Exploiting deep learning and augmented reality in fused deposition modeling: a focus on registration

International Journal on Interactive Design and Manufacturing (IJIDeM)
https://doi.org/10.1007/s12008-022-01107-5

ORIGINAL PAPER

Exploiting deep learning and augmented reality in fused deposition modeling: a focus on registration

Leonardo Tanzi[1] · Pietro Piazzolla[1] · Sandro Moos[1] · Enrico Vezzetti[1]

Received: 28 April 2021 / Accepted: 1 November 2022

Abstract
The current study aimed to propose a Deep Learning (DL) based framework to retrieve in real-time the position and the rotation of an object in need of maintenance from live video frames only. For testing the positioning performances, we focused on intervention on a generic Fused Deposition Modeling (FDM) 3D printer maintenance. Lastly, to demonstrate a possible Augmented Reality (AR) application that can be built on top of this, we discussed a specific case study using a Prusa i3 MKS FDM printer. This method was developed using a You Only Look Once (YOLOv3) network for object detection to locate the position of the FDM 3D printer and a subsequent Rotation Convolutional Neural Network (RotationCNN), trained on a dataset of artificial images, to predict the rotations' parameters for attaching the 3D model. To train YOLOv3 we used an augmented dataset of 1653 real images, while to train the RotationCNN we utilized a dataset of 99.220 synthetic images, showing the FDM 3D Printer with different orientations, and fine-tuned it using 235 real images tagged manually. The YOLOv3 network obtained an AP (Average Precision) of 100% with Intersection Over Unit parameter of 0.5, while the RotationCNN showed a mean Geodesic Distance of 0.250 ($\sigma = 0.210$) and a mean accuracy to detect the correct rotation r of 0.619 ($\sigma = 0.130$), considering as acceptable the range $[r - 10, r + 10]$. We then evaluate the CAD system performances with 10 non-expert users: the average speed improved from 9.61 ($\sigma = 1.53$) to 5.30 ($\sigma = 1.30$) and the average number of actions to complete the task from 12.60 ($\sigma = 2.15$) to 11.00 ($\sigma = 0.89$). This work is a further step through the adoption of DL and AR in the assistance domain. In future works, we will overcome the limitations of this approach and develop a complete mobile CAD system that could be extended to any object that presents a 3D counterpart model.

Keywords Deep learning · Neural network · Augmented reality · CAD assistance

1 Introduction

In recent decades, technology has helped several procedures to improve massively; in particular, significant progress has been achieved with Deep Learning (DL) [1] paradigms. DL, i.e., the area of Machine Learning dealing with neural networks, has acquired a fundamental role in various environments, and many different remarkable applications have been implemented, from the medical domain [2–4] to natural language processing [5] and even gaming [6]. In parallel, the same enhancement was brought in by Augmented Reality (AR) in a wide range of fields such tourism [7], education [8], surgery [9], and manufacturing [10]. In particular, one of the many possible tasks which could be improved by DL and AR is the maintenance area, which refers to all actions that aim to restore any functionality of a product in its life cycle. The actions that can be performed to restore product functionality can be technical, administrative, and managerial [11].

Studies on DL and AR used independently in maintenance showed promising results to improve human performance in performing technical maintenance tasks, improving the administration of maintenance operations, and supporting maintenance managerial decision-making. This is specially important because modern production machines are becoming increasingly complex, often integrating advanced components that require a high level of expertise in the skills necessary for their maintenance. In order to allow technicians to work on equipment they are not familiar with, it has been assessed in the literature [14] that DL and AR systems allow faster problem diagnosis, which in turn may lead to faster repair tasks. In fact, the augmentation is particularly

✉ Leonardo Tanzi
leonardo.tanzi@polito.it

[1] Department of Management, Production and Design Engineering, Polytechnic University of Turin, Turin, Italy

Published online: 17 November 2022

helpful for immediately identifying the involved components and for providing the professional with extra details regarding diagrams or process to be followed. Additionally, more repairing activities can be accomplished by a variety of experts with varying levels of skill while maintaining the same standard of quality, which reduces the requirement for a single expert person who must travel from place to place to perform maintenance. Unfortunately, because to the difficult trade-off between using 3D data and the speed required for real-time elaboration, the joint use of DL and AR in this industry is still undervalued.

In this study, we moved the first step in this context, as we aim to demonstrate how to apply DL to retrieve the position and the rotation data of the machine in need of maintenance from live video frames only. With those data, it will be possible to implement an AR system to give guidance to non-expert users during a specific maintenance operation. To test our framework, we chose to use FDM 3D printers. This choice was due to two main reasons: firstly, they are ubiquitous and relatively cheap printers, primarily used by non-expert users that our system could facilitate; secondly, as in our research group we work with additive manufacturing, this software could help new members when facing such problems.

The presented framework leverages two different Convolutional Neural Networks (CNN) to determine the position and rotation data of the machine to be maintained. The two nets work on frames from a live video stream, like the one that may be obtained using any standard smartphone. The first neural network, a YOLOv3 [12] architecture, is applied to obtain the object localization inside the image. Object detection has become particularly efficient thanks to CNN, which made it possible to analyze images using a sliding window method. The first family of CNN-based algorithms to achieve noteworthy results in this discipline were Region-CNN [31]. Unfortunately, they were not real-time appropriate, and this limitation was overcome by the introduction of YOLO, since it is both quick and precise. Because of this, we choose to use YOLO without making any changes to the original method. Once the object has been found, the rest of the image is cropped, and the resulting ROI (Region Of Interest) is then passed to a second CNN, called RotationCNN, trained on a synthetic dataset generated with Blender [13] to predict the rotation of the printer on the three cartesian axes.

We tested YOLOv3 with AP (Average Precision) value, a metric that defines the quality of an object detector, and the RotationCNN with values of Geodesic Distance, a positive amount corresponding to the length of the geodesic arc connecting the two rotations expressed in quaternions. With the information of position ad scale retrieved from the YOLOv3's output and the ones of rotations predicted by the RotationCNN, an AR application can be implemented to project the 3D model directly onto its real-world counterpart. In Sect. 5, we propose an example of such an application.

As a case study, we chose a specific FDM 3D Printer, the Prusa i3 MKS model. We implemented a particular maintenance activity, such as the replacement of filament, which is divided into several steps explained in detail further on. We selected this procedure as it is one of the most frequently handled. Nevertheless, this method works with any procedure and any FDM 3D Printer (or generic object as well) of which the 3D model is available. We then asked five non-expert users to perform the procedure with the help of the CAD system and five more without the CAD system but instead using the official Prusa i3 MKS's instruction manual to demonstrate the validity of our system. The basic idea idea is that evaluating the ability of the user is also an indirect metric of the positioning method in a real situation, as it shows how the positioning network is behaving. Of course, if the position is not retrieved correctly, the performance of the users would not improve by using the assistance tool.

