Annals of Nuclear Energy 213 (2025) 111080

Contents lists available at ScienceDirect

Annals of Nuclear Energy

journal homepage: www.elsevier.com/locate/anucene

Semi-automatic image annotation using 3D LiDAR projections and depth camera data

Pei Yao Li [a,*], Nicholas A. Parrilla [a], Marco Salathe [a], Tenzing H. Joshi [a], Reynold J. Cooper [a], Ki Park [b], Asa V. Sudderth [b]

[a] Lawrence Berkeley National Laboratory, 1 Cyclotron Rd, Berkeley, CA, 94720, USA
[b] Nevada National Security Sites (NLV Facility), 232 Energy Way, North Las Vegas, NV, 89030, USA

ARTICLE INFO

Keywords:
Computer vision
Object recognition neural networks
LiDAR-assisted image annotation
Nuclear safeguards

ABSTRACT

Efficient image annotation is necessary to utilize deep learning object recognition neural networks in nuclear safeguards, such as for the detection and localization of target objects like nuclear material containers (NMCs). This capability can help automate the inventory accounting of different types of NMCs within nuclear storage facilities. The conventional manual annotation process is labor-intensive and time-consuming, hindering the rapid deployment of deep learning models for NMC identifications. This paper introduces a novel semi-automatic method for annotating 2D images of nuclear material containers (NMCs) by combining 3D light detection and ranging (LiDAR) data with color and depth camera images collected from a handheld scan system. The annotation pipeline involves an operator manually marking new target objects on a LiDAR-generated map, and projecting these 3D locations to images, thereby automatically creating annotations from the projections. The semi-automatic approach significantly reduces manual efforts and the expertise in image annotation that is required to perform the task, allowing deep learning models to be trained on-site within a few hours. The paper compares the performance of models trained on datasets annotated through various methods, including semi-automatic, manual, and commercial annotation services. The evaluation demonstrates that the semi-automatic annotation method achieves comparable or superior results, with a mean average precision (mAP) above 0.9, showcasing its efficiency in training object recognition models. Additionally, the paper explores the application of the proposed method to instance segmentation, achieving promising results in detecting multiple types of NMCs in various formations.

1. Introduction

Over the past decade, immense progress in object recognition has been made by utilizing deep learning algorithms. This growth has permitted machines to recognize and localize target objects in real time. This ability finds various applications to improve process efficiency in nuclear safeguards by simplifying time-consuming inspection tasks (Haddal and Hayden, 2018). One of these applications, crucial to nuclear material accountancy, is object localization and counting. For example, a typical nuclear storage facility could contain hundreds of nuclear material containers (NMCs) lying on the floor or stacked on shelves. Having a trained object recognition model combined with other sensor data can be used to locate and estimate the number of these objects and simplify the workload of inspectors considerably (Salathe et al., 2024). However, preparing an annotated ground truth dataset to train these object recognition neural network models is a tedious and manually intensive task. The current standard object annotation process requires the masks or bounding boxes outlining the target objects to be manually placed onto the images one by one using bounding box/mask labeling software. Only after that can the annotated images be used to train a deep-learning computer vision model for object recognition. By automating the annotation process, the entire process from data annotation to object recognition can become more streamlined, and manual efforts can be minimized.

This paper presents a novel method to create annotated images of new objects requiring little human intervention. First, the inspector scans the room with a handheld multi-sensor system and collects a dataset of target containers. The inspector then marks the locations of the new target objects on a map created with the LiDAR data. The locations of the objects are then transferred to the images and the objects' annotations are automatically created. This procedure will allow deep-learning models of new types of NMCs to be trained on-site within a few hours. On-site model training is important as nuclear facilities are

* Corresponding author.
E-mail address: pyli@lbl.gov (P.Y. Li).

https://doi.org/10.1016/j.anucene.2024.111080
Received 7 February 2024; Received in revised form 9 August 2024; Accepted 18 November 2024
Available online 4 December 2024

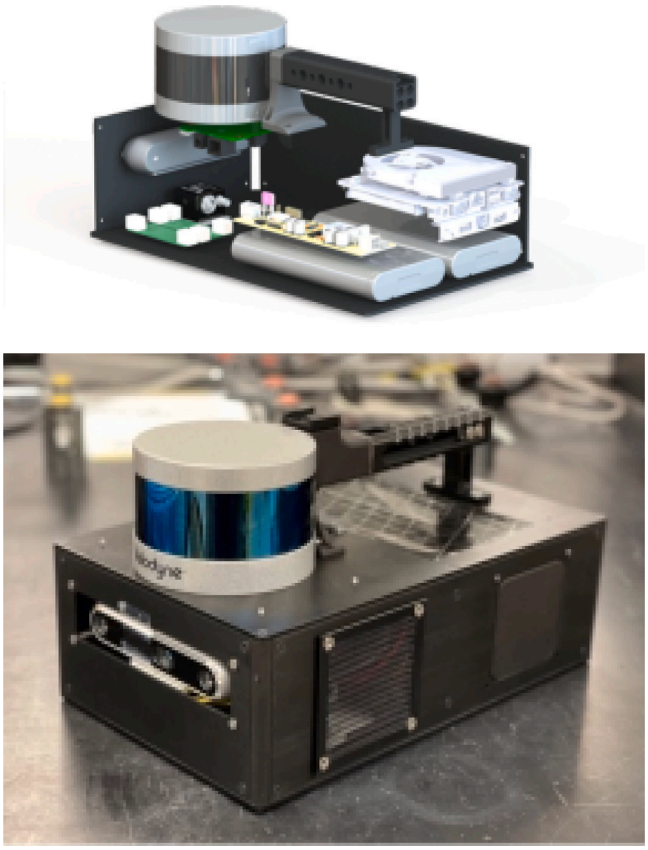


Fig. 1. Rendering (top) and photograph (bottom) of the CLAMP system.

sensitive and carrying facility imagery outside or uploading onto the cloud is often not possible.

The following sections will describe the hand-held system and the collection and annotation of relevant data with the semi-automatic annotation procedure. It then compares the performance of neural networks trained with semi-automatically annotated data, commercially annotated data, and manually annotated data. It will be shown that the semi-automatic annotation process produces annotated data of similar quality compared to the manually annotated process. This new annotation procedure would allow for thousands of images to be annotated in less than 20 min instead of hours, and then allow for the object recognition neural network model to be trained within the same day the data is collected.

2. Relevant work

Detecting uncommon objects with specific characteristics such as NMCs differs from object recognition of common everyday objects encountered in fields such as autonomous vehicles and robotics. Training data for common objects such as people, vehicles, and bicycles can all be found in open-source annotation datasets such as COCO and ImageNet (Lin et al., 2014; Deng et al., 2009; Tian et al., 2020). However, there is no existing annotated dataset for any type of NMC. Furthermore, although most classes of NMC have the same shape and color, the subtle difference in size and they way their bolts look are the only few features that distinguish them from each other. Object recognition labels are generally either in a rectangular bounding box format or polygon mask format that outlines the edges of the object shape. The quality of a new ground truth dataset with annotated NMCs will significantly impact the performance of the object recognition model trained on it.

Although there can be drawbacks associated with human-annotated datasets including time, effort, and errors, this method is often still used for annotation of new objects that require domain-specific knowledge (Wei et al., 2022). Commonly used methods involved using labeling tools such as “Labelimg” or “CVAT” to display the images one by one and have an annotator manually place and adjust the size of the bounding boxes or polygon masks on target objects (Tzutalin, 2015; Intel, 2020). Many studies have investigated the impact of variances and inaccuracies of human-drawn bounding boxes on neural network performance (Wu et al., 2023) and instructions for how to draw optimal bounding boxes are readily available (Rizzoli, 2021). To achieve consistent performance with such a manually annotated dataset, the annotation task needs to be performed by someone trained in annotating objects for deep learning models. This is not desirable in safeguard applications as the operators in the field are typically not trained in proper image annotation techniques, nor do they have the time to do so.

Many commercial and open-source 2D image and 3D point cloud annotation tools are available (The MathWorks, Inc., 2022; OÜ, 2023). The annotation step can also be outsourced to annotation services with detailed instructions guiding bounding box drawing such as Amazon Sagemaker (Amazon Web Services, Inc., 2023b). However, the turnaround time to receive the annotations back from commercial services is generally a bit longer and more error-prone than labeling them internally. Using external services is also not feasible for annotating classified images.

Efficient annotation approaches minimizing human involvement are discussed in the literature. One method is to use neural networks to detect objects in images and draw bounding boxes or masks around them or, in the case of 3D, implement neural network object recognition on LiDAR point clouds (Chen et al., 2022). These neural networks often leverage automation techniques to produce training data; for example, using similarity metrics to validate object type (Radenović et al., 2019), or using features of the data, such as whether or not an object is moving, to generate a label for that object. Other methods utilize key points of an object to define the physical extent and highlight features of an object (You et al., 2020). There have also been explorations on semantically segmenting the precise object contours via foreground segmentation given approximate object locations (e.g. manually labeled bounding boxes) in a point cloud, and prior knowledge of object shapes (Chen et al., 2014). Recently, large commercial companies have branched into the image annotation space, such as META, with the “Segment Anything” tool which can create mask segmentation of an object after a single click by the user; however, it requires a GPU to efficiently run. Furthermore, the segmented masks do not contain any class labels (Kirillov et al., 2023).

While these methods hold promise for speeding up the annotation process or improving precision, many still depend on COCO and ImageNet annotations, and humans are generally required to validate or edit the annotations. Furthermore, typical safeguard scenarios require operations in areas with many other similarly shaped and colored objects; so these methods would not produce high accuracy in locating and annotating NMCs. Hence, an algorithm that can fill a niche of efficiently annotating static objects of simple shapes and minimizes manual effort, time, and annotation domain expertise while retaining label accuracy is desirable.

