Training with synthetic images for object detection and segmentation in real machinery images

Alonso J. Cerpa Salas
Department of Computer Science
Universidad Católica San Pablo
Arequipa, Peru
e-mail: alonso.cerpa@ucsp.edu.pe

Graciela Meza-Lovon
Department of Computer Science
Universidad Católica San Pablo
Arequipa, Peru
e-mail: gmezal@ucsp.edu.pe

Manuel E. Loaiza Fernández
Department of Computer Science
Universidad Católica San Pablo
Arequipa, Peru
e-mail: meloaiza@ucsp.edu.pe

Alberto Raposo
Tecgraf Institute
Pontifical Catholic University of Rio de Janeiro
Rio de Janeiro, Brazil
e-mail: abraposo@tecgraf.puc-rio.br

***Abstract*—Over the last years, Convolutional Neural Networks have been extensively used for solving problems such as image classification, object segmentation, and object detection. However, deep neural networks require a great deal of data correctly labeled in order to perform properly. Generally, generation and labeling processes are carried out by recruiting people to label the data manually. To overcome this problem, many researchers have studied the use of data generated automatically by a renderer. To the best of our knowledge, most of this research was conducted for general-purpose domains but not for specific ones. This paper presents a methodology to generate synthetic data and train a deep learning model for the segmentation of pieces of machinery. For doing so, we built a computer graphics synthetic 3D scenery with the 3D models of real pieces of machinery for rendering and capturing virtual photos from this 3D scenery. Subsequently, we train a Mask R-CNN using the pre-trained weights of COCO dataset. Finally, we obtained our best averages of 85.7% mAP for object detection and 84.8% mAP for object segmentation, over our real test dataset and training only with synthetic images filtered with Gaussian Blur.**

I. Introduction

During the last decade, great progress has been made on deep learning techniques such as Convolutional Neural Networks (CNN). However, for a successful generalization, these techniques require many training examples. To speed up the creation of a dataset, some researchers and enterprises contract services such as Google Cloud or Amazon WebServices, by which people are paid for labeling the data. However, this causes research projects to increase their costs.

Alternatively, the creation of synthetic data by using a renderer has been applied to speed up both data collection and labeling of examples in some domains, such as object segmentation. For this purpose, a renderer needs a 3D model of the object, along with several backgrounds. Both the object and the background are appropriately located in a scene, generating a synthetic image. In addition to the created image, a learning algorithm requires the output attribute. For image segmentation, that attribute is obtained by rendering the model of the object, without the background. In this way, the renderer generates an image with the silhouette or mask of the object. The process of generating and labeling data as described above has advantages over the manual one: 1) It is less time-consuming. 2) Human errors are avoided.

On the other hand, the synthetic generation also has a disadvantage: The generated synthetic data will not look 100% like the real data. Hence, training a model with synthetic data would not give the same results as training the same model with real data, i.e., the model trained with synthetic data will not adequately predict when tested with real data. This problem is known as the reality gap [1] [2]. There are two ways to deal with the reality gap problem in the domain of object segmentation when deep neural networks are used. The first way consists in increasing the similarity between the synthetic data and the real data, which can be carried out by creating a virtual scene similar to the image background of the real scene by using a photorealistic renderer [3]. The second way to overcome the reality gap is to use Domain Randomization (DR) [1], which creates new images by introducing and randomizing parameters in the scene. Examples of forms of DR include varying the lighting parameters, varying the textures of the objects, varying the backgrounds of the scene, applying different types of noises, applying different types of filters such as Gaussian blur, introducing distracting objects into the scene. Thus, DR makes the model exposed to many variations of the synthetic data, in this way when the model is exposed to real data, the model will assume that real data is only a version of the synthetic data with particular parameters, such as real textures, real illumination, or real filters. That is how the model will be able to bridge the reality gap, thanks to Domain Randomization (DR).

This work aims to generate a set of synthetic data of machinery pieces to evaluate if the training with this data overcomes the reality gap problem. We take aspects from both approaches above mentioned. So, we built a renderer based on the Phong Reflection Model [4] using OpenGL, and then we used the Mask R-CNN model [5] to perform the segmentation task. The experiments were carried out to evaluate whether applying Gaussian Blur to the synthetic images during the training stage achieves a precision similar to that using real data. The model was trained with three types of distinct pieces of machinery. Besides, tests were carried out by mixing the real data with the synthetic one to measure the importance of real data in the increase of the model's precision.