The code to generate the synthetic dataset given a 3D model is available here: github.com/leonardotanzi/3d-render.

2 Related works

As stated in [14], the use of AR in maintenance ranges from dis/assembly to repair, diagnosis, and training. Repair operations are defined as actions aimed at restoring the functional properties of a device [15]. Diagnosis refers to maintenance activities that aim to assess the current state of the product and analyze the causality of deterioration and functional degradation [16]. Training refers to processes that aim to transfer maintenance skills to technicians [17]. Regarding dis/assembly, which is the area where our application resides, as early as 1997, Azuma [18] stated that overlaying a 3D animated drawing could facilitate assembly processes compared to traditional user manuals. In [19], the authors demonstrate a straightforward AR approach that overlays virtual arrows and text on top of the real environment. In [20], the authors used the Hand Held Display (HHD) to perform maintenance tasks on consumer devices by showing the task description at the bottom of the display and providing a few buttons to navigate through the procedure. In [17] the authors showed an effort in providing different levels of instructions. They proposed two levels of guidance: a strong one that supports the user through each step, and a soft one that offers more high-level information and is designed for more experienced users. In [21], the authors incorporated into the AR procedure the ability to provide real-time feedback on the operation. Through the position and orientation of the components, they were able to show warning messages to correct the assembly procedure. Finally, a slightly different approach was proposed in [22], where the authors developed an AR application to simulate

the assembly procedure during the initial component design phase. They also estimated the forces involved in assembly by considering the stiffness, shapes, and contact surfaces of both the real component and the virtual prototype.

In all these applications, when AR is applied to a specific situation, the goal is fusing 3D objects with real-time images taken by the camera. The challenge was how to correctly align the virtual objects with their real-world equivalent. Estimating the 6D position of an object from an image is a central problem in Computer Vision (CV). It affects many domains such as robotics, autonomous driving, medicine, industrial inspection, and virtual/augmented reality applications, widely used in the entertainment and medical care industries [23–25]. The problem consists of determining the 3D rotation and translation of an object whose shape is known with respect to the camera, using observable details from the reference image. However, solving this problem is not trivial. Due to self-occlusions or symmetries, objects cannot be clearly and unambiguously identifiable, assuming an ambiguous position. In addition, image conditions are not always optimal in terms of illumination and occlusions between depicted objects [26]. In these situations, it is often necessary to add an earlier semantic segmentation or object detection step to identify the area of the image that contains the object, before estimating its position. Although researchers have studied this problem for many years, it has experienced something of a renaissance with the advent of DL. Older pose estimation methods were based on geometric approaches, trying to establish correspondences between 3D models and corresponding 2D images of objects using manually annotated local features. With untextured or geometrically complex objects, it is not easy to select local features. In these cases, although matching is time-consuming, it can fail and provide a result that is not always accurate [27]. In opposition to these methods, researchers have introduced other strategies, relying on representations of 2D objects from different viewpoints, and comparing them with the original image to determine location and orientation. These methods are very susceptible to variations in illumination and occlusions even though they can handle untextured objects and require many comparisons to achieve a certain level of accuracy, increasing the runtime [25]. With the spread of DL, researchers have introduced new strategies to achieve this goal, improving the traditional methods, and making them more efficient and performant. The basic idea of systems involving CNN is to learn a mapping function between the image, and the 6D position of the object, from images that have three-dimensional position annotations. These methods can achieve very high levels of accuracy but need a lot of data to accurately train the network to work well in real-world cases. One particular approach based on DL relies on a synthetic dataset to train a CNN to predict the object's rotation, such as in [28, 29]. After the literature review, the approach proposed in these two papers seemed the most suitable for our system, due to its flexibility and extendibility to new objects or procedures.

Nevertheless, the first relies on Faster-CNN for object detection, an algorithm that is 8 times slower than YOLO, as demonstrated in the original YOLO paper, and can't be thus used for a real-time application. The second utilized for training both synthetic data and ∼22K images real images from PASCAL 3D+, which made the algorithm once again dependent on real data, which are costly to collect.

In our work, we proposed an approach that overcomes these two obstacles by implementing a real-time detection and an overlay phase dependent on a very small amount of real-images for fine-tuning. The main contribution of this paper is in presenting a novel approach that combines two different neural networks to predict the rotation and position of a generic machine in real-world space, leveraging only RGB data from a live video stream. The predicted data can then be used for AR applications like, for example, those that support nonskilled people with maintenance operations, such as the one presented in Sect. 5.

3 Methods

The general method of our framework is detailed in Fig. 1. The original RGB image is firstly passed to an object detection algorithm, in our case YOLOv3, which returns the ROI (Region Of Interest) related to the object detected. From this bounding box, we can obtain the values of scale and position of the 3D model. The cropped area is then passed to the RotationCNN, which returns the rotations values along the X, Y, and Z axes. This information is combined to overlay the 3D model on the 2D stream. All these steps will be discussed broadly in the following sub-sections.

3.1 Object detection

The first step of our approach aimed to locate the 3D Printer object in the video stream. The reason is two folds:

1. Allow the RotationCNN to concentrate solely on the printer without getting confused by the noisy background;
2. Retrieve the positions and scale values from the bounding box vertexes' coordinates. In particular, the center of the bounding box is used as an anchor point to attach the 3D model, and the scaling is computed comparing with a specific ratio of the width of the bounding box and the width of the 3D model.

As the application aimed to be real-time, we chose to use YOLO (You Only Look Once) algorithm [30], in particular,

