Instance Segmentation in Industry 5.0 Applications Based on the Automated Generation of Point Clouds

Cristina R. Monsone[1], Ádám B. Csapó[2]

[1] Széchenyi István University
Doctoral School of Multidisciplinary Engineering Sciences
Győr, Hungary
monsone.cristina@sze.hu

[2] Corvinus University of Budapest
Corvinus Institute for Advanced Studies /
Institute of Data Analytics and Information Systems
Budapest, Hungary
adambalazs.csapo@uni-corvinus.hu

Abstract: In this paper, we explore the utility of classical neural network-based approaches, originally designed for processing 2D images, in semantic segmentation and object recognition tasks within the context of 3D point cloud images generated from handheld video recordings. Our investigation centers around the use of a custom-created, small-sized training dataset, consisting of 108 RGB images of humans and cobots in diverse industrial settings. This dataset allows us to demonstrate that flexible segmentation and recognition applications can be built even with a restricted dataset developed using widely available low-cost tools and modern convolutional neural net architectures. Downstream benefits of the approach include the ability to detect humans and human gestures, as well as to rapidly prototype digital twins in Industry 5.0 environments.

Keywords: industrial image datasets; point-cloud generation; convolutional neural networks; instance segmentation

1 Introduction

Industry 5.0 (I5.0) builds on the foundations of Industry 4.0 in a way that transcends mere technological advancements and instead emphasizes a collaborative approach where technology serves humanity. In particular, a key focus in I5.0 is

on “Human- Centric Integration”, i.e. on human needs throughout the manufacturing process, in accordance with the principles outlined by the United Nations in its Sutstainable Development Goal 9 [1][2]. Collaboration with cobots (collaborative robots) is one key feature of this integration, and the ability to recognize and understand both human and cobot behaviour in this context is essential both from a functionality and a safety perspective [3][4].

Approaches within artificial Intelligence (AI) and machine learning (ML) are natural candidates for implementing such capabilities [5]. In an industrial context, artificial vision tasks involve both the acquisition of suitable images, and the interpretation of their content, with the primary objective of controlling, inspecting, measuring and analyzing processes, thereby making more informed decisions. The main benefits of artificial vision systems in industry can be summarized as follows [6]:

- Quality Improvement: By precisely analyzing images, these systems enhance product quality.
- Defect Prevention: Early detection of defects minimizes production issues.
- Traceability Compliance: Systems ensure adherence to traceability requirements.
- Waste Reduction: Efficient processes lead to less material waste.
- Increased Yield and Productivity: Optimized operations result in higher output.
- Reduced Human Error: Automation reduces the risk of human mistakes.
- Cost and Time Savings: Compared to manual methods, processing times and costs decrease significantly.

Training AI for computer vision, however, presents challenges. Achieving an acceptable level of accuracy demands a substantial volume of labelled data to train neural networks. This labelled data consists of images annotated by individuals who have already identified and marked relevant elements. The effort involved in curating such data often incurs disproportionate costs and therefore hinders the creation of sufficiently large datasets [7] [8].

Another critical challenge arises during the transition from traditional to smart manufacturing. Here, the goal is to digitize industrial machines and other objects relevant to the environment —- a process that often requires the combination of 2D images and video to reconstruct 3D representations. Point Cloud technology emerges as a valuable approach for generating 3D models suitable for instance segmentation using Convolutional Neural Networks (CNNs). Such smart models find applications in tasks like the development of Digital Twins [9, 10, 11, 12, 13].

In this paper, we focus on artificial vision using a variety of input modalities (e.g. RGB images, handheld videos, automatically generated point clouds) and modern neural network architectures to leverage tradeoffs between these challenges. In particular, we address the challenge of data availability by proposing a workflow that enables a smooth transition between these different input modalities, for subsequent application in a transfer learning context. The ability to apply a neural network trained on 2D images in the context of 3D point clouds suggests that it should be possible to generate semantically interpretable 3D representations for use in virtual reality (VR) based digital twin scenarios. This feature is crucial for industries where existing industrial equipment can be upgraded to enable smart manufacturing. This transformation offers dual benefits: cost reduction and optimization of the production process. Furthermore, ensuring improved safety conditions during human-robot interactions is another relevant aspect.

The structure of this paper is as follows. In Section 2, we provide an overview of the technologies and approaches constituting the proposed workflow – including data collection, 3D point cloud generation from videos, dataset augmentation and instance segmentation. In Section 3, we describe an application scenario involving collaborative robots in which we applied the workflow. Finally, in Section 4 we provide a summary of the metrics we used to evaluate our approach, and in Section 5 we present our experimental results.

2 Constituent Technologies of the Proposed Workflow

Our approach, centered on the generation of 3D point clouds and subsequent inference using pre-trained convolutional neural networks, is developed through 6 key steps, as described below. One of the main goals is to define a workflow – called "DTnet" – to support the rapid prototyping of digital twins in virtual reality environments, as shown on Figure 3.

2.1 Key steps of the workflow

1. Creation of a custom dataset: To train our models, in our previous work we created a custom dataset specifically tailored to industrial cobot scenarios. The original dataset contained 108 images, which were augmented to 244 (referred to as Dataset244). Pre-processing techniques, including auto-orientation and resizing to 640-by-640 pixels, were applied. Additionally, standard data augmentation methods such as 90-degree clockwise and counter-clockwise rotation and up to 2.2 px blur were used. Besides object detection, we also applied instance segmentation to the images in this dataset [13].