This paper is organized as follows. We describe the related work in Section II by examining the techniques that generate and label synthetic data for various tasks including object detection and segmentation. In Section III, we present the proposed methodology. First, we describe the Phong reflection model used to build the renderer and to obtain the synthetic data. Second, we introduce the Mask R-CNN network used for the segmentation of the images contained in the dataset. The experiments and results to assess whether the generated synthetic dataset, along with the training of Mask R-CNN network, allows us to overcome the reality gap are given in Section IV. Finally, the conclusions are drawn in Section V.

II. Related Work

This section describes the latest techniques for generating and labeling synthetic data, and shows, in Table I, a summary of the research jobs described below.

Ruiz et al. [6] proposed to generate labeled data of general-purpose objects (e.g., hairdryers, chairs) and objects of the self-driving domain (e.g., roads, trees, cars). The tasks they focused on were object detection, pose estimation, semantic segmentation, object segmentation, depth estimation, camera pose estimation, and 3D box estimation. Ruiz et al. used the Blender renderer to generate their data, and they also created their Python scripts to manage the camera, lighting, models, model positions, and orientations.

Another multi-tasking technique is called Falling Things, which was proposed by Tremblay et al. [7]. They used the NVIDIA Deep learning Dataset Synthesizer (NDDS) [12] tool to create a synthetic dataset for object pose estimation tasks with six degrees of freedom, object detection, and object segmentation. Falling Things generates synthetic data for YCB objects [13], which are general-purpose objects such as small boxes and cans of food. The difference between Falling Things [7] and the previous work by Ruiz et al. [6] is that Falling Things use a two-fold approach to generate the data. A part of their synthetic dataset was obtained by applying domain randomization, and the other part by photorealism. This two-fold approach helps to take advantage of the best of two worlds, to achieve state-of-the-art performance in object pose estimation with six degrees of freedom, as observed in a study of Falling Things by Tremblay et al. [2]. In the study, they demonstrated the effectiveness of Falling Things, testing the model on a robot capable of picking up and placing the YCB objects [13] with which it was synthetically trained.

Continuing the work of Falling Things [7], Tremblay et al. [8] used NVIDIA Deep learning Dataset Synthesizer to create a new synthetic dataset, which included cuboid objects, such as cars or toy cubes. The synthetic objects were employed for the network to learn to identify their poses. Besides, the authors made the program observe through a camera a human demonstration of a task which consisted in stacking several colored cubes vertically. Then, they used the recognition of the cubes the program learned and generated a plan (readable by a human being) from the recognized demonstration. Finally, the program executed the generated plan, and thus it learned to stack the cubes from the human demonstration captured by a camera. Tremblay et al. also demonstrate the effectiveness of their technique by testing it with a robot and real cuboid objects, as done in Falling Things [2].

Gaidon et al. [3] presented a technique called Virtual KITTI, which employs photorealism as the primary approach to close the reality gap. The tasks that can be performed with Virtual KITTI are object detection, tracking, scene, instance segmentation, depth image, and optical flow. The dataset contains models of cars, track, trees, houses, among others, necessary to train the models useful for self-driving cars. Another advantage of synthetic data in this work is that it allows data to be generated with particular climates such as fog or rain on demand, not as in reality, where these climates are less likely to occur.

There are particular scenarios for the generation of these synthetic datasets. In the first one, the objects are shiny and do not have texture. Chen et al. [9] addressed this problem by generating synthetic data of metal components using a renderer that estimates the 6D pose of an object. For doing so, they divided the 6D pose estimation task into three subtasks, namely, object segmentation, keypoint detection, and finally, 6D pose estimation. For object segmentation, they trained a Mask R-CNN [5] model with the help of synthetic data and its annotations, and by doing so, they detected the bounding box and obtained object segmentation. For keypoint detection subtasks, they used a Stacked Hourglass Network [14] to detect key points in the segmented objects obtained previously. For the last subtask, 6D pose estimation, they used the key points of the object and the 3D model to find the 6D pose of the object.

