



# Multi-view damage inspection using single-view damage projection

R. E. van Ruitenbeek<sup>1</sup> · S. Bhulai<sup>1</sup>

Received: 17 July 2021 / Revised: 6 December 2021 / Accepted: 28 February 2022 / Published online: 11 April 2022  
© The Author(s) 2022

## Abstract

Single-view computer vision models for vehicle damage inspection often suffer from strong light reflections. To resolve this, multiple images under various viewpoints can be used. However, multiple views increase the complexity as multi-view training data, specialized models, and damage re-identification over different views are required. In addition, traditional point cloud applications require large computational power, being impractical for edge computing. Therefore, multi-view damage inspection has not yet found its way into practical applications. We present a novel approach that projects the results from widely available single-view computer vision models onto 3D representations, to combine the detections from various viewpoints. With this, we leverage all advantages of multi-view damage inspection, without the need for multi-view training data and specialized models or hardware. We conduct a practical evaluation using a drive-through camera setup, to show the applicability of the methods in practice. We show that our proposed method successfully combines similar damages across viewpoints, reducing the number of duplicate damages by almost 99%. In addition, we show that our approach reduces the number of false positives by 96%. The proposed method leverages the existing single-view training data and single-view deep learning models to make multi-view inspection more accessible for practical implementations.

**Keywords** Damage detection · Inspection · Multi-view · 3D models · Ray tracing · Vehicles

## 1 Introduction

Human decision-making often relies on various visual viewpoints [1]. Despite this, most computer vision models and visual inspection systems are still relying on a single viewpoint [2]. Seeland and Mäder [1], Van Ruitenbeek [3], Carrasco and Mery [4], and Wang et al. [5] point out that visual damage inspection is strongly influenced by the camera angle and light reflection. As a result, accurate localization and classification of damages remain challenging. Therefore, a multi-view inspection system that mimics human decision-making could improve the detection performance. However, using multiple camera locations and viewpoints increases the complexity as damages can be identified multiple times. Therefore, an identification of similar detections between different viewpoints is required.

Both Stent et al. [6] and Tang et al. [7] show that fragmenting original images into smaller-sized tiles increases the recall on small objects. Although they used multiple fragments, they did not include multiple viewpoints. The effect of multi-view image classification or object localization is already slightly touched upon in previous research, showing that the use of multiple viewpoints can significantly improve both the classification and localization accuracy [8–10].

Although multi-view damage detection can result in improvements for both precision and recall, Nassar et al. [11] point out that alignment of detections over different images can be challenging, especially for situations, such as assembly lines, where the object of interest moves along the cameras. The overlap between different viewpoints, as well as the moving object within a static environment, complicates the widely implemented image stitching approaches [11]. Alternative approaches use point clouds for damage inspection [12,13]. Point clouds have the benefit to firstly combine the multiple viewpoints to a point cloud, to eventually detect the objects or damages in 3D. The construction of point clouds from a multi-view camera setup is widely studied [13–17]. However, construction of these point clouds is

---

✉ R. E. van Ruitenbeek  
revanruitenbeek@gmail.com

S. Bhulai  
s.bhulai@vu.nl

<sup>1</sup> Department of Mathematics, Vrije Universiteit Amsterdam,  
1111 De Boelelaan, 1081 HV Amsterdam, The Netherlands

costly on high resolutions. This makes point clouds less suitable for the identification of small damages.

We propose the use of ray tracing to construct a fast multi-view detection system. Our proposed solution projects damages from each individual viewpoint onto a 3D vehicle representation. With this, each face in the 3D representation receives an associated class probability from the single-view detection model. The final multi-view prediction is obtained by directly weighting the class probabilities on the 3D representation. Therefore, our novel approach solely requires a single-view CNN and a 3D representation of the object. With this, our method largely benefits from the extensively researched and optimized single-view object detection models of the last decade. While focusing on vehicle inspection, our generic approach can transform any single-view inspection model into a multi-view inspection system without the need to modify the existing model or underlying training data.

Our proposed method enables the re-identification of similar damages without requiring a minimum amount of overlapping context between the individual views. Even more, it does not require the vehicle of interest to be captured in the same environment by each camera, as our approach is independent of surroundings and solely requires the location and orientation for both the camera and vehicle.

We contribute to previous research in three ways. Firstly, we implement a single-view inspection system and quantify how our single view CNN benefits from improved label consistency, by revising labels using multiple annotation rounds. Secondly, we present a uniform way to combine inspections from different viewpoints, without the need for multi-view training data or hardware for depth estimation, as used by Kim et al. [16], Liu et al. [18], Chen et al. [19], Deng and Czarnecki [20]. Lastly, we evaluate our approach in practice, on more than 3000 images, and show that our approach successfully merges similar damages across different viewpoints. Our approach successfully merges almost 99% of the duplicate directions across all viewpoints and is able to remove 96% percent of the false positives due to the multi-view setup.

We first describe related work in Sect. 2, followed by a detailed explanation of our proposed multi-view inspection in Sect. 3. We conduct a practical evaluation of our proposed method and present the findings in Sect. 4.

## 2 Related work

Various research focuses on multi-view object detection to increase the detection and localization accuracy. Some encode multiple viewpoints in a single CNN [7,8,10,11], whereas others combine the detections from each individual viewpoint in a later stage [2,4,21]. Alternative approaches

combine multiple viewpoints into a point cloud to perform object detection directly on a point cloud level [12–17].

### 2.1 Multi-view object detection

Tang et al. [7] applied multi-view object detection to improve detection capabilities for small objects. They segmented a single image into multiple overlapping fragments and applied either Single Shot Detector (SSD) [22] or You Only Look Once (YOLO) v2 [23] on each segment. Although they used a fragmented object detection, they did not use different camera locations or camera angles, making the merging step relatively easy.

Accounting for different viewpoints, Zhao et al. [8] proposed M-YOLO, to achieve more positioning accuracy. They detect an object from two different viewpoints, using a front and a side view. They map the two-dimensional bounding boxes onto the reconstructed three-dimensional scene, to form the three-dimensional object box. They perform the mapping using the transformation matrix between the viewpoints, in combination with the three-dimensional space coordinates of the camera.

Nassar et al. [11] developed a multi-view object detection model to accurately identify trees within the landscape. Their task differs from the object detection of Savarese and Fei-Fei [9] and Liebelt and Schmid [10], since their images are not captured in a fixed setting. They extended the SSD model of Liu et al. [22] to receive camera pose information. They developed a Geo Regression network to estimate the real-world geographical positions of the objects. Nassar et al. [11] argue that strong changes in viewpoints, light conditions, and scale complicate the detection task. Furthermore, they point out that training a multi-view object detection model requires large-scale multi-view object detection datasets, which are mostly not available.