3. Data collection

3.1. Sensor system

The Container Localization and Mapping Platform (CLAMP), shown in Fig. 1, was used to collect the datasets discussed in this work. It is a free-moving scanning system, that was designed to map an indoor or outdoor facility and determine how many nuclear material containers

(NMCs) are present in near real-time when carried through a facility during an inspection.

Installed sensors are an Advanced Navigation Spatial Inertial Navigation System INS (Advanced Navigation, 2023), an Intel RealSense D455 depth camera (Intel Corporation, 2023), and a 16-beam Velodyne PUCK LITE LiDAR (Velodyne Lidar, Inc., 2023). For performing all real-time computational tasks the sensors are connected to an Intel NUC i7 edge computer (Simply NUC, 2023). The system is powered by two 99 Wh batteries providing a multi-hour battery life and enabling hot swapping of batteries to assure uninterrupted operation beyond the battery life. Including the two batteries, the system weighs around 8 lbs. In Fig. 1, a CAD drawing of the design and a photograph of the realized system is shown.

The system's software stack is dockerized (Docker, Inc., 2023), assuring stable and consistent operation. It leverages the Robot Operating System (ROS) (Open Source Robotics Foundation, Inc., 2023) to collect time-synchronized sensor data, to process data in real-time for data quality checks, and to store raw data streams on disk. A local web-server hosts the User Interface (UI) which gives an operator the ability to start and stop measurements and to inspect some of the sensor outputs to assure consistent sensor data quality.

3.2. Data collection

The CLAMP unit was brought to the Nevada National Security Sites in September 2021 and in May 2024. For each configuration of containers, the facility was scanned with the CLAMP unit which simultaneously collects color images, depth images, and 3D LiDAR data. In Fig. 3 a top-down map from a simple measurement scenario is shown, and a more challenging scenario is displayed in Fig. 8.

Empty AT-400 containers, stainless steel drums that are typically used to hold fissile materials (Gilbert et al., 2016), were the only type of container available during the first data collection campaign in 2021. Fig. 2 shows an example of AT-400 containers in the facility. These containers serve as an example NMCs and are small enough to be reconfigured without the need for heavy machinery. During the measurements, 12 AT-400 containers were available and 11 different configurations of the 12 containers were laid out for data collection. Some configurations had very simple container formations where the containers were all unobstructed standing upright and individually scattered across the floor. Some other configurations had more complex container formations where containers were tightly packed in a big pile with some laid down on the side and some standing upright. Additionally, in some scenarios, the containers were purposely obstructed by chairs, pelican cases, and other smaller cylinders during data collection. The containers were always placed on the floor and never stacked in this measurement campaign.

In the 2024 measurement campaign, data on 5 additional types of cylindrical containers were added to further test the feasibility of the semi-automatic annotation to container localization methodology. All container types' names and their approximate sizes are displayed in Appendix A. Some data was collected indoors and some outdoors and some AT-400 containers were stacked together on shelves as shown by Fig. 4.

3.3. Data processing

The LiDAR and inertial measurement unit (IMU) data for each dataset were processed offline with faster-lio (Bai et al., 2022), a LiDAR-Inertial Odometry toolkit to create a high-fidelity three-dimensional map of the scene in the form of a point cloud. This point cloud is essential for the NMC mapping, localization, and counting analysis, but will be used in the presented work mainly for the semi-automatic annotation procedure. LiDAR-Inertial Odometry additionally calculates the poses of the system at any given point in time, and thus the camera's location and viewing angle for every image. Furthermore,

Fig. 2. Example of AT-400 containers in a facility.

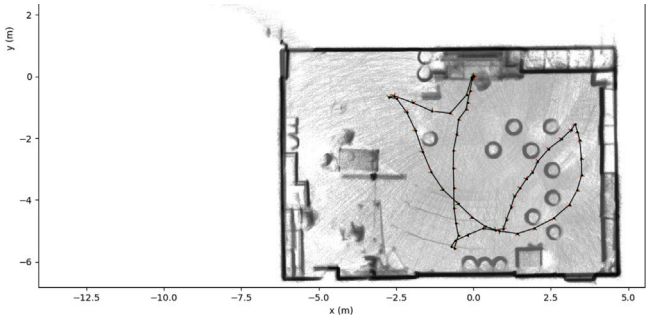


Fig. 3. A top-down view of the resulting point cloud, scale in meters. The black line displays the trajectory of the system during the measurement.

Fig. 4. Stacked containers in the 2024 measurement campaign.

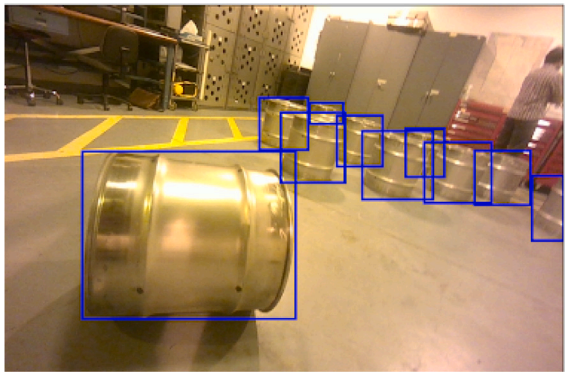


Fig. 5. Example of AWS annotated images.

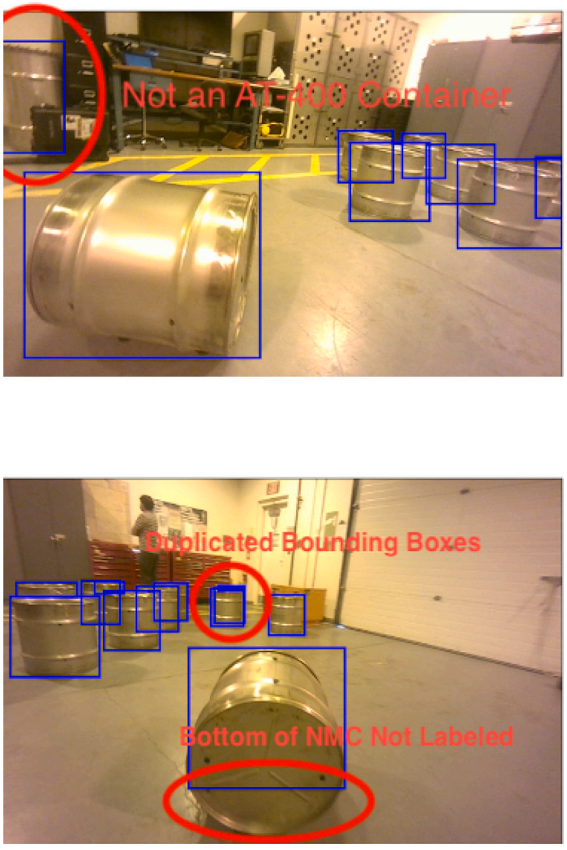


Fig. 6. Examples of mislabeled AWS annotations: Top, containers mislabeled as AT-400 Container by Amazon SageMaker. Bottom, AWS annotations with duplicated bounding boxes per container and a bounding box not completely covering a container.

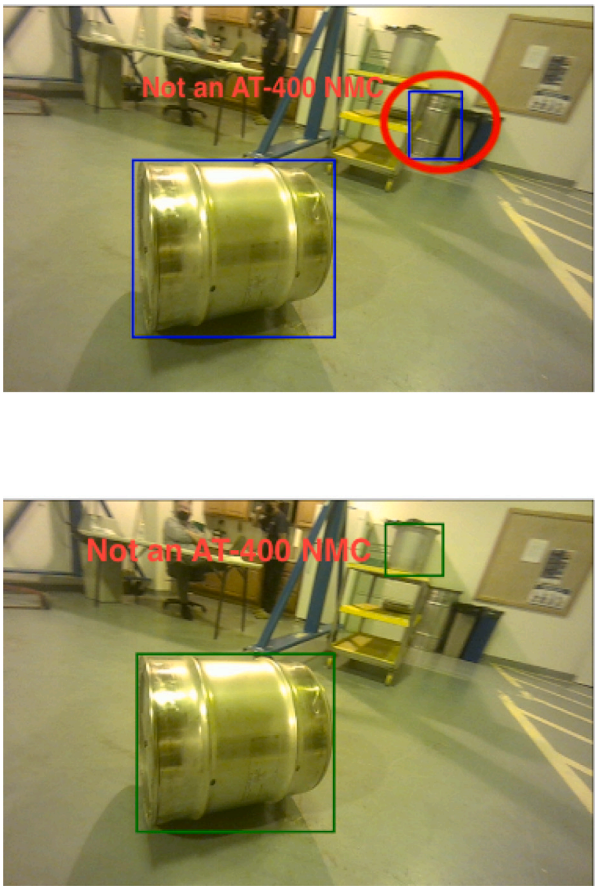


Fig. 7. Examples of Amazon SageMaker producing different incorrect annotations on the same image: Top, the first round of AWS annotations result. Bottom, the second round of AWS annotations result.

thanks to the Realsense camera's ability to sense depth, each color image is accompanied by a depth image, defining pixel by pixel how far away from the camera the represented objects are.

The AT-400 container was selected as the benchmark container and it is also the most abundant in the collected data. From the 11 different scans of the 2021 measurement campaign, 785 images of AT-400 containers were selected to comprise the *benchmark dataset*. This dataset is randomly split into 262 training images, 262 validation images, and 261 testing images.

After the 2024 campaign, a second *extended dataset* was created to test the semi-automatic annotation performance for labeling various types of containers by adding 200 images from the 2024 campaign to each of the original benchmark dataset's training, validation, and testing images. Hence, the extended dataset consists of 462 training images, 462 validation images, and 461 testing images.