2. Training of a YOLO (You Only Look Once [14]) based model on the previ-

ous dataset.[13]

3. Recording videos of humans and cobots in industrial scenes. We recorded a total of 700 frames, which were never shown to the YOLO model during the initial training process. Our videos were recorded with the following parameters:

    - Frame size: 1920 x 1080
    - Codec: AAC H-264
    - Color: HD (1-1-1)
    - Total time: 62 seconds

    We refer to this dataset as the Cobot700 dataset. However, we also subsampled this dataset to keep only 150 images, so as to facilitate speed and accuracy of the automatic generation of point clouds in the next step (this subsampling led to the Cobot150 dataset)

4. Automatic generation of point clouds based on those frames, making use of the continuity of the video recordings and cutting-edge point cloud generation solutions. To create 3D point cloud images, we evaluated several software options. Notable applications for this task include Autodesk ReCap, Leica Cyclone, PointCab, Pix4D, RhinoPointCloud, and Agisoft Metashape4 [15] [16]. Among these, we opted for Agisoft Metashape[1] due to its support for various input data sources, including videos from ground-based cameras.

5. Definition of YOLOv8 Properties for inference and segmentation: We configured the properties of the YOLOv8 model in PyTorch to suit our specific task of 3D point cloud image recognition. YOLOv8 has proven effective in 2D object detection, and we adapt it for our novel context using our inference [13][17] [18].

6. Rapid prototyping of digital twins based on localized and segmented cobots, matched up with the pointcloud data.

2.2 Creation and training of custom dataset

As mentioned in the previous section, in our previous work we created a custom industrial dataset based on Cobot images, demonstrating how specific settings and labelling parameters are suitable to obtain satisfactory training metrics leveraging convolutional neural networks (CNNs) with a limited number of images [13]. In our current work, this Dataset244 is used to validate the proposed workflow.

[1] https://www.agisoft.com/features/professional-edition/

2.3 Video pre-processing and image fragmentation for 3D point cloud generation

To be able to generate and work with models in 3D, we processed the frames in our video recordings so as to obtain 3D point cloud models. In this section, we detail the steps we took to achieve this.

2.3.1 Multi-spectral video image processing

To prepare our videos for Metashape, we converted the .mp4 files into .mov files with identical characteristics but a smaller file size. Subsequently, we fragmented the .mov files into Multichannel TIFF images (Metashape natively supports the handling of multispectral data saved as multichannel single-page TIFF files). This workflow remains consistent with the standard approach used for RGB photos, with the exception of an additional primary channel selection step performed after image addition to the project.

The key steps involved in processing multispectral images using Metashape, then, are as follows:

1. Loading Images into Metashape: We imported the multispectral images into the Metashape software environment.
2. Image Inspection and Camera Alignment: After loading, we carefully inspected the images, removing manually any unnecessary sections of model geometry. Subsequently, we had Metashape align the camera poses on the different images to ensure accurate geometric correspondence. Metashape also aligns the cameras into calibration groups automatically based on the image resolution and/or EXIF metadata like camera type and focal length.
3. Dense Point Cloud and Mesh Generation: Metashape was employed to create a dense point cloud and a 3D polygonal mesh model from the aligned images.
4. Texture Generation: The software's texture generation feature allowed us to create various types of textures for the model.
5. Tiled Model and Digital Elevation Model (DEM) Construction: We built a tiled model and derived a digital elevation model (DEM) from the processed data.
6. Exporting Results: Finally, the results were exported for further analysis and visualization.

In the following, we provide more information on the key concepts that arise during the above steps.

2.3.2 Sparse and dense point cloud generation

It is important to highlight the difference between the concepts of “sparse point clouds” and "dense point clouds".

A *sparse point cloud* is generated during the initial alignment step in the photogrammetric workflow. It relies on tie points (common features detected across multiple images) to establish geometric correspondence between images and its primary purpose is to determine which image pairs overlap sufficiently. These overlapping pairs then serve as a basis for subsequent depth map generation. Thus, the goal of the sparse point cloud is to provide a rough representation of the scene’s 3D structure. Distortion parameters and camera positions are still unknown at this stage.

In contrast, a *dense point cloud* is created after the alignment step: depth maps are computed for overlapping image pairs, and these depth maps are then merged to form the dense point cloud. The dense point cloud aims for high fidelity and accuracy. It captures detailed 3D information about the scene. In particular, it contains a significantly larger number of points, represents the fine details of the object or environment, it is used for creating 3D models, orthomosaics, and digital elevation models (DEMs). It requires accurate camera positions and distortion parameters.

In our case, these processes, including the camera alignment, are automatically performed based on EXIF metadata[19, 20]. Thus, photo alignment is determined by the camera’s position at the time of image capture. This latter is defined by both interior and exterior orientation parameters, which depend on the original camera’s point of view.

The outcome of this processing step includes estimated exterior (translation and rotation) and interior camera orientation parameters, along with a tie point cloud containing triangulated positions of matched image points.

2.3.3 Accuracy of point cloud image through tie points

A tie point in a digital image corresponds to the same location in an adjacent image. Typically expressed as a pair, tie points serve to connect images and facilitate the creation of mosaics. This value plays a crucial role in defining the image cloud; specifically, the sparse point cloud consists of tie points. Each tie point corresponds to a feature that has been identified in multiple photos and is deemed a valid match. When operating at the High accuracy setting, the software processes photos at their original size. However, we should note that the Medium setting results in image downscaling by a factor of 4 (reducing each side by 2 times). To construct a dense point cloud image from our video, Metashape calculates both exterior and interior image orientation parameters.

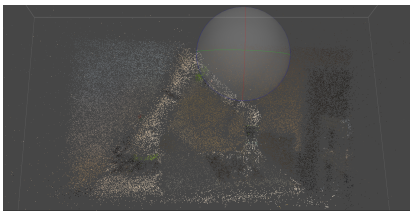

Figure 1
Typical example of a sparse 3D point cloud image from the Cobot150 dataset.