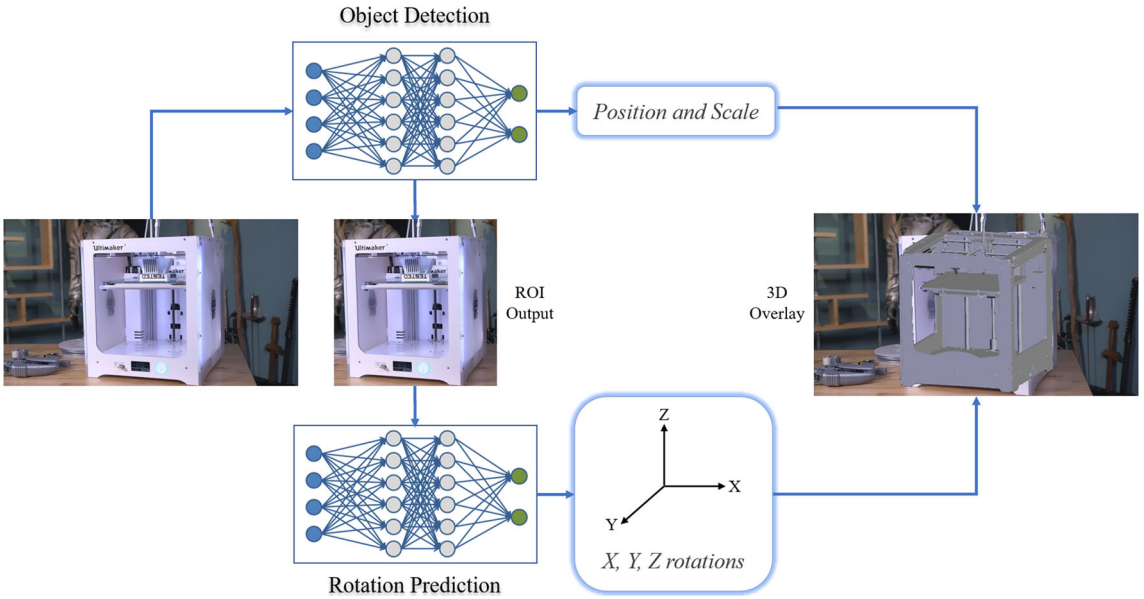

Fig. 1 Full pipeline of the proposed approach. The frame is passed to an object detection algorithm which returns the cropped area related to the detected 3D printer and the position and scale information. Its output is passed to a CNN, which predicts the rotation values, according to the coordinates system shown in the figure. This information is then combined to correctly overlay the 3D model to the 2D stream with specific instructions and highlighted components. However, for clarity, in this example is shown the whole 3D model instead of a specific highlighted part

YOLOv3 [12], which, compared to the latest object detection framework such as Faster-RCNN [31], is less precise in some aspects, for example, it struggles with small objects within the image, but an order of magnitude faster. Prior detection systems repurpose classifiers or localizers to perform detection. They apply the model to an image at multiple locations and scales, with high scoring regions of the image considered as detections. YOLO uses a totally different approach with just a single neural network that divides the image into regions and predicts bounding boxes (weighted by the predicted probabilities) and probabilities for each region.

3.2 Rotation CNN

For the RotationCNN, which takes as input the area around the bounding box returned by YOLOv3, we used a ResNet50 [32] with three different branches for X, Y, and Z axes rotation. We choose ResNet50 after comparing it with two state-of-the-art model, InceptionV3 [33] and ViT-B16 [34], in terms of accuracy, geodesic distance and the number of iterations per seconds. Each branch has the same structure, which contains: a Dense layer with 4096 neurons and a ReLU (Rectified Linear Unit) activation function, a Batch Normalization layer [35], a Dropout layer [36] with a random parameter of 0.5, and a Dense output layer, with several neurons equal in number to the specific axis' range of rotations and a Softmax activation function. We solved the X, Y and Z axes rotation value estimation as a classification problem. We subdivided the set of possible rotation values along an axis according to the possible configurations of rotations that the 3D printer can possibly assume during this specific procedure. We considered 40 classes for X-axis rotation (from $-$ 10° to 30°), 60 for Y-axis rotation (from $-$ 30° to 30°), and 120 for Z-axis rotation (from $-$ 60° to 60°). Therefore, the neuron with the highest probability according to the Softmax activation function will fire and produce the corresponding rotation value as output. The architecture of the model is shown in Fig. 2.

3.3 Datasets

Three datasets have been used in this work. For training YOLOv3, we collected and tagged with bounding boxes 545 images showing the FDM 3D printers in different positions and rotations, of which 20% were kept away for testing. Each image was then resized to 461 $\times$ 461 pixels, and augmentation was applied with random horizontal flip, random zoom crop (0 to 45%), and random rotation ($-$ 15° to 15°), resulting in a total of 1653 images. To train the RotationCNN, we artificially build a synthetic dataset through Blender to make the rotation estimation easier by using as inputs 17 3D models of different FDM printers, and 36.500 real backgrounds taken from SUN Database [37]. According to meaningful rotation values of the FDM 3D printers, we considered the following ranges in degrees for the three rotation axes: [$-$ 10°, 30°] for X-axis, [$-$ 30°, 30°] for Y-axis, and [$-$ 15°,

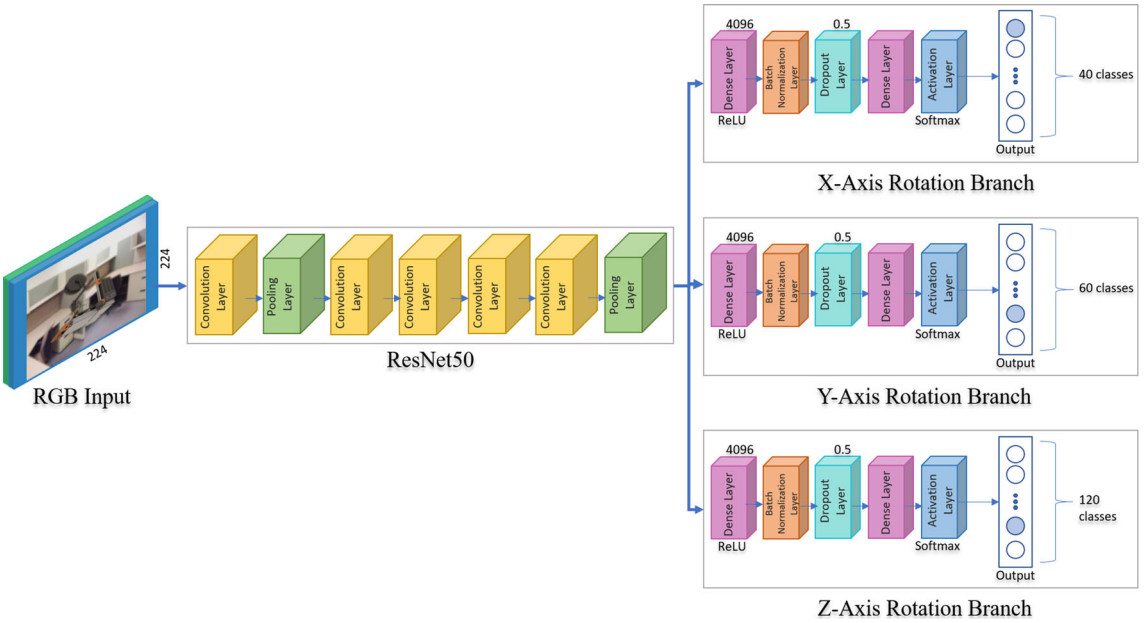

Fig. 2 Architecture of the RotationCNN. The network takes as input the RGB images, which are passed through the first layers of ResNet50. The last layer was substituted with three branches composed of a Dense layer, a Batch Normalization layer, a Dropout layer, and a final Dense layer with n neurons, where n is the range of the rotation along the specific axis

15°] for Z-axis. We generated a render for each combination of X and Z rotation values and 1/3 of Y rotation values and randomly changed lighting conditions and the scene's background. With this process, we obtained a synthetic dataset of 94.220 images for training and 5000 for testing. Finally, we also created a dataset of 235 real-images of FDM 3D printers tagged with values of rotations, to fine-tune the network with 150 images and test the overall performances with the remaining 85 images.