3.4. Ground truth and annotation

3.4.1. Amazon SageMaker annotation

The imagery from the benchmark dataset was passed to Amazon Web Services (AWS) Sagemaker Ground Truth annotation service (Amazon Web Services, Inc., 2023b). Amazon Sagemaker Ground Truth annotation service was chosen because it is a very reputable annotation service; thus, it can serve as a benchmark for commercial annotation services. The annotation outputs, in the form of bounding boxes placed around each container, are visualized in an example image in Fig. 5. The AWS annotated images originally had 4906 labels spread over the 785 images that were used. However, it was found that some of the images contained erroneous annotations. Some containers of a different type present in the scene were often mislabeled as AT-400 containers and some images had duplicate bounding boxes drawn around the same container. Examples of such false labels are shown in Fig. 6. Furthermore, when 122 of these images were sent to AWS again, they were annotated differently than the first time, containing different sets of false labels. Examples of differences between the two AWS annotation results can be found in Fig. 7. A second AWS annotated dataset was created, containing 122 images from the second annotation request and the remaining 663 images are kept identical to the first AWS annotated dataset.

3.4.2. Manual annotation

Considerable effort was put into attempting to clean up errors in the AWS annotations, including multiple rounds of reviews that resulted in manually removing hundreds of false annotations from each set of AWS annotations. However, it was found that duplicated bounding boxes closely overlapping on top of each other were too difficult to find and correct; thus, all 785 images were manually annotated with bounding boxes by the authors instead. Therefore, there are four sets of annotations used in this work: Two annotation sets comprise the original, unaltered AWS annotations, one fully manually labeled set, and one with semi-automatic labeled annotations. The manually labeled annotation set is believed to be the most accurate one and is used to evaluate and compare the performances of neural networks trained with annotations created with different methods.

4. Methodology

The proposed annotation pipeline can be summarized as follows: A dataset is collected using a new type of NMC or other target objects placed on the ground. The dataset is processed and a point cloud is assembled by running the LiDAR-Inertial Odometry algorithm. For each new type of target object, the operator needs to define the geometric shape and dimensions in an configuration file. The point cloud is then presented to the operator in the forms of both a top-down and a side-view map and allowing the operator to reduce the full map to a volume containing only objects of interest. This typically removes clutter from the ceiling and walls so the operator can better zoom in on the containers. For the second step, the zoomed-in top-down view of the point cloud is presented to the operator again. This time, the operator selects the poses of all objects of interest. At that point, the operator intervention is completed and the algorithm creates a 3D point cloud model of the object based on the shape and dimensions defined in the configuration file at the selected location with the selected orientation. Since a dataset generally contains thousands of images, a blur filter can select the least blurry images from each video as training images. Then, the 3D model is projected onto the images. Finally, based on these projections, the annotations are created and serve as training data for deep-learning obejct recognition models. As such, the pipeline cannot be classified as fully automatic, but also requires considerably less user intervention compared to manual annotation, ergo the semi-automatic designation of this approach. All scripts for this semi-automatic pipeline are written in Python3. In the following sections, the various steps will be presented in more detail.

For this benchmark evaluation, clicking was considered sufficient for localizing the containers. The blur filters were not used on the benchmark dataset as images were pre-selected and no further filtering was done to ensure the exact same images were labeled by different annotation methodologies for comparison purposes. The rest of the semi-automatic annotation pipeline was the same as the production annotation steps described above.

4.1. Estimation of the target object locations

4.1.1. Crop the XYZ ranges of the scene

The full point cloud data of the facility is very large and hinders the user from focusing on objects of interest. Therefore, the X, Y, and Z ranges of the point cloud scene are first cut by having the operator click on two corners of an axis-aligned rectangular cuboid, both in the top-down view and the side-view of the point cloud scene. Two clicks are required for each view to uniquely define the cuboid. In the top-down view, the operator clicks on the top left and the bottom right corner which indicates the minimum and maximum X and Y range to crop the point cloud scene as shown in Fig. 8. In the side-view, the operator indicates the minimum and maximum Z ranges. After applying these cuts, the following plots are limited to the cropped area, allowing the user to zoom-in on the objects of interest.

4.1.2. Selection of object poses

The selection of an object's locations and orientation happens by presenting the operator with a top-down point cloud view of the facility and having an operator manually click on a few points on the object's surface. It is assumed that the objects to be annotated will mainly be placed on the ground. However, objects stacked on top of each other can also be annotated by editing an CSV output file if there are known heights of the stacked objects. The operator first inputs the target objects' dimensions and shapes into a CSV file. The currently supported shapes are upright cylinders, sideways cylinders, rectangles, and squares. Other shapes can be added by appending codes in the script that create 3D geometric models of the user-defined shapes. Containers that are standing upright and containers lying on their sides are considered to be two different shapes in the operator input CSV file. To select each shape, the operator first presses an associated hotkey defined in the configuration CSV file; the title of the figure will reflect the shape that is currently being selected each time a hotkey is pressed. All points clicked are assumed to be for that selected shape unless the user presses a new hotkey. In the top-down view, the operator clicks once for each upright container in the center of the circles forming the base of the cylinder. The operator needs to click twice on the two midpoints of the cylinder's base for containers lying on the side as shown in Fig. 8; this defines the orientation of the container on the XY plane of the scene. Although the annotation tool also worked well with other geometric shapes, for the purpose of NMC inventory accounting, it is assumed that the only objects of interest are cylindrical.

4.1.3. Computation of estimated centers of containers

Based on the points clicked by the operator, the containers' centers in terms of the X and Y coordinates are automatically calculated. It is assumed that containers are either upright (sitting on the ground plane) or sideways, lying on the ground plane. The ground plane is found by finding the plane with the largest support in the point cloud through the plane segmentation methodology included in Open3D (Zhou et al., 2018). For an upright container, the central axis is aligned with the Z axis of the scene; thus, the orientation is unambiguous. Its X and Y center is where the point is clicked, and the Z center is found by adding half the height of the container to the ground plane's Z value. For sideways containers, the X and Y center is the halfway point between the two clicked coordinates; the Z coordinate for the center is calculated by adding the radius of the container to the ground plane's Z value. Once the exact center and orientation of each object is known, a 3D point cloud model of the respective object is placed at that place in the scene. These 3D models consist of a few thousand points that will be projected point by point to camera images.

4.2. Selection of least blurry frames

The images from the CLAMP unit are not collected with a video stabilizer; thus; some of the frames are very blurry. There are two options included in the labeling tool for selecting images to be annotated. One is to select the least blurry frame of every N frame where N is a user-defined integer. The second one is to state the minimum number of training images to produce, and the tool will automatically calculate the value of N that produces the minimum defined number of images after looping through the entire video. The least blurry frame is found by calculating the Laplacian for every N image and selecting the image with the maximum Laplacian value (Sagar, 2020).

4.3. 2D projection of point cloud and occlusion removal

The model containers in point cloud form are projected onto each 2D image in the video given the trajectory of the system (position and orientation as a function of time) and the timestamp of each video frame. The point cloud in the scene's 3D coordinate defined with respect to the global scene. It is first transformed into the camera's coordinates, and then hidden point removal is performed to only keep fractions of the container point clouds that are visible in the camera image (Katz et al., 2007). The surviving points are projected to pixel coordinates using the OpenCV *projectPoints* function (Bradski, 2000) to form a 2D pixel-mask of the containers. The 2D mask consists of pixels in the image that map to the geometric 3D point clouds placed in the scene.

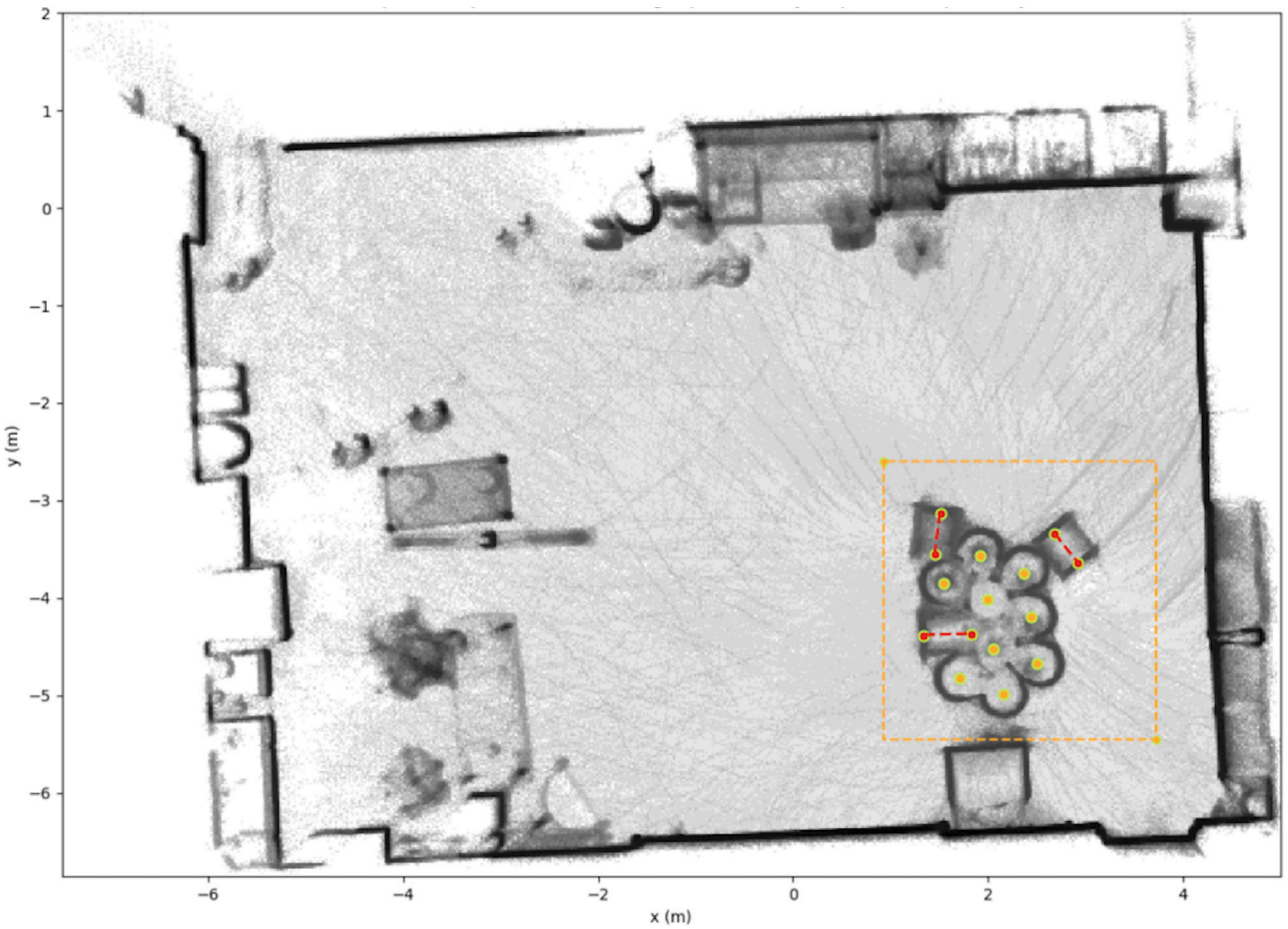


Fig. 8. Top-down view of the point cloud scene: Two clicks define the X and Y boundaries (shown as an orange rectangular box). Containers are labeled by individual clicks (shown in red and yellow) after zooming in on the rectangular box.