For our video, we used Metashape to align a total of 150 frames from the Cobot150 dataset. In the process, a total of 57,543 tie points were detected; the software generated 396,926 projections based on the camera positions and orientations, and computed 150 depth maps. The resulting point clouds contained 176,651 points, which represents a medium quality level.

2.3.4 Resulting 3D point cloud images

At the end of the workflow described in the previous sections, we obtained 3D point cloud images from a video of a cobot working in real-time. In particular,

a typical sparse point cloud image obtained for our cobot video can be seen on Figure 1. The same images have also been generated as dense point clouds, as shown on Figure 2. We also generated virtual-reality-ready Potree files of resulting 3D point clouds.

It is important to highlight that point clouds can be extremely useful in applications such as robotics, augmented reality, and self-driving [21]. For our research, deep learning models represent an effective data-driven approach for acquiring information from 3D point cloud data, leveraging Convolutional Neural Networks (CNNs) [21][22][23]. Through these 3D point cloud images and our trained custom dataset, Dataset244, we were able to validate our workflow as demonstrated in the next section.

2.4 Segmentation with YOLOv8 in Pytorch

In AI vision applications, the task of object segmentation entails assigning each pixel of an image to a category of object; whereas instance segmentation entails assigning a separate bounding box or a separate mask to each instance of the same object category.

Both object and instance segmentation are widely researched topics in the context

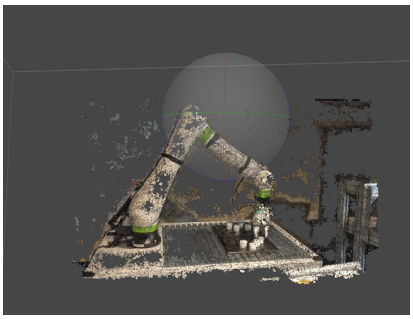

Figure 2
Typical example of a dense 3D point cloud from the Cobot150 dataset. As shown on the figure, dense point clouds contain more details, including textures besides the tie point locations.

of 3D point clouds [24, 25, 26, 27], and deep learning models are particularly well-suited to this application [28].

Taking this into consideration, we chose to use YOLOv8 (You Only Look Once version 8) [17] to make inferences on our 3D point cloud images. Although the model was initially intended for object detection tasks, it also proved to be useful for our goal of segmenting 3D point clouds, especially with its new features (for another recent summary of the model's capabilities, see [29]):

- **Mosaic Data Augmentation** Mosaic data augmentation involves blending four images into a new one to provide the model with richer contextual information during training. Notably, this augmentation technique is applied until the last ten training epochs, contributing to improved performance.
- **Anchor-Free Detection** The model directly predicts an object's mid-point, reducing the number of bounding box predictions. This approach accelerates Non-max Suppression (NMS), a crucial pre-processing step that filters out incorrect predictions.
- **Built-In Support for 3D Point Clouds** The model is capable of performing inference on 3D point clouds without modification.

For this study, we utilized our Cobot videos, our customized Cobot Dataset244 and the results obtained from our previous research.

2.5 Possibilities for VR integration

By saving the resulting neural network model in a TFLite file, and exporting the resulting segmented point clouds as potree files, it is possible to develop a

Model parameters for a virtual reality (VR) prototype using convolutional neural networks (CNNs) : DTnet"

Computational language

Python	12 torch-2.0.1+cu118 CUDA:0 (Tesla T4, 15102MiB)

Hyper-parameters for images and video to create Dataset for CNNs based trainig model

Minimum requirement of images for dataset validated	**Number**	**Size**	**Color**
	>106,	> 1200x800,	RGB
Video parameters	**Time**	**Size**	**Color -Codec**
	< 24 sec	1920x1980	HD, AAC-H-264

CNN training algorithm settings

Algorithm	Yolov8		
Detection	BBox and Mask		
Images in dataset	106	640x640	RGB
Video format and size	Mp4	1920x1980	HD
Software for annotation	Roboflow		
Type of annotation	Instance segmentation		
Classes for annotation	3: Cobot, person, display		
Distribution of Classes subsets for training	3: Cobot, person, display		
Training data	10 with 500 steps, 261 layers,11791257 parameters, 11791241 gradients, 42.7 GFLOPs		
Data augmentation	Mosaic and Albumation		
Confidence -recall values	0.238 for Box and 0.165 for Mask		
Precision all classes	0.634 for Box and 0.635 for Mask		
mAP all classes	0.796		
IoU	Threshold of 0.5		

Export setting to create VR image when using Point Cloud

Type	**Potree**

Setting to export model training as API for Android application

Export Phyton. ply scripts file as Tensorlite

Settings to create Point Cloud from custom video for VR

Software	Agisoft Metashape	**Alignment memory usage**	166.90 MB
Cameras	150	**Date created**	2022:11:26 12:13:20
Aligned cameras	150	**Software version**	1.8.4.14856
Point Cloud	Point Cloud	**File size**	12.31 MB
Points	57,543 of 136,482	**Depth Maps**	Depth Maps
RMS reprojection error	1.264 (4.7116 pix)	**Count**	150
Max reprojection error	9.99569 (141.637 pix)	**Depth maps generation parameters**	Depth maps generation parameters
Mean key point size	4.24893 pix	**Quality**	Medium
Point colors	3 bands, uint8	**Filtering mode**	Moderate
Key points	No	**Max neighbors**	16
Average tie point multiplicity	5.08175	**Processing time**	40 minutes 10 seconds
Alignment parameters		**File size**	7.17 MB
Accuracy	High	**Dense Point Cloud**	Dense Point Cloud
Generic preselection	Yes	**Points**	176,651
Reference preselection	Sequential	**Point colors**	3 bands, uint8
Sequential			
Key point limit	400,000	**Depth maps generation parameters**	Depth maps generation parameters
Key point limit per Mpx	30,000	**Quality**	Medium
Tie point limit	200,000	**Filtering mode**	Moderate
Exclude stationary tie p.	No	**Max neighbors**	16
Guided image matching	Yes	**Processing time**	40 minutes 10 seconds
Adaptive camera model fitting	Yes	**Dense cloud generation parameters**	Dense cloud generation parameters
Matching time	1 minutes 37 seconds	**Processing time**	55 seconds
Matching memory usage	363.09 MB	**Date created 2022:11:26 12:07:25**	2022:11:26 12:07:25
Alignment time	5 minutes 16 seconds	**Software version**	1.8.4.14856
Alignment memory usage	166.90 MB	**File size**	2.54 MB
Date created	2022:11:26 12:13:20	**System**	System
Software version	1.8.4.14856	**Software name**	Agisoft Metashape
File size	12.31 MB	**OS Mac OS 64 bit**	OS Mac OS 64 bit
Depth Maps	Depth Maps	**RAM 1**	16.00 GB
Count	150	**CPU**	Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz

Figure 3
Parameters for "DTNet" workflow for the rapid prototyping of digital twins in virtual reality environments.

workflow for the rapid prototyping of digital twins. We refer to this workflow as the "DTNet Workflow" (Digital Twin Net Workflow). It is important to highlight that the integration of Digital Twins (DT) and Virtual Reality (VR) can create not only a digital platform to boost the industrial process but can also serve as a digital training platform for industrial operators. This platform allows operators to experience a more lifelike environment for digital learning [30].

The parameters used for the entire workflow are shown on Figure 3.

3 Validation of Workflow Through a Sample Use-Case

3.1 Creation of a cobot dataset for instance segmentation of 3D point cloud images

Training datasets are pivotal in machine learning, as they enable relevant features to be extracted for future generalization. Further, a robust dataset not only solves this original problem but also supports the development of new applications combined with transfer learning [31, 6]. Unfortunately, publicly available datasets suitable for instance segmentation, not to mention instance segmentation in industrial scenarios, remain scarce [32],[33] [34].

Expanding upon the results of our previous work, we employed our custom dataset

of 108 images (Dataset108) to fine-tune the YOLO-v8 architecture [13]. Now, unlike in our previous approach, the dataset augmentation was achieved directly through the YOLOv8 model itself.

Our initial step was to use Roboflow Annotate [35], a tool designed for labeling and annotation for various tasks, including segmentation. In particular, we applied to our dataset, Segment Anything Model (SAM) for Smart Polygon Annotation and generated records in the COCO format (Common Objects in Context). For computer vision applications, this format is frequently used as a dataset and benchmark. Based on JSON, its annotation format offers structured data about object instances, together with segmentation masks, bounding boxes, and other pertinent information.

As opposed to our earlier labelling, which took into account terms like “Joint”, “Link”, and “Cobot”, in this case we have used the terms “Display” and “Person” besides “Cobot”.

The definition of these classes comes from the need to improve gesture recognition and body tracking for interactive systems, like virtual reality and smart home devices, as demanded by I5.0 in the field of human-machine interaction. In particular, we highlight that cobots work alongside humans in shared workspaces. Ensuring their safety and preventing collisions is paramount. By accurately detecting and differentiating between humans and cobots, we can implement safety protocols: for instance, if a human gets too close to a cobot, it can slow down or stop functioning to avoid unintended contact. This collaboration enhances productivity and efficiency while minimizing risks.

Additionally, we have transitioned from object labeling to semantic labeling. This segmentation aims to recognize a 3D point cloud image generated from real-time video captured by an Industrial Cobot, through YOLOv8. Instance segmentation extends beyond simple object detection by providing not only bounding boxes but also detailed shapes for each prediction. Instead of merely defining a bounding box using a center point, width, and height, instance segmentation involves complex image annotation. Specifically, we use polygons to delineate each object individually, along with its precise bounding box coordinates and pixel-wise segmentation mask [36].

Figure 4 shows an example of a segmented image from Dataset244. Altogether, within the original 108 images, we segmented 53 instances of “Person”, 125 instances of “Cobot”, and 27 instances of “Display”.

3.2 Data augmentation techniques adopted for our model

As mentioned in the section 3.1, we used YOLOv8 to apply augmentation to the dataset, using the two different approachesof mosaic and albumentation based augmentation.

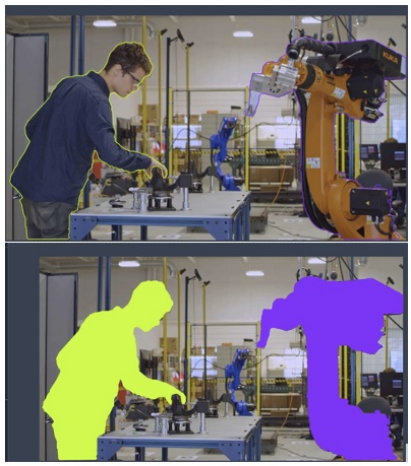

Figure 4
Example segmentation of Dataset108

To use the "Mosaic" approach, we defined in the config.yaml file the mosaic parameter to 0.3, which means that this method was used only 30% of the time.

In addition, "Albumentation" settings included:

- Blur (with a probability of 0.01 and blur limits ranging from 3 to 7)
- MedianBlur (with a probability of 0.01 and blur limits ranging from 3 to 7)
- To Gray (with a probability of 0.01)
- CLAHE (with a probability of 0.01, clip limit between 1 and 4.0, and a tile grid size of 8x8)

Both augmentation methods were used in the case of the input images and their corresponding label mask.

