Synthetic Data Generation for Surface Defect Detection

Déborah Lebert, Jeremy Plouzeau, Jean-Philippe Farrugia, Florence Danglade, Frédéric Merienne

Synthetic data generation for surface defect detection*

Déborah Lebert[1], Jérémy Plouzeau[1], Jean-Philippe Farrugia[2], Florence Danglade[1], and Frédéric Merienne[1]

[1] ENSAM, LISPEN, institut image, Chalon sur Saône, France
deborah.lebert@ensam.eu, Florence.Danglade@ensam.eu, Jeremy.PLOUZEAU@ensam.eu, frederic.merienne@ensam.eu
[2] LIRIS, Lyon, France
jean-philippe.farrugia@univ-lyon1.fr

Abstract. Ensuring continued quality is challenging, especially when the customer satisfaction is basically the provided service. It seems to become easier with new technologies like Artificial Intelligence. But to design an intelligent assistant, field data are necessary but not always available. Synthetic data are largely used to replace real data. Made with a Generative Adversarial Networks or a rendering engine, they aim to be as efficient as real ones to train a Neural Network. When synthetic data generation meet the challenge of object detection, its capacity to deal with defect detection challenge is unknown. Here we demonstrate how to generate these synthetic data to detect defects. Through iterations we apply different methods from literature to generate synthetic data for object detection. From how to extract a defect from the few data we have, to how to organize the scene before data synthesis. Our study suggests that defect detection may be performed by training an object detector neural network with synthetic data and gives a protocol to do so even if at this point, no field experiments have been conducted to verify our detector performances under real conditions. This experiment is the starting point to develop a mobile and automatic defect detector that might be adapted to ensure new product quality.

Keywords: Database Generation · Synthetic Data · Defect Detection

1 Introduction

When customer satisfaction is the main objective of a product, constant improvement in quality control not only during its production but also all along its life circle become essential. Our study focuses on ensuring cosmetic continued quality during the life of the product thought intermittent inspection of all products. But these inspections are time consuming and limited by the product uptime. The challenge here is then to develop an assistant for inspectors that would detect and differentiate defects from normal wear. Our assistant must then

* Supported by organization ENSAM

be based on visual features like detector using Neural Network (NN) are. However data collection for deep learning is time-consuming. Therefore, our assistant must train using generated data [1]. Our assistant should be able to detect each defect whatever its shape is. Thus during our study we will see how to generate synthetic data in order to train a NN.
This paper starts with a state of the art on object detection using synthetic data in section 2. Section 3 aims to contextualize our project by highlighting its scientific issues, section 4 presents the generation and customization process. Finally, section 5 presents the first results on real data. To finish, we will conclude and introduce our future development on this topic in section 6.
In the following, the term "SD" will be used to designate Synthetic Data.

2 Background in defect detection using synthetic data

To satisfy the increasing need of ensured quality, companies are including more validation processes from random inspection to computer vision solution in order to get a global idea of the quality of all pieces. For example computer vision can be used to detect structure defect by comparing the actual piece with its digital twin [2] which is a numerical representation of the part designed and modified according to the part evolution. It can also be used to detect surface defect using a NN trained with a set of real pictures of defects as in [3].

Our project requires to create an original dataset, but it can become time consuming especially when dealing with new classes as “wearing a mask” [1] or specific ones as chemical classification. In addition, deep learning requires a large quantity of data that must be annotated which is most of the time manually performed. Then SD appears to be a possible solution to these problems. So we choose to generate our own dataset using previous work on SD generation.

First used in economic field in the form of de-identified data [4], SD are now used to face NN challenges. In fact SD play an important role when data are nonexistent or unavailable for privacy reasons. In his book [4], K. El Emam highlights the main difficulty when dealing with SD which is “showing that the results from the SD are similar to the results from the real data“. Through this study, we will try to fit this definition with our SD.
Through studies we identify two ways to generate SD. The first is Generative Adversarial Network (GAN), first introduced in 2014 by Goodfellow and al. [5]. GAN is already used for defect detection as in [6]. It can be considered as synthetic data augmentation using one generative and one discriminant network. Indeed generative part creates new images based on provided dataset when discriminant one checks its plausibility. Then it needs a large amount of real data, as seen in [6] where S. Jain and co. used 5400 images to train their GAN.
In our context, approximately fifty pictures of defects are available. Therefore a GAN cannot be used.

Unlike the first method, the second one do not need real pictures of the target but a numerical representation of it. Indeed in industry 4.0 it is easy to directly

get our target digital twin or to build it using some pictures. Then it is possible to use any rendering engine to generate data as in [8]. For instance, M. Johnson-Roberson and al. used a video game (GTA V) to generate traffic dataset in [7], and J. Cohen used a CAO software to do so in [8]. In our context we choose to focus on this second methodology since 3D models of our targets are directly available and defect generation can be done from images or mathematical models.

As highlighted in [9] and [16], SD generation strategy depends on features we want to detect. Thus realism requirement seems to be determinant for synthetic data effectiveness as emphasized in most studies about SD generation [10] [11]. Since our data aim to be used to train a NN, their realism ie. their ability to appear to people as real pictures instead of generated ones [12] will not be studied here. Indeed, as highlighted in [13], if generated SD are supposed to be used for training then plausibility of generated scene is more important than its global realism. But when GAN are creating realistic data thanks to real ones, manual generation of SD has to simulate it through environment randomization and 3D model precision. That is why we will consider realism as the accurate, detailed, unembellished depiction [3] of our real object.

Consequently, when identifying an object from its texture, it is necessary to get an accurate capture of it. As seen in [11] it is possible to use real images to "cut and paste" real object form pictures to synthetic images or to directly use a textured 3D scan [15]. Indeed some studies proved SD realism importance for NN training, first regarding environment in [17] and [18], then concerning the targeted object in [13] and [19]. Additionally, some studies worked on photorealism in SD as in [10], [8] and [14] in which teams enhanced it by adding noise. Equally J. Hodapp and al. [11] demonstrated that a good real/synthetic data distribution can improve NN performances: their best distribution is 5% of real data for 47,5% of SD and 47,5% of "mixed" data or "cut-and-paste" data in which SD are generated with CAO software without any photorealism requirement, realism is achieved with "cut-and-paste" dataset.

3 Scientific issues

We saw that defect detection with NN was becoming common. However the use of SD in this field stays limited to GAN generated ones. Our designed detector is supposed to take part into maintenance procedures. However in our case, taking pictures of parts would be time-consuming not only to collect pictures but also to perform pre-process procedures to de-identify and annotate data. Moreover an important challenge is to differentiate real defects from regular attrition. A defect can be defined as an unexpected difference of color on a homogeneous surface. Here we are considering only 2 different surface defects: cracks and spots which occur mainly on plastic surfaces. The purpose of our work is then to determine how to generate an effective synthetic database. Since cracks and spots are surface defects they can be projected on a plane. This association may then

[3] definition from `https://www.britannica.com/`

be manipulated as an object, therefore we choose to generate SD using methods from literature seen in section 2. Breaks could be added to the defect list but they will not be studied here since detecting breaks cannot be compared with object detection and should be done using 3D comparison as in [2].

4 Our synthetic data generator and training

We decided to generate SD as in [8] and [7] where both teams where using SD to train an object detector, but to detect cracks and spots.
Our first step is to define a workflow to generate SD. We design our generation process using previous work on SD generation results [11] [8], given in figure 1. The scene is composed of our targeted defect and may include some other 3D object as distractors. Then we decide to simulate a real environment by varying lighting parameters (nature of sources, orientation and light color), background and post-processing parameters.

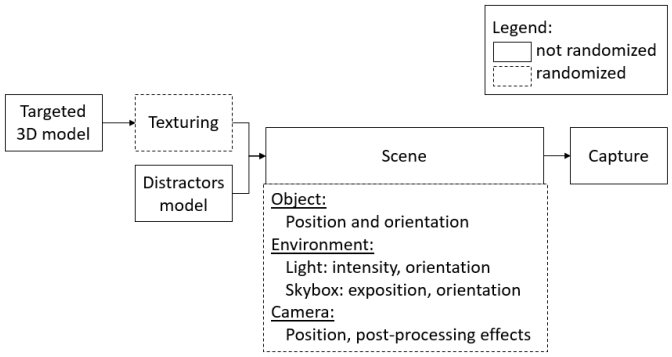


Fig. 1. Adopted workflow for synthetic data generation based on 3D Model

Our scene design aims to meet realism requirements previously defined ie. to get an accurate, detailed and unembellished depiction of the targeted object in a realistic and plausible environment. In order to do so we decide to use a game development platform (`https://unity.com`) as in [7] with its Universal Renderer Pipeline to get access to more post-processing parameters and high definition materials. Our scene parameters can be divided into two groups: extrinsic parameters and intrinsic ones. Extrinsic parameters are not modifying the defect but are essential for the scene plausibility: architecture of the scene and environment conditions, while intrinsic parameters are directly related to the aspect of the defect: its shape. Regarding extrinsic parameters, to meet the environment realism requirement, the targeted object is located in an empty space surrounded by a High Dynamic Range Image (HDRI). A HDRI is either

a panoramic picture or a cubemap contaning a large amount of data like the brightness, which can be used for the illumination of a virtual scene. The environment variability is simulated by rotating this skybox. Lightning environment is simulated by varying exposure of the skybox from 1 to 3 and by rotating an extra light source and randomizing its intensity from 1 to 10. Finally random noises are added as post-processing effect to simulate the user impact on the quality of the picture with motion blur (intensity between 0 and 0.2) or unmanageable environmental conditions like dust simulated with grain (intensity between 0 and 1). But to ensure scene plausibility, some distractors must be added. These 3D objects are used to prevent misclassification during detection by adding current geometry in the training dataset. Since the detector would be used during maintenance session, we pick around 20 distractors, mostly tools like screw-driver. Finally, the base material used for the projection of defects can be considered as an intrinsic and extrinsic parameter since its contrast with the defect will influence the detection however the detection should not depend on materials. That is why we decided to use a library of every material that could be used for all inspected parts and to check manually the contrast between defects and base materials.
Regarding intrinsic parameters, to meet defect realism, we decided to use "cut-and-paste" method [11] using defect pictures from online databases [4] (figure 3a) to get a mask (figure 3b) to inlay in our SD adding some random distortion and material to synthetically vary the shape of defect. For cracks, we isolated 8 different basic shapes from which we created around 100 different shapes. For spots, we got 10 different shapes from previous databases, and we created around 150 different shapes.
Then, data are generated according to the previous workflow, an example of generated data is given in figure 2 (figure 1) in which all extrinsic parameters are randomized and intrinsic parameters are set manually to ensure the defect visibility (defect proportion, defect contrast with its material).

Every set of data is evaluated through performances of the NN. First, we manually run our detector on the limited amount of field data in order to identify improvement axes. When no other axe of improvement can be highlighted by this method we check our dataset plausibility by measuring the mean average precision and the Intersection over Union (IoU) score of each NN trained with our sets. Here the IoU loss we use, is presented in [20] as the "computing process [that] will trigger the calculation of the four coordinate points of the bounding box by executing IoU with the ground truth". In order to do so we create a test set of real data to get around 70 real pictures. Finally the verified dataset is used to train the final multiple class detector adding some multiple class generated pictures. Then performances of the final detector are checked on every classes thanks to the previous test-set before. The final test will be to check the performances of our detector in real conditions ie. handled by the final operator in real environment. Since our work does not aim to improve actual NN architecture,

[4] `http://defectsdatabase.npl.co.uk/defectsdb/defects_query.php` and `https://www.kaggle.com/yidazhang07/bridge-cracks-image`

Fig. 2. Example of generated data

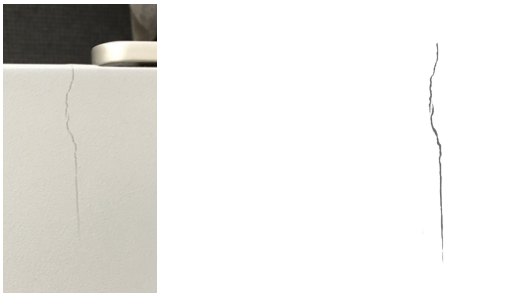
we set up supervised training for both object and defect detection reusing the well known regression-based architecture: YOLO v4 [5] [20] trained in Darknet environment [21]. Studies do not agree on a fixed number of images required for training, thus we decided to use at least 2,000 pictures for training (80% for training and 20% for during training testing) as seen in training YOLO v4 on custom dataset project[6]. In our case we decided to generate 1,000 pictures at first per class and to increase the set at each iteration depending on deficiencies of NN trained with the previous generation. To identify it easily we chose to work on each class independently before mixing training to generate our final detector. To speed up training process we decided to fine-tune our model using weights pre-trained for object detection.

5 Results

Our evaluation process is divided into 3 steps: first we check our detector result on a set of SD, then we use a limited amount of real pictures we retrieved to identify axes of improvement. and finally, we run the detection on a video and on the field experiment afterwards. This improvement process was more focused on improving defect detection since the part detection gets an average acuracy of 98% for 8 different pats.
Regarding defect, the first generation of SD shows defects on a big planer surface, we can directly see on the result on real data that the NN doesn't detect the defect (figure 4a), but it detects a false positive defect. Our objective was then to detect the defect and to reduce this false positive detection. In order to do so we decided to add some distractors and to reduce the plane size in our scene, even if our false positive detection are mainly related to the scene composition and

[5] YOLO is a real-time object detector, previous work [1] [16] [14] used different version of this NN to perform object detection

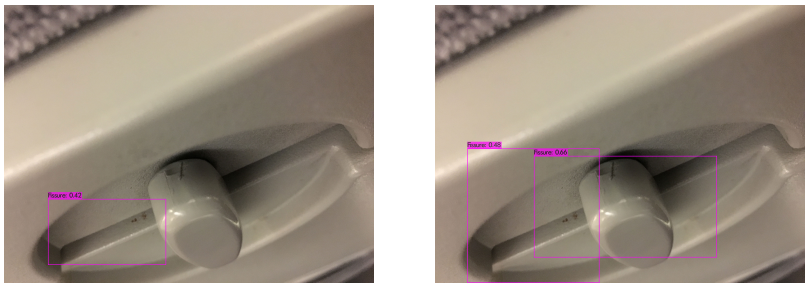
[6] `https://github.com/AlexeyAB/darknet`



(a) Original defect

(b) Extracted defect

Fig. 3. "Cut-and-paste" defect



(a) Detection results with the first generation

(b) Detection results with the second generation

Fig. 4. Results obtained after 3 iterations of our improvement process

to shadows. We choose distractors related to maintenance operations in order to prevent false positive detection. Finally, our defects and parts detector get a F1-score of 88% with a precision of 90% and a recall of 87% when testing its performances on the synhtetic test set. The next step will be to run detection on a on the field video, a capture of the resulting video is given in figure 5.

6 Conclusion and future work

Regarding application development, we defined and validated both SD generation workflow and a protocol to improve generated dataset especially through an upgrade of environment plausibility. Indeed, performances of our surface defect detector on SD enable us to validate our process according to K. El Emam's definition of SD [4]. Now, we must measure our detector performances with a proper protocol, to qualitatively check our synthetic data efficiency and ensure that our device become an effective assistance for inspection. This will imply

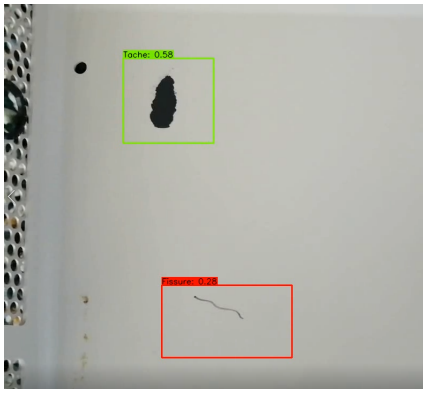


Fig. 5. Defect and object detection on real armrest

to study our application impact on inspection preparation, inspection and its results compilation as well as user's impact on performances of the detector. Furthermore some extra work can be done on the training stage, indeed recent studies show good performances on detecting surface defects using unsupervised learning [22]. Thus a comparaison between both training strategies using SD must help to design an accurate surface defect detector.

References

1. J. Yu and W. Zhang, Face Mask Wearing Detection Algorithm Based on Improved YOLO-v4, DOI 10.3390/s21093263.
2. I. Jovančević, J.J. Orteu, T. Sentenac and R. Gilblas, Inspection d'un aéronef à partir d'un système multi-capteurs porté par un robot mobile, *14ème Colloque Méthodes et Techniques Optiques pour l'Industrie* (2015), `https://hal.archives-ouvertes.fr/hal-01350898`.
3. D. Tabernik and S. Šela and J. Skvarč and D. Skočaj, Segmentation-based deep-learning approach for surface-defect detection, *Journal of Intelligent Manufacturing* (2019) vol.31 pp.759–776, DOI 10.1007/s10845-019-01476-x.
4. K. El Emam, Accelerating AI with Synthetic Data, `https://www.nvidia.com/fr-fr/deep-learning-ai/resources/accelerating-ai-with-synthetic-data-ebook/`ebook link, (2020), ch.1&2.
5. I. Goodfellow, J. Pouget-Abadie, M. Mirza, S. Ozai, A. Courville, Y. Bengio, Generative adversarial networks, *In Advances in neural information processing systems 27*, p.139-144, vol. 63, (2014), DOI 10.1145/3422622.
6. S. Jain, G. Seth, A. Paruthi, U. Soni, G. Kumar, Synthetic data augmentation for surface defect detection and classification using deep learning, *Journal of Intelligent Manufacturing*, (2020), DOI 10.1007/s10845-020-01710-x.
7. M. Johnson-Roberson, C. Barto, R. Mehta, S.N. Sridhar, K. Rosaen, R. Vasudevan, Driving in the Matrix: Can virtual worlds replace human-generated annotations for real world tasks?, *2017 IEEE International Conference on Robotics and Automation (ICRA)*, p.746-753 (2017), DOI 10.1109/ICRA.2017.7989092.

8. J. Cohen, C. Crispim-Junior, C. Grange-Faivre, L. Tougne, CAD-based Learning for Egocentric Object Detection in Industrial Context, *15th International Conference on Computer Vision Theory and Applications*, pp.644-651 (2020), DOI 10.5220/0008975506440651.
9. J. Tremblay, A. Prakash, D. Acuna, M. Brophy and V. Jampani, Training Deep Networks with Synthetic Data: Bridging the Reality Gap by Domain Randomization, *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp.1082-10828 (2018), DOI 10.1109/CVPRW.2018.00143.
10. P. F. Proença and Y. Gao, Deep Learning for Spacecraft Pose Estimation from Photorealistic Rendering, *2020 IEEE International Conference on Robotics and Automation (ICRA)* (2020) pp.6007-6013, DOI 10.1109/ICRA40945.2020.9197244.
11. J. Hodapp, M. Schiemann, C. Arcidiacono, M. Reichenbach and V. Bilous, Advances in Automated Generation of Convolutional Neural Networks from Synthetic Data in Industrial Environments, *Hawaii International Conference on System Sciences* (2020) pp.1278-1286, DOI 10.24251/HICSS.2020.565.
12. S. Fan, T.T. Ng, J. S. Herberg, B. L. Koenig, C. Y.-C. Tan and R. Wang, An Automated Estimator of Image Visual Realism Based on Human Cognition, *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 4201-4208 (2014), DOI 10.1109/CVPR.2014.535
13. T. Hodan, V. Vineet, R. Gal, E. Shalev, J. Hanzelka, T. Connell, P. Urbina, Sudipta N. Sinha and B. Guenter, Photorealistic Image Synthesis for Object Instance Detection, (2019), arXiv 1902.03334.
14. J. Huh, K. Lee, I. Lee, and S. Lee, A Simple Method on Generating Synthetic Data for Training Real-time Object Detection Networks, *2018 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)* (2015) pp.1518-1522, DOI 10.23919/APSIPA.2018.8659778.
15. M.Z. Wong, K. Kunii, M. Baylis, W.H. Ong, P. Kroupa and S. Koller, Synthetic dataset generation for object-to-model deep learning in industrial applications, *PeerJ Computer Science 5* (2019), DOI 10.7717/peerj-cs.222.
16. J. Jing, D. Zhuo, H. Zhang, Y. Liang and M. Zheng,Fabric defect detection using the improved YOLOv3 model, *Journal of Engineered Fibers and Fabrics, vol.15* (2020), DOI 10.1177/1558925020908268.
17. X. Peng, B. Sun, K. Ali and K. Saenko, Learning Deep Object Detectors from 3D Models, *2015 IEEE International Conference on Computer Vision (ICCV)* (2015) pp.1278-1286, DOI 10.1109/ICCV.2015.151.
18. K. Sarkar., K. Varanasi. and D. Stricker., Trained 3D Models for CNN based Object Recognition, *Proceedings of the 12th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 5: VISAPP, (VISIGRAPP 2017)*, DOI 10.5220/0006272901300137.
19. C. M. de Melo, B. Rothrock, P. Gurram, O. Ulutan and B. S. Manjunath, Vision-Based Gesture Recognition in Human-Robot Teams Using Synthetic Data, *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2020) pp.10278-10284, DOI 10.1109/IROS45743.2020.9340728.
20. Alexey Bochkovskiy and Chien-Yao Wang and Hong-Yuan Mark Liao, YOLOv4: Optimal Speed and Accuracy of Object Detection, (2020), arXiv 2004.10934.
21. Joseph Redmon, Darknet: Open Source Neural Networks in C, `http://pjreddie.com/darknet/` (2013-2016).
22. K. Roth, L. Pemula, J. Zepeda, B. Schölkopf, T. Brox and P. V. Gehler, Towards Total Recall in Industrial Anomaly Detection, *CoRR - Volume abs/2106.08265, 2021*, `https://arxiv.org/abs/2106.08265`.