4.3.1. Occlusion removal

Hidden point removal only accounts for occlusion between the 3-D model containers and does not account for other objects in the scene; thus, container point clouds hidden behind annotated objects are removed, but points hidden behind unannotated objects such as chairs are not removed by the hidden point removal step. To remove container masks occluded by unannotated objects, the depth images are used to determine the distance of each camera pixel from the camera location. Any pixel with a distance less than the closest edge of the container including a 25 cm margin must belong to other objects occluding the container from the camera's field of view. Hence, those pixels are removed from the 2D masks of the containers. The 25 cm margin was obtained through trial and error. The size of the margin can be easily modified in the configuration file.

4.3.2. Hole filling in point cloud projections

A hull is created with the *convexHull* function in OpenCV based on the pixel mask resulting from the previous step. This hull is filled using the *fillPoly* function (Bradski, 2000). This step is necessary because the pixel masks contain holes or gaps due to sparsity in the 3D point cloud geometric model. The resulting hull fills the entire space that is inside the outer edges of the pixel mask. The filled polygon mask is projected onto the selected training image for visualization and inspection. This process is done for all the NMCs in all images.

4.4. Creation of bounding boxes and mask annotations

Deep learning models for computer vision typically require the training data to contain image annotations defined in a text file format. Currently, two different formats are supported: One format is to train the YOLOv7 object detection model, and the other format is to train the YOLOv7 instance segmentation model (Wang et al., 2023). The YOLOv7 object detection training label format requires the class index, the normalized x and y centers of the bounding box, and the normalized width and height of the bounding box. The rectangular bounding boxes' edges are produced by finding the minimum and maximum x and y coordinate values of each container's 2D mask. The class index of the object is a user-input non-negative integer value. The YOLOv7 instance segmentation training label format requires the class index and the x and y coordinates of a polygon that bounds the 2D mask's outer edge. The x and y coordinates of the 2D mask outline are produced using the *findContour* function in OpenCV (Bradski, 2000). The training images and the annotations need to have the same file names; the tool uses the timestamp of the images as the file names for both the images and annotations.

4.5. Quality check on annotations

At this step, the annotations can be used to directly train a computer vision model. However, it is good practice to check the annotations to make sure the bounding boxes and masks are correct. Some of the depth values in the depth images are erroneous and this can lead to masks occluded by certain objects not being correctly removed (see an example annotated image in Fig. 10). These images are best removed from the final training dataset through manual inspection. A results viewer algorithm is run to play the annotated images as a video where both the masks and bounding boxes are drawn on the images. The operator can pause the video to remove an image, or manually go through the images

Table 1
AT-400 object detection model performance comparison.

Training and validation dataset	Hyper-parameter	Train mAP @0.5IOU	Validate mAP @0.5IOU	Test recall@0.2 confidence threshold	Test precision@0.2 confidence threshold	Test mAP @0.5IOU
First original AWS annotations set	Default	0.849	0.851	0.852	0.934	0.900
Second original AWS annotations set	Default	0.851	0.853	0.866	0.926	0.894
Manual annotations	Default	0.852	0.853	0.865	0.906	0.908
Semi-automatic annotation tool annotations	Default	0.886	0.887	0.870	0.924	0.899
First original AWS annotations set	Optimized	0.851	0.853	0.865	0.928	0.908
Second original AWS annotations set	Optimized	0.852	0.854	0.870	0.917	0.908
Manual annotations	Optimized	0.851	0.854	0.866	0.931	0.916
Semi-automatic annotation tool annotations	Optimized	0.877	0.880	0.887	0.908	0.914

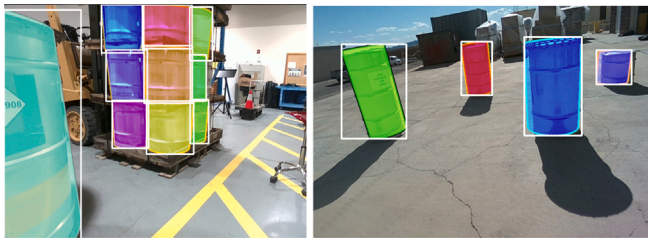


Fig. 9. Examples of semi-automatically annotated images. Left, containers in stacks. Right, four different types of containers.

one by one. The viewer was implemented in Python and allows for images with issues to be removed from the dataset by pressing a hotkey during the review process. This process usually takes a few minutes for each scan of hundreds of images. The remaining annotated images can be used as the training dataset for either training an object detection model or an instance segmentation model. Examples of training images with both bounding boxes and masks produced by the semi-automatic labeling tool are shown in Fig. 9.

4.6. Deep learning computer vision models

To save time training a neural network and reduce the amount of training data necessary to achieve good performance, a technique called transfer learning (Bozinovski, 2020) is leveraged. It requires a neural network that is already trained on a public dataset with most of the layers frozen, and only the last couple of layers are allowed to freely update weights.

4.6.1. AWS rekognition

The first deep learning model considered was AWS "Rekognition" (Amazon Web Services, Inc., 2023a), a customizable and pre-trained computer vision platform. Rekognition offers ease of use and direct linking to other AWS services such as image storage locations and custom labels from automation tasks. The user just needs to point to the locations and specify which sets are for training and testing. Rekognition then employs transfer learning and outputs the newly trained model with performance metrics. Although Rekognition demonstrated good performance, the lack of user control over model hyperparameters and training aspects prohibited improving the model. Furthermore, the Rekognition model is commercial and incurs costs to both train and run on new datasets.

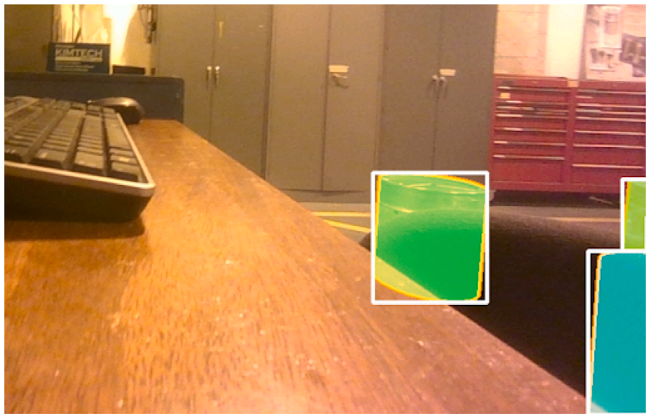


Fig. 10. An example of occlusion detection errors due to depth measurement inaccuracies resulting in three containers masks incorrectly displayed when they should be mostly behind the chair.

4.6.2. YOLOX and YOLOv7

On the other hand, YOLO models are a widely recognized open-source deep learning computer vision model producing state-of-the-art detection accuracies. YOLOX was initially used to benchmark against Rekognition and it was soon replaced with YOLOv7 which offered improved performance and training speed compared to YOLOX (Ge et al., 2021; Wang et al., 2022). YOLOv7 is a state-of-the-art computer vision model that performs object detection tasks with the high accuracy and short prediction time (viso.ai, 2023). YOLOv7's model architecture has three main components, a backbone, a neck, and a head. The backbone is used to extract features of an image, the neck combines these features and analyzes them, the head predicts the locations and classes of target objects, and outputs bounding boxes encompassing the objects (Wang et al., 2022; Roboflow, Inc., 2023). The instance segmentation model released by YOLOv7 has BlendMask integrated into YOLOv7 (Chen et al., 2020), and outputs a more precise location of objects by producing a mask on the pixels that belong to the target object.