3.4 Training, metrics, and framework

YOLOv3 was trained for 500 epochs with the first 49 layers frozen and a batch size of 32 and 100 additional epochs with all layers un-frozen and a batch size of 16. The metric used to evaluate the object detection is the Average Precision (AP) criterium defined in the PASCAL VOC 2012 [38] competition. First, the neural net detection results were sorted by decreasing confidence and were assigned to ground-truth objects. We had a match when the IoU (Intersection over Union) was more significant than a certain threshold. The IoU is defined as:

$$IoU = \frac{AOverlap}{AUnion}$$

where *AOverlap* is the area of overlap between the predicted bounding box and the ground truth, and *AUnion* is the area of union between the predicted bounding box and the ground truth. This metric is normalized in the interval [0, 1], with 0 meaning no overlap and 1 meaning a perfect overlay. In the PASCAL VOC criterium, the IoU threshold was set to 0.5. In this work, as we were looking for a higher precision to overlap the 3D model, we also presented the values of AP using 0.6 and 0.7 thresholds. Using this approach, we calculated the precision/recall curve. Then, we computed a version of the measured precision/recall curve with precision monotonically decreasing and calculated the AP as the area under this curve by numerical integration.

The RotationCNN was trained with the synthetic dataset for 20 epochs with a batch size of 32 and Adam optimizer with a learning rate of 0.0001 and fine-tuned with real images for 100 epochs with the same batch size but Adam optimizer with a learning rate of 0.000001 and with all the layers except the three final branches frozen. The difference between the predicted and the actual rotations was computed by converting the values of the rotations in quaternions and then calculating the geodesic distance between two quaternion coordinates $q1$ and $q2$. To get a distance between two unit quaternions, you have to rotate both of them such that one of them becomes the identity element. To do this for our pair $q1$ and $q2$, we simply multiplied $q1$ by $q2$'s inverse from the left

$$Q = (inverse(q2) * q1)$$

and normalize the obtained quaternion Q through L2 normalization:

$$geoDist = L2(Q) = \sqrt{Q \cdot Q}$$

The metric is a positive amount corresponding to the length of the geodesic arc connecting *q1* to *q2*. We choose the geodesic distance as is the most common metric used in literature to evaluate the difference between two angles. Finally, to evaluate the performance of the different networks, we used the number of iterations per seconds, defined as:

$$it/s = \frac{n}{\text{sec}}$$

where n is the number of iterations, and *sec* is the time unit. In this case, one iteration consists of predicting the rotation's angles given an input image. As the frame rate metric depends on different aspects, such as the complexity of the mesh, the rendering engine used, the hardware specifics, the particular implementation of the pipeline, etc., these factors can determine a strong fluctuation of the metric, for this reason we preferred to opt for a metric independent of these parameters. We empirically noticed that our frame rate was acceptable if we kept the *it/s* greater than 5; this evaluation is not indicative but sufficient for us to obtain a real-time validation. We used Keras [39], an open-source neural-network library written in Python, running on top of TensorFlow and, on Windows 10 Pro with NVIDIA Quadro P4200.

4 Results

4.1 Object detection

After the training phase, YOLO obtained a test loss of 11.353. We then computed the AP with IoU thresholds of 0.5, 0.6 and 0.7, shown in light blue as the area under the curve of the precision/recall curve in Fig. 3a–c, respectively. These values of AP were 100% with 0.5 as IoU threshold, 95.68% with 0.6 as IoU threshold, and 83.27% with 0.7 as IoU threshold.

4.2 RotationCNN

The results of the comparison between different networks are showed in Table 1.

In addition, in Table 2 are showed the extensive results for the most performing network, ResNet50, chosen as the best compromise between accuracy, geodesic distance and number of iterations per seconds.

We first tested the network with 5000 synthetic images generated with Blender. The accuracies for X, Y, and Z, were computed as the number of correct predictions over the total number of samples, using different acceptable ranges: exact prediction, with an error in the range $[-5, +5]$ and with an error in the range $[-10, +10]$, were 0.852 ($\sigma = 0.147$), 0.999 ($\sigma = 0.001$) and 1 ($\sigma = 0$) respectively. We also computed the geodesic distance, which resulted in an average of 0.0038 ($\sigma = 0.005$).

We then test this same network with 85 real images tagged manually. The accuracies were 0.007 ($\sigma = 0.005$), 0.113 ($\sigma = 0.038$) and 0.231 ($\sigma = 0.052$) and the geodesic distance 0.454 ($\sigma = 0.217$).

We finally fine-tuned the network with 150 images and tested with the same 85 images as before, resulting in an accuracy of 0.188 ($\sigma = 0.044$), 0.443 ($\sigma = 0.077$), and 0.619 ($\sigma = 0.130$) respectively and a geodesic distance of 0.250 ($\sigma = 0.210$).

5 Case study

To validate our method, we chose a specific FDM 3D Printer, the Prusa i3 MKS model, and implemented a specific maintenance action, such as the replacement of filament, as it is one of the most frequent to be handled. Nevertheless, this system works for any procedure and any FDM 3D Printer (or generic object as well) of which the 3D model is available.

5.1 Procedure

After the extraction of the bounding box and the rotations value, the 3D model is attached at the center of the bounding box with the predicted rotations. The tool is then used to guide the user in the filament substitution. The phases, also underlined in Fig. 4, are:

(1) Press the **button** and search in the menu screen "Unload the **filament**"
(2) Press the **button** and specify the material to unload
(3) Press the **button** and wait until the acoustic signal
(4) Press the **button** to eject the **filament**
(5) Pull the **filament** upwards
(6) Replace the **coil**
(7) Insert the new **filament** into the **extruder**'s **filament** hole
(8) Search in the menu **screen** for "Load the **filament**"
(9) Press the **button** and check if the extruded **filament** has the correct color
(10) If *Yes*, clean the **extruder**, if *No*, repeat step 9

5.2 Evaluation

Finally, we asked five non-expert users to perform the procedure with the help of the CAD System and five more without