An alternative approach is taken by Kaichi et al. [2], where a rotating table is used to capture different viewpoints for surface inspection on industrial parts. They used a fixed camera and light source location and used 3D CAD models, in combination with the predefined rotations, to track scratches and dents over different views. Rubino et al. [21] used a minimum of 2 and ideally 3 viewpoints to construct a 3D object location. They used conics instead of bounding boxes from the 2D images to formulate a quadric ellipsoid of the object location. Although they did not focus on surface inspection and solely focused on combining detections of large objects, they were able to successfully merge detections across different viewpoints.

To overcome the need for optical and geometrical parameters when combining multiple images, Carrasco and Mery [4] developed Automatic Multiple View Inspection (AMVI). Their proposed AMVI model firstly identifies defects in each image, then extracts keypoints of the object, and uses the key-

points to match the identifications across the images. They validate their proposed method on aluminum wheels, while rotating the wheel. The AMVI method takes approximately 38 s to combine the detections across 3 views, making it less applicable in real situations.

## 2.2 Object detection using point clouds

Constructing point clouds or 3D geometry from multiple viewpoints has been widely studied. Fender and Müller [24] present Velt, a uniform framework to combine multiple RGB-D viewpoints into a single point cloud. They show that multiple viewpoints can successfully be combined on both high and low resolutions. Similarly, Lemkens et al. [25] evaluate the construction of point clouds using multiple RGB-D viewpoints and specifically focused on cross-camera interference and noise influence when constructing the point cloud. They showed that a larger angle between the light source and camera increases the noise and a smaller distance between the individual cameras increases the interference.

Other research uses depth map fusion for 3D reconstructions from multiple viewpoints [26]. The proposed solution of Weder et al. [26] constructs voxelgrids on  $320 \times 240$  pixels at 15FPS using an Nvidia Titan GPU, therefore being applicable for real-time processing on low resolutions.

A variety of researchers focus solely on RGB images to overcome the need for additional depth information and, therefore, specialized cameras or sensors [27,28]. Wen et al. [27] implement a single-view CNN to predict semantic and geometry features, which are used to leverage cross-view feature pooling. With this, they achieve an impressive speed of 0.32 s to generate a single mesh from 3 viewpoints on a  $214 \times 214$  pixel resolution, using an NVIDIA Titan Xp.

Much research has been done on object detection in point clouds [13–17]. Both Li et al. [14], Feng et al. [15] focus on indoor object detection and exploit the relationship between objects to improve the 3D object detection. Kim et al. [16] use RGB-D images for mask segmentation. They generate multiple foreground masks for each located bounding box and use a corresponding point cloud to estimate the 3D location using a Support Vector Machine (SVM) formulation.

Madrigal et al. [12] presented a method for recognizing surface defects in 3D point clouds. They introduced a Model Point Feature Histogram (MPFH) to detect defects using principal components and classify the point clouds into primitives for final surface defect recognition. Their 3D surface inspection has higher discriminative capabilities than single-view surface inspection. In line with Madrigal et al. [12], Makuch and Gawronek [13] automate inspections of cooling towers using point cloud analysis. They construct high-quality point clouds with fixed 3D laser scanners with a 0.1mm accuracy. Using a sequence of algorithms, among

which PCA, a 100% recognition rate is achieved for the targeted damage types.

## 2.3 Ray tracing

Ray Casting, initially pioneered by Appel [29], enables to find the closest object on the path of a ray. His approach enabled for the first time to go beyond traditional-shaped cones and spheres and provided a way to trace rays for differently shaped objects. Foley et al. [30] extended the algorithm of Appel [29] by adding recursive ray tracing to trace reflection, refraction, and/or shadow rays emerging from each hit. This extension enables more realistic image rendering with shadows and reflections. As both ray tracing and ray casting require the calculation for ray-primitive intersection, the complexity is  $O(\text{rays} \times \text{primitives})$ . By this, a brute force ray tracing algorithm is computationally expensive. To reduce this, several algorithms have been proposed to minimize the number of triangle validations per ray.

Bentley [31] proposed a multidimensional binary search tree (kd-tree) to consequently split each dimension of the search space to construct the binary tree. The binary search algorithm requires on average  $O(\log(n))$  operations for search actions. As kd-trees enable range operations, it is especially of use for ray casting/tracing. Other approaches, considered less efficient, are the Bounding Volume Hierarchy (BVH) of Rubin and Whitted [32] and Three-Dimensional Digital Differential Analyzer (3DDDA) [33].

## 3 Methodology

Figure 1 presents the same damage using two different viewpoints, indicating the complexity of multi-view detection systems. The complexity increases with the number of viewpoints and the number of identified damages per viewpoint. To overcome this, we use single-view object detection, in combination with a 3D object representation, to generate multi-view object detection. We apply a sequence of operations to project the single-view damage locations onto the 3D object. These operations are graphically summarized in Fig. 2. We define the camera and vehicle scene representation and its mathematical formulation in Sect. 3.1. Using this formulation, the bounding boxes of the single-view damage detection (step one) are projected onto the vehicle by ray tracing in step two. Each projection is weighted with a bivariate Gaussian distribution to generate a weighted class probability onto the vehicle in step three. The multi-view detection is constructed by averaging the bivariate Gaussian distributions over all detections and taking the dominant class per face in step four. In this step, we provide both the single-class and multi-class damage predictions.

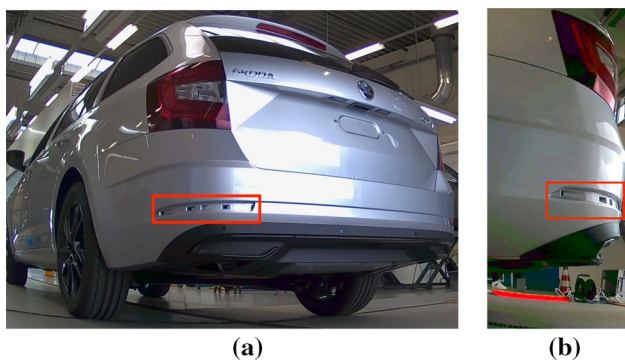


Fig. 1 Single damage from two different viewpoints

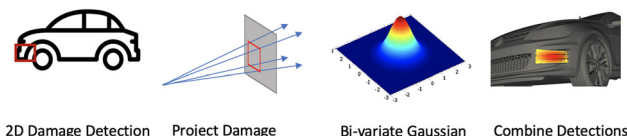


Fig. 2 Graphical representation of single view damage projection onto 3D objects

We evaluate the effectiveness of the proposed approach in a practical setting, described in Sect. 3.5.