5. Results

5.1. Evaluation of YOLOv7 object detection models trained with various annotation methods

To compare different annotation methodologies' impact on object detection model training, the YOLOv7 object detection model is trained using the two AWS annotated datasets, the manually annotated dataset and the semi-automatically annotated dataset. The ground truth used

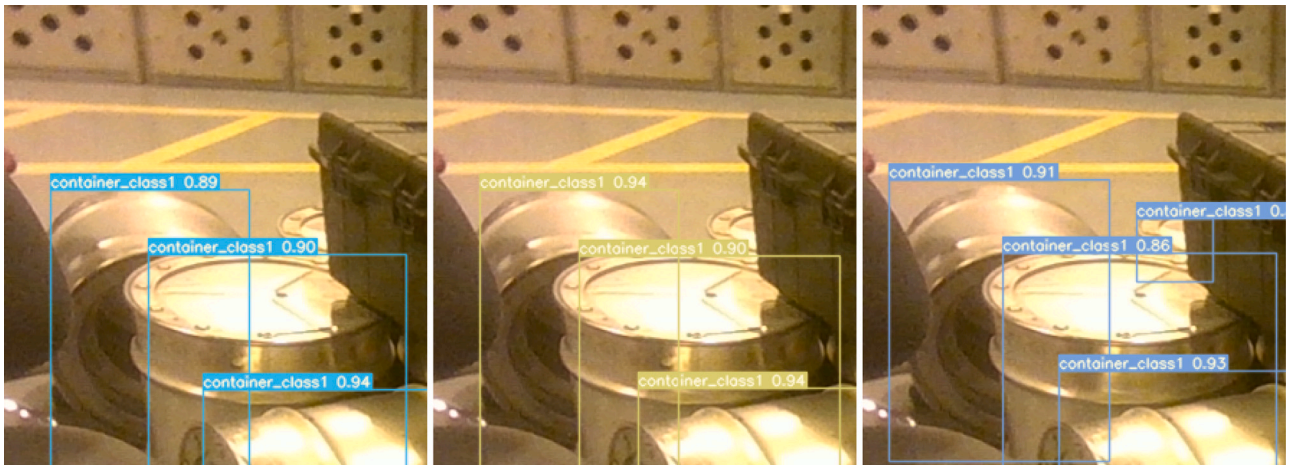


Fig. 11. Left, image inference from the best AWS annotations trained model with a false negative. Middle, image inference from manual annotations trained model with a false negative. Right, image inference from semi-automatic annotations trained model.

for testing are the 261 manually annotated testing images. For consistency, all testing mAP report performance of the trained models on the same 261 manually annotated images.

5.1.1. YOLOv7 object detection model hyperparameters and runtime

For simplicity, all models are trained with the default image size of 640 × 640 pixels, for 250 epochs, with a batch size of 12. The models were initially trained using the default YOLOv7 hyperparameters. This assures consistent outcomes when comparing model performance for different annotation methods. To further improve the models' detection performances, these models were also trained with a set of optimized data-augmentation hyperparameters tuning the model to use more data augmentation than with the default hyperparameters. The optimized hyperparameters were found by systematic searches and successively adjusting data-augmentation hyperparameters that improved the yolov7 object detection models' performances. The tuning of the hyperparameters was done with training images on AT-400 containers that were not part of the benchmark dataset, but it improved performance for all the models trained in this study as well and was added for completeness. With more data augmentation, containers are pasted and rotated more times in the training images than models with default hyperparameters; thus, the quality of the annotations will have a greater impact on the model's performance than models trained with less data augmentation. Both the default and the optimized data-augmentation hyperparameters are shown in Appendix B.

The training converges for all eight models, the epoch vs. loss curve can be found in Appendix C. Each epoch took about 8 s to train, thus the total time required to train a model was around 40 min per model. The inference time is 5.6 ms per image, which allows for real-time inferencing. All timing data were measured on a single Nvidia GeForce RTX 4070 Ti GPU, and will be different on an edge device deployable in the field.

5.1.2. Metrics

The training, validation, and testing results for the models trained with both the default and the optimized hyperparameters are shown in Table 1. The precision is the ratio of the total number of true positive detections over the total number of detections. The recall is the ratio of the number of true positive detections versus the number of ground truth objects. Both metrics depend on the chosen confidence threshold. By varying the threshold value, different tradeoffs between precision and recall can be achieved. A threshold-independent metric is the mAP@0.5IOU, i.e. the mean average precision (mAP) of the detections with at least a 50 percent intersection over union (IOU) between the predicted bounding box and the ground truth bounding box averaged over all classes. The mAP is the metric that generally is optimized during network training and used for leaderboard scoring in computer vision challenges.

5.1.3. Performances comparison of various annotation methodologies

The mAPs shown in Table 1 are the average values from running the model 10 times with 10 different random seeds; with the difference between the 10 runs being around 0.01. Thus, comparing model performances using these results is very consistent and reliable. Using default hyperparameters, the manually annotated dataset performs the best as expected, and the semi-automatically annotated dataset performs similarly to the first AWS annotation set, with the second AWS annotation set performing the worst. However, when comparing the four models trained using the optimized hyperparameters, the semi-automatic annotated dataset model performed in second place and very close to the manually labeled model, with an average test mAP being only 0.2 percent lower; these two models clearly outperform the models trained with AWS-labeled datasets. Overall, the mAP@0.5IOU of all models are relatively similar and are all above 90 percent, which is a very good score for object detection tasks.

Fig. 11 shows example images, where the containers are missed by the best model trained using the original AWS annotations, but is successfully predicted by the model trained using manual annotations and semi-automatic annotations. Fig. 12 shows examples where a false positive prediction was made by models trained using the AWS annotations and the manual annotations, but not the semi-automatic annotations.

5.2. Evaluation of YOLOv7 instance segmentation models trained with various datasets

YOLOv7 instance segmentation models have also been trained with transfer learning using semi-automatic annotated images. While AWS is capable of annotating objects in this way, this was not done here due to the many mistakes found in their object detection bounding box annotations. Manually labeling images with instance segmentation annotation as polygon masks would be time intensive and was not done either. Thus, no AWS or manually annotated dataset is available to compare instance segmentation performance with. The evaluation mainly focuses on exploring the difference in accuracy and run time of the object detection model versus the instance segmentation model, as well as the impact of the complexity of NMC formations in the training data on the trained models' performances. All training, validation, and testing image annotations resulting from the semi-automatic annotation tool were manually reviewed for correctness.

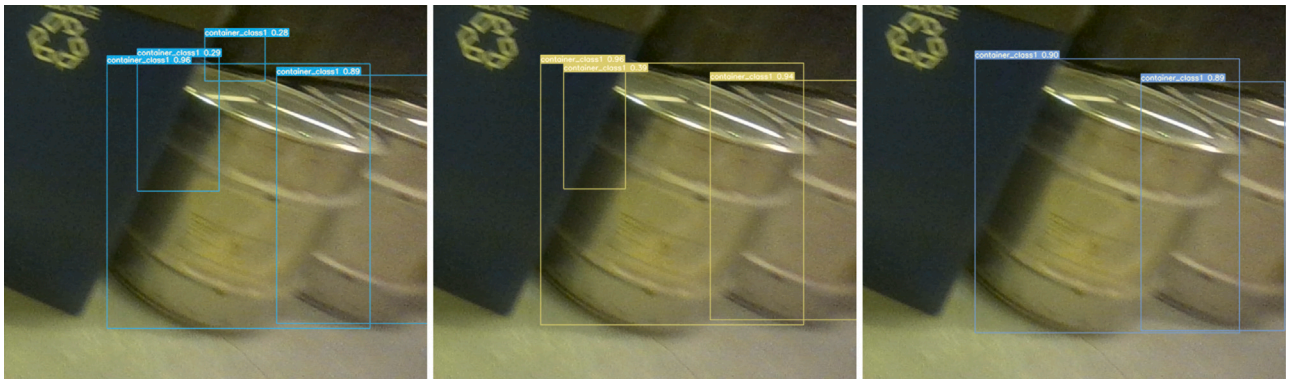


Fig. 12. Left, image inference from the best AWS annotations trained model with two false positives. Middle, image inference from manual annotations trained model with a false positive. Right, Image Inference from Manual Semi-automatic Annotations Trained Model.

5.2.1. YOLOv7 instance segmentation model hyperparameters and runtime

All Models are trained with images of size 640 × 640 pixels, for 350 epochs, with a batch size of 12. The model with the benchmark dataset was initially trained with default data-augmentation hyperparameters, then these hyperparameters were tuned to optimize the validation results. Both the results with default hyperparameters and with tuned hyperparameters are reported in Table 2. Since the optimized hyperparameters performed better, all models trained using other datasets were trained directly on the optimized hyperparameters. Both sets of hyperparameters are shown in Appendix B.

The instance segmentation models were all trained and inferenced using a single Nvidia GeForce RTX 4070 Ti GPU, and took about 13 s per epoch to train; thus, the total time required to train a model with transfer learning was close to 80 min. All models converged during training, the epoch vs. loss plots can be found in Appendix C. The inference time is about 8.7 ms per image, which allows for real-time inferencing. The durations required for both training and inferencing are slightly longer than the durations for object detection models. The instance segmentation model automatically produces bounding box annotation, derived from the masks predicted by the model. The mAPs of these bounding boxes are slightly lower for the testing images than what was observed from the object detection models. Thus, if it is not necessary to obtain an exact mask on the target objects' locations, the object detection model with bounding box outputs is more efficient and accurate.

5.2.2. Performance comparison of various training datasets

Further explorations were made with the instance segmentation model to compare the performance of models trained using different complexities of container formations in training images. The instance segmentation models trained and validated on the benchmark dataset comprised 11 scans of various container formations. Another two models were only trained on images of unobstructed containers not placed in large piles. This comparison aims to find out how much the model performance is impacted if the training images contain the same objects but only in relatively simple formations compared to covering all the formations present in the testing images. The number of training and validation images remained at 262 each for the two new training datasets, but they were all selected from a single scan instead of 11 different scans. In this case, the least blurry image filter was used to select the least blurry image out of every 10 frames. The training images and validation images were randomly selected by splitting the full dataset. One dataset had 12 containers, all individually scattered on the floor. The other dataset had 12 unobstructed containers placed in rows, so it is closer to a pile formation. The results for these two datasets are shown in the last two rows of Table 2. Some example model predictions for models trained with different datasets and hyperparamaters are also shown in Figs. 13 and 14. The two models perform significantly worse than the model trained with the benchmarking dataset. While their testing precisions at a confidence threshold of 0.2 were only slightly reduced, the recall values were almost a factor of two smaller. This shows that the neural network can correctly place masks on the images but is much more likely to miss or misidentify containers in arrangements that are not part of the training dataset. It can be concluded that collecting data of various formations is necessary to achieve good model performance even with lots of data augmentation included in the training.