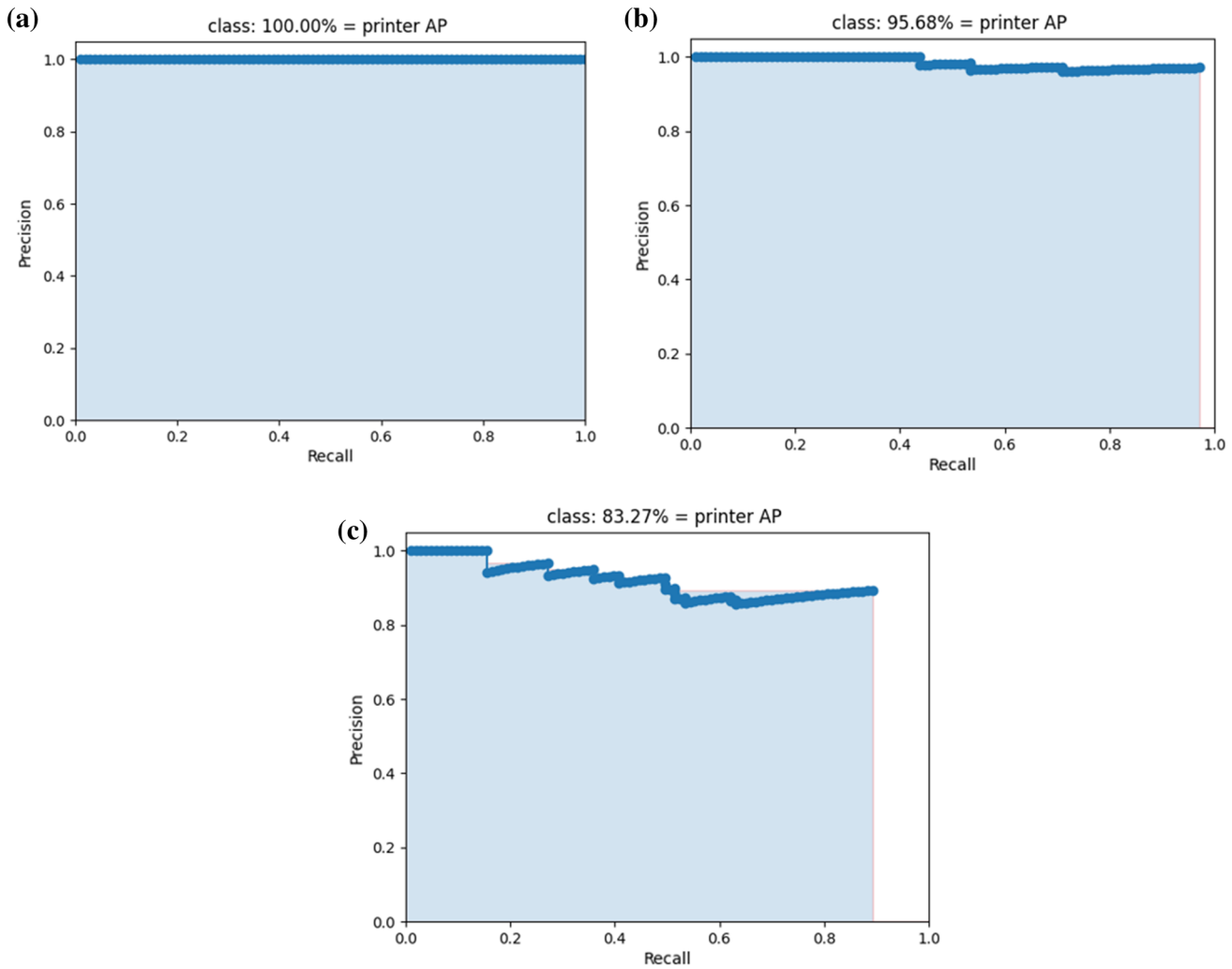

Fig. 3 Precision/recall curve with IoU thresholds of 0.5 (**a**), 0.6 (**b**) and 0.7 (**c**). The AP is shown in light blue as the area under the curve

Table 1 Values of mean accuracies together geodesic distance

Architecture	Accuracies	Accuracies [− 5, + 5]	Accuracies [− 10, + 10]	Geodesic distance	*it/s*
ResNet50	0.188 (σ = 0.044)	0.443 (σ = 0.077)	0.619 (σ = 0.130)	0.250 (σ = 0.210)	7
InceptionV3	0.158 (σ = 0.043)	0.343 (σ = 0.098)	0.531 (σ = 0.160)	0.299 (σ = 0.206)	6
ViT-B16	0.180 (σ = 0.049)	0.402 (σ = 0.080)	0.601 (σ = 0.123)	0.255 (σ = 0.203)	2

All values are shown with related standard deviation. The accuracies were computed as the number of correct predictions over the total number of samples, using different acceptable ranges: exact prediction, with an error in the range [− 5, + 5] and in the range [− 10, + 10]**,** while the geodesic distance is computed as the positive amount corresponding to the length of the geodesic arc connecting the two rotation's angles. *It/s* are iterations per second, where one iteration is considered as a forward pass in the full pipeline

Table 2 Values of accuracies for each axes together with mean accuracy and geodesic distance

	Accuracies			Accuracies [− 5, + 5]			Accuracies [− 10, + 10]			Geodesic distance	# of images
Axes	X	Y	Z	X	Y	Z	X	Y	Z		
Synthetic dataset	0.644	0.961	0.950	0.998	0.999	0.998	1	1	1	0.0038 ($\sigma = 0.005$)	5000
	0.852 ($\sigma = 0.147$)			0.999 ($\sigma = 0.001$)			1 ($\sigma = 0$)				
Real dataset	0.000	0.012	0.007	0.060	0.151	0.125	0.164	0.290	0.239	0.454 ($\sigma = 0.217$)	85
	0.007 ($\sigma = 0.005$)			0.113 ($\sigma = 0.038$)			0.231 ($\sigma = 0.052$)				
Real dataset (after fine tuning)	0.130	0.237	0.196	0.334	0.502	0.491	0.435	0.713	0.707	0.250 ($\sigma = 0.210$)	85
	0.188 ($\sigma = 0.044$)			0.443 ($\sigma = 0.077$)			0.619 ($\sigma = 0.130$)				

All values are shown with related standard deviation for tests performed with a synthetic dataset, a real dataset, and the same real dataset after fine-tuning the network. The accuracies for *X*, *Y*, and *Z*, were computed as the number of correct predictions over the total number of samples, using different acceptable ranges: exact prediction, with an error in the range [− 5, + 5] and in the range [− 10, + 10], while the geodesic distance is computed as the positive amount corresponding to the length of the geodesic arc connecting the two rotation's angles