The second scenario happens when the objects are reflective, i.e., when their texture is the same as some region of the scene. Hartwig et al. [10] dealt with this type of scenario by using a technique in which a dataset is built with reflective objects from bathroom furniture (sinks, taps, urinals, toilets, etc.) to perform object detection. The authors carried out three experiments that include domain randomization and photorealism. In the first one, the authors rendered the models with a Blinn-Phong [15] reflection model; in the second experiment, they added domain randomization to the previous experiment; and in the third one, they used a physics-based renderer, in order to capture photorealism, along with domain randomization. The last experiment provided the best results for predicting real data.

Similarly, Hinterstoisser et al. [11] generated synthetic data from general-purpose objects (toys and tableware) and also from industrial objects, for the task of object detection and segmentation. Furthermore, Hinterstoisser et al. froze the layers

TABLE I
TECHNIQUES FOR SYNTHETIC DATASET GENERATION AND LABELING

	Dataset Objects	2D Box	3D Box	Mask	Depth Image	Optical Flow
Ruiz et al. [6]	General-purpose Objects and Objects for Self-Driving Cars	X	X	X	X	
Falling Things [7]	General-purpose Object (YCB Objects)	X	X	X	X	
Trembay et al. [8]	Cuboid Objects	X	X	X	X	
Virtual KITTI [3]	Objects for Self-Driving Cars	X	X	X	X	X
Chen et al. [9]	Shiny Metallic Objects without Texture (Metallic Parts)	X		X		
Hartwig et al. [10]	Reflective Objects (Bath Furniture)	X				
Hinterstoisser et al. [11]	General-purpose Objects (Toys and Tableware) and Industrial Objects	X		X		
Ours	Metallic Objects with Shiny Parts (Machinery Objects)	X		X		

of the feature extractor and only modified the remaining ones. Finally, they applied Gaussian Blur to the synthetic images, which helped considerably to improve the results.

Our work is similar to that of Hinterstoisser et al. [11], since we also perform object detection and segmentation with a trained model with synthetic images obtained from a renderer with Phong Illumination [4]. However, our work differs in two aspects. The first one is that we apply our method on metal pieces of mining machinery (with shiny parts). The second is that Hinterstoisser et al. [11] focus both on training a Faster-CNN network and on changing the parameters such as light color, illumination, noise, pose, background and blur, during the creation of the synthetic images. Notwithstanding, in our work, we train a different model, called Mask R-CNN [5] network, varying the pose, the background, and blur parameters. However, we noticed that applying the blur is very significant in our dataset, so we experimented with this parameter extensively by varying its kernel size. We work with objects (pieces and components) of heavy machinery. The 3D models we used correspond to perfect versions of these objects, i.e., objects that have not been used in real life and are not wasted or deteriorated. However, the images used in real tests correspond to used objects, probably wasted deteriorated, and therefore they differ from the synthetic models of images. Hence, applying Gaussian Blur helps to reduce the gap or difference between the synthetic and real images used in our work. This fact can be corroborated by our experiments.

III. PROPOSED METHODOLOGY

Our methodology, explained in detail in the next section, comprises two modules: Generation of synthetic data and labels; and Training of Mask R-CNN network.

- The first module aims at obtaining synthetic data for three types of pieces of mining machinery. For doing so, we implement a renderer based on the Phong Reflection Model [4] using OpenGL. In addition, we apply Gaussian Blur, and vary parameters such as the number of backgrounds. We apply translation and rotation transformations to the objects (3D models) to create the synthetic dataset. Both the translation and the rotation are performed randomly with a uniform distribution, and within an angle range in order to the object not to be rendered outside the viewing area of the virtual camera that generates the synthetic images. Furthermore, it is important to mention that the synthetic images' backgrounds are different from those of the real test images.
- The second module aims at detecting pieces of machinery and subsequently at obtaining the segmentation of these pieces. For this purpose, we use Mask Region-based Convolutional Neural Network (Mask R-CNN) [5], which carries out two procedures. The first one provides proposals about the regions where there might be an object. These regions are framed in bounding boxes. The second procedure performs three tasks: it a) predicts the objects' classes, b) it improves the bounding boxes, and c) it generates masks of the objects.

A. Generation of Synthetic Data and Labels

Synthetic images and mask images were generated by using a renderer (implemented in C++ with OpenGL) that uses the Phong Reflection or Illumination Model [4]. The following subsections discuss the Phong Reflection Model, the mask image generation, the synthetic image generation, and the mask image-to-label conversion process.

