Journal of Nondestructive Evaluation (2022) 41:21
https://doi.org/10.1007/s10921-022-00851-8

Target Detection by Target Simulation in X-ray Testing

Domingo Mery[1] · Alejandro Kaminetzky[1] · Laurence Golborne[1] · Susana Figueroa[1] · Daniel Saavedra[1]

Received: 28 April 2021 / Accepted: 25 January 2022 / Published online: 14 February 2022

Abstract
In X-ray testing, the aim is to inspect those inner parts of an object that cannot be detected by the naked eye. Typical applications are the detection of targets like blow holes in casting inspection, cracks in welding inspection, and prohibited objects in baggage inspection. A straightforward solution today is the use of object detection methods based on deep learning models. Nevertheless, this strategy is not effective when the number of available X-ray images for training is low. Unfortunately, the databases in X-ray testing are rather limited. To overcome this problem, we propose a strategy for deep learning training that is performed with a low number of target-free X-ray images with superimposition of many simulated targets. The simulation is based on the Beer–Lambert law that allows to superimpose different layers. Using this method it is very simple to generate training data. The proposed method was used to train known object detection models (e.g. YOLO, RetinaNet, EfficientDet and SSD) in casting inspection, welding inspection and baggage inspection. The learned models were tested on real X-ray images. In our experiments, we show that the proposed solution is simple (the implementation of the training can be done with a few lines of code using open source libraries), effective (average precision was 0.91, 0.60 and 0.88 for casting, welding and baggage inspection respectively), and fast (training was done in a couple of hours, and testing can be performed in 11ms per image). We believe that this strategy makes a contribution to the implementation of practical solutions to the problem of target detection in X-ray testing.

Keywords X-ray testing · X-ray simulation · Object detection · Casting inspection · Welding inspection · X-ray baggage security

1 Introduction

In X-ray testing, there are certain applications in which we have to identify—non-destructively—a target located inside an object [1] as illustrated in Fig. 1. In general, automated visual inspection problems have been tackled recently using computer vision methods based on deep learning models (see for example the product quality control of pelletization process [2], detection of defects [3–5], and inspection of optical components [6]). Moreover, object detection approaches based on deep learning [7] have established themselves as state-of-art in computer vision applications that require the identification and the location of a target. They have been used in many applications such as remote sensing [8] and gear pitting detection [9] among others. In this paper, we deal with the target detection problem in X-ray testing giving examples on three application fields: detection of discontinuities in casting and welding inspection, and recognition of prohibited items in baggage inspection.

Casting inspection is used in the production of light aluminum alloys (e.g. wheels for the automotive industry), where cracks or voids can be formed inside the workpiece. In the last years, automated methods based on classic computer vision [10,11], multiple views [12], computed tomography [13] and deep learning [14–17] have been proposed.

Welding inspection is used used for detecting those defects in the petroleum, chemical, nuclear, naval, aeronautics and civil construction industries, among others. A mandatory inspection using X-ray testing is required in order to detect defects like porosity, inclusion, lack of fusion, lack of penetration and cracks. Semi-automated methods based on classic computer vision [18,19] and deep learning [20–22] have been reported.

Baggage inspection is used as a security check to protect public entrances (e.g. government buildings, airports, etc.) where all baggage are scanned to detect prohibited items like

✉ Domingo Mery
domingo.mery@uc.cl

1 Department of Computer Science, Pontificia Universidad, Católica de Chile, Santiago, Chile

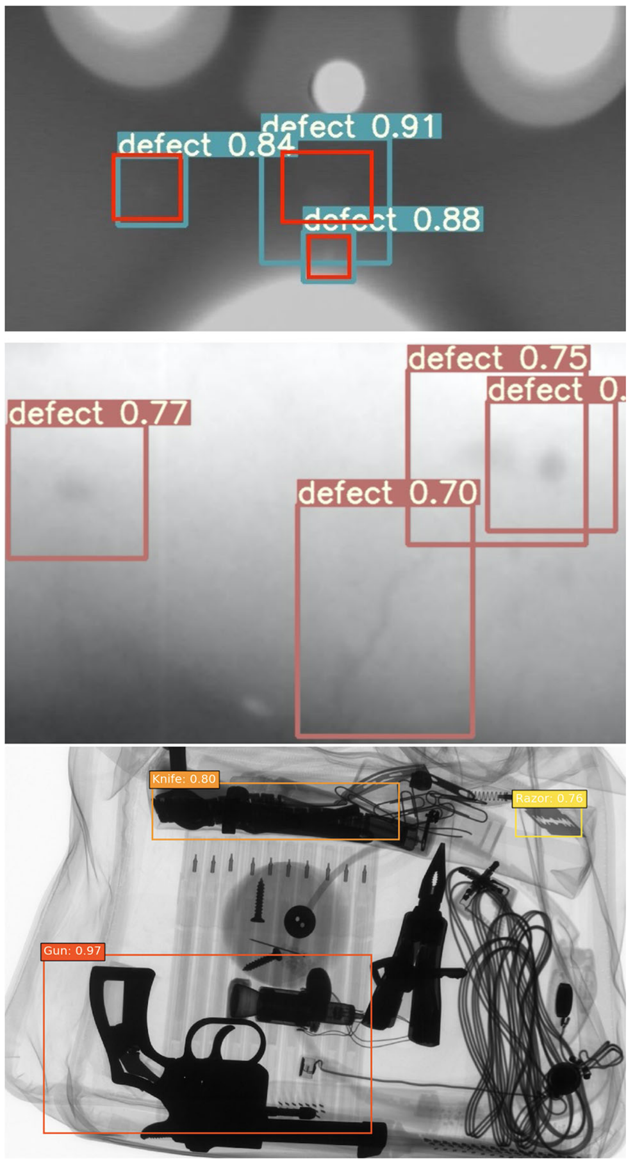


Fig. 1 Examples of real X-ray images used in our experiments. Top: Casting inspection: Detection of defects using YOLOv5s is shown in cyan, and ground truth in red. Middle: Welding inspection: Detection using YOLOv3-Tiny is shown in orange (ground truth is evident). Bottom: Baggage inspection. Detection using SSD (ground truth is evident)

handguns or explosives [23]. Recently, methods based on classic computer vision and deep learning have been proposed. In single views using mono-energy and dual-energy images, we can mention [24–31] respectively. Besides, there are some contributions based on GAN's (Generative Adversarial Networks) [32] to generate synthetic X-ray images that can be used as data augmentation in the training stage [33,34].

From the literature review, we can see that although many computer vision algorithms have been developed in the last decades, the performance of them is far from perfect, and visual inspection is still required in certain cases. In recent years, we have witnessed how methods based on deep learning have significantly boosted the performance in several areas of computer vision such as object detection. Object detection can be used in the mentioned applications of X-ray testing to detect targets (such as defects or prohibited items). In object detection, more than one object can be recognized in an image, and the location of each recognized object is given by a bounding box that encloses the detected object [7]. The most representative methods in object detection are those based on a single convolutional neural network (CNN) [35], i.e. a deep neural network with several layers, that are trained to both location and classification, i.e. prediction of bounding boxes, and estimation of the class probabilities of the detected bounding boxes. This group of approaches corresponds to the state of the art in detection methods, because they are very effective and very fast. They are the best performing as stated in [36], where the most well known are: YOLO [37–41],[1] EfficientDet [42], RetinaNet [43] and SSD [44].

In X-ray testing, however, object detection is only possible when the number of X-ray images is high enough to train a model. Unfortunately, the annotated datasets in X-ray testing are rather limited, and the number of available X-ray images is low. To overcome this problem, we propose a strategy based on simulation of very realistic targets to build the training dataset. Thus, training is performed with a low number of target-free X-ray images with superimposition of simulated targets. Whereas, testing is performed on real X-ray images ensuring an evaluation in real scenarios.