5.3. Evaluation of annotation methodology on various container types

A feasibility study of the semi-automatic annotation methodology on various container types was done after the 2024 campaign. The annotated, combined dataset was trained with both the YOLOv7 object detection model and the YOLOv7 instance segmentation model. Since it has been shown that the labeling tool is able to produce annotation quality that is on par with manual annotation, ground truth was labeled first with the semi-automatic annotation tool and was inspected manually for correctness. All modelLos are trained with the default image size, default hyperparameters, and with a batch size of 12 on a single Nvidia GeForce RTX 4070Ti GPU. The object detection model was trained for 500 epochs or until convergence with about 15 s per epoch (full training took about 2.2 h in total). The model required more epochs to train than training with the benchmark dataset with only AT-400 containers due to the increased number of container classes. The instance segmentation model was trained for 350 epochs to reach convergence with about 26 s per epoch (the full training took about 2.6 h in total). The loss vs. epochs plots for these models can be found in Appendix C. The inference times of the trained models are similar to the previously trained object detection and instance segmentation models; thus, still feasible for real-time detection.

5.3.1. Model performance

The object detection model's results are shown in Table 3 and the instance segmentation model's results are shown in Table 4. It is evident that both the object detection model and the instance segmentation model have learned to recognize the different classes of containers. The object detection model still outperforms the instance segmentation model. The testing mAPs of the models are all above 0.9 even though the test images contain complex scenarios such as containers stacked together, and outdoor and indoor scenes that have not been part of the training data. Some examples of mask predictions on test images are shown in Fig. 15. The bottom image shows inference on drone footage taken outdoors, with the container placed further away than typically the case during training. Nevertheless, the network was able to correctly identify the container with relatively high confidence.

Table 2
AT-400 instance segmentation model performance comparison.

Training dataset	Hyper-parameter	Train mAP @0.5IOU	Validate mAP @0.5IOU	Test Recall @ 0.2 confidence threshold	Test precision @ 0.2 confidence threshold	Test mAP @ 0.5IOU	Test mAP (Bounding Boxes) @0.5IOU
Original benchmark dataset	Default hyperparameter	0.875	0.839	0.762	0.903	0.822	0.870
Original benchmark dataset	Optimized hyperparameter	0.867	0.876	0.756	0.918	0.859	0.883
Unobstructed individually positioned containers	Optimized hyperparameter	0.831	0.788	0.406	0.868	0.477	0.561
Unobstructed containers positioned in rows	Optimized hyperparameter	0.980	0.951	0.482	0.849	0.564	0.616

Table 3
Performance of object detection model (default hyperparameter) trained on multiple container classes.

Object class	Train mAP@ 0.5IOU	Validate mAP@ 0.5IOU	Test recall@ 0.2 confidence	Test precision@ 0.2 confidence	Test mAP@ 0.5IOU
AT-400	0.882	0.894	0.843	0.928	0.894
ES-3100	0.952	0.97	0.949	0.946	0.97
9978	0.923	0.953	0.935	0.965	0.953
9975	0.872	0.91	0.836	0.96	0.91
DPP1	0.996	0.978	1	0.847	0.978
DT23	0.946	0.915	0.718	1	0.915
All	**0.928**	**0.937**	**0.88**	**0.941**	**0.937**

For both models, the classification accuracy on AT-400 containers stayed on par with models that were trained with only AT-400 containers. This demonstrates that the model successfully learned newly added classes that look similar to AT-400 containers without losing accuracy in detecting the original AT-400 container classes.

6. Discussions

6.1. Difference between various annotation techniques

Overall, the models trained with all annotation methods performed very similar with only small differences. The semi-automatic annotations outperformed the AWS annotations, a standard commercial annotation service, and it performed only marginally worse than annotations that are entirely done manually. Thus, it is a viable technique to replace the traditional annotation methodology for training computer vision models on objects with shapes that can be easily replicated in point cloud forms. The precision of models trained with the semi-automatic labels is generally worse than models trained with the other annotation methods. This is expected as the occlusion removal of the tool is not perfect, and the model trained with the tool could potentially produce false positive labels. However, models trained with semi-automatic labels consistently had higher recalls. This could be because the tool uses a more consistent method of annotating objects, so the model can learn the pattern of the containers better and misses less of them. The neural network, trained on semi-automatically labeled data, was also able to distinguish with high fidelity between different types of containers. This is crucial for applications in safeguards. The new annotation methodology allows hundreds to a few thousands of images to be annotated in 20 min on a typical laptop. An object detection model can be trained in less than 2 h with these annotations using a single Nvidia GeForce RTX 4070 Ti GPU. The entire process from image annotation to trained object-detecting neural network is reduced from weeks to a few hours. Furthermore, as has been shown in the previous section, models trained on semi-automatically labeled images performs better than models trained on commercial annotation service and is on par with manual annotations. The semi-automatic approach also has the benefit of producing both bounding boxes and polygon masks at once; whereas, it would take even more effort and cost to create instance segmentation labels with traditional labeling methods. The semi-automatically created annotations can be used in most situations without an intensive manual review, making it an even simpler procedure and suitable for use at a nuclear facility to label new container types.

6.2. Other methods explored - Fully automatic annotation

Another method explored to locate the NMCs was using convolution to automatically locate objects of cylindrical shapes in situations where they are lying on the ground and with the base of the cylinder aligned with the ground. In this particular scenario, the centers can be found by first aggregating the point cloud into an XY histogram. Then 2D convolution between the histogram and a circular kernel was performed to find cylindrical clusters of points. The center of mass of each cluster is the estimated center of the cylinders. Although this methodology was able to locate the majority of the upright containers in each dataset, one or two containers were sometimes missed. It is also not able to capture containers that are lying on its side. Therefore, the semi-automatic annotation process is more accurate and robust.

6.3. Future improvement and limitations

6.3.1. Explore methods to refine container masks

To refine the masks labeled on target objects, Segment Anything could be added as a last step to automatically carve out container edges from bounding boxes or masks to produce tightly bounded masks on target objects, instead of solely relying on the 3D point cloud models for mask projection (Kirillov et al., 2023).

Optionally, it is also possible to further refine the object's pose by performing an iterative closest point (ICP) search (Zhang et al., 2021). ICP calculates a translation and rotation of a source point set/clouds to minimizes the difference between the source and target point sets/clouds. To implement the ICP algorithm, the 3D geometric models of hollow containers are placed at the positions marked by operators, and are used as sources to match the target containers in the actual LiDAR point cloud. It aims to find the pose of a real-scene container by rotating and translating a model container. This would further reduce the burden on operators as it can find the exact poses of matching cylindrical shapes in the point cloud.

The ICP feature has already been implemented in the script but was not used for this study as it needs further improvements. Currently,

Table 4
Performance of instance segmentation model (default hyperparameter) trained on multiple container classes.

Object class	Train mAP@ 0.5IOU	Validate mAP@ 0.5IOU	Test recall@ 0.2 confidence	Test precision@ 0.2 confidence	Test mAP@ 0.5IOU	Test mAP (Bounding boxes)@ 0.5IOU
AT-400	0.839	0.865	0.76	0.894	0.814	0.86
ES-3100	0.975	0.971	0.94	0.928	0.954	0.955
9978	0.928	0.933	0.925	0.973	0.952	0.952
9975	0.846	0.878	0.849	0.958	0.88	0.912
DPP1	0.995	0.995	1	0.765	0.964	0.907
DT23	0.906	0.925	0.661	0.799	0.735	0.735
All	**0.915**	**0.928**	**0.856**	**0.886**	**0.883**	**0.887**

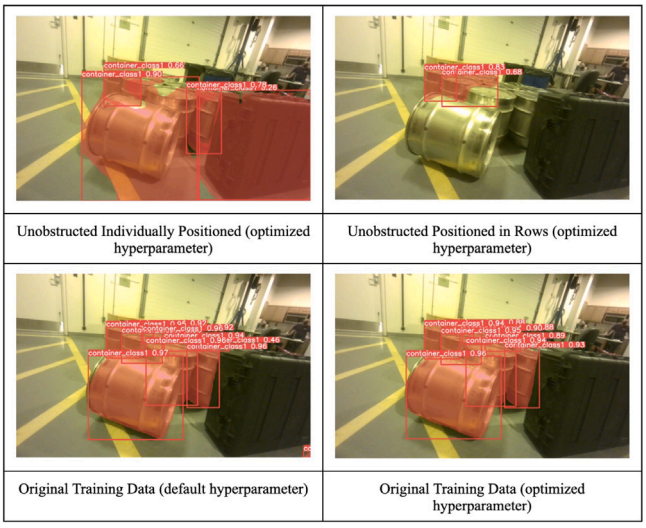


Fig. 13. Example of testing inference result on models trained with 4 different training data, where the model trained with simple formations dataset detected pelican case as a container.

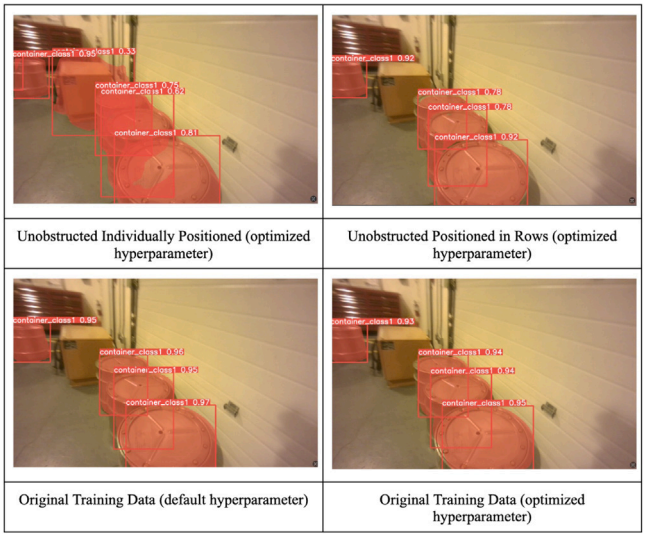


Fig. 14. Example of testing inference result on models trained with 4 different training data, where the model trained with individually placed containers performed significantly worse.

both global and local ICP registration are implemented in the current pipeline respectively. The global ICP registration method chosen is a random sample consensus (RANSAC) based feature matching method implemented in Open 3D (Steinbrucker et al., 2011; Rusu et al., 2009; Choi et al., 2015). The local ICP registration method chosen is the

Fig. 15. Predictions on test images of multiple classes (top) and on unseen outdoor scene (bottom).

point-to-plane ICP algorithm, also implemented in Open 3D (Chen and Medioni, 1992; Besl and McKay, 1992; Park et al., 2017). Fig. 16 shows the container location before and after the ICP step. In this particular case, where the operator did not define the initial position of containers well, ICP helped to find a more accurate position for the container.

However, it has been observed that the current ICP methodology is prone to errors. Point cloud registration via ICP is error-prone in orienting objects when the object's point cloud is noisy or close to other objects. Fig. 17 illustrates some objects (blue, purple, red) being oriented incorrectly as a result of using the described ICP algorithm. Fine-tuning the hyperparameters of the ICP algorithm did not show much improvement, nor did running local ICP only, or Rigid Coherent Point Drift (Myronenko and Song, 2010), another powerful registration approach.

It should be also noted that registration techniques in general do not seem to perform well on incomplete point clouds or shapes lacking features. If there are many objects near a fraction of the scene may never be visible from the LiDARs perspective, leading to challenging situations for registration approaches. As such it is not clear if cylindrical shapes, although often clearly visible to a human, can be reliably identified and oriented with current computer vision techniques.

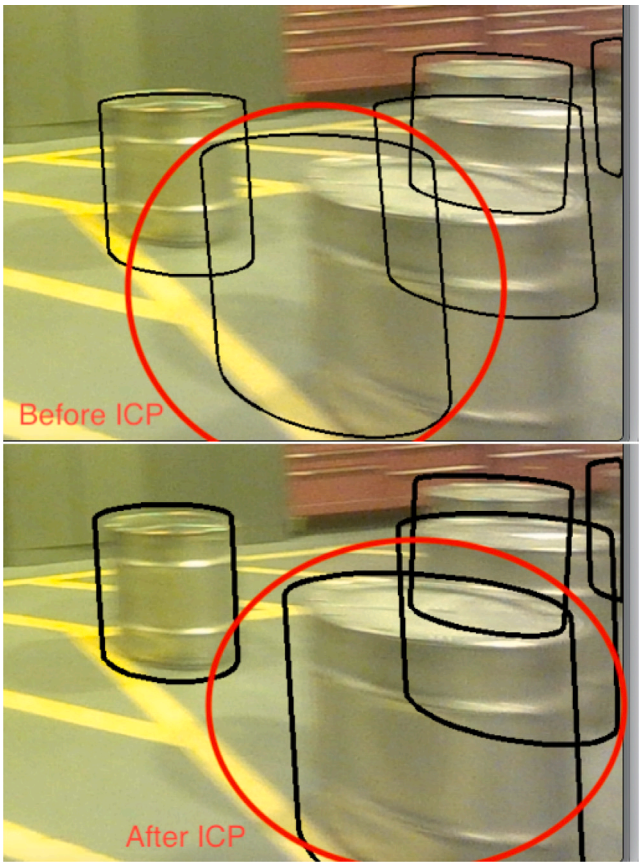


Fig. 16. Top, manually estimated NMC locations before ICP. Bottom, more accurate NMC locations after ICP.

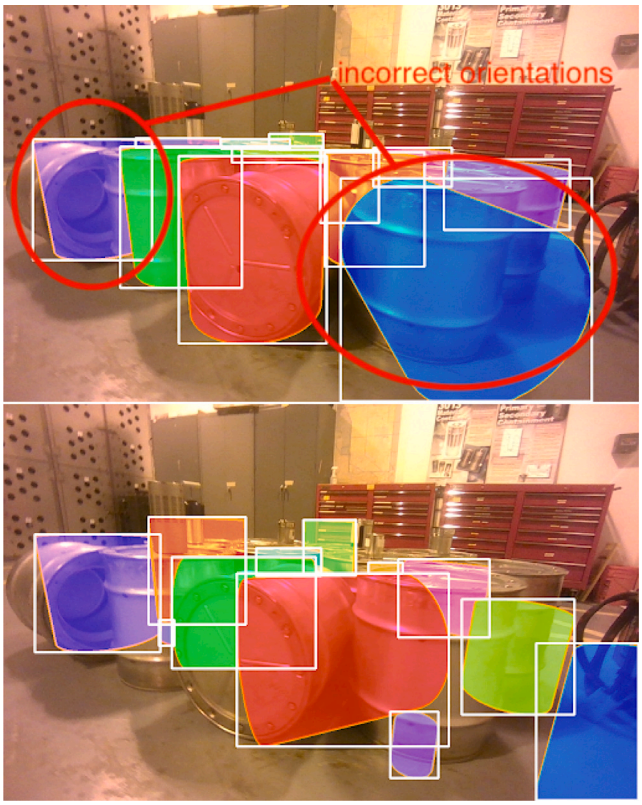


Fig. 17. Top, mis-orientation of objects by current ICP algorithm. Bottom, registration results from rigid CPD with noticeably worse accuracy than current ICP algorithm.

6.3.2. Improvement in depth measurement accuracy

Another limitation of the presented work is that the depth camera's measurements are not very accurate. It is possible to obtain more accurate depth information from a LiDAR point cloud. This could potentially improve the tool's ability to perform more accurate occlusion removal and annotations like the ones shown in Fig. 10 will be corrected.

6.3.3. Produce pose-estimation model training data

A feature that can be added in the future is to produce keypoint annotations of target objects along with the existing bounding boxes and masks. This would allow pose-estimating neural network to also be trained from the semi-automatic annotation pipeline's output without having to creating pose annotation manually. A pose-estimation model can define the positions and orientations of target objects, which can be helpful to more robustly localize an object in 3D. Producing the poses along with the masks and bounding boxes would not add much additional runtime. A pose-estimation model can also be applied to reduce any potential misplaced masks due to clicking errors of the operators and further refine the objects' orientations as discussed in Section 6.3.1.

7. Conclusion

A highly automated method of LiDAR-assisted novel object annotation was presented that provides high-throughput generation of training images for 2D object detection models. One key application for this tool is the inventorying of nuclear material containers (NMCs), where training data for new object types can be created much faster than the standard traditional method of manual annotation or waiting for turnaround of commercial annotation services. Aside from labeling cylindrical containers, this method can be used to annotate any shape that can be described by a 3D geometric point cloud representation. This annotation method requires input on approximate container locations in a 3D map, which is provided by an operator, as such it is a semi-automatic process instead of fully automated. Based on user input of each container's poses, and the scanning system's pose derived from LiDAR mapping, container 3D models are projected onto individual images to create 2-D masks and bounding boxes. The containers' poses can also be added into the output in future work. It takes about 20 min to annotate a scanned dataset of around 800 image frames running on a standard laptop's CPU. The performance of the annotation tool was demonstrated to be on par in quality with manual annotation and surpass the quality of Amazon Web Services Machine Learning-assisted annotations. This evaluation has also demonstrated that the semi-automatic annotation methodology can annotate various types and sizes of nuclear material containers with high quality. It was also found that the quality and variation of the training dataset collected significantly impacted the results of the inferences by the trained deep learning model. Using an over-simplistic target object setup when collecting training images does not yield a robust object recognition model even with high data-augmentation.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was performed under the auspices of the US Department of Energy by Lawrence Berkeley National Laboratory under Contract

DE-AC02-05CH11231. The project was funded by the US Department of Energy, National Nuclear Security Administration, Office of Defense Nuclear Nonproliferation Research and Development (DNN R&D).

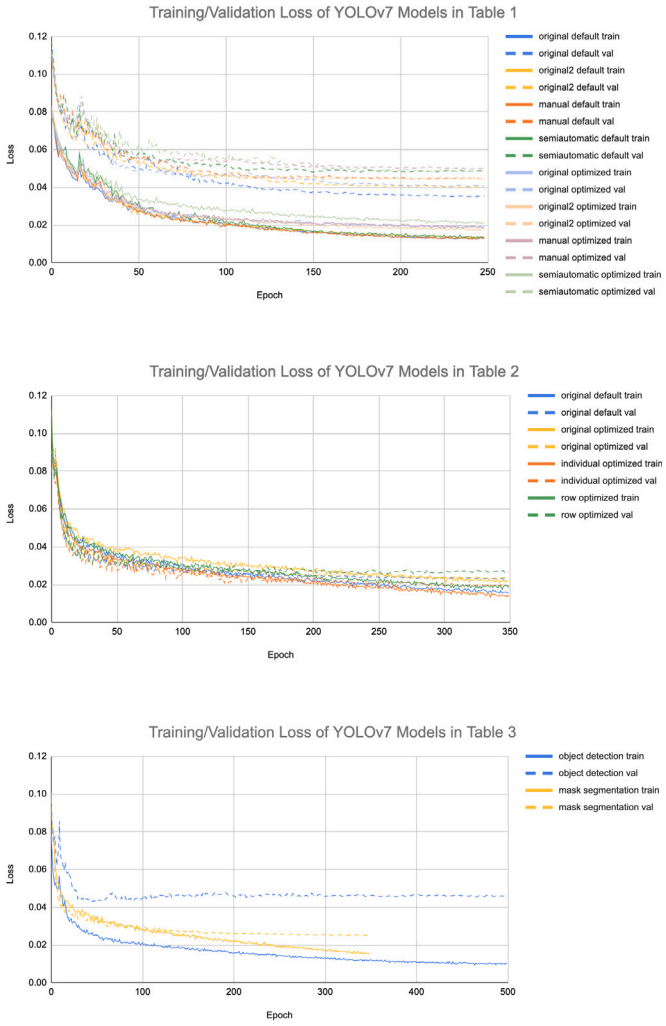
Appendix A. Cylindrical container dimensions