3.3 Setting of parameters to train Dataset108

After creating our unique dataset and our 3D point cloud images based on the video recordings (as discussed in Section 2), we used the Ultralytics version of YOLOv8.0.28, which is based on the most recent developments in deep learning and computer vision, to train our model on just the Dataset244 dataset. We trained the model using version 12 torch-2.0.1+cu118 (Tesla T4, 15102MiB).

In the initial training phase, we focused on studying the impact of Mask R-CNN hyperparameters on the final model. Each model obtained during this phase resulted from 10 training epochs, with parameters set to $patience = 5$ and the capabilities of the GPU allowed us to extend the batch size to 16 .Additionally, to expedite the training process, we adjusted the image shape by using a 3-channel image of 640x640 pixels for both training and validation. Each epoch consisted of 500 steps. Based on this, we trained 261 layers containing $11,791,257$ parameters.

4 Metrics considered to evaluate performance of our training model

To validate our assumptions and for further evaluations, we considered the following metrics:

- **Precision Box / Mask** measures the proportion of true positive bounding boxes (correctly predicted objects) out of the total predicted bounding boxes (true positives + false positives) providing metrics on how many boxes are correct. The mask, similar to precision box, measures the correctness of predicted masks for segmentation. Thus, precision has to do with the ability of the model to report only bounding boxes or masks that really do exist.

$$\text{Precision(Box/Mask)} = \frac{TP(B/M)}{TP(B/M) + FP(B/M)} \tag{1}$$

  where

  - TP(B/M): True positive bounding boxes in box or mask case
  - FP(B/M): False positive bounding boxes in box or mask case

- **Recall**, also called sensitivity, measures the proportion of true positive bounding boxes out of the total ground truth bounding boxes and/or true positive masks out of the total ground truth masks (true positives + false negatives) providing metrics on how many objects we correctly detected. Thus, recall has to do with the ability to identify bounding boxes / masks among all of the bounding boxes / masks.

$$\text{Recall(Box/Mask)} = \frac{TP(B/M)}{TP(B/M) + FN(B/M)} \tag{2}$$

- The **F1 score** combines precision and recall into a single metric using their harmonic mean. This can be an especially useful metric when class distribution is imbalanced.

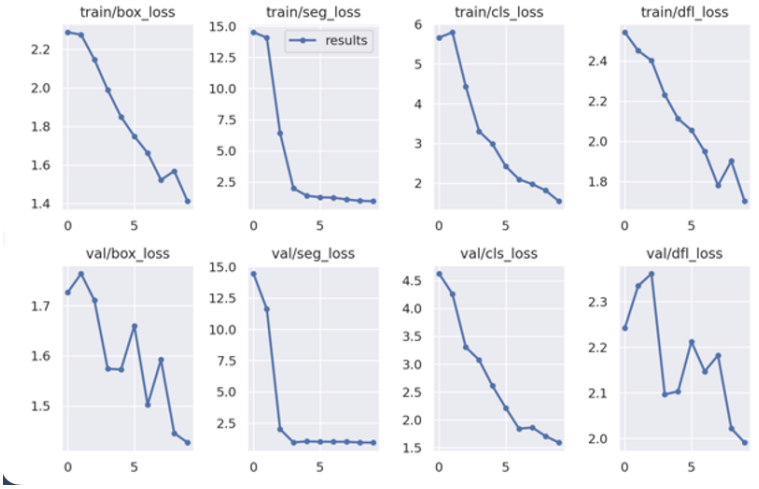

Figure 5
Training set and validation set loss values during the training epochs.

$$F1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3}$$

- The **Intersection over Union (IoU)** quantifies the overlap between a predicted bounding box or mask and the corresponding ground truth (actual) bounding box or mask in terms of number of pixels or some other metric of spatial extent (however, since this is a ratio, it is dimensionless). The IoU value ranges from 0 to 1, specifically if IoU = 0: No overlap (completely disjoint regions), if IoU = 1: Perfect overlap (predicted and ground truth regions are identical). A higher IoU indicates better alignment between the predicted and actual regions. It helps evaluate the accuracy of localization and segmentation models. *In this work, if the IoU between a predicted bounding box and the ground truth bounding box is greater than our predefined threshold (0.5), we consider it a correct detection.*

- **Average precision**, or AP is the area under the precision-recall curve obtained by plotting the precision and recall for different threshold values of IoU. The point of this metric is that if a low IoU is tolerated, even small overlaps will be categorized as correct, meaning that recall can be higher (in general), however, in general this is also at the expense of precision. Hence, it is possible to plot the precision values corresponding to different recall values for different threshold values. The area under this curve is the average precision.

- **mAP**, or **Mean Average Precision**, is used for evaluating object detection models. This is the mean of all average class precisions across all classes:

$$mAP = \frac{1}{N_{\mathrm{c}}} \sum_{n_c=1}^{N_{\mathrm{c}}} \frac{1}{N_{\mathrm{gt}}^{n_c}} \sum_{r} AP_{n_c}(r) \tag{4}$$

  where

  - N_C: Number of object classes
  - $N_{\mathrm{gt}}^{n_c}$: Number of ground truth instances for class (n_c)
  - $AP_{n_c}(r)$: Average precision for class (n_c) at IoU threshold (r)

  In the remainder of this paper, when the value of r is restricted to a single value, 0.5, we will write mAP as mAP50. When r ranges between e.g. 0.5 and 0.95, we will write mAP50-95.

All these metrics play a crucial role in quantifying a model's scalability, affecting storage capacity and computational complexity during the learning process. In particular, they help to assess the performance of object detection and segmentation models, considering both precision and recall. Our goals are to achieve through the proposed model higher values with regular handling for precision, recall, F1 score, IoU and mAP.

5 Experimental results

5.1 Training losses

Figure 5 shows the values of location (box), segmentation (seg), classification (cls) and distribution focal loss (dfl) in different training epochs. The top row of graphs represents results for the training set, while the bottom row represents the validation set. As we can see, these losses show a steady decline on both the training and validation set, although there are more fluctuations in the case of the validation set, given that these examples were not used to update the model.