### 3.1 Camera scene

We define the scene and mathematical formulation to be independent of the number of cameras, camera placement, camera angles, and vehicle make or model. We solely assume the location of the vehicle and cameras within the scene to be known.

We define the scene as a  $xyz$ -plane, where the vehicle is orientated in the  $z$  direction as visualized in Fig. 3a.  $\mathbf{O}_{c_i} = \langle x_{c_i}, y_{c_i}, z_{c_i} \rangle$  defines the origin of camera  $i$  within the  $xyz$ -plane and  $\Theta_{c_i} = \langle \theta_{c_i}^y, \theta_{c_i}^z \rangle$  the angle of camera  $i$  with respect to the  $y$ -axis and  $z$ -axis, respectively. Both  $\mathbf{O}_{c_i}$  and  $\Theta_{c_i}$  are of major importance for the proposed multi-view inspection and can be obtained at installation of the cameras.

The camera angle encoding is graphically presented in Fig. 3b.

### 3.2 Single-view damage detection

Our approach aims to provide a fast and scalable multi-view inspection system, without the requirement of extensive multi-view training data or multi-view computer vision models. We aim to achieve this by leveraging existing single-view object detection models and combining the individual detections afterward. This approach removes the need for advanced hardware such as lidar sensors to estimate depth in images, as implemented by Kim et al. [18], Liu et al. [16], Chen et al. [19], and Deng and Czarnecki [20]. We use YOLOv5l, developed by Jocher et al. [34], for the single-view

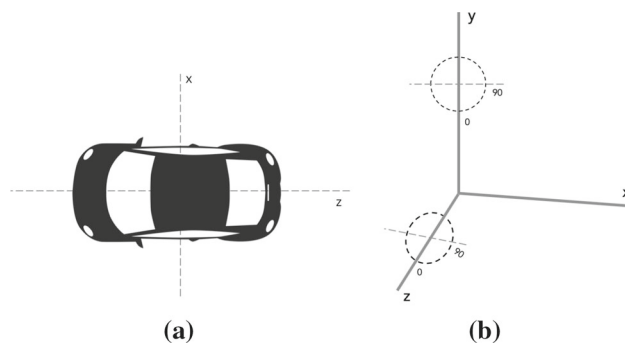


Fig. 3 Scene representation with vehicle orientation in  $xy$ -plane (a) and angle encoding in  $xyz$ -plane (b)

damage detection. Although we outline the proposed method using object detection, our method can easily be adapted to instance segmentation models.

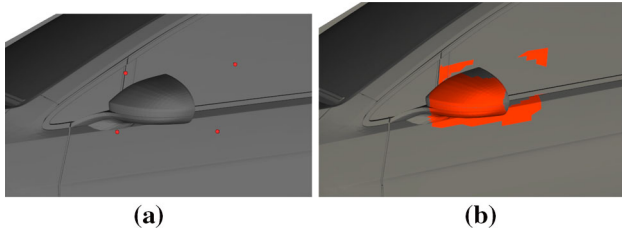
We aim to construct a fast multi-view object detection algorithm, which can be executed on the edge with either GPU or CPU devices. To achieve this, we avoid the computationally expensive point clouds constructions. In line with this, we leverage bounding box models instead of instance segmentation model to reduce the computational complexity. To date, the state-of-the-art object detection model (YOLOv5) achieves 50.1  $AP_{box}$  with 218.8 GFLOPS and 167 FPS (V100), whereas the state-of-the-art instance segmentation model (SOLO V2) achieves 41.4  $AP_{box}$  with 10 FPS (V100) [34,35].

### 3.3 Bounding box projection

We denote the damage locations, obtained from the single-view damage detection model, by  $\mathbf{d}_{ijk} = \langle x_1, y_1, x_2, y_2 \rangle$ , with damage  $k$  of camera  $i$  in iteration  $j$ . Iteration  $j$  defines the capture count of the camera over time. We encode the coordinates as the top left and right bottom corners of the bounding box and use  $\Delta \mathbf{d}_{ijk} = \langle \Delta d_{ijk}^x, \Delta d_{ijk}^y \rangle = \langle x_2 - x_1, y_2 - y_1 \rangle$  for its dimensions. The image dimensions are represented by  $\mathbf{D}_{c_i} = \langle H_{c_i}, W_{c_i} \rangle$ , where we assume a constant image dimension over different iterations, but not necessarily across different cameras. Similarly, we define the camera viewing angle by  $\alpha_{c_i} = \langle \alpha_{c_i}^y, \alpha_{c_i}^z \rangle$ . The camera viewing angle depends on the camera type, size, and focal length of the lens used within the camera. Therefore, this variable is determined upfront since it is specified by the manufacturer.

The use of rectangular boxes adds a redundancy around the location of interest. This results in lower localization accuracy of the projected area. Figure 4a illustrates the incorrect projection when solely using bounding box coordinates. Wang et al. [36], Li et al. [37], and Liu et al. [38] tried to overcome this redundancy of bounding boxes by encoding the rotation angle in the prediction. Despite this rotation angle,





**Fig. 4** Projection 2D damage location on a 3D vehicle with incorrect prediction when using bounding box coordinates (a) and improved localization when using all pixels within the bounding box (b)

the projection is still affected when projecting on outward facing parts of the vehicle, such as mirrors.

To overcome this, we project the bounding box onto the vehicle by ray tracing through each individual pixel within the bounding box. This approach ensures that the full location can be extrapolated, as displayed in Fig. 4b. The downside of this approach is an increased number of rays from 4 to  $\Delta d_{ijk}^x \cdot \Delta d_{ijk}^y$ . This implementation is further described in Sect. 3.3.1. To further reduce the effect of redundancy around the damage object, we weight the area within the bounding box with a bivariate Gaussian distribution to decrease the class probability as the distance to the bounding box center increases, which is further described in Sect. 3.3.2.

### 3.3.1 Ray tracing projection

We project the bounding box from the single-view image onto the 3D representation of the vehicle. This step is performed using ray tracing, where the ray angles are obtained from two known components. Firstly, we obtain the angle of each pixel with respect to the camera center. Secondly, the camera angles with respect to the  $xyz$ -plane are added to the ray directions.

We index all pixels within the bounding box  $\mathbf{d}_{ijk}$  and define this matrix as  $\mathbf{M}_{ijk}$  (Eq. 1). The distance of each pixel with respect to the image center is then expressed by Eq. 2. The first term normalizes the pixel coordinates of  $\mathbf{M}_{ijk}$  with respect to the bounding box dimension, where subtracting 0.5 shifts the normalized pixels to the origin. The angles of the pixels with respect to the image center are subsequently obtained by multiplying the normalized pixel coordinates with the camera viewing angle ( $\alpha_i$ ) in Eq. 3.

$$\mathbf{M}_{ijk} = \begin{bmatrix} x_u \\ y_v \end{bmatrix}_{u=x_1, x_1+1, \dots, x_2, v=y_1, y_1+1, \dots, y_2}^T, \tag{1}$$

$$\tilde{\mathbf{M}}_{ijk} = \frac{\mathbf{M}_{ijk}}{\Delta \mathbf{d}_{ijk}} - \frac{1}{2}, \tag{2}$$

$$\tilde{\mathbf{M}}_{ijk}^\theta = \tilde{\mathbf{M}}_{ijk} \cdot \alpha_i. \tag{3}$$

In the second step, the pixel angles within the  $xyz$ -plane are obtained by adding the camera orientation ( $\Theta_{c_i}$ ). This results in the angle matrix  $\tilde{\mathbf{M}}_{ijk}^{\Theta_{xyz}}$ , consisting of a single ray angle per pixel. This final computation is given in Eq. 4, leading to the ray directions.

$$\tilde{\mathbf{M}}_{ijk}^{\Theta_{xyz}} = \tilde{\mathbf{M}}_{ijk}^\theta + \Theta_{c_i}. \tag{4}$$

Using ray tracing, we shoot rays under all angles of  $\tilde{\mathbf{M}}_{ijk}^{\Theta_{xyz}}$ . Fortunately, Foley et al. [30] developed a real-time ray tracing algorithm to determine the ray-triangle intersection point. Using this, we obtain the face matrix  $\mathbf{F}_{ijk}$ , containing a 1 if ray  $r$  hits face  $l$ , and 0 otherwise. We represent  $\mathbf{F}_{ijk}$  as a  $n_f \times n_r$  matrix with  $n_f$  the number of faces and  $n_r$  the number of rays.

### 3.3.2 Probability map

We weight each bounding box projection with a bivariate Gaussian distribution to increase the weight given to the center of the box. We combine the projected bounding boxes from different cameras and iterations and weight each detection with its class probability. We identify the damaged vehicle areas on the 3D representation by computing the class probability  $p^\gamma$  per face  $l$  using Eq. 5.

Each ray is weighted by the bivariate Gaussian probability density function, given in Eq. 6, with  $\mathbf{x}$  the normalized pixel distances toward the center ( $\tilde{\mathbf{M}}_{ijk}$ ). This results in the intensity matrix  $\mathbf{F}_{ijk1 \times n_r}$  which we expand to  $n_f \times n_r$  to eventually multiply using the Hadamard product with  $\mathbf{F}_{ijk}$ . As a result, the numerator defines the damage intensity obtained from ray  $r$  on face  $l$ .

The number of rays that hit a 3D face strongly depends on the size of the face. To compensate for this, we average over the number of ray hits per face in the denominator. Based on this, each face receives the average hit intensity from all rays that project onto the face. Figure 5 shows the resulting heatmap projection for two different damages, based on Eq. 5.

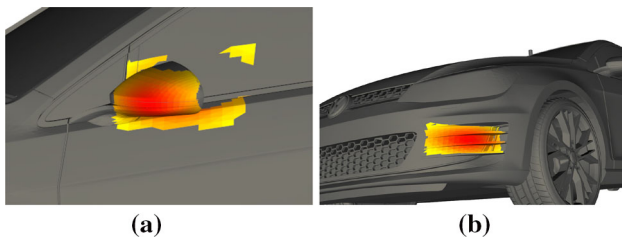
$$p_{ijk}^\gamma = \frac{\mathbf{1}_{n_f \times 1} \mathbf{F}_{ijk} \circ f(\tilde{\mathbf{M}}_{ijk}) \mathbf{1}_{n_f \times 1}}{\mathbf{F}_{ijk} \mathbf{1}_{n_f \times 1}}, \tag{5}$$

$$f(\mathbf{x}) = \frac{1}{2\pi} |\Sigma|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \mu)^T \Sigma^{-1}(\mathbf{x} - \mu)\right), \tag{6}$$

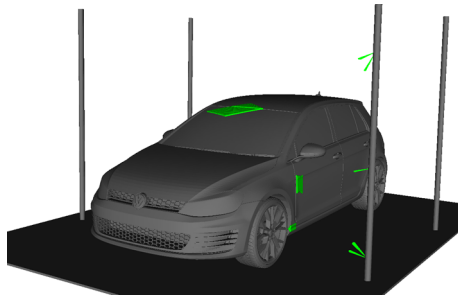
$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \quad \mu = \begin{bmatrix} \mu_{x_1} \\ \mu_{x_2} \end{bmatrix}, \quad \Sigma = \begin{bmatrix} \sigma_{x_1}^2 & \rho \sigma_{x_1} \sigma_{x_2} \\ \rho \sigma_{x_1} \sigma_{x_2} & \sigma_{x_2}^2 \end{bmatrix}.$$

### 3.4 Combine multiple views

We combine multiple camera viewpoints to move from single-view damage detection to multi-view damage detec-



**Fig. 5** Damage projection with ray tracing on each pixel, weighted by a heatmap probability. Damage located on left mirror (a) and left front bumper (b)



**Fig. 6** Damage projection from three viewpoints with rays departing from camera

tion. A non-max suppression is not suitable as it does not account for the irregular shapes of the 3D vehicle representation. Fortunately, we can directly apply Eq. 5 on all damages from the single-view model for all available viewpoints.

A bias arises toward false negative detections from the single-view model. To overcome this bias, we average the detection probability ( $p_{ijk}^\gamma$ ) over the number of cameras that have a non-blocking view on the damaged face. We denote the number of cameras that could have seen the damaged face  $l$  by  $t_l$ . We obtain the vector  $\mathbf{t}$  by projecting a single ray to the face from each camera and counting the number of hits on the face. Using this, we define the multi-view class prediction by  $\tilde{p}_l^\gamma$ , obtained by Eq. 7 (Fig. 6).

$$\tilde{p}_l^\gamma = \frac{1}{t} \sum_{ijk} p_{ijk}^\gamma \tag{7}$$

The final dominant class prediction for face  $l$  ( $\gamma_l$ ) is then defined by Eq. 8, where the background class is set to 0. For multi-class prediction, the classes are solely defined by  $\{p_l^\gamma \mid p_l^\gamma > \tau\}$ . Redundant predictions are easily removed by the threshold variable  $\tau$ , which serves as a confidence threshold. In other words, if the damage does not surpass threshold  $\tau$ , the damage is market by Eq. 8 as background.

$$\gamma_l = \begin{cases} \arg \max_{\gamma} p_l^\gamma & ; \{p_l^\gamma \mid p_l^\gamma > \tau\} \neq \emptyset, \\ 0 & ; \{p_l^\gamma \mid p_l^\gamma > \tau\} = \emptyset. \end{cases} \tag{8}$$

### 3.5 Practical evaluation

We construct a camera gate to evaluate the proposed solution in practice. We configure the camera gate and scene by manually selecting  $\mathbf{O}_{c_i}$  and  $\Theta_{c_i}$ . We place three cameras at 20, 100, and 200 centimeters from the ground surface and place these cameras at the following locations:  $(-175, -100)$ ,  $(-175, 100)$ ,  $(175, -100)$ , and  $(175, 100)$ . The selected cameras are from the same manufacturer, which reports the viewing angle  $\alpha_{c_i} = \langle 58, 80 \rangle$  and the resolution  $\mathbf{D}_{c_i} = \langle 1920, 2560 \rangle$ .

We compared our multi-view approach with a single-view inspection process. The comparison is conducted using 50 vehicles, 12 cameras, and 5 captures per camera. Each vehicle is captured while driving through the gate, with a speed of approximately 5 kilometers an hour. With this, each vehicle scan consists of exactly 60 images. Therefore, we conduct the evaluation on 50 vehicle scans with a total sample size of 3000 images.

As described in Sect. 3.2, YOLOv5 [34] is used to train the single-view object detection model. A total of 42,313 images are used for the training of the single-view object detection model. These images are captured by various users, with various mobile phones in different settings. This ensures that the dataset contains different camera angles, image resolutions, vehicle types, vehicle colors, and lightning and background conditions. This results in a diverse dataset, to ensure that the single-view model is not solely trained on a single camera setup. Therefore, the single-view model is more robust against different camera angles and light conditions, when implemented in the multi-view approach.

A total of 42,313 images are manually and disjoint labeled by four individuals in a first annotation round. All annotations are validated in a second annotation round by shuffling the label sets over different individuals. We use this extensive approach in an attempt to improve label quality and consistency. During the second annotation round, a total of 26.2% of the images received a change for at least one of the labels (e.g., remove/add bounding box or change of bounding box shape). To validate if the consistency improved, we conducted an additional round with a randomized subset of 1000 images, resulting in a significantly lower change rate of 2.8%.

The full labeling process results in a total of 119,701 bounding boxes. We used transfer learning from the COCO trained weights and fine-tuned the detection head of YOLOv5 for 100 epochs. The full network is then trained for an additional 50 epochs. We make use of Stochastic Gradient Descent, image size 640, Mosaic augmentation, and apply a grid search for  $\{(lr, bs); lr \in \{1e^{-4}, 1e^{-3}, 1e^{-2}, 1e^{-1}\}, bs \in \{8, 16, 32, 64\}\}$ . The learning rate, momentum, image scale, rotation, sheering, and HSV Saturation are optimized using Bayesian Optimization.

We apply the Gaussian heatmap using  $\mu_1 = \mu_2 = 0$ ,  $\sigma_1 = \sigma_2 = 5$ , and  $\rho = 1$ . We use a relatively high  $\sigma$ , to put slightly more attention on the bounding box center compared with the corners. To enable fast ray tracing, we leverage the latest<sup>1</sup> ray tracing implementation of Parker et al. [39].

## 4 Results

We first outline the performance for single-view damage detection and present the performance improvement that is achieved from the label revision. Secondly, we draw the comparison between the single-view damage detection approach and our proposed multi-view damage detection method.

### 4.1 Single-view damage detection

Table 1 presents the damage detection performance after parameter optimization. The performed hyperparameter optimization yields  $lr = 1e^{-3}$ ,  $bs = 32$ ,  $momentum = 0.89$ ,  $image\ scale = \pm 40\%$ ,  $rotation = \pm 20^\circ$ , and  $shearing = \pm 5^\circ$ . The results show that an improved label consistency from the second annotation round reflects in a strong performance improvement. The recall and precision improve with almost 23% and 12%, respectively. The annotation revision does not solely improve the precision and recall, but also improves the alignment of the bounding box. Figure 7 presents an excerpt of the label revision, where 2 false positives are removed and the alignment of one bounding box is improved.

Table 2 presents the confusion matrix for single-view damage inspection. A total of 3000 images of 50 unique vehicles are used in the evaluation. The test set contains 931 damages, corresponding to 224 unique damages (e.g., some images are captured multiple times under different angles). The presented results seem quite fair at first sight, but the underlying problem directly arises when the individual detections are aggregated per unique damage. Table 3 shows that 34 unique damages (15%) have not been identified on any of the views. Furthermore, it points out that 73%<sup>2</sup> of the unique damages are identified more than once. Based on the used test set, a human operator will receive each damage on average 3.5 times. These duplicate identifications directly affect the applicability in real-world situations, since duplicates have to be removed manually.

### 4.2 Multi-view damage detection

Table 3 shows that the single-view approach contains many damages which are identified more than once. A strong



**Fig. 7** Excerpt of a modified class label with iteration 1 (a) and iteration 2 (b). 2 removed false positives and 1 revised bounding box dimension

**Table 2** Single-view confusion matrix for 3000 Ultra HD images of 50 unique vehicles, under confidence threshold 0.2

	Actual	
	Damaged	Undamaged
Prediction		
Damaged	773	312
Undamaged	158	–

A total of 931 damages are present in the ground truth test dataset. The number of damages per image ranges between 0 and 3

reduction in duplicate identifications is achieved using the multi-view approach, where only 8 damages are marked as two separate damages, while belonging to the same unique damage. In other words, our proposed method enables us to reduce the number of duplicate identifications from 576<sup>3</sup> to 8. This strong reduction in duplicate detections is achieved since the single-view model is not able to identify if the damage in the first viewpoint is equivalent with the damage in the second viewpoint. Furthermore, Table 4 shows that the number of false positives is reduced by 79% for the multi-view approach, compared with the single-view approach. This reduction is again obtained from the ability to share information across the different viewpoints. We previously argued that the viewpoint matters for the detection of damages, since light reflections can result in false positives. Applying the multi-view model enables to evaluate and average the damage from multiple viewpoints, according to Eq. 7. This approach reduces the number of false positives notably, since a false positive from one viewpoint is outweighed by the other viewpoints.

Table 3 shows that a lower confidence threshold reduces the number of false negatives (0 identifications) for the multi-view approach, while keeping the property to remove almost all duplicate identifications. In addition, a lower confidence threshold reduces the number of false negatives at a slightly increased number of false positives. The effect of a lower confidence threshold on the number of false positives is presented in Table 4. Therefore, we can state that our approach retains

<sup>1</sup> Version 7.0.

<sup>2</sup>  $(224 - 34 - 27)/224$ .

<sup>3</sup> 773 identifications - 197 unique detections.

**Table 1** Performance comparison of single-view damage detection, before and after label revision

Iteration	Algorithm	$AP_{5:95}$ (%)	$AP_{.5}$ (%)	Precision (%)	Recall (%)
1	YOLOv5l	16.8	31.9	28.09	40.16
2	YOLOv5l	23.4	38.5	31.36	49.32

**Table 3** Number of identifications for all 224 unique damages, based on 3000 images, for varying confidence thresholds for the single-view detection model

Approach	Conf	Number of identifications									
		0	1	2	3	4	5	6	7	8	9+
Single-view	0.20	34	27	26	29	36	24	15	9	8	12
Multi-view	0.20	61	155	8	0	0	0	0	0	0	0
Multi-view	0.15	33	183	8	0	0	0	0	0	0	0
Multi-view	0.10	27	187	10	0	0	0	0	0	0	0
Multi-view	0.05	26	187	11	0	0	0	0	0	0	0

**Table 4** Number of false positives, based on 224 unique damages with 3000 images

Conf	0.2	0.15	0.10	0.05
Single-view	81			
Multi-view	17	26	39	44

the same properties as the single-view approach in terms of the false positive/false negative trade-off, while being able to remove almost all duplicate detections.

It has to be noted that our proposed approach does not solely remove duplicate identifications from multiple views; it also directly removes almost all false positives on the surrounding of the vehicle. The trained single-view damage inspection model generates false positives on the surroundings when cracks or scratches are present in the concrete walls and floors of the setup. A total of 71 false positives have been generated on the surrounding, based on the 3000 test images. The multi-view algorithm automatically removes 68 of the false positives (96%) without the need for any additional algorithms such as vehicle segmentation, which is normally required.

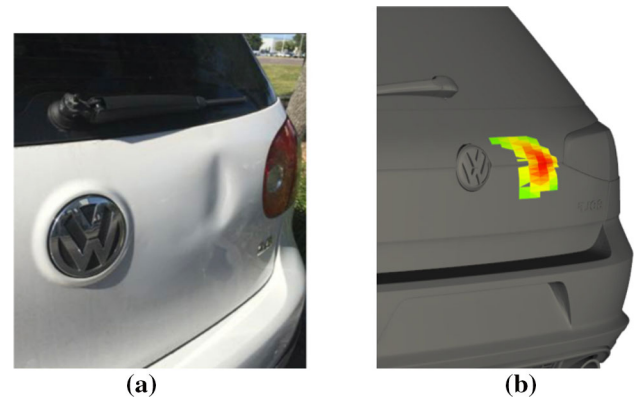
Both Figs. 8 and 9 visualize the multi-view prediction for a single dent. The heatmap predictions are combined from multiple damages, where the damage of Fig. 8a is seen by 6 out of the 6 cameras and the damage of Fig. 9a by 4 out of 5 cameras. Both visualizations are constructed with probability threshold 0.10.

### 4.3 Robustness of the proposed model

We previously presented how a change in the confidence level of the single-view model (YOLOv5l) impacts the number of duplicate detections (Table 3) and the number of false positives (Table 4). Besides this, other factors play an important role in the performance of the multi-view model. Chang-



**Fig. 8** Excerpt of a single-view (a) and associated multi-view prediction (b)



**Fig. 9** Excerpt of a single-view (a) and associated multi-view prediction (b)

ing the underlying training dataset for the single-view model largely influences the outcome of the multi-view model.

Table 5 shows that a smaller single-view training dataset directly effects the performance of the multi-view model. This results from a decreased diversity and a decreased detection performance. As a result, the number of false positives increases and the number of false negatives increases. However, the number of duplicate dictations does not increase to large extend. With the original dataset, a total of 8 damages are not successfully merged, against 18 with a dataset of 10,000 images. In addition, the number of false positives from the multi-view model with only 10,000 training images



**Table 5** Effect of the training dataset size for the single-view damage detection model on the number of duplicate identifications and the number of false positives, under confidence threshold 0.2

# Images	Number of identifications						False positives
	0	1	2	3	4	5	
42,313	61	155	8	0	0	0	17
20,000	103	113	10	0	0	0	44
10,000	129	77	18	0	0	0	76

A total of 224 damages are present in the ground truth

**Table 6** Computation time complexity for both GPU and CPU implementation. Reported in seconds

Stage	GPU 50 vehicles	GPU 1 vehicle	CPU 1 vehicle
Single-view	15.69	0.31	25.8
Preprocessing	1.28	0.03	0.18
Ray tracing	43.09	0.86	27.11
Post-processing	4.03	0.08	0.84
Total time	64.09	1.28	53.93

is still lower than the single view model trained on the full dataset. Showing that our proposed model can already add value for smaller datasets.

#### 4.4 Computation time

We evaluate the computational time of our proposed solution on a Nvidia RTX 3090 GPU with an AMD 5800 CPU and 64GB memory. We break the computation time into two segments: the single-view stage and the multi-view stage. We report the total time to process the 50 vehicles (3000 images), as well as the average time per vehicle (60 images). The resulting computational time for both the GPU and the CPU implementation is presented in Table 6. The largest computational components are the single-view YOLOv5l model and the ray tracing component. The ray tracing implementation requires a relatively low number of rays to process. There are no rebounds required to find the 3D location and the rays are only computed over the pixels of the damage locations. In our practical evaluation, the single-view model predicts 1085 (773 + 312) bounding boxes, resulting in 1085 boxes for ray tracing. Despite this, the ray tracing component is still relatively large. This is mainly due to context setup required in our proposed solution. The majority of the ray tracing time is used for loading the 3D vehicle context from disk and into the render environment. Therefore, the mentioned results are an indicator for the performance which are not yet optimized.