Container name	Approximate height (cm)	Approximate diameter (cm)
AT-400	50	46
ES-3100	50	110
9978	48	91
9975	53	91
DPP1	76	110
DT23	85	93

Appendix B. Model hyperparameters

Hyperparameter list	Default - Object detection	Optimized - Object detection	Default - Instance segmentation	Optimized - Instance segmentation
lr0	0.01	0.01	0.01	0.01
lrf	0.1	0.1	0.1	0.1
momentum	0.937	0.937	0.937	0.937
weight-decay	0.0005	0.0005	0.0005	0.0005
warmup-epochs	3.0	3.0	3.0	3.0
warmup-momentum	0.8	0.8	0.8	0.8
warmup-bias-lr	0.1	0.1	0.1	0.1
box	0.05	0.05	0.05	0.5
cls	0.3	0.3	0.3	0.3
cls-pw	1.0	1.0	1.0	1.0
obj	0.7	0.7	0.7	0.7
obj-pw	1.0	1.0	1.0	1.0
iou-t	0.20	0.2	0.2	0.2
anchor-t	4.0	4.0	4.0	4.0
fl-gamma	0.0	0.0	0.0	0.0
hsv-h	0.015	0.015	0.015	0.015
hsv-s	0.7	0.7	0.7	0.7
hsv-v	0.4	0.4	0.4	0.4
degrees	0.0	45.0	0.0	45.0
translate	0.2	0.1	0.1	0.1
scale	0.9	0.9	0.9	0.9
shear	0.0	0.0	0.0	0.0
perspective	0.0	0.0001	0.0	0.0001
flipud	0.0	0.0	0.0	0.0
fliplr	0.5	0.5	0.5	0.5
mosaic	1.0	1.0	1.0	1.0
mixup	0.15	0.1	0.1	0.1
copy-paste	0.0	0.8	0.1	0.8
paste-in	0.15	0.15		
loss-ota	1	1		

Appendix C. Training and validation loss curves



Data availability

Data will be made available on request.

References

2023. Advanced navigation, spatial. Available: https://www.advancednavigation.com/inertial-navigation-systems/mems-gnss-ins/spatial/.

Amazon Web Services, Inc., 2023a. Amazon rekognition. Available: https://aws.amazon.com/rekognition/. (Accessed 6 December 2023).

Amazon Web Services, Inc., 2023b. Amazon SageMaker. Available: https://aws.amazon.com/pm/sagemaker. (Accessed 6 December 2023).

Bai, C., Xiao, T., Chen, Y., Wang, H., Zhang, F., Gao, X., 2022. Faster-LIO: Lightweight tightly coupled lidar-inertial odometry using parallel sparse incremental voxels. IEEE Robot. Autom. Lett. 7 (2), 4861–4868. http://dx.doi.org/10.1109/LRA.2022.3152830.

Besl, P.J., McKay, N.D., 1992. A method for registration of 3D shapes. PAMI.

Bozinovski, S., 2020. Reminder of the first paper on transfer learning in neural networks, 1976. Informatica (Slovenia) 44, URL https://api.semanticscholar.org/CorpusID:227241910.

Bradski, G., 2000. The OpenCV library. Dr. Dobb's J. Softw. Tools.

Chen, L.-C., Fidler, S., Yuille, A.L., Urtasun, R., 2014. Beat the MTurkers: Automatic image labeling from weak 3D supervision. pp. 3198–3205. http://dx.doi.org/10.1109/CVPR.2014.409.

Chen, Y., Medioni, G.G., 1992. Object modelling by registration of multiple range images. Image Vis. Comput. 10 (3).
Chen, X., Mersch, B., Nunes, L., Marcuzzi, R., Vizzo, I., Behley, J., Stachniss, C., 2022. Automatic labeling to generate training data for online LiDAR-based moving object segmentation. CoRR abs/2201.04501 arXiv:2201.04501 URL https://arxiv.org/abs/2201.04501.
Chen, H., Sun, K., Tian, Z., Shen, C., Huang, Y., Yan, Y., 2020. BlendMask: Top-down meets bottom-up for instance segmentation. arXiv:2001.00309.
Choi, S., Zhou, Q.-Y., Koltun, V., 2015. Robust reconstruction of indoor scenes. In: CVPR.
Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., Fei-Fei, L., 2009. ImageNet: A large-scale hierarchical image database. In: Proceedings of CVPR09.
Docker, Inc., 2023. Docker.
Ge, Z., Liu, S., Wang, F., Li, Z., Sun, J., 2021. YOLOX: Exceeding YOLO series in 2021. arXiv:2107.08430.
Gilbert, A.J., McDonald, B.S., Deinert, M.R., 2016. Advanced algorithms for radiographic material discrimination and inspection system design. Nucl. Instrum. Methods Phys. Res. B 385, 51–58. http://dx.doi.org/10.1016/j.nimb.2016.07.013.
Haddal, R., Hayden, N.K., 2018. Autonomous systems artificial intelligence and safeguards.
Intel, 2020. CVAT: Computer vision annotation tool. https://github.com/openvinotoolkit/cvat.
Intel Corporation, 2023. Intel RealSense depth camera D455.
Katz, Tal, A., Basri, R., 2007. Direct visibility of point sets. SIGGRAPH.
Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S., Berg, A.C., Lo, W.-Y., Dollár, P., Girshick, R., 2023. Segment anything. arXiv:2304.02643.
Lin, T.-Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Zitnick, C.L., Dollár, P., 2014. Microsoft COCO: Common objects in context.
Myronenko, A., Song, X., 2010. Point set registration: Coherent point drift. IEEE Trans. Pattern Anal. Mach. Intell. 32 (12), 2262–2275. http://dx.doi.org/10.1109/TPAMI.2010.46.
Open Source Robotics Foundation, Inc., 2023. Robot operating system (ROS).
OÜ, S., 2023. A complete solution for LiDAR annotation. Available: https://supervisely.com/labeling-toolbox/3d-lidar-sensor-fusion/. (Accessed 6 December 2023).
Park, J., Zhou, Q.-Y., Koltun, V., 2017. Colored point cloud registration revisited. In: ICCV.
Radenović, F., Tolias, G., Chum, O., 2019. Fine-tuning CNN image retrieval with no human annotation. IEEE Trans. Pattern Anal. Mach. Intell. 41 (7), 1655–1668. http://dx.doi.org/10.1109/TPAMI.2018.2846566.
Rizzoli, A., 2021. Bounding box annotation. URL https://www.v7labs.com/blog/bounding-box-annotation.
Roboflow, Inc., 2023. What is YOLOv7? A complete guide. Available: https://blog.roboflow.com/yolov7-breakdown/. (Accessed 6 December 2023).
Rusu, R., Blodow, N., Beetz, M., 2009. Fast point feature histograms (FPFH) for 3D registration. In: ICRA.
Sagar, 2020. Laplacian and its use in blur detection. Available: https://medium.com/@sagardhungel/laplacian-and-its-use-in-blur-detection-fbac689f0f88. (Accessed 6 December 2023).
Salathe, M., Yao Li, P., Parrilla, N.A., Joshi, T.H.Y., Cooper, R.J., Park, K., Sudderth, A.V., 2024. Multi-sensor fusion for nuclear material container counting and inventory. In: INMM 2024. https://resources.inmm.org/annual-meeting-proceedings/multi-sensor-fusion-nuclear-material-container-counting-and-inventory.
Simply NUC, 2023. NUC 11 performance i7.
Steinbrucker, F., Sturm, J., Cremers, D., 2011. Real-time visual odometry from dense RGB-D images. In: ICCV Workshops.
The MathWorks, Inc., 2022. Lidar labeler (r2023b). Available: https://www.mathworks.com. Accessed 6 December 2023.
Tian, Z., Shen, C., Wang, X., Chen, H., 2020. BoxInst: High-performance instance segmentation with box annotations. arXiv:2012.02310.
Tzutalin, 2015. LabelImg. https://github.com/tzutalin/labelImg.
Velodyne Lidar, Inc., 2023. Puck LITE.
viso.ai, 2023. YOLOv7: The most powerful object detection algorithm (2024 guide). Available: https://viso.ai/deep-learning/yolov7-guide/. (Accessed 6 December 2023).
Wang, C.-Y., Bochkovskiy, A., Liao, H.-Y.M., 2022. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. arXiv:2207.02696.
Wang, C.-Y., Bochkovskiy, A., Liao, H.-Y.M., 2023. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. CVPR.
Wei, J., Zhu, Z., Cheng, H., Liu, T., Niu, G., Liu, Y., 2022. Learning with noisy labels revisited: A study using real-world human annotations. arXiv:2110.12088.
Wu, D., Chen, P., Yu, X., Li, G., Han, Z., Jiao, J., 2023. Spatial self-distillation for object detection with inaccurate bounding boxes. arXiv:2307.12101.
You, Y., Lou, Y., Li, C., Cheng, Z., Li, L., Ma, L., Lu, C., Wang, W., 2020. KeypointNet: A large-scale 3D keypoint dataset aggregated from numerous human annotations. CoRR abs/2002.12687 arXiv:2002.12687 URL https://arxiv.org/abs/2002.12687.
Zhang, J., Yao, Y., Deng, B., 2021. Fast and robust iterative closest point. IEEE Trans. Pattern Anal. Mach. Intell. 1. http://dx.doi.org/10.1109/tpami.2021.3054619.
Zhou, Q.-Y., Park, J., Koltun, V., 2018. Open3D: A modern library for 3D data processing. arXiv:1801.09847.