Our contributions to the detection of targets in X-ray testing is fourfold:

(i) We develop a simple, effective, and fast deep learning strategy that can be used in the detection of targets in X-ray testing (see some results in Fig. 1).
(ii) Due to the low number of available real targets, we propose a very useful training/testing strategy, in which only simulated targets are used in training, and only real targets in testing (see Fig. 2). This practice avoids overfitting. The training stage requires a low number of real X-ray images and no manual annotations because a simulation model is used to superimpose targets onto the X-ray images.
(iii) There are some simulation models that have been specifically used to simulated defects in castings [45], in weldings [46] and threat objects in baggage inspection [31,47], however, in our work, we propose a unified algorithm (see Algorithm 1) for simulation that can be used in X-ray testing that superimposes two X-ray images in general. Besides, a new simulation approach is presented

[1] YOLOv5 was released in June 2020. The GitHub repository is available on https://github.com/ultralytics/yolov5.

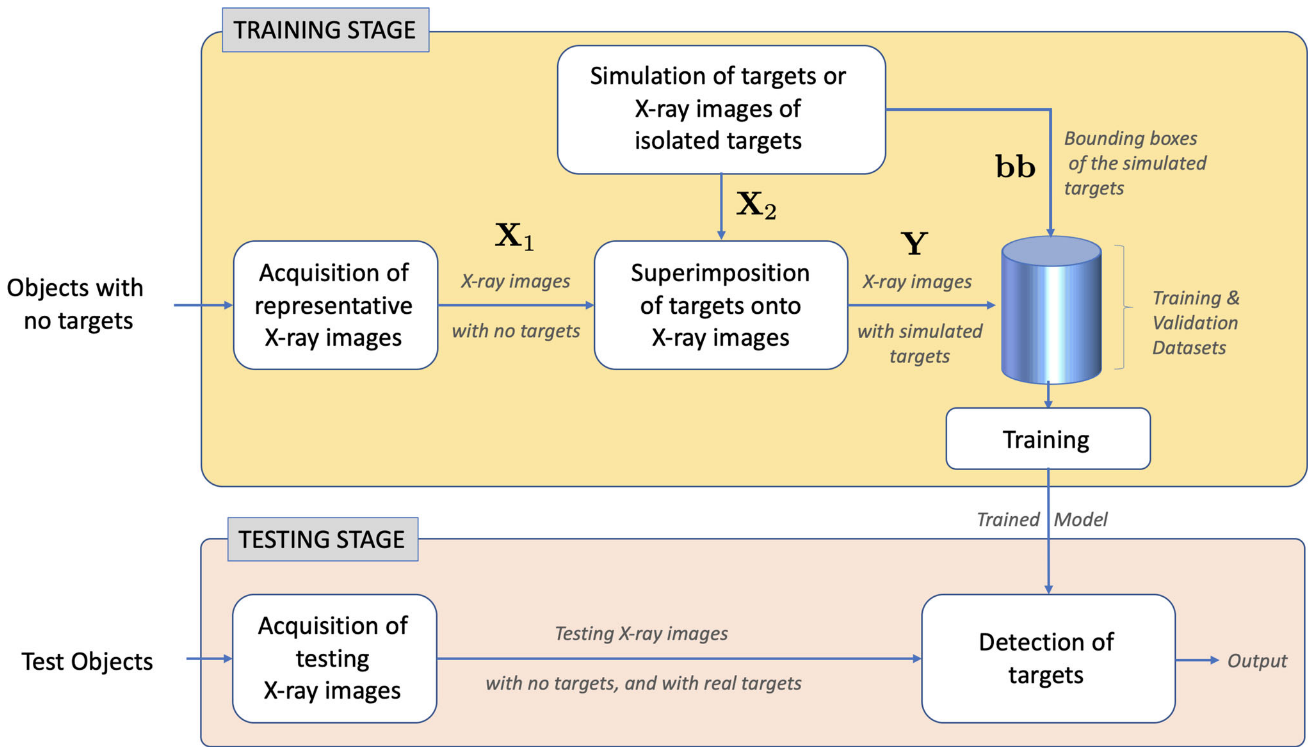

TRAINING STAGE
Simulation of targets or X-ray images of isolated targets
$\mathbf{X}_2$
bb
Bounding boxes of the simulated targets
Objects with no targets
Acquisition of representative X-ray images
$\mathbf{X}_1$
X-ray images with no targets
Superimposition of targets onto X-ray images
$\mathbf{Y}$
X-ray images with simulated targets
Training & Validation Datasets
Training
Trained Model
TESTING STAGE
Test Objects
Acquisition of testing X-ray images
Testing X-ray images with no targets, and with real targets
Detection of targets
Output

Fig. 2 Block diagram of proposed method. See examples of images $\mathbf{X}_1$, $\mathbf{X}_2$, $\mathbf{Y}$, and **bb** in Fig. 3

that generates very realistic defects (see stick example in Fig. 3)

(iv) The implementation of training methods is based on well-established deep object detection libraries avoiding practical settings in training or testing stages. The code and the datasets used in this paper are available on a public repository.[2] We believe that this practice, which is very common in other fields of computer vision, should be more common in X-ray testing. Thus, anyone can reproduce the results reported in this paper, or re-use the code in other inspection tasks. Moreover, training and testing can be executed in Python in any browser with no intricate configuration and free access to GPUs using Google Colab.[3]

2 Proposed Method

The general overview of our proposed method is presented in Fig. 2. As usual, the recognition approach has two stages: training and testing. In our method, training is performed using real X-ray images with simulated targets only (see Fig. 3). Thus, no real image with targets is used for training purposes, because in this kind of problem the amount of real targets is very low. On the other hand, testing is carried out using real X-ray images with real targets. Thus, the reported performance on the testing dataset corresponds to a real scenario. In this section, we explain the simulators we use (Sect. 2.1), and the training and testing processes (Sect. 2.2).

2.1 Simulators

For the simulation of X-ray images, used later to build the training dataset, we used the superimposition idea presented in [45,47], in which a simulated X-ray image is created by combining two X-ray images.[4] In this section, we present a unified approach for the superimposition and a new simulation approach of defects based on a strategy using 'sticks' that are located randomly.

2.1.1 Unified Model

The unified model is a unique simulation approach that can be used to simulate X-ray images (for training purposes). The central idea of the method is to generate a simulated X-ray image ($\mathbf{Y}$) from two X-ray images: an image ($\mathbf{X}_1$) with no target, and an image ($\mathbf{X}_2$) with the target, as illustrated in

[2] URL of the repository will be available after publication.

[3] See http://domingomery.ing.uc.cl/material.

[4] Simulators based on GAN have been proposed (see for example [16]) however, we don't use them in the experiments, because the superimposition of ellipsoidal defects achieved higher performance.

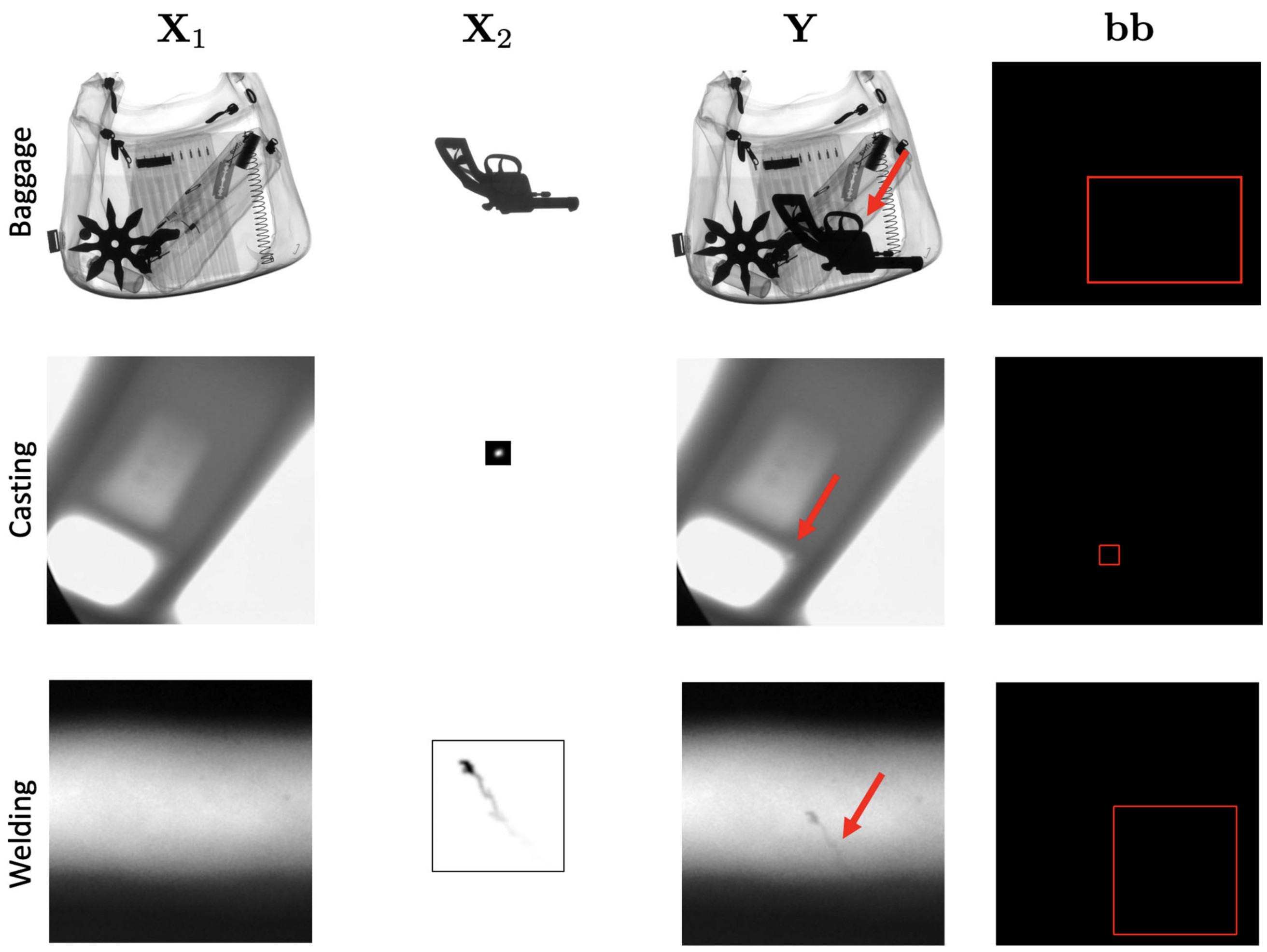


Fig. 3 Simulation of targets for baggage inspection (target = handgun), for casting inspection (target = hole using ellipsoidal model), and for welding inspection (target = crack using stick model): $\mathbf{X}_1$ Original X-ray image with no target. $\mathbf{X}_2$ Simulated targets or X-ray image of isolated target. $\mathbf{Y}$ X-ray image with simulated target (superimposition of target onto the original X-ray image). **bb** Bounding boxes of the simulated targets. Using this method it is very simple to generate training data, i.e. X-ray images with (simulated) targets and location of the targets, where the bounding boxes are obtained with no manual annotation. The red arrows show the simulated targets

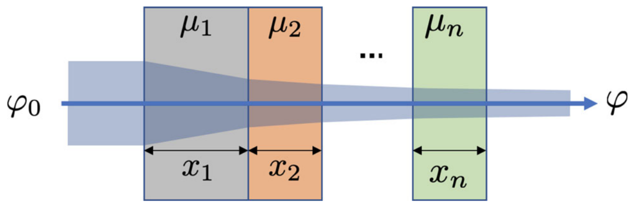


Fig. 4 Beer–Lambert Law in n Materials

Fig. 2. In our approach, $\mathbf{X}_1$ is a real X-ray image, and $\mathbf{X}_2$ can be whether a real X-ray image of the isolated target (with no background), or a simulated X-ray image of the target. Thus, we can locate the target image in any position of the real X-ray image to obtain as many images as we need for training.

The unified model is based on the use of the Beer–Lambert law [48] that characterizes the distribution of X-rays through matter as shown in Fig. 4:

$$\varphi = \varphi_0 \exp\left(-\sum_{i=1}^{n} \mu_i x_i\right), \tag{1}$$

with μ_i the absorption coefficient of matter i with thickness x_i, φ_0 the incident energy flux density, and φ the energy flux density at the output. In general, we know that a grayscale X-ray image Y is followed by the linear model:

$$Y = a \cdot \varphi + b, \tag{2}$$

where a and b are constant parameters of the capture model. With the Beer–Lambert law and the linear scaling, we can

Algorithm 1 – Superimposition of X-ray images

```
Input:
  $\mathbf{X}_1$ : X-ray image of the object
  $\mathbf{X}_2$ : X-ray image of the target
  $(i_1, j_1)$ : location (left corner up) of the target
  $(b, c)$ : linear parameters of the model (see (5))
  ---------------------------------------------
  $\mathbf{Y} \leftarrow \mathbf{X}_1$                                   ▷ initialization of output image
  $(n_2, m_2) \leftarrow$ size of image $\mathbf{X}_2$
  $(i_2, j_2) \leftarrow (i_1 + n_2 - 1, j_1 + m_2 - 1)$
  $\mathbf{X}'_1 \leftarrow \mathbf{X}_1(i1 \cdots i2, j1 \cdots j2)$       ▷ location of the target
  $\bar{\mathbf{X}}'_1 = (\mathbf{X}'_1 - b)/c$                          ▷ normalization
  $\bar{\mathbf{X}}_2 = (\mathbf{X}_2 - b)/c$                            ▷ normalization
  $\mathbf{bb} \leftarrow [j_1\ i_1\ j_2\ i_2]$                          ▷ bounding box $(x_1, y_1, x_2, y_2)$
  $\mathbf{Y}(i1 \cdots i2, j1 \cdots j2) = c\bar{\mathbf{X}}'_1\bar{\mathbf{X}}_2 + b$   ▷ superimposition
  ---------------------------------------------
Output:
  $(\mathbf{Y}, \mathbf{bb})$                                            ▷ output X-ray image and bounding box
```

model the image X_i of matter i only with $\varphi_i = \varphi_0 e^{-\mu_i x_i}$ as:

$$X_i = a \cdot \varphi_i + b = ce^{-\mu_i x_i} + b, \tag{3}$$

where $c = a\varphi_0$. It is clear, from (1), (2), and (3) that output image Y can be rewritten as a contribution of n layers:

$$Y = c\prod_{i=1}^{n} \bar{X}_i + b \quad \text{with} \quad \bar{X}_i = \frac{X_i - b}{c}. \tag{4}$$

Parameters b and c can be estimated as follows: The maximal grayvalue, $g_{\max}$, typically 255, can be achieved when no object is present, i.e. $x = 0$. That means, $g_{\max} = c + b$. On the other hand, the minimal grayvalue, $g_{\min}$, typically 0, can be achieved when the thickness of the object to be irradiated is maximal, *i.e.,* $x = x_{\max}$. That means, $g_{\min} = c\gamma + b$, with $\gamma = e^{-\mu x_{\max}}$ for a known absorption coefficient μ. Using $g_{\min} = 0$ and $g_{\max} = 255$, we obtain:

$$b = \frac{255}{1 - 1/\gamma} \quad \text{and} \quad c = \frac{255}{1 - \gamma}. \tag{5}$$

Now, we can use (4) to compute output image Y from input X-ray images X_1 and X_2, that means for $n = 2$. Algorithm 1 shows the implementation of this approach. The simplicity of the proposed method is evident. In following sections, we will see how to use this method in practice.

It is worthwhile to mention that Bremsstrahlung effect [48] is not considered in the model of equation (1). The reason of this is because in the simulation process—when using the superimposition of the proposed model onto the original image produced with real X-rays, the Bremsstrahlung spectrum is part of the real X-rays with which the original image was produced. Although this is indeed a simplification of the problem, we believe that the results obtained do not significantly affect the appearance of the simulated defects.

2.1.2 Simulation in Baggage Inspection

This strategy was presented in [47] for baggage inspection, where $\mathbf{X}_1$, the X-ray image of a luggage with no threat object,[5] and $\mathbf{X}_2$, the X-ray image of an isolated threat object, are defined. For this end, we use the images of $\mathbb{GDX}$ray dataset [49] that contains X-ray images of isolated threat objects (like handguns, razor blades and knifes) located inside a sphere of expanded polystyrene (EPS) with a very low absorption coefficient. An example for baggage inspection is shown in Fig. 3 (see first row). In this example, $\mathbf{X}_2$ is a real X-ray image of a handgun in an EPS sphere. The superimposition of both images is achieved by Algorithm 1 that uses (4).

2.1.3 Simulation of Ellipsoidal Defects

This simulation model was originally presented in [45] and evaluated in the detection of defects in aluminum castings using object detection methods in [17]. For the sake of completeness, we include this section a summary of this strategy.

The method defines an ellipsoidal cavity in 3D space. Here, we reformulated it using our unified model presented in Sect. 2.1.1 and Fig. 4, where the X-ray imaging process can be modeled using $\mu_1 = \mu$ (the absorption coefficient) and x_1 (the thickness) of the aluminum casting, and for the cavity we can use $\mu_2 = \mu_1$ and $x_2 = -d$ to obtain

$$\varphi = \varphi_0 e^{-\mu(x_1 - d)}. \tag{6}$$

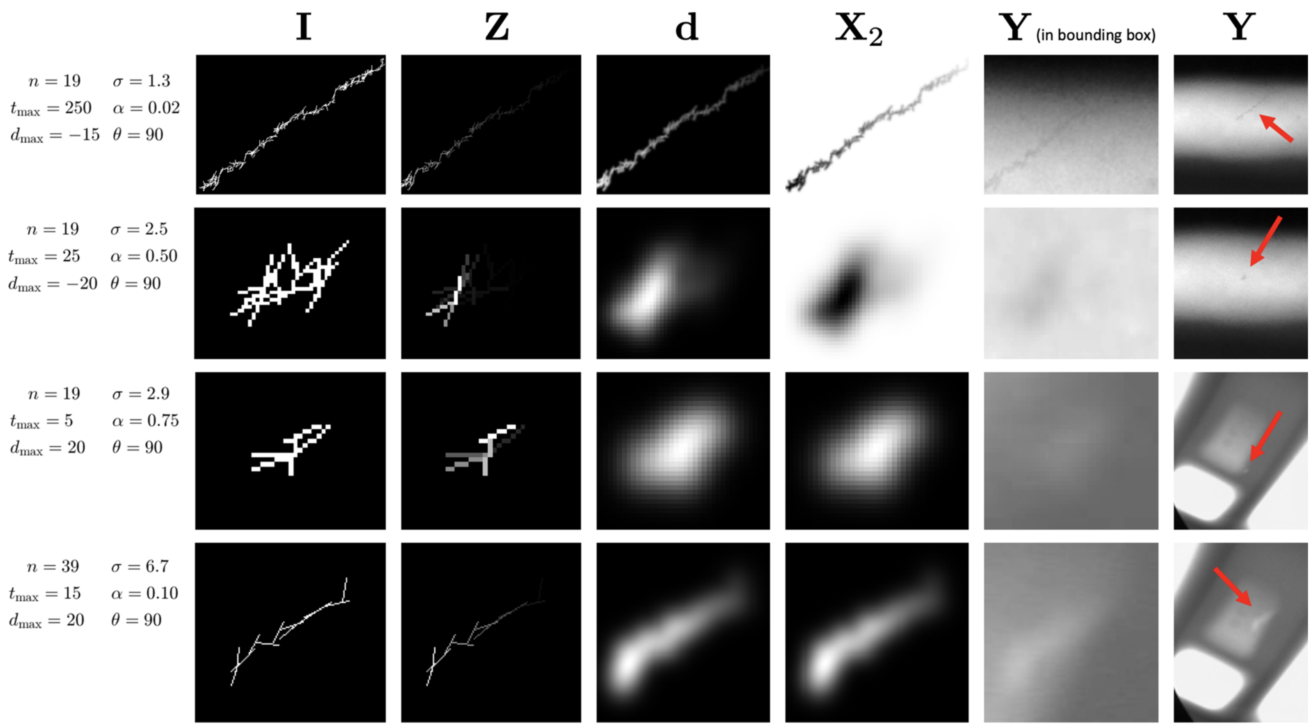
where d is the thickness of the cavity.

The definition of the ellipsoidal cavity includes orientation and size of each axis. The modeled ellipsoid is projected onto a real X-ray image $\mathbf{X}_1$ as follows: (i) for each pixel of the X-ray image we estimate the X-ray beam that goes from the X-ray source to the pixel, and (ii) we compute d, the length intersection of the X-ray beam with the ellipsoid, i.e. distance between the two intersection points of the surface of the ellipsoid. That means, for each pixel (i, j) of image $\mathbf{X}_1$ we have the corresponding thickness of the ellipsoidal cavity $d(i, j)$. Thus, the simulated X-ray image for the project ellipsoidal defect can be computed from (3) by:

$$\mathbf{X}_2 = ce^{\mu\mathbf{d}} + b, \tag{7}$$

where $\mathbf{d}$ is a matrix of the same size of $\mathbf{X}_1$ that contains the values d for each pixel. If $d(i, j) = 0$, that means when there is no intersection, then $X_2(i, j) = c + b$. Therefore, from (4), $\bar{X}_2(i, j) = 1$, and $Y(i, j) = X_1(i, j)$, which means in those pixels that there is no intersection with the ellipsoid,

[5] In this case, $\mu_1 x_1$ represents $\sum_j \mu_j x_j$ including all cluttered objects j that lie on the X-ray beam [47].



I
Z
d
X2
Y (in bounding box)
Y
n = 19 σ = 1.3 tmax = 250 α = 0.02 dmax = −15 θ = 90
n = 19 σ = 2.5 tmax = 25 α = 0.50 dmax = −20 θ = 90
n = 19 σ = 2.9 tmax = 5 α = 0.75 dmax = 20 θ = 90
n = 39 σ = 6.7 tmax = 15 α = 0.10 dmax = 20 θ = 90

Fig. 5 Simulation of defects using the proposed stick approach. For the visualization, the grayvalues of all images are linearly scaled between the lowest value in black and the highest value in white. See Algorithm 2 for an explanation of each variable

the output X-ray image is the original real image $Y = X_1$. In other words, this approach modifies only those pixels in which the ellipsoid is projected.

An example of this method using Algorithm 1 is illustrated in Fig. 3 for a casting. All details of the computation of the projection of the ellipsoid are given in [45].

2.1.4 Simulation of Defects Using ‘Sticks’

In previous section, we simulated a defect in 3D space that is projected onto the X-ray image. For the projection, we compute the thickness d (the length of the intersection of the X-ray beam with the simulated 3D defect) in every pixel of the X-ray image, and the simulated image is obtained using (7). In this section, we present a new method for simulation of realistic defects. Rather than simulating a 3D shape for the defect and computing its projected thickness d onto the X-ray image, we simulate the projected thickness d directly. Thus, we obtain the 2D matrix $\mathbf{d}$ that contains the required values d to simulate the output image $\mathbf{X}_2$ according to (7). The key idea of our approach is to superimpose a set of connected ‘sticks’ (binary images of lines in random orientations) processed by a low-pass filter. An overview of this idea is shown in Fig. 5, where the first column shows the ‘sticks’ and the third column shows the low-pass filtered image that corresponds to thickness matrix $\mathbf{d}$. Here, the method is presented in more detail.

The method consists of an iterative approach. At the beginning, an empty binary image $\mathbf{I}_0$ with a seed pixel is defined. Additionally, an empty image $\mathbf{Z}_0$ of the same size is created. In iteration t, we add a segment ℓ_t of n pixels, that we call a *stick*, in a random direction that touches one of the last pixels of $\mathbf{I}_t$ using a Boolean operation ‘or’ $\oplus$:

$$\mathbf{I}_{t+1} = \mathbf{I}_t \oplus \ell_t. \tag{8}$$

At the same time, matrix $\mathbf{Z}$ is updated as follows:

$$\mathbf{Z}_{t+1} = (1 + \alpha \cdot \text{rand})\mathbf{Z}_t + \mathbf{I}_{t+1}, \tag{9}$$

where α is a latency factor. After $t_{\max}$ iterations, $\mathbf{Z}$ is smoothed using a Gaussian mask,[6] and reoriented to a desired direction Finally, the grayvalues are non-linearly scaled between 0 and $d_{\max}$, where $d_{\max}$ is the maximal allowed thickness of the simulated defect.

[6] Gaussian filtering is used to blur the sticks. This blurriness does not depend on the sharpness of the X-ray image. The Gaussian filtering eliminates the high frequencies of the perfect lines of the sticks (see image $\mathbf{Z}$ in Fig. 5). Thus, the tiny lines are replaced by thicker and blurred regions (see image $\mathbf{d}$ in Fig. 5), whose appearance is similar to the real defects. Parameter σ of the Gaussian mask is set manually according to the thickness of the defect we want to simulate. For instance, for small defects σ can be a low number (e.g. 2.9 pixels as shown in third row of Fig. 5). On the other, for thicker defects, v can be larger (e.g., 6.7 pixels as shown in fourth row of Fig. 5).

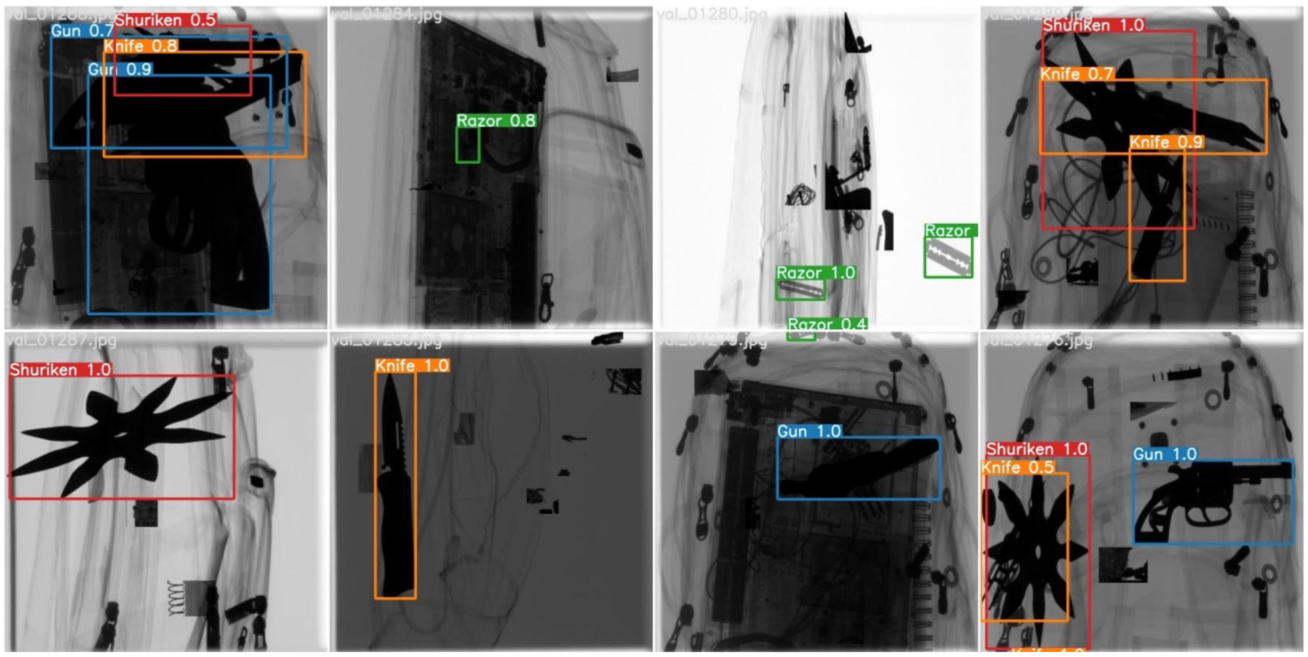


Fig. 6 Some simulated X-ray images used in the training subset of baggage inspection as validation images. The bounding boxes and confidence score were computed by method YOLOv5s

The algorithm of the simulation approach is presented in Algorithm 2. An implementation is given in our webpage[2]. Some examples in castings and welds with these simulated defects are given in Fig. 5. They were generated by Algorithm 1. It is worthwhile to mention that for $d_{\max} > 0$ (or $d_{\max} < 0$) the defects are brighter (or darker) than the background. We can observe that the defects are very realistic, and indeed they are more lifelike than the perfect ellipsoids of previous approach.

This method has been designed to simulate defects such as porosity and cracks mainly. However, other kinds of defects, like lack of fusion or lack of penetration in welds, could be simulated as well using our method, because they show patterns similar to those that can be achieved by superimposing sticks.

2.2 Training and Testing

The detection model is trained using real X-ray images with simulated targets only (see Figs. 2 and 3) as follows:

1. Representative X-ray images of the testing object with no targets are selected.
2. In each representative X-ray, targets are simulated and located in a random place, with a random orientation and sometimes a random size. The idea is to superimpose many targets onto the target-free X-ray images.
3. For each simulated target, a bounding box is defined as a rectangle that encloses the target. Using these three steps, it is very simple to generate training data. Now, we have X-ray images with many (simulated) targets with their locations, where the bounding boxes are obtained with no manual annotation.
4. We split the X-ray images (with simulated targets) in a set for training purposes and a set for validation purposes.
5. The detection model is trained using training and validation sets. Details of numbers of images and simulated targets per image are given in Sect. 3.

The trained model is tested on X-ray images that may or may not contain real targets. In our testing strategy, the idea is to estimate the performance on a real scenario. Thus, in testing dataset no simulated target is used. To build the testing dataset, we need X-ray images of the same test object type with real targets that are manually annotated by human operators. In the testing stage, it is necessary to measure the performance and the computational time as well.

3 Experimental Results

In this Section, we present the achieved results in casting inspection (Sect. 3.1), welding inspection (Sect. 3.2) and baggage inspection (Sect. 2.1.2). In our experiments, we used $\mathbb{GDX}$ray dataset of X-ray images for non-destructive test-

Algorithm 2 – Simulation of stick-defects

```
Input:
$(N, M)$ : maximal size in pixels of the simulated image
$(s_i, s_j)$: seed pixel of the simulation
$n$ : size in pixels of the segment
$\theta$ : maximal angle of the segment
$\phi$ : final orientation of the simulated structure
$\sigma$ : size in pixels of the Gaussian mask
$\alpha$ : latency factor
$\mu$ : absorption coefficient
$t_{\max}$ : maximal number of iterations
$d_{\max}$ : maximal size in mm of the defect
$(b, c)$ : linear parameters of the model (see (5))

$\mathbf{I} \leftarrow$ zeros$(N, M)$                          ▷ binary initialization
$\mathbf{I}(s_i, s_j) \leftarrow 1$                              ▷ seed
$\mathbf{Z} \leftarrow$ zeros$(N, M)$                          ▷ double initialization
for $t \leftarrow 1, t_{\max}$ do                               ▷ iterations
    $\theta_t \leftarrow$ random$(0, \theta)$                  ▷ angle of the segment
    $\mathbf{T} \leftarrow$ zeros$(n, n)$ + line(center,border-pixel at $\theta_t$)
    $(i, j) \leftarrow$ pixels where $\mathbf{I} = 1$
    $(i', j') \leftarrow$ one of the last pixels of arrays $(i, j)$
    $\mathbf{H} \leftarrow$ zeros$(N, M)$
    $\mathbf{H}(i', j') \leftarrow 1$
    $\mathbf{H} \leftarrow \mathbf{H} \oplus \mathbf{T}$       ▷ dilation
    $\mathbf{I} \leftarrow \mathbf{I} + \mathbf{H}$            ▷ logical or
    $\mathbf{Z} \leftarrow (1 + \alpha \text{ rand })\mathbf{Z} + \mathbf{I}$   ▷ update of $\mathbf{Z}$
$\mathbf{G} \leftarrow$ filtering($\mathbf{Z}$, Gaussian Mask($\sigma$))   ▷ smooth of $\mathbf{Z}$
$\theta_Z \leftarrow$ orientation of $\mathbf{Z}$
$\mathbf{G}_r \leftarrow$ rotate($\mathbf{G}, \phi - \theta_Z$)   ▷ rotation
$\mathbf{A} \leftarrow \sqrt{\mathbf{G}_r}$                    ▷ non-linear scale
$\mathbf{D} \leftarrow d_{\max}(\mathbf{A} - \mathbf{A}_{\min})/(\mathbf{A}_{\max} - \mathbf{A}_{\min})$   ▷ linear scale
$\mathbf{d} \leftarrow$ rectangular window where $\mathbf{D} > 0$   ▷ bounding box
$\mathbf{X}_2 \leftarrow ce^{\mu\mathbf{d}} + b$               ▷ simulation of X-ray process

Output:
$\mathbf{X}_2$                                              ▷ simulated X-ray image
```

ing [49].[7] In following sub-sections, we describe in further details the datasets, the simulation process, the experiments and the results. The performance is evaluated using the average precision (AP) that is obtained by thresholding the normalized area of intersection over union (IoU) at a IoU threshold α [50].

3.1 Casting Inspection

In casting inspection, we follow the experimental protocol suggested in [17] in which object detection methods were tested using as training data simulated ellipsoidal defects. In our experiments, we included the defects simulated by the stick-model.

- **Datasets** From $\mathbb{GDX}$ray, we used series C0001 that contains 72 X-ray images of a specific casting type with an annotated ground truth that includes real defects. In series C0001, there is a unique casting piece that is radiographed from different points of view. Defects on these castings have a round shape with a diameter $\emptyset = 2.0$–7.5 mm. Many of them are in positions of the casting which were known to be difficult to detect (for example at edges of regular structures).
- **Training and validation subsets** To build the training and validation subsets, we use the following procedure: *i) Pre-processing* The size of the images of series C0001 is 572×768 pixels. We resized them by a factor of two to 1144×1536 pixels. *ii) Selection* Series C0001 has 72 X-ray images. For each resized X-ray image, we randomly select 100 windows of 640×640 pixels, where there are no real defects. Some examples are given in Figs. 3 and 5. From the 100 images, 90 are selected for the training subset and 10 for the validation subset. *iii) Simulation:* In each selected window of the previous step, we simulate defects using the ellipsoidal model (see Sect. 2.1.3) and the stick model (see Sect. 2.1.4). For this purpose, we use algorithms 1 and 2. For each X-ray image 20 defects were simulated in average. The location, size and orientation of the defects was set randomly. In our experiments, we set randomly the number of simulated defects per image (from 2 to 20). The location of them were defined randomly. For the ellipsoidal model, the size and orientation of the three axes of the ellipsoid were set randomly from 1 to 9 mm and from 0 to 360^0 respectively. For the stick model we used the same parameters mentioned in Fig. 5 in third row (due to the random nature of the algorithm, each simulated stick-defect is different). The parameters were manually defined to obtain simulated defects similar to the real ones. Additionally, for each simulated defect we store the coordinates of the bounding box that encloses it. Summarizing, for series C0001, we have 7200 X-ray images of 640×640 pixels with around 80,000 simulated defects (with no real defects).
- **Testing subset** Similarly to training and validation subsets, to define the testing subset we use the following steps: *i) Pre-processing:* The size of the images of series C0001 is 572×768 pixels. We resized them by a factor of two to 1144×1536 pixels. *ii) Selection:* Series C0001 has 72 X-ray images. For each resized X-ray image, we randomly select 10 windows of 640×640 pixels, in which there may be real defects. Summarizing, for series C0001, we have 720 X-ray images of 640×640 pixels with around 650 real defects. It is worthwhile to mention, that in testing dataset there are no simulated defects.
- **Experiments and Results** In our experiments, we tested methods based on YOLO, RetinaNet and EfficientDet,

[7] $\mathbb{GDX}$ray is a public dataset for X-ray testing with around 20.000 X-ray images that can be used free of charge, for research and educational purposes only.

Table 1 AP in casting inspection for different α* values

Method	Sim**	0.1	0.2	0.25	0.33	0.5
YOLOv3-SPP	E [17]	0.9253	0.9020	0.8735	0.7377	0.2564
	S	0.9068	0.8667	0.8550	0.8073	0.6387
YOLOv3-Tiny	E [17]	0.9258	0.9061	0.8753	0.7471	0.2496
	S	0.9132	0.8698	0.8525	0.7969	0.6472
YOLOv5s	E [17]	0.9212	0.8883	0.8825	0.7946	0.3140
	S	0.9000	0.8775	0.8694	0.8333	0.6345
YOLOv5l	E [17]	0.9148	0.8850	0.8724	0.7654	0.3291
	S	0.8961	0.8672	0.8305	0.7660	0.5436
YOLOv5m	E [17]	0.9142	0.8868	0.8716	0.7704	0.3039
	S	0.8969	0.8589	0.8201	0.7641	0.5554
YOLOv5x	E [17]	0.9151	0.8891	0.8741	0.7932	0.3210
	S	0.8640	0.8361	0.7921	0.7264	0.5301

* α is the threshold used in intersection over union criterion simulation
** E for ellipsoidal and S for stick models

however, the last two ones achieved low performance[8]. We included two experiments (see Table 1): in the first one, reported in [17], the simulated defects used in the training stage were based on ellipsoids ('Sim=E') outlined in Section 2.1.3, whereas in the second one, the simulatds defects were based on sticks ('Sim=S') presented in Section 2.1.4. In both experiments, the testing dataset was the same: only real X-ray images with no simulated defects as explained above. See an example in Fig. 1. The results are summarized in Table 1 for $\alpha = 0.1, 0.2, 0.25, 0.33, 0.5$. Since the defects are very small (many of them are 20×20 pixels), and the resolution of the manual annotation is poor, a IoU threshold $\alpha =$ 0.25–0.33 is adequate. We conclude: *i*) the effectiveness of the method is evident, and *ii*) the simulation strategy based on sticks is better than the one based on ellipsoids.

3.2 Welding Inspection

We use the same approach explained Sect. 3.1:

- **Datasets** From $\mathbb{GDX}$ray, we used series W0001 that contains 10 X-ray images of welds with annotated real defects. From them we test on following six images: W0001_000X.png for X = 1, 2, 3, 5, 6, and 8.
- **Training and validation subsets** To build the training and validation subsets, we use the following procedure: *i*) *Pre-processing* Each image X, is resized by a factor of two. *ii*) *Simulation* In each image X, we simulate 40 defects in places where there is no real defects using the stick model (see Sect. 2.1.4). The location, size, shape and orientation of the defects were set randomly. We repeat this step six times. The ellipsoidal model was not used because it cannot simulate cracks. *iii*) *Selection:* For each of the six times, we randomly select 500 windows of 320 $\times$ 320 pixels, where there are no real defects. That means, for each image X we have 3000 windows of 320 $\times$ 320 pixels with simulated defects. They are resized again by a factor of two to 640 $\times$ 640 pixels because object detection methods work better with this size. Some examples are shown in Figs. 3 and 5. From them, 2700 images were selected for training and 300 for validation. All simulated defects were partitioned in small segments using watershed algorithm [51] to avoid large bounding boxes that contain thin cracks in diagonal. Finally, for each simulated defect we store the coordinates of the bounding box that encloses it. Since the X-ray images X were taken from very different welding processes (very different welded pipelines), in this step, we train one model for each image X. Summarizing, for each image X of series W0001, we have 3000 X-ray images of 640 $\times$ 640 pixels with around 7100 simulated defects in average.
- **Testing subset** Similarly to training and validation subsets, to define the testing subset we use the following steps: *i*) *Pre-processing* For each image X, we resized the images of series W0001 by a factor of four. *ii*) *Selection* For each resized X-ray image, we randomly select 1000 windows of 640 $\times$ 640 pixels, in which there may be real defects. For the ground truth, we have a binary image where the pixels are '1' if they belong to a defect. Summarizing, for each image X of series W0001, we have 1000 X-ray images of 640 $\times$ 640 pixels with around 7600 real defects. It is worthwhile to mention that in the testing dataset there are no simulated defects.
- **Experiments and results** In our experiments, we only tested YOLOv3-Tiny and YOLOv5s detectors because they achieved a good performance in the detection of

[8] For example, we obtained $AP = 0.53$ for RetinaNet and and $AP = 0.46$ for EfficientDet at $\alpha = 0.33$ using ellipsoidal defects.

Table 2 AP in welding inspection

Method	X=1	X=2	X=3	X=5	X=6	X=8
YOLOv5s	0.4663	0.5746	0.4899	0.5551	0.6932	0.6268
YOLOv3-Tiny	0.4902	0.6126	0.4320	0.6159	0.6430	0.7933
Train-defects	12,819	4549	6612	4710	4696	5336
Val-defects	1425	482	802	479	533	556
Test-defects	39,140	658	1423	2039	2353	90

defects in casting inspection as we reported in the previous section. See some results in Fig. 1. For the computation of precision-recall values, we count the pixels that were correctly and wrongly detected because the shape of the defects are so intricate that it is very difficult to determine where a defect starts and ends. The effectiveness of the proposed method is shown in Table 2. In average (of the six images), YOLOv3-Tiny achieves and average precision of 0.60.

3.3 Baggage Inspection

For baggage inspection, we use the approach outlined in Sect. 3.3 in which we superimpose X-ray images of isolated threat objects onto real X-ray images of backpacks with no threat objects. We follow the experimental protocol suggested in [25][9] whose details we describe below.

- **Datasets** From $\mathbb{GDX}$ray, we used three groups of series of X-ray images: G1) backpacks with no threat objects (series B0085), G2) isolated target objects (series B0049 for handguns, B0051 for razor blades, B0052 for shuriken, B0076 for knifes, and B0082 for non-threat objects) and G3) real X-ray images of backpacks with threat objects (series B0046). The first two groups are used to build the training/validation subset. The last group is used as testing subset.
- **Training and validation subsets** To build the subset for training purpose, we superimpose the X-ray images of group G2 onto the X-ray images of group G1 in random positions using Algorithm 1. In this case, we have four classes ('Gun', 'Knife', 'Razor Blade' and 'Shuriken'). Some examples are given in Fig. 6. In this dataset, we generate 9615 X-ray images, where there are around 6,000 threat objects for each class. From this dataset, we randomly select 90% for training and 10% for validation.
- **Testing subset** We used the images of group G3 as testing subset. In this group, there were 210 guns, 24 knives, 78 razor blades, and 33 shuriken.
- **Experiments and results** In our experiments, we tested SSD and YOLO detectors. The achieved results are summarized in Table 3. An example is shown in Fig. 1. Similar results have been reported in [25] for the first three methods. The last five methods (based on YOLOv4 and YOLOv5), however, correspond to new experiments.

[9] Specifically, we conducted Experiment 'A' and Testing Subset 1 of [25].

Table 3 AP in baggage inspection for $\alpha^* = 0.5$

Method	Gun	Knife	Blade	Shuriken	mAP
SSD [25]	0.8857	0.0917	0.5037	0.6147	0.5239
YOLOv2 [25]	0.7893	0.6921	0.4151	0.7002	0.6492
YOLOv3 [25]	0.9517	0.9206	0.8156	0.8517	0.8849
YOLOv4	0.9957	0.9092	0.7145	0.9006	0.8800
YOLOv5s	0.9010	0.8390	0.1750	0.7570	0.6680
YOLOv5m	0.9210	0.8510	0.2030	0.7560	0.6830
YOLOv5l	0.9290	0.8750	0.1980	0.7690	0.6930
YOLOv5x	0.9250	0.8450	0.2250	0.7690	0.6910

α is the threshold used in intersection over union criterium

3.4 Discussion

In our experiments, we implemented several object detectors based on YOLO, SSD, RetinaNet and EfficientDet. In the reported results, it is evident that the achieved performance of YOLO-based detectors is very good. RetinaNet and EfficientDet achieved a low performance in the detection of small defects, probably because they have been designed for larger objects. To overcome this problem, we could increase the resolution of the training images, but that would increase the training time considerably.

We believe that the proposed methodology could satisfy the requirements in the industry due to the following three attributes: *i*) *Simplicity* The construction of the training dataset is very simple because we only need a low number of target-free X-ray images and a simulation process for including simulated targets in the dataset. That means, no manual annotation is required. *ii*) *Effectiveness* The performance of the methods was high enough. The average precision was 0.91, 0.60 and 0.88 for casting, welding and baggage inspection respectively. Probably, a better performance could have been achieved in welding inspection if the simulation had been able to simulate all possible shapes and sizes of defects. It is known that the wide variety of possible defects in welds makes detection very challenging. *iii*) *Speed* The computational time in training and testing stage are very low: We need a 1–3 h for training, and it can be used in real time inspection to aid human operators (about 11 ms per image).

4 Conclusions

Our methodology provides an effective way to overcome the data scarcity problem in X-ray testing. We proposed a training strategy for target detection using target-free X-ray images with superimposition of simulated targets. Thus, no manual annotations are required because the locations of all simulated targets are known. In addition, testing is carried out using real X-ray images with real targets in the automated inspection of aluminum castings, welds, and baggage. That means, the testing stage corresponds to real scenarios. Besides, a new simulation approach is presented that generates very realistic defects (based on superimposition of 'stick'-pattern located randomly). This model has been successfully used in the detection of discontinuities in casting and welding inspection.

For the detection of targets, we used well-established object detection methods (YOLO, SSD, RetinaNet and EfficientDet), all of them developed in the last 3 years with many examples in public repositories that were adapted to our task. The implemented strategies have been simple, effective, and fast. The training stage requires a relatively small number of X-ray images. In addition, on the testing dataset (with real targets), the achieved performance was very high, and the computational time is very low (the solution can be used in real time).

The code and the datasets used in this paper are available on a public repository, so anyone can reproduce (and improve) the reported results, or reuse the code in other inspection tasks.

Acknowledgements This work was supported in part by Fondecyt Grant, No. 1191131 from National Science Foundation of Chile.

References

1. Mery, D., Pieringer, C.: Computer Vision for X-ray Testing, 2nd edn. Springer, Basel (2021)
2. Duan, J., Liu, X.: Online monitoring of green pellet size distribution in haze-degraded images based on vgg16-lu-net and haze judgment. IEEE Trans. Instrum. Meas. **70**, 1–16 (2021)
3. Zhang, D., Gao, S., Yu, L., Kang, G., Wei, X., Zhan, D.: Defgan: Defect detection gans with latent space pitting for high-speed railway insulator. IEEE Trans. Instrum. Meas. **70**, 1–10 (2020)
4. Yang, J., Fu, G., Zhu, W., Cao, Y., Cao, Y., Yang, M.Y.: A deep learning-based surface defect inspection system using multiscale and channel-compressed features. IEEE Trans. Instrum. Meas. **69**(10), 8032–8042 (2020)
5. Gong, X., Su, H., Xu, D., Zhang, J., Zhang, L., Zhang, Z.: Visual defect inspection for deep-aperture components with coarse-to-fine contour extraction. IEEE Trans. Instrum. Meas. **69**(6), 3262–3274 (2020)
6. Hou, W., Tao, X., Xu, D.: Combining prior knowledge with CNN for weak scratch inspection of optical components. IEEE Trans. Instrum. Meas. **70**, 1–11 (2020)
7. Liu, L., Ouyang, W., Wang, X., Fieguth, P., Chen, J., Liu, X., Pietikäinen, M.: Deep learning for generic object detection: A survey. Int. J. Comput. Vis. **128**(2), 261–318 (2020)
8. Lu, X., Ji, J., Xing, Z., Miao, Q.: Attention and feature fusion SSD for remote sensing object detection. IEEE Trans. Instrum. Meas. **70**, 1–9 (2021)
9. Xi, D., Qin, Y., Luo, J., Pu, H., Wang, Z.: Multipath fusion mask R-CNN with double attention and its application into gear pitting detection. IEEE Trans. Instrum. Meas. **70**, 1–11 (2021)
10. Jin, C., Kong, X., Chang, J., Cheng, H., Liu, X.: Internal crack detection of castings: A study based on relief algorithm and Adaboost-SVM. Int. J. Adv. Manuf. Technol. **108**, 1–10 (2020)
11. Cogranne, R., Retraint, F.: Statistical detection of defects in radiographic images using an adaptive parametric model. Signal Process. **96**, 173–189 (2014)
12. Mery, D.: Inspection of complex objects using multiple-X-ray views. IEEE/ASME Trans. Mechatron. **20**(1), 338–347 (2015)
13. Bandara, A., Kan, K., Morii, H., Koike, A., Aoki, T.: X-ray computed tomography to investigate industrial cast Al-alloys. Prod. Eng. **14**(2), 147–156 (2020)
14. Du, W., Shen, H., Fu, J., Zhang, G., He, Q.: Approaches for improvement of the X-ray image defect detection of automobile casting aluminum parts based on deep learning. NDT E Int. **107**, 102144 (2019)
15. Ferguson, M., Ak, R., Lee, Y.-T. T., Law, K. H.: Automatic localization of casting defects with convolutional neural networks. *2017 IEEE International Conference on Big Data*. IEEE, pp. 1726–1735 (2017)
16. Mery, D.: Aluminum casting inspection using deep learning: A method based on convolutional neural networks. J. Nondestruct. Eval. **39**(1), 12 (2020)
17. Mery, D.: Aluminum casting inspection using deep object detection methods and simulated ellipsoidal defects. Mach. Vis. Appl. **32**(3), 1–16 (2021)
18. Shao, J., Du, D., Chang, B., Shi, H.: Automatic weld defect detection based on potential defect tracking in real-time radiographic image sequence. NDT E Int. **46**, 14–21 (2012)
19. Baniukiewicz, P.: Automated defect recognition and identification in digital radiography. J. Nondestruct. Eval. **33**(3), 327–334 (2014)
20. Hou, W., Wei, Y., Guo, J., Jin, Y., et al.: Automatic detection of welding defects using deep neural network. J. Phys. **933**, 012006 (2018)
21. Pan, H., Pang, Z., Wang, Y., Wang, Y., Chen, L.: A new image recognition and classification method combining transfer learning algorithm and mobilenet model for welding defects. IEEE Access (2020)
22. Suyama, F.M., Delgado, M.R., da Silva, R.D., Centeno, T.M.: Deep neural networks based approach for welded joint detection of oil pipelines in radiographic images with double wall double image exposure. NDT E Int. **105**, 46–55 (2019)
23. Mery, D., Saavedra, D., Prasad, M.: X-ray baggage inspection with computer vision: A survey. IEEE Access **8**, 145–620 (2020)
24. Mery, D., Svec, E., Arias, M., Riffo, V., Saavedra, J.M., Banerjee, S.: Modern computer vision techniques for X-ray testing in baggage inspection. IEEE Trans. Syst. Man Cybern. **47**(4), 682–692 (2016)
25. Saavedra, D., Banerjee, S., Mery, D.: Detection of threat objects in baggage inspection with X-ray images using deep learning. Neural Comput. Appl. **18**, 1–17 (2020)

26. : Akçay, S., Kundegorski, M. E., Devereux, M., Breckon, T. P.: Transfer learning using convolutional neural networks for object classification within X-ray baggage security imagery. In *Image Processing (ICIP), 2016 IEEE International Conference on*. IEEE, 2016, pp. 1057–1061
27. Akcay, S., Kundegorski, M.E., Willcocks, C.G., Breckon, T.P.: Using deep convolutional neural network architectures for object classification and detection within X-ray baggage security imagery. IEEE Trans. Inf. Forensic Secur. **13**(9), 2203–2215 (2018)
28. Akcay, S., Breckon, T. P.: An evaluation of region based object detection strategies within X-ray baggage security imagery. In: *Image Processing (ICIP), 2017 IEEE International Conference on*. IEEE, 2017, pp. 1337–1341
29. Miao, C., Xie, L., Wan, F., Su, C., Liu, H., Jiao, J., Ye, Q.: Sixray: A large-scale security inspection X-ray benchmark for prohibited item discovery in overlapping images. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2119–2128 (2019)
30. Aydin, I., Karakose, M., Erhan, A.: A new approach for baggage inspection by using deep convolutional neural networks. In *2018 International Conference on Artificial Intelligence and Data Processing (IDAP)*. IEEE, pp. 1–6 (2018)
31. Bhowmik, N., Wang, Q., Gaus, Y. F. A., Szarek, M., Breckon, T. P.: The good, the bad and the ugly: Evaluating convolutional neural networks for prohibited item detection using real and synthetically composited X-ray imagery. arXiv:1909.11508 (2019)
32. Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative Adversarial Networks. arXiv:1406.2661 (2014)
33. Akcay, S., Atapour-Abarghouei, A., Breckon, T. P.: Ganomaly: Semi-supervised anomaly detection via adversarial training. arXiv:1805.06725 (2018)
34. Sangwan, D., Jain, D. K.: An evaluation of deep learning based object detection strategies for threat object detection in baggage security imagery. Pattern Recogn. Lett. (2019)
35. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press, Cambridge (2016)
36. Zhao, Z., Zheng, P., Xu, S., Wu, X.: Object detection with deep learning: A review. IEEE Trans. Neural Netw. Learn. Syst. **30**(11), 3212–3232 (2019)
37. Redmon, J., Divvala, S. K., Girshick, R. B., Farhadi, A.: You only look once: Unified, real-time object detection. *CoRR* In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2015)
38. Redmon, J., Farhadi, A.: YOLO9000: Better, faster, stronger. *CoRR*, In: Proceedings of the IEEE conference on computer vision and pattern recognition (2016)
39. Redmon, J., Farhadi,A.: Yolov3: An incremental improvement. *CoRR*, vol. arXiv:1804.02767 (2018)
40. Bochkovskiy, A., Wang, C.-Y., Liao, H.-Y. M.: YOLOv4: Optimal speed and accuracy of object detection (2020)
41. G. J. et al.: *ultralytics/yolov5*, 2020 (released in June 2020), https://github.com/ultralytics/yolov5
42. Tan, M., Pang, R., Le, Q. V.: EfficientDet: Scalable and efficient object detection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* pp. 10781–10790 (2020)
43. Lin, T., Goyal, P., Girshick, R. B., He, K., Dollár, P.: Focal loss for dense object detection. *CoRR*, vol. arXiv:1708.02002 (2017)
44. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S. E., Fu, C., Berg, A. C.: SSD: single shot multibox detector. *CoRR*, vol. arXiv:1512.02325 (2015)
45. Mery, D.: A new algorithm for flaw simulation in castings by superimposing projections of 3D models onto X-ray images. In *Proceedings of the XXI International Conference of the Chilean Computer Science Society (SCCC-2001)*. Punta Arenas: IEEE Computer Society Press, 6–8 Nov. 2001, pp. 193–202
46. Mery, D., Hahn, D., Hitschfeld, N.: Simulation of defects in aluminum castings using cad models of flaws and real X-ray images. Insight **47**(10), 618–624 (2005)
47. Mery, D., Katsaggelos, A.: A logarithmic X-ray imaging model for baggage inspection: Simulation and object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 57–65 (2017)
48. Als-Neielsen, J., McMorrow, D.: Elements of Modern X-ray Physics, 2nd edn. Wiley, Hoboken (2011)
49. Mery, D., Riffo, V., Zscherpel, U., Mondragón, G., Lillo, I., Zuccar, I., Lobel, H., Carrasco, M.: GDXray: The database of X-ray images for nondestructive testing. J. Nondestruct. Eval. **34**(4), 1–12 (2015)
50. Szeliski, R.: Computer Vision: Algorithms and Applications, 2nd edn. Springer, New York (2020)
51. Gonzalez, R., Woods, R.: Digital Image Processing, 3rd edn. Prentice Hall, Pearson (2008)