\sqrt{a}, \sqrt{1-a}\right) \tag{3}$$

where $a = \sin^2\left(\frac{\Delta\phi}{2}\right) + \cos(\phi_1) \cdot \cos(\phi_2) \cdot \sin^2\left(\frac{\Delta\lambda}{2}\right)$ and $\Delta\phi$ and $\Delta\lambda$ are the differences in latitude and longitude, and r is the Earth's radius.
2. **Sort Agents**: Agents are sorted based on their distance from the user. The closest agent is contacted first.
3. **Handle Availability**: If an agent rejects the request or is at full capacity, the system moves to the next closest agent.

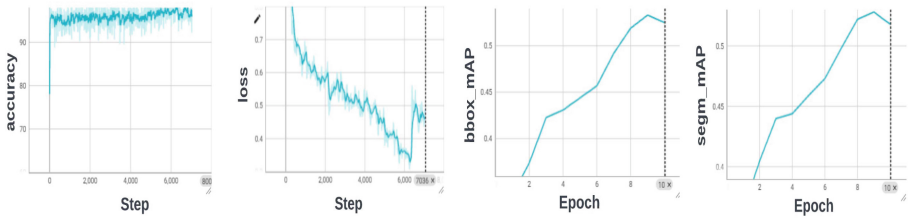
4.6 Report

Once the repair is completed, the user is notified and the detailed cost breakdown, including repair costs and service charges, is presented for review and then added to the monthly report of the user's organization.

5 Results

5.1 Object Detection and Instance Segmentation

For this research we used CarDD dataset [20]. Models were trained using an NVIDIA Tesla T4 GPU with a batch size of 4 for 10 epochs. The learning rate was set to 8e-05, using the Adam optimizer with a weight decay of 0.05. The loss functions included RPN classification loss, RPN bounding box regression loss, main classification loss (Cross-Entropy Loss), main bounding box regression loss (L1 Loss), and mask prediction loss (Mask Loss). The total loss, which is the sum of these individual losses, provided a comprehensive measure of the model's performance across different stages of object detection and segmentation. Figure 4 presents key metrics evaluating the model's performance per batch during training, with a batch size of 4.



(a) Accuracy (b) Loss (c) bbox mAP (d) seg. mAP

Fig. 4. Training Phase Result Metrics. Accuracy (Fig. 4a) shows a consistent upward trend, indicating improving model performance. The loss (Fig. 4b) declines steadily, reflecting increasingly accurate predictions and convergence. Bounding Box Mean Average Precision (bbox mAP) (Fig. 4c) indicates improving object detection performance, with steady increases in precision and recall. Segmentation Mean Average Precision (seg. mAP) (Fig. 4d) shows consistent improvement in identifying and segmenting objects

Testing Phase. For the testing phase of the SAARTHI framework we evaluated the model's performance using Mean Average Precision (mAP) and Intersection over Union (IoU) metrics. mAP measures the average precision across different classes, indicating better model accuracy with higher values. IoU measures the overlap between predicted and ground truth bounding boxes, with specific thresholds (e.g., 0.5, 0.75) determining correct predictions.

The overall bounding box mAP (bbox mAP) was 0.538 (Table 1), with high precision at IoU thresholds of 0.5 and 0.75. The model showed good accuracy for large, medium, and small objects. For segmentation masks (seg. mAP), the overall mAP was 0.519 (Table 2), also demonstrating high precision at IoU thresholds of 0.5 and 0.75, and good performance across different object sizes. Figures 6 and 8 illustrate the model's performance with examples of original images and damage detection outputs. These results indicate that the model performs well, particularly for larger objects and at an IoU threshold of 0.5. Figure 5 represents results of damage detection and segmentation for an input image.

Table 1. bbox mAP

mAP	Value
Overall	0.538
IoU 0.5	0.746
IoU 0.75	0.560
mAP Large	0.537
mAP Medium	0.309
mAP Small	0.314

Table 2. seg. mAP

mAP	Value
Overall	0.519
IoU 0.5	0.716
IoU 0.75	0.525
mAP Large	0.545
mAP Medium	0.236
mAP Small	0.171

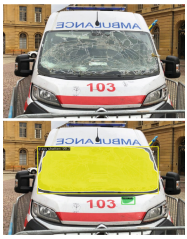
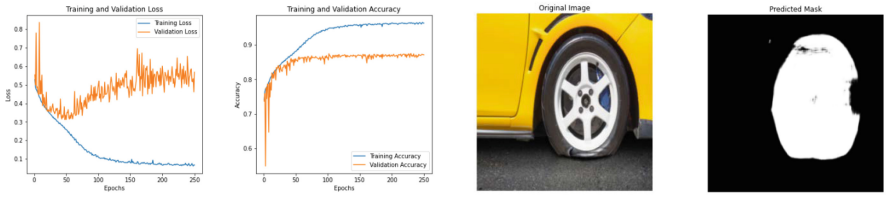


Fig. 5. Results

5.2 Salient Object Detection

Table 3 summarizes the model's performance for salient object detection during testing. The average loss of 0.5826 indicates a good match between predicted and actual values. The model achieves an accuracy of 0.8645, reflecting a high proportion of correct predictions. Precision and recall, both at 0.87, suggest the model effectively identifies true positives with balanced accuracy. The F1-score of 0.87 confirms the overall robust performance of the model in detecting salient objects. Figure 6 represents training phase results (loss, accuracy) and predicted segmentation mask for an input image.