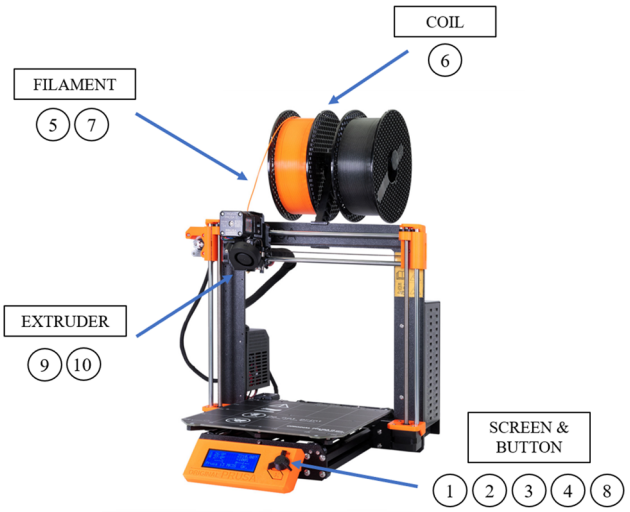

Fig. 4 In the specific operation of filament replacement four components of the Prusa i3 MKS are involved: the Screen and the Button for steps 1, 2, 3, 4, and 8, the Filament for steps 5 and 7, the Coil for step 6 and the Extruder for steps 9 and 10. These components are highlighted during the corresponding step

the CAD system but using the official Prusa i3 MKS's instructions manual. This method of validation is more accurate, because if we asked the same ten people to perform the operation with and without the CAD help the second time they would be facilitated by the fact that they have already done the operation once. Our system applies instead to users who have never performed the operation. The video stream with the overlay model and the textual information was provided on the PC screen, and the user has simply to press a generic button when he/she finished a step. Figure 5 shows the ten different augmented steps, while Fig. 6 shows the process of a single step. The image is first passed to the object detector (1), which extracts the ROI (2). This information is used to overlay the 3D model (3) on the video stream and finally the RotationCNN is used to retrieve the actual values of rotations. The model is rotated accordingly, and text instructions are also added (4). The specific component involved in the current precision was highlighted in orange, while the other components were kept in grey. The text instructions were the same as defined above. To evaluate the quality of the intervention, we chose two parameters suggested by an expert in 3D printing in our research group: the time spent and the number of actions to perform the whole operation. An action is defined as a single step performed. With these two metrics we could evaluate both the speed and the precision (i.e., how many times a step had to be repeated). An expert user performs the whole procedure in the ten actions defined above with an average time of approximately 2 min. The five users performing the operations consulting the instructions manual obtained a mean speed of 9.61 ($\sigma = 1.53$), and the mean number of actions was 12.60 ($\sigma = 2.15$), while the five users helped by the CAD system obtained a mean speed of 5.30 ($\sigma = 1.30$) and the mean number of action was 11.00 ($\sigma = 0.89$). Results are resumed in Table 3.

6 Discussion

In this work, we proposed a framework to assist non expert users in maintenance procedures of FDM 3D printers, through an AR overlay of the printer's 3D model based on DL algorithms. After a literature review, we chose to implement a two steps approach. The first phase involved an object detection algorithm, specifically YOLOv3, to locate the area related to the generic FDM 3D printer and obtain the value of the 3D model position and scale. This network was trained using 1653 augmented images together with their bounding boxes. In the second phase, we passed the output of YOLOv3

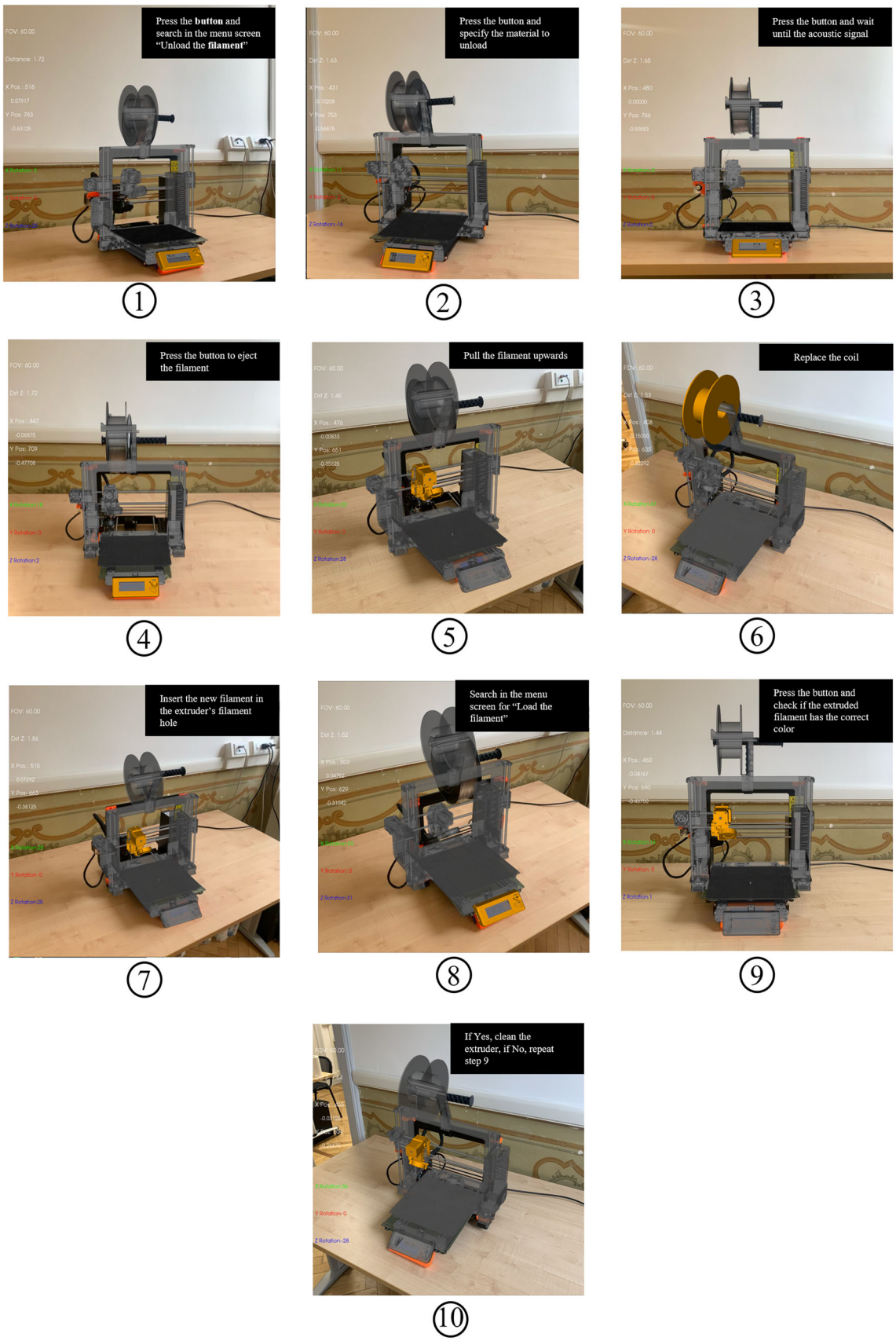

Fig. 5 The ten steps of the filament substitution procedure. In each step, the specific component involved in the current precision was highlighted in orange, while the other components were kept in grey. The text instructions were the same defined above

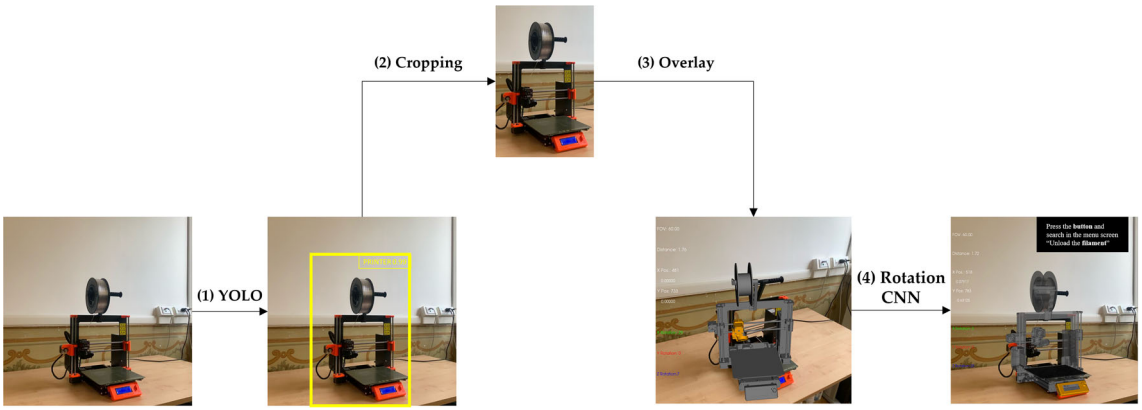

Fig. 6 The process of retrieving the position and rotation information in all the steps. Here is shown the one related to the first step, composed by the sequence of YOLO's object detection and rotation prediction with the RotationCNN

Table 3 Values of speed (in minutes) and a number of actions performed by the ten non-expert users with the help of the user manual or with the help of our CAD system

Users	User manual						CAD system					
	1	2	3	4	5	Mean	6	7	8	9	10	Mean
Speed (minutes)	9.08	10.11	7.41	9.33	12.12	**9.61 ($\sigma = 1.53$)**	7.21	4.32	5.23	3.56	6.20	**5.30 ($\sigma = 1.30$)**
Number of actions	11	14	12	10	16	**12.60 ($\sigma = 2.15$)**	12	10	11	10	12	**11.00 ($\sigma = 0.89$)**

In the bold values are showed the means and standard deviations for each distribution

to a RotationCNN, which predicted the rotations values along the three axes, *X*, *Y,* and *Z*. This second network was trained with 94.220 synthetic images produced with Blender and fine-tuned with 150 real images tagged manually. The performances of these two networks are discussed in Table 2 and Sect. 4. Results YOLOv3 detected 100% of the FDM 3D printer with an IoU threshold of 0.5, the official threshold used in the PASCAL VOC challenge, and obtained good results even if we increased the value of IoU. The most critical network was the RotationCNN, as the prediction of rotations is far more complex than the object detection task. We tested three different network, ResNet50, InceptionV3 and ViT-B16, after selecting ResNet50 as the most performing one. In fact, ResNet50 achieved similar accuracies and geodesic distances as ViT-B16 while performing three times as many operations per second. We used two metrics: the accuracy, considering different acceptable ranges: exact prediction, with an error in the range $[-5, +5]$ and in the range $[-10, +10]$**,** and the geodesic distance. Testing with 5000 synthetic images, we obtained values of accuracies close to 1 and a very low geodesic distance of 0.0038 ($\sigma = 0.005$), showing that the network actually learned; the main question was if he was able to generalize the results with authentic images. Without fine-tuning, the performances of the network were very poor, with accuracies close to 0 and a geodesic distance of 0.454 ($\sigma = 0.217$). After fine-tuning the network with 150 real images tagged manually, we obtained the following results: the three values of accuracy were 0.188 ($\sigma = 0.044$), 0.443 ($\sigma = 0.077$), and 0.619 ($\sigma = 0.130$) respectively and the geodesic distance was 0.250 ($\sigma = 0.210$). These results are good enough to implement our methodology.

To test our system, we also presented a case study with a specific FDM 3D printer, the Prusa i3 MKS. We asked five non-expert users to perform the procedure with the help of the CAD System and five more without the CAD system but using the official Prusa i3 MKS's instructions manual and evaluate the performances with speed and number of operations to complete the whole procedure. The mean improvement given by using our tool was 4 min and 31 s in speed and *1.6* in number of operations. The concept of evaluating AR-assisted maintenance was inspired by the realization that measuring user competence also serves as an indirect indicator for measuring the performance of the positioning method in actual use. Naturally, employing the support tool would not increase the users' performance if the position is not correctly obtained, as shown in Table 3.

7 Conclusions and future works

In this paper, we demonstrate how to apply DL to retrieve the position and the rotation data of the machine in need of maintenance from live video frames only. With those data, it will be possible to implement an AR system to give guidance

to non-expert users during a specific maintenance operation. of an FDM 3D printer. We also presented a simple AR application leveraging our system, to support unskilled people in printer maintenance.

Even if we showed our system's performance, there are still several limits to overcome. First of all, the Rotation-CNN performances are acceptable, but it still struggles in detecting the precise value of the rotation. Secondly, we provided the users with a GUI directly on the PC screen: we think the performances could vastly improve with a mobile application. Thirdly, the operation itself was quite simple to perform. Thus the practical improvement of the use of our tool may seem low.

In future works, we plan to compare the performances of our methodology to other similar, implementing a more complex application for mobile smartphones. Focusing on different models of FDM 3D printers, the improvement will include more complex maintenance operations.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s12008-022-01107-5.

Funding Open access funding provided by Politecnico di Torino within the CRUI-CARE Agreement. This research received no external funding.

Data availability Not available.

Code availability Not available.