1) Phong Reflection Model: This renderer implements the Phong Reflection Model [4], which is a model of local illumination of points on a surface that is grounded on the observation that shiny surfaces have small intense specular highlights, in opposition to, opaque surfaces which have great highlights. Based on that observation, the model combines the diffuse reflection of rough surfaces and the specular reflection of shiny surfaces in order to simulate the form in which lights are reflected on a surface. The model also includes an ambient component to take into account the amount of light that is scattered throughout the whole scene.

The illumination of each surface point I_p is estimated in (1).

$$I_p = k_a i_a + \sum_{l \in \mathcal{L}} (k_d(\hat{\mathbf{L}}_l \cdot \hat{\mathbf{N}}) i_{l,d} + k_s(\hat{\mathbf{R}}_l \cdot \hat{\mathbf{V}})^{\alpha} i_{l,s}), \tag{1}$$

where, k_a is a specular reflection constant; k_d is a diffuse reflection constant; k_s is an ambient reflection constant; α is a shininess constant of the surface material; i_a is the intensity of the ambient (a) light; $i_{l,d}$ is the intensity of the diffuse (d) component of the light source l; $i_{l,s}$ is the intensity of the

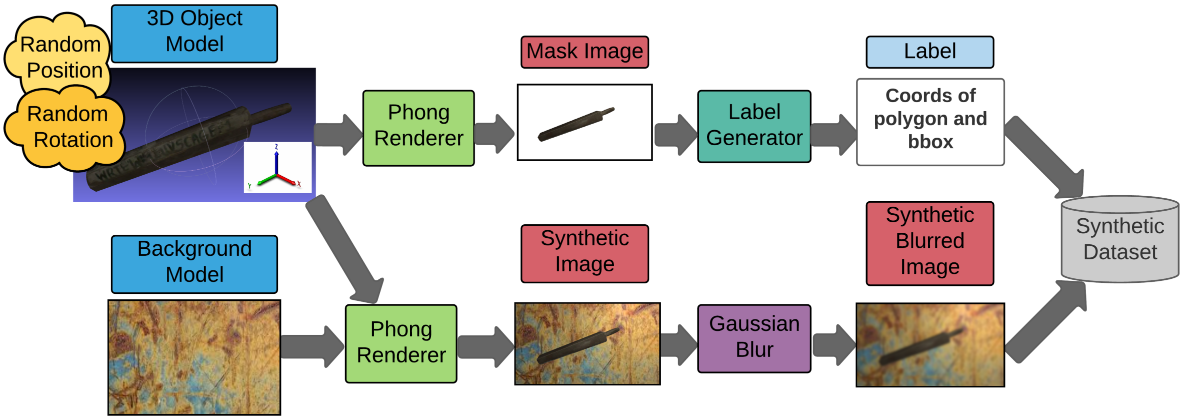

Fig. 1. Architecture of the synthetic dataset generation

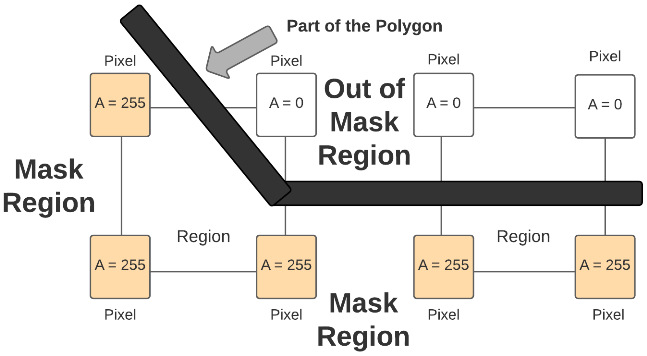

Fig. 2. Marching Squares Method [16].

specular (s) component of the light source l; $\mathcal{L}$ is the set of all light sources; $\hat{\mathbf{L}}_l$ is the direction vector from the point p on the surface toward the light source l; $\hat{\mathbf{N}}$ is the normal at point p on the surface; $\hat{\mathbf{R}}_l$ is the direction vector of a perfectly reflected ray of light from point p on the surface, which is computed via $\hat{\mathbf{R}}_l = 2(\hat{\mathbf{L}}_l \cdot \hat{\mathbf{N}})\hat{\mathbf{N}} - \hat{\mathbf{L}}_l$, $\hat{\mathbf{V}}$ is the direction vector from point p on the surface to the virtual camera.