(a) Training Loss (b) Training Accu. (c) Input Image (d) Predicted Mask

Fig. 6. SOD Training phase result 6a 6b and sample input image results 6c 6d

Table 3. Salient Object Detection results metrics (Testing Phase)

Average Loss	Accuracy	Precision	Recall	F1-score
0.5826	0.8645	0.87	0.87	0.87

5.3 SAARTHI User Interface

Figure 7 represents a few snippets of the end-to-end SAARTHI user interface. For more details, demo and code, please refer to sites.google.com/view/saarthi-home

Fig. 7. SAARTHI User Interface

6 Conclusion

The SAARTHI framework effectively addresses the challenges of emergency vehicle accidents by utilizing advanced AI technologies for damage assessment, standardized repair cost estimations, and expedited vehicle recovery. This framework reduces downtime and enhances operational readiness through real-time assessment, detailed reporting, and chatbot assistance. While the focus is on external damages visible in images, future work will integrate telematics data and sensor readings for a holistic approach, including internal damages.

Integrating SAARTHI with IoT technologies for real-time monitoring and predictive maintenance will further improve emergency vehicle readiness and efficiency.

References

1. Sanddal, T., Sanddal, N., Ward, N., Stanley, L.: Ambulance crash characteristics in the us defined by the popular press: a retrospective analysis. Emerg. Med. Int. **2010**, 525979 (2010). https://doi.org/10.1155/2010/525979
2. Donoughe, K., Whitestone, J.J., Gabler, H.C.: Analysis of firetruck crashes and associated firefighter injuries in the united states. In: Annals of Advances in Automotive Medicine. Association for the Advancement of Automotive Medicine. Annual Scientific Conference, vol. 56, pp. 69–76 (2012). https://api.semanticscholar.org/CorpusID:17994834

3. Rivara, F., Mack, C.: Motor vehicle crash deaths related to police pursuits in the United States. Inj. Prev. **10**, 93–5 (2004). https://doi.org/10.1136/ip.2003.004853
4. Miller, T., Bhattacharya, S., Zaloshnja, E., Taylor, D., Bahar, G., David, J.: Costs of crashes to government, United States. In: 2008, Annals of Advances in Automotive Medicine/Annual Scientific Conference ... Association for the Advancement of Automotive Medicine. Association for the Advancement of Automotive Medicine. Scientific Conference, vol. 55, pp. 347–355 (2011)
5. He, K., Zhang, X., et al.: Deep residual learning for image recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE (2016). https://doi.org/10.1109/cvpr.2016.90
6. He, K., Gkioxari, G., DollÃr, P., Girshick, R.: Mask r-cnn (2017). arXiv:1703.06870
7. Chen, K., et al.: Mmdetection: open mmlab detection toolbox and benchmark (2019). arXiv:1906.07155
8. Bodla, N., Singh, B., Chellappa, R., Davis, L.S.: Soft-nms – improving object detection with one line of code. In: 2017 IEEE International Conference on Computer Vision (ICCV). IEEE (2017). https://doi.org/10.1109/iccv.2017.593
9. Hosang, J., et al.: Learning non-maximum suppression. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE (2017). https://doi.org/10.1109/cvpr.2017.685
10. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation (2014). arXiv:1311.2524
11. Hafiz, A.M., Bhat, G.M.: A survey on instance segmentation: state of the art. Int. J. Multimedia Inf. Retr. **9**(3), 171–189 (2020). https://doi.org/10.1007/s13735-020-00195-x
12. Wang, X., et al.: Cardd: a new dataset for vision-based car damage detection. IEEE Trans. Intell. Transport. Syst. **24**(7), 7202–7214 (2023). https://doi.org/10.1109/tits.2023.3258480
13. Author, B.: Vehicle damage detection and analysis using convolutional neural networks. IEEE Trans. Intell. Transport. Syst. (2020). https://ieeexplore.ieee.org/document/9752971
14. Author, C.: Damage classification in vehicles using deep learning techniques. IEEE Trans. Veh. Technol. (2021). https://ieeexplore.ieee.org/document/10105039
15. Redmon, J., Farhadi, A.: Yolov3: an incremental improvement. arXiv preprint arXiv:1704.04503 (2018)
16. Zhou, Q., et al.: An automatic surface defect inspection system for automobiles using machine vision methods. Sensors **19**, 644 (2019). https://doi.org/10.3390/s19030644
17. MMlabs and mmdetection contributors. MMdetection documentation (2024). https://mmdetection.readthedocs.io/en/latest/
18. Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft COCO: common objects in context. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8693, pp. 740–755. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10602-1_48
19. Ronneberger, O., Fischer, P., Brox, T.: U-Net: convolutional networks for biomedical image segmentation. In: Navab, N., Hornegger, J., Wells, W.M., Frangi, A.F. (eds.) MICCAI 2015. LNCS, vol. 9351, pp. 234–241. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24574-4_28
20. Wang, X., Li, W., Wu, Z.: CarDD dataset. https://drive.google.com/file/d/1bbyqVCKZX5Ur5Zg-uKj0jD0maWAVeOLx/view?pli=1
21. BotUI contributors, BotUI—A JavaScript framework to create conversational UIs (2024). https://botui.org/docs

22. Google Developers, Google Maps API (2024). Accessed 28 May 2024. https://developers.google.com/maps/documentation
23. Robusto, C.C.: Haversine formula. Am. Math. Monthly **64**(1), 57–59 (1957)
24. Nichat, M.: Landmark based shortest path detection by using dijkestra algorithm and haversine formula. Int. J. Eng. Res. Appl. (IJERA) **3**(3), 162 (2013). ISSN: 2248-9622