5.2 Precision, Recall and Mean Average Precision

The performance of our object detection model, fine-tuned on the original Dataset108 (with augmentations by YOLOv8), and assessed on the point cloud images can be summarized as follows:

- Precision(Box): 0.777 for all instances, 0.647 for cobots and 0.907 for persons.

- Recall(Box) : 0.852 for all instances, 0.815 for cobots and 0.889 for persons.
- mAP50: 0.869 for all instances, 0.755 for cobots and 0.984 for persons.
- mAP50–95: 0.467 for all instances, 0.35 for cobots and 0.584 for persons.
- Precision(Mask): 0.708 for all instances, 0.619 for cobots and 0.796 for persons.

This demonstrates that our model performs reasonably well across all classes, with high precision and recall.

The mAP50 score indicates good accuracy at a moderate IoU threshold. However, the mAP50–95 score suggests that performance drops significantly at higher IoU thresholds. Cobots have slightly lower performance compared to all instances, but considering that we trained a small datset, this is to be expected.

5.3 True positives and false negatives

We generated a confusion matrix to show the number of correct and incorrect predictions made by our classifier per category type.

For our model, we obtained:

- ***Cobot:*** True Positive (TP): 0.74 (correctly identified as Cobot). False Negative (FN): 0 (actually a Cobot but predicted as Person or Background).
- ***Person:*** TP: 0.78 (correctly identified as Person). FN: 0.22 (actually a Person but predicted as a Cobot or Background).

5.4 F1-confidence curves

As mentioned in section 4, in the context of instance segmentation tasks, we must consider an critical threshold of overlap: Intersection over Union (IoU). IoU quantifies the overlap between predicted object regions and ground truth annotations. For the graphs presented here, an IoU threshold of 0.5 was employed.

Figure 6 shows F1 values for different confidence levels of box and mask prediction. The blue line represents the cobot class, the orange line represents the person class and the grey line represents all classes combined.

We obtained:

- The average F1 score for all classes is 0.82, with a standard deviation of 0.238 in the case of box predictions.

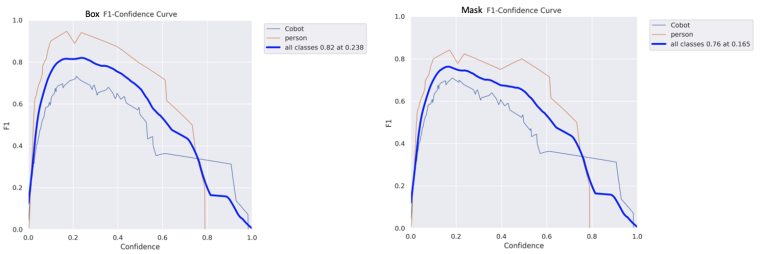

Figure 6
F1-confidence curves for box and mask predictions.

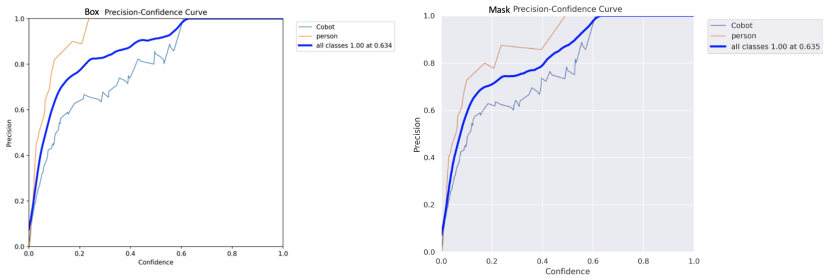

Figure 7
Precision-confidence curves for box and mask predictions.

- The average F1 score for all classes is 0.76, with a standard deviation of 0.165 in the case of mask predictions.

These curves help evaluate the trade-off between confidence (how certain the model is about its predictions) and F1 score (a measure that balances precision and recall). The higher the F1 score, the better the model's performance.

5.5 Precision and recall related curves

Figure 7 shows precision values for different confidence levels of box and mask prediction. The blue line represents the cobot class, the orange line represents the person class and the grey line represents all classes combined.

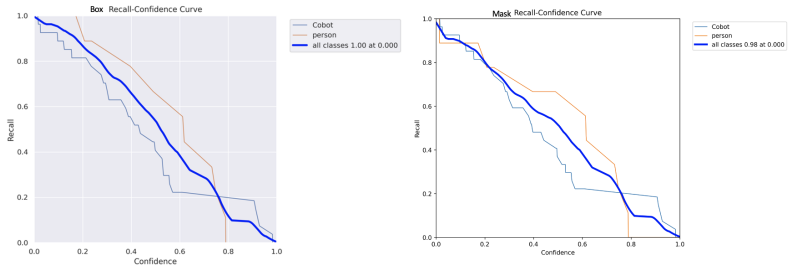

Figure 8
Recall-confidence curves for box and mask predictions.

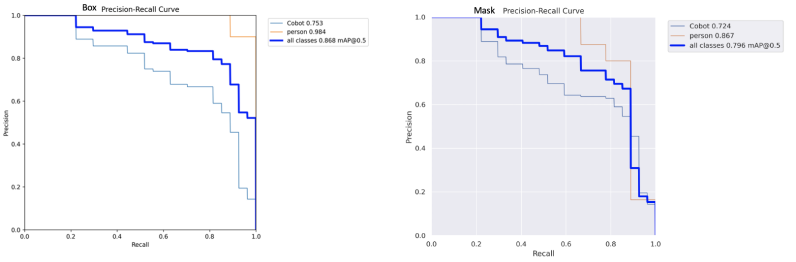

Figure 9
Precision-recall curves for box and mask predictions.