## 5 Discussion

Our implementation uses pre-constructed 3D models, to reduce computation time. Therefore, the mirror or wiper can be differently orientated between the real situation and pre-constructed 3D models. As a result, the damage projection becomes slightly less accurate for these areas. The same occurs for different rim types, which is not taking into account in our approach. The latter can easily be resolved in further research as most vehicles have a fixed set of available rims, giving the option to render 3D models for each make, model, and rim combination.

The detail level of the 3D object affects the damage projection accuracy. More fine-tuned 3D models lead to more detailed damage projections. Contrary to this, a more detailed 3D model increases the number of faces, which increases the computational time of the ray tracing. Future work can focus on quantifying the trade-off between 3D model granularity and computational time.

## 6 Conclusion

We presented how computer vision models benefit from improved label consistency, by comparing the detection performance before and after label revision from a second annotator. We showed that both the recall and precision improve with, respectively, almost 23% and 12%.

Furthermore, we presented a novel approach to leverage existing single-view training data and single-view deep learning models for multi-view damage inspection. Our practical evaluation, conducted in a dynamic environment, shows that similar damages are successfully grouped together, therefore reducing the number of duplicate identifications by 99%. Furthermore, we showed that the multi-view approach reduces the false positive rate by more than 96%, while maintaining the same number of true positives. Even more, we show that almost all false positives on the surroundings are automatically removed, without the need for additional segmentation models. Therefore, single-view damage detection models can significantly benefit from our novel multi-view inspection methodology without requiring specialized depth hardware, multi-view deep learning architectures, or multi-view training data.

The proposed method requires relatively low computational power, as solely viewpoints with identified damage are used in the merging process. Therefore, ray tracing is solely performed on the damaged areas, making it applicable for CPU and edge devices.

With the reduced number of duplicate detections, the reduced number of false positives, and the low computational requirements, we are able to reduce the barrier for practical implementation of multi-view damage inspection systems.

**Acknowledgements** We would like to thank Pon Holdings B.V. for contributing to this research by providing a test location, hardware, and test vehicles.

**Author Contributions** REvR involved in conceptualization; methodology; formal analysis and investigation; writing—original draft preparation; writing—review and editing. SB took part in supervision.

**Funding** Not applicable.

**Availability of data and materials** Not applicable.

## Declarations

**Conflict of interest** The authors declare that they have no conflict of interest.

**Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

## References

- Seeland, M., Mäder, P.: Multi-view classification with convolutional neural networks. *PLoS ONE* **16**(1), 1–17 (2021). <https://doi.org/10.1371/journal.pone.0245230>
- Kaichi, T., Mori, S., Saito, H., Sugano, J., Adachi, H.: Multi-view surface inspection using a rotating table. *Electron. Imaging* **2018**, 1–5 (2018)
- Van Ruitenbeek, R.E.: Vehicle damage detection using deep convolutional neural networks. Vrije Universiteit Amsterdam, technical report (2019). [https://beta.vu.nl/nl/Images/stageverslag-ruitenbeek\\_tcm235-919695.pdf](https://beta.vu.nl/nl/Images/stageverslag-ruitenbeek_tcm235-919695.pdf)
- Carrasco, M., Mery, D.: Automatic multiple view inspection using geometrical tracking and feature analysis in aluminum wheels. *Mach. Vis. Appl.* **22**(1), 157–170 (2011). <https://doi.org/10.1007/s00138-010-0255-2>
- Wang, H., Xiong, Z., Finn, A.M., Chaudhry, Z.: A context-driven approach to image-based crack detection. *Mach. Vis. Appl.* **27**(7), 1103–1114 (2016). <https://doi.org/10.1007/s00138-016-0779-1>
- Stent, S., Gherardi, R., Stenger, B., Cipolla, R.: Detecting change for multi-view, long-term surface inspection. *Br. Mach. Vis. Assoc. Soc. Pattern Recognit.* **12**, 1–127 (2015)
- Tang, C., Ling, Y., Yang, X., Jin, W., Zheng, C.: Multi-view object detection based on deep learning. *Appl. Sci.* **8**(9), 1423 (2018)
- Zhao, X., Jia, H., Ni, Y.: A novel three-dimensional object detection with the modified You Only Look Once method. *Int. J. Adv. Robot. Syst.* **15**(2), 172988141876550 (2018). <https://doi.org/10.1177/1729881418765507>
- Savarese, S., Fei-Fei, L.: Multi-view object categorization and pose estimation. *Stud. Comput. Intell.* **285**, 205–231 (2010). [https://doi.org/10.1007/978-3-642-12848-6\\_8](https://doi.org/10.1007/978-3-642-12848-6_8)
- Liebelt, J., Schmid, C.: Multi-view object class detection with a 3D geometric model. In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 1688–1695 (2010)
- Nassar, A., Lefevre, S., Wegner, J.D.: Simultaneous multi-view instance detection with learned geometric soft-constraints. In: *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2019-October. Institute of Electrical and Electronics Engineers Inc., pp. 6558–6567 (2019)
- Madrigal, C.A., Branch, J.W., Restrepo, A., Mery, D.: A method for automatic surface inspection using a model-based 3D descriptor. *Sensors (Switzerland)* **17**(10), 2262 (2017)
- Makuch, M., Gawronek, P.: 3d point cloud analysis for damage detection on hyperboloid cooling tower shells. *Remote Sens.* **12**(10), 1542 (2020)
- Li, Y., Ma, L., Tan, W., Sun, C., Cao, D., Li, J.: Grnet: geometric relation network for 3d object detection from point clouds. *ISPRS J. Photogramm. Remote Sens.* **165**, 43–53 (2020)
- Feng, M., Gilani, S.Z., Wang, Y., Zhang, L., Mian, A.: Relation graph network for 3d object detection in point clouds (2019)
- Kim, B.S., Xu, S., Savarese, S.: Accurate localization of 3D objects from RGB-D data using segmentation hypotheses. In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 3182–3189 (2013)
- Vetrivel, A., Gerke, M., Kerle, N., Nex, F., Vosselman, G.: Disaster damage detection through synergistic use of deep learning and 3d point cloud features derived from very high resolution oblique aerial images, and multiple-kernel-learning. *ISPRS J. Photogramm. Remote Sens.* **140**, 45–59 (2018)
- Liu, L., Lu, J., Xu, C., Tian, Q., Zhou, J.: Deep fitting degree scoring network for monocular 3D object detection. In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2019-June, 4, pp. 1057–1066 (2019). [arXiv:1904.12681](https://arxiv.org/abs/1904.12681)
- Chen, X., Ma, H., Wan, J., Li, B., Xia, T.: Multi-view 3D object detection network for autonomous driving. In: *Proceedings of 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-Janua. Institute of Electrical and Electronics Engineers Inc., pp. 6526–6534 (2017). [arXiv:1611.07759](https://arxiv.org/abs/1611.07759)
- Deng, J., Czarniecki, K.: MLOD: A multi-view 3D object detection based on robust feature fusion method. In: *2019 IEEE Intelligent Transportation Systems Conference, ITSC 2019*. Institute of Electrical and Electronics Engineers Inc., pp. 279–284 (2019). [arXiv:1909.04163](https://arxiv.org/abs/1909.04163)
- Rubino, C., Crocco, M., Del Bue, A.: 3d object localisation from multi-view image detections. *IEEE Trans. Pattern Anal. Mach. Intell.* **40**(6), 1281–1294 (2018)
- Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., Berg, A.C.: SSD: single shot MultiBox detector. In: *European Conference on Computer Vision*, vol. 9905. Springer, pp. 21–37 (2016). [https://doi.org/10.1007/978-3-319-46448-0\\_2](https://doi.org/10.1007/978-3-319-46448-0_2)
- Redmon, J., Farhadi, A.: YOLO9000: better, faster, stronger. In: *Proceedings of 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-January. Institute of Electrical and Electronics Engineers Inc., pp. 6517–6525 (2017)
- Fender, A., Müller, J.: Velt: a framework for multi rgb-d camera systems, pp. 73–83 (2018)
- Lemkens, W., Kaur, P., Buys, K., Slaets, P., Tuytelaars, T., Schutter, J.: Multi rgb-d camera setup for generating large 3d point clouds, pp. 1092–1099 (2013)
- Weder, S., Schönberger, J.L., Pollefeys, M., Oswald, M.R.: Routefusion: learning real-time depth map fusion (2020)
- Wen, C., Zhang, Y., Li, Z., Fu, Y.: Pixel2mesh++: multi-view 3d mesh generation via deformation, pp. 1042–1051 (2019)

28. Chang, A.X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., Xiao, J., Yi, L., Yu, F.: Shapenet: an information-rich 3d model repository (2015)
29. Appel, A.: Some techniques for shading machine renderings of solids. In: Proceedings of the April 30–May 2, 1968, Spring Joint Computer Conference on - AFIPS'68 (Spring). New York, New York, USA: Association for Computing Machinery (ACM), p. 37 (1968). <http://portal.acm.org/citation.cfm?doid=1468075.1468082>
30. Foley, J.D., Foley, J.D., Whitted, T.: An improved illumination model for shaded display (1980). <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.156.1534>
31. Bentley, J.L.: Multidimensional binary search trees used for associative searching communications of the ACM, vol. 18, no. 9, pp. 509–517 (1975). <http://portal.acm.org/citation.cfm?doid=361002.361007>
32. Rubin, S.M., Whitted, T.: A 3-dimensional representation for fast rendering of complex scenes. ACM SIGGRAPH Comput. Graph. **14**(3), 110–116 (1980)
33. Fujimoto, A., Tanaka, T., Iwata, K.: Arts: accelerated ray-tracing system. IEEE Comput. Graph. Appl. **6**(4), 16–26 (1986)
34. Jocher, G., Stoken, A., Borovec, J., NanoCode012, Christopher-STAN, Changyu, L., Laughing, Tkianai, YxNONG, Hogan, A., Lorenzomamma, AlexWang1900, Chaurasia, A., Diaconu, L., Marc, Wanghaoyang0106, M15ah, Doug, Durgesh, Ingham, F., Frederik, Guilhen, Colmagro, A., Ye, H., Jacobsolawetz, Poznan-ski, J., Fang, J., Kim, J., Doan, K., L. Y.: ultralytics/yolov5: v4.0 - nn.SiLU() activations, Weights & Biases logging, PyTorch Hub integration. Jan 2021. <https://zenodo.org/record/4418161>
35. Wang, X., Zhang, R., Kong, T., Li, L., Shen, C.: SOLOv2: dynamic and fast instance segmentation (2020). [arXiv:2003.10152](https://arxiv.org/abs/2003.10152)
36. Wang, Y., Wang, L., Lu, H., He, Y.: Segmentation based rotated bounding boxes prediction and image synthesizing for object detection of high resolution aerial images. Neurocomputing **388**, 202–211 (2020)
37. Li, S., Zhang, Z., Li, B., Li, C.: Multiscale rotated bounding box-based deep learning method for detecting ship targets in remote sensing images. Sensors (Switzerland) **18**(8), 2700 (2018)
38. Liu, L., Pan, Z., Lei, B.: Learning a rotation invariant detector with rotatable bounding box. [arXiv:1711.09405](https://arxiv.org/abs/1711.09405) (2017)
39. Parker, S.G., Bigler, J., Dietrich, A., Friedrich, H., Hoberock, J., Luebke, D., McAllister, D., McGuire, M., Morley, K., Robison, A., Stich, M.: OptiX: a general purpose ray tracing engine. In: ACM SIGGRAPH 2010 Papers, SIGGRAPH 2010, vol. 29, no. 4. Association for Computing Machinery, Inc, pp. 1–13 (2010). <https://dl.acm.org/doi/10.1145/1778765.1778803>

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

**R. E. van Ruitenbeek** M.Sc. is a technical lead and co-founder at Lensor. He obtained both his B.Sc. and M.Sc. Cum Laude in Business Analytics at the Vrije Universiteit Amsterdam. His passion for research is reflected by receiving the best paper award for Business Analytics in 2019 and by his current research as a PhD student at the Vrije Universiteit Amsterdam. His research is focused on automatic damage inspection on vehicles using computer vision.

**Prof. Dr. S. Bhulai** Bhulai is a full professor of Business Analytics at Vrije Universiteit Amsterdam. He studied “Mathematics” and “Business Mathematics and Informatics” and obtained a PhD on Markov decision processes for the control of complex, high-dimensional systems. He is co-founder of the Amsterdam Center for Business Analytics (ACBA), co-founder of the postgraduate programme Business Analytics/Data Science, and also co-founder of PersonalAIze. Sandjai's research is on the interface of mathematics, computer science, and operations management. His specialization is in decision making under uncertainty, optimization, data science, and business analytics. His current research projects focus on HR analytics, social media analytics, predictive analytics, dynamic pricing, and planning and scheduling in complex systems. His passion for research is also reflected in his teaching for which he received the “Excellence in Teaching Award” for best lecturer of both the faculty and the university in 2006. Together with Prof. Dr. Rob van der Mei, he has been awarded the Huibregtsen prize 2021 for innovative scientific research with a major societal impact.