Declarations

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

References

1. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**(7553), 436–444 (2015)
2. Tanzi, L., Vezzetti, E., Moreno, R., Aprato, A., Audisio, A., Massè, A.: Hierarchical fracture classification of proximal femur X-ray images using a multistage deep learning approach. Eur. J. Radiol. **133**, 109373 (2020)
3. Tanzi, L., Piazzolla, P., Porpiglia, F., Vezzetti, E.: Real-time deep learning semantic segmentation during intra-operative surgery for 3D augmented reality assistance. Int. J. CARS **16**, 1435–1445 (2021). https://doi.org/10.1007/s11548-021-02432-y
4. Tanzi, L., Audisio, A., Cirrincione, G., Aprato, A., Vezzetti, E.: Vision transformer for femur fracture classification. Injury **53**(7), 2625–2634 (2022)
5. Young, T., Hazarika, D., Poria, S., Cambria, E.: Recent trends in deep learning based natural language processing. IEEE Comput. Intell. Mag. **13**(3), 55–75 (2018)
6. Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., Chen, Y., Lillicrap, T., Hui, F., Sifre, L., van den Driessche, G., Graepel, T., Hassabis, D.: Mastering the game of go without human knowledge. Nature **550**(7676), 354–359 (2017)
7. Cranmer, E.E., Tom Dieck, M.C., Fountoulaki, P.: Exploring the value of augmented reality for tourism. Tour. Manag. Perspect. **35**, 100672 (2020)
8. Hughes, C.E., Stapleton, C.B., Hughes, D.E., Smith, E.M.: Mixed reality in education, entertainment, and training. IEEE Comput. Graph. Appl. **25**(6), 24–30 (2005)
9. Gribaudo, M., Piazzolla, P., Porpiglia, F., Vezzetti, E., Violante, M.G.: 3D augmentation of the surgical video stream: toward a modular approach. Comput. Methods Programs Biomed. **191**, 105505 (2020)
10. Nee, A.Y.C., Ong, S.K., Chryssolouris, G., Mourtzis, D.: Augmented reality applications in design and manufacturing. CIRP Ann. **61**(2), 657–679 (2012)
11. Komonen, K.: A cost model of industrial maintenance for profitability analysis and benchmarking. Int. J. Prod. Econ. **79**(1), 15–31 (2002)
12. Redmon, J., Farhadi, A.: YOLOv3: an incremental improvement. arXiv:1804.02767 [cs] (2018)
13. Community, B.O.: Blender: a 3D modelling and rendering package. Stichting Blender Foundation, Amsterdam (2018)
14. Palmarini, R., Erkoyuncu, J.A., Roy, R., Torabmostaedi, H.: A systematic review of augmented reality applications in maintenance. Robot. Comput. Integr. Manuf. **49**, 215–228 (2018)
15. Wójcicki, T.: Supporting the diagnostics and the maintenance of technical devices with augmented reality. Diagnostyka **15**(1), 43–47 (2017)
16. Regenbrecht, H., Baratoff, G., Wilke, W.: Augmented reality projects in the automotive and aerospace industries. IEEE Comput. Graph. Appl. **25**(6), 48–56 (2005)
17. Webel, S., Bockholt, U., Engelke, T., Gavish, N., Olbrich, M., Preusche, C.: An augmented reality training platform for assembly and maintenance skills. Robot. Auton. Syst. **61**(4), 398–403 (2013)
18. Azuma, R.T.: A survey of augmented reality. Presence Teleoper Virtual Environ. **6**(4), 355–85 (1997)
19. Lee, S.G., Ma, Y.-S., Thimm, G.L., Verstraeten, J.: Product lifecycle management in aviation maintenance, repair and overhaul. Comput. Ind. **59**(2), 296–303 (2008)
20. Sanna, A., Manuri, F., Lamberti, F., Paravati, G., Pezzolla, P.: Using handheld devices to support augmented reality-based maintenance and assembly tasks. In: 2015 IEEE International Conference on Consumer Electronics (ICCE), pp. 178–9 (2015)
21. Westerfield, G., Mitrovic, A., Billinghurst, M.: Intelligent augmented reality training for motherboard assembly. Int. J. Artif. Intell. Educ. **25**(1), 157–172 (2015)
22. Wang, X., Ong, S.K., Nee, A.Y.C.: Real-virtual components interaction for assembly simulation and planning. Robot. Comput. Integr. Manuf. **41**, 102–114 (2016)
23. Li, X., Cai, Y., Wang, S., Lu, T.: Learning category-level implicit 3D rotation representations for 6D pose estimation from RGB

images. In: 2019 IEEE International Conference on Robotics and Biomimetics (ROBIO), pp. 2310–2315 (2019)
24. Muñoz, E., Konishi, Y., Beltran, C., Murino, V., Bue, A.D.: Fast 6D pose from a single RGB image using cascaded forests templates. In: 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 4062–4069 (2016)
25. Liu, F., Fang, P., Yao, Z., Fan, R., Pan, Z., Sheng, W., Yang, H.: Recovering 6D object pose from RGB indoor image based on two-stage detection network with multi-task loss. Neurocomputing **337**, 15–23 (2019)
26. Zuo, G., Zhang, C., Liu, H., Gong, D.: Low-quality rendering-driven 6D object pose estimation from single RGB image. In: 2020 International Joint Conference on Neural Networks (IJCNN), pp. 1–8 (2020)
27. Zhao, W., Zhang, S., Guan, Z., Luo, H., Tang, L., Peng, J., Fan, J.: 6D object pose estimation via viewpoint relation reasoning. Neurocomputing **389**, 9–17 (2020)
28. Josifovski, J., Kerzel, M., Pregizer, C., Posniak, L., Wermter, S.: Object detection and pose estimation based on convolutional neural networks trained with synthetic data. In: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 6269–6276 (2018)
29. Su, H., Qi, C.R., Li, Y., Guibas, L.J.: Render for CNN: viewpoint estimation in images using CNNs trained with rendered 3D model views. In: 2015 IEEE International Conference on Computer Vision (ICCV), pp. 2686–2694 (2015)
30. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: unified, real-time object detection. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 779–788 (2016)
31. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: towards real-time object detection with region proposal networks. IEEE Trans. Pattern Anal. Mach. Intell. **39**(6), 1137–1149 (2017)
32. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770–778. IEEE (2016). Available from: http://ieeexplore.ieee.org/document/7780459/
33. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2818–26.. IEEE, Las Vegas, NV, USA (2016) [cited 2019 Nov 25]. Available from: http://ieeexplore.ieee.org/document/7780677/
34. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N.: An image is worth 16 × 16 words: transformers for image recognition at scale. In: International Conference on Learning Representations (2021). Available from: https://openreview.net/forum?id=YicbFdNTTy
35. Ioffe, S., Szegedy, C.: Batch normalization: accelerating deep network training by reducing internal covariate shift. In: Bach, F., Blei, D. (eds) Proceedings of the 32nd International Conference on Machine Learning, pp. 448–456. PMLR, Lille, France (2015). (Proceedings of Machine Learning Research; vol. 37). Available from: http://proceedings.mlr.press/v37/ioffe15.html
36. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. **15**(56), 1929–1958 (2014)
37. Xiao, J., Hays, J., Ehinger, K.A., Oliva, A., Torralba, A.: SUN database: large-scale scene recognition from abbey to zoo. In: 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 3485–3492 (2010)
38. Everingham, M., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A.: The PASCAL visual object classes challenge 2012 (VOC2012) results. Available from: http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html
39. Chollet, F. et al.: Keras (2015). Available from: https://keras.io