As we can see, the model achieves higher precision for higher confidence levels. This is to be expected, as the model becomes more selective, and so is less likely to report false positives.

In contrast, as shown on Figure 8, the higher the confidence level, the lower the recall, as the more restrictive model is more apt to classify real positive examples as negative, thereby increasing the number of false negatives.

This inverse relationship is also demonstrated on Figure 9, which shows that as the recall increases, the precision also increases for the different categories.

Considering that a good classifier will maintain both a high precision and high recall, our conclusions are:

- For the box method:

  - Cobot: Mean Average Precision (mAP) score: 0.753. As recall increases, the precision decreases gradually.
  - Person: mAP score: 0.984. Precision remains high even at higher recall levels.
  - All Classes (combined): mAP score: 0.868. Precision declines as recall increases

- For the mask method:
  - Cobot: mAP score: 0.724. A similar trend as in the "Box" method – precision decreases with increasing recall.
  - Person: mAP score: 0.867. Precision remains relatively stable across recall levels.
  - All Classes (combined): mAP score: 0.796. Precision decreases as recall increases, but not as sharply as in the "box" case.

These results demonstrate that both methods show trade-offs between precision and recall.

The "mask" method generally achieves slightly lower precision but maintains better overall performance across classes.

5.6 Overall conclusions

In summary, we have shown that:

- The bounding box method for object localization achieves high precision for person detection (mAP score: 0.984) and a slightly lower precision for cobot detection (mAP score: 0.753).
- The mask method for object localization, which incorporates pixel-level segmentation masks, balances precision and recall more evenly; hence, we have seen better overall performance for all classes combined (mAP score: 0.796)

Typical examples of segmentation results are shown on Figure 10.

Conclusions

In our study, we evaluated the classification of 3D point clouds from handheld videos using a model combining instance segmentation, data augmentation, and transfer learning. The model showed promising results in real-time labeling, particularly for Cobot images, in line with Industry 5.0 instance segmentation concerns. Despite a small dataset, our model maintained stable performance and demonstrated potential in bridging 2D and 3D image processing for object recognition.

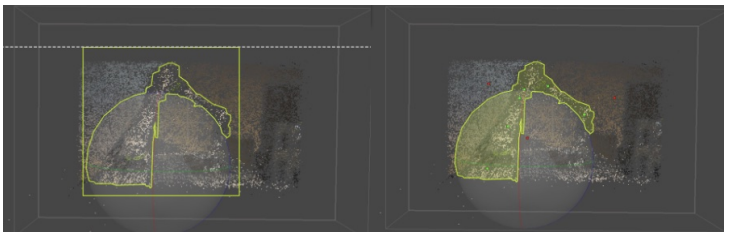

Figure 10
Example of a successful segmentation of a cobot from a 3D point cloud.

A key finding is the precision-recall trade-off, influenced by the dataset size and model architecture, highlighting the importance of IoU and appropriate confidence levels in model evaluation. Moving forward, expanding the dataset and refining the model could improve accuracy, especially in video applications where precision is critical. Our research contributes to advancing deep learning techniques for 3D point cloud processing.

Of course, estimating uncertainties in a custom model's predictions is a difficult task, which is nevertheless crucial for safety-critical applications and decision-making. For these reasons, in our future research we intend to focus on enhanced CNN model architectures such as graph convolutional neural networks (GCNNs), which could perhaps be more specifically tailored to 3D point cloud data.

References

[1] J. Gupta and C. Vegelin, "Sustainable development goals and inclusive development," *International environmental agreements: Politics, law and economics*, vol. 16, pp. 433–448, 2016.

[2] A. Kusiak, "Smart manufacturing," in *Springer Handbook of Automation*, pp. 973–985, Springer, 2023.

[3] J. Qi, L. Ma, Z. Cui, and Y. Yu, "Computer vision-based hand gesture recognition for human-robot interaction: a review," *Complex & Intelligent Systems*, vol. 10, no. 1, pp. 1581–1606, 2024.

[4] N. V. N. Vemuri, "Enhancing human-robot collaboration in industry 4.0 with ai-driven hri," *Power System Technology*, vol. 47, no. 4, pp. 341–358, 2023.

[5] E. Costa, "Industry 5.0 and sdg 9: a symbiotic dance towards sustainable transformation," *Sustainable Earth Reviews*, vol. 7, no. 1, p. 4, 2024.

[6] C. Monsone and Á. B. Csapó, "Charting the state-of-the-art in the application of convolutional neural networks to quality control in industry 4.0 and smart

manufacturing," in *2019 10th IEEE International Conference on Cognitive Infocommunications (CogInfoCom)*, pp. 463–468, IEEE, 2019.

[7] S. Borkman, A. Crespi, S. Dhakad, S. Ganguly, J. Hogins, Y.-C. Jhang, M. Kamalzadeh, B. Li, S. Leal, P. Parisi, *et al.*, "Unity perception: Generate synthetic data for computer vision," *arXiv preprint arXiv:2107.04259*, 2021.

[8] J. Moehrmann and G. Heidemann, "Efficient annotation of image data sets for computer vision applications," in *Proceedings of the 1st International Workshop on Visual Interfaces for Ground Truth Collection in Computer Vision Applications*, pp. 1–6, 2012.

[9] L. Gonzo, A. Simoni, M. Gottardi, D. Stoppa, and J.-A. Beraldin, "Smart sensors for 3d digitization," in *IMTC 2001. Proceedings of the 18th IEEE Instrumentation and Measurement Technology Conference. Rediscovering Measurement in the Age of Informatics (Cat. No. 01CH 37188)*, vol. 1, pp. 117–122, IEEE, 2001.

[10] P. Vivet, G. Sicard, L. Millet, S. Chevobbe, K. B. Chehida, L. A. Cubero, M. Alegre, M. Bouvier, A. Valentian, M. Lepecq, *et al.*, "Advanced 3d technologies and architectures for 3d smart image sensors," in *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 674–679, IEEE, 2019.

[11] D. Moon, S. Chung, S. Kwon, J. Seo, and J. Shin, "Comparison and utilization of point cloud generated from photogrammetry and laser scanning: 3d world model for smart heavy equipment planning," *Automation in Construction*, vol. 98, pp. 322–331, 2019.

[12] C. R. Monsone and J. Jósvai, "Challenges of digital twin system," *Acta Technica Jaurinensis*, vol. 12, no. 3, pp. 252–267, 2019.

[13] C. R. Monsone and A. Csapo, "A Cobot Image Dataset for the Creation of Digital Twins in Industrial Robot Applications," in *12th IEEE International Conference on Cognitive Infocommunications: CogInfoCom 2021*, pp. pp. 847–852, 2021.

[14] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[15] S. Catalucci, R. Marsili, M. Moretti, and G. Rossi, "Comparison between point cloud processing techniques," *Measurement*, vol. 127, pp. 221–226, 2018.

[16] A. Rahman and A. Cahyono, "Analysis of 3-d building modeling using photogrammetric software: Agisoft metashape and micmac," in *IOP Conference Series: Earth and Environmental Science*, vol. 1276(1), p. 012044, IOP Publishing, 2023.

[17] G. Jocher, A. Stoken, J. Borovec, *et al.*, "ultralytics/yolov5: v4. 0-nn. silu () activations, weights & biases logging, pytorch hub integration (version v4. 0). zenodo," 2021.

[18] G. Jocher, A. Stoken, J. Borovec, L. Changyu, A. Hogan, *et al.*, "ultralytics/yolov5: v3. 1-bug fixes and performance improvements," 2020.

[19] D. Gangwar and A. Pathania, "Authentication of digital image using exif metadata and decoding properties," *International Journal of Scientific Research in Computer Science, Engineering and Information Technology (IJSR CSEIT)*, vol. 3, no. 8, pp. 335–341, 2018.

[20] C. Zheng, A. Shrivastava, and A. Owens, "Exif as language: Learning cross-modal associations between images and camera metadata," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6945–6956, 2023.

[21] S. Qiu, S. Anwar, and N. Barnes, "Dense-resolution network for point cloud classification and segmentation," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 3813–3822, 2021.

[22] J. Hong, K. Kim, and H. Lee, "Faster dynamic graph cnn: Faster deep learning on 3d point cloud data," *IEEE Access*, vol. 8, pp. 190529–190538, 2020.

[23] T. T. Mac, C.-Y. Lin, N. G. Huan, L. Duc, P. C. H. Nhat, and H. H. Hai, "Hybrid slam-based exploration of a mobile robot for 3d scenario reconstruction and autonomous navigation," *Acta Polytech. Hung*, vol. 18, pp. 197–212, 2021.

[24] E. Agapaki and I. Brilakis, "Instance segmentation of industrial point cloud data," *Journal of Computing in Civil Engineering*, vol. 35, no. 6, p. 04021022, 2021.

[25] Y. Liao, H. Zhu, Y. Zhang, C. Ye, T. Chen, and J. Fan, "Point cloud instance segmentation with semi-supervised bounding-box mining," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 12, pp. 10159–10170, 2021.

[26] B. Yang, J. Wang, R. Clark, Q. Hu, S. Wang, A. Markham, and N. Trigoni, "Learning object bounding boxes for 3d instance segmentation on point clouds," *Advances in neural information processing systems*, vol. 32, 2019.

[27] W. Wang, R. Yu, Q. Huang, and U. Neumann, "Sgpn: Similarity group proposal network for 3d point cloud instance segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2569–2578, 2018.

[28] T. Kontogianni, E. Celikkan, S. Tang, and K. Schindler, "Interactive object segmentation in 3d point clouds," *arXiv*, 2023.

[29] V. Tadic, A. Odry, Z. Vizvari, Z. Kiraly, I. Felde, and P. Odry, "Electric vehicle charging socket detection using yolov8s model," *Acta Polytechnica Hungarica*, vol. 21, no. 10, 2024.

[30] A. Martínez-Gutiérrez, J. Díez-González, P. Verde, and H. Perez, "Convergence of virtual reality and digital twin technologies to enhance digital operators' training in industry 4.0," *International Journal of Human-Computer Studies*, vol. 180, p. 103136, 2023.

[31] P. Baranyi, A. Csapo, and G. Sallai, *Cognitive Infocommunications (CogInfoCom)*. Springer, 2015.

[32] Y. He, H. Yu, X. Liu, Z. Yang, W. Sun, Y. Wang, Q. Fu, Y. Zou, and A. Mian, "Deep learning based 3d segmentation: A survey," *arXiv preprint arXiv:2103.05423*, 2021.

[33] P. Wang, Y. Tang, F. Luo, L. Wang, C. Li, Q. Niu, and H. Li, "Weed25: A deep learning dataset for weed identification," *Frontiers in Plant Science*, vol. 13, p. 1053329, 2022.

[34] H. Choi, E. Hosseini, S. R. Alvar, R. A. Cohen, and I. V. Bajić, "A dataset of labelled objects on raw video sequences," *Data in Brief*, vol. 34, p. 106701, 2021.

[35] B. Dwyer, J. Nelson, J. Solawetz, *et al.*, "Roboflow (version 1.0)[software](2021)."

[36] D. Bolya, C. Zhou, F. Xiao, and Y. J. Lee, "Yolact: Real-time instance segmentation," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 9157–9166, 2019.