2) Mask Image Generation: Our renderer generates mask images, which are made up of single objects. For this purpose, the object model is located at a random position and rotation in the scene (as seen in Fig. 1). Also, the renderer converts the background of the mask image into transparent, updating the color buffer for R, G, B, and A (alpha) channels to zero, i.e., it sets R=0, G=0, B=0, A=0.

Also, when the renderer generates a mask image, the following process occurs. The vertex shader processes each vertex of the object's 3D model and returns a set of valid vertices. Then a fragment shader updates the values of the color buffer of the valid vertices by modifying the illumination value I_p via (1) for channels R, G, B, and setting the value of channel A to 255.

In this way, those pixels that are modified by the interaction between the shaders and the model of the object differentiate from the unmodified pixels in the fact that the first ones have their alpha channel equal to 255, and the others have their alpha equal to 0.

3) Label Generation: From Image Mask to Label: The generation of synthetic data includes the creation of the labels, which are used during training of the object segmentation network explained in the next module. In our proposal, the label is a polygon, with x and y coordinates from the image, which covers the object contained both in the mask image and its corresponding synthetic image. To find this polygon, we use the mask image and apply the Marching Squares algorithm, a particular case of the Marching Cubes algorithm [16]. Fig. 2 illustrates the Marching Squares algorithm. The entire mask image is divided into rectangular regions, which are processed one at a time. The changes from 0 to 255 in channel A (alpha) determine what is inside or outside the polygon. The output of every rectangular region contributes with a segment of the polygon. Those segments are put together, obtaining the complete polygon and hence also the label.

4) Synthetic Image Generation: Another task that the renderer carries out is the generation of synthetic images. A synthetic image is made up of an object and a background. We provide the renderer with a set of background images, which, as mentioned before, are different from the real test set backgrounds; subsequently, the renderer randomly selects a background and adds it to the scene immediately after generating the mask image. In the end, we obtain pairs of mask-synthetic images as illustrated in Fig. 3, where the mask image provides the label for the model training, i.e., the output attribute, and the synthetic image works as the training image, i.e., the input attribute.

B. Training of the Mask R-CNN Network

In this module, we used Mask R-CNN [5] in order to deal with the problem of instance segmentation, which is the process of detecting and delineating each distinct object of an image. For doing so, Mask R-CNN performs two procedures.

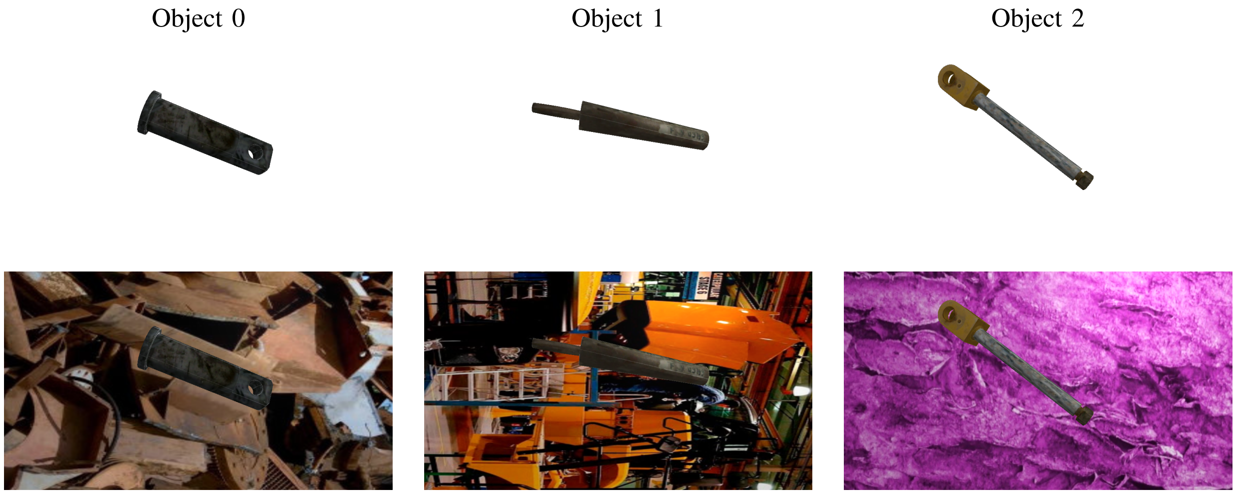

Fig. 3. In the first row there are samples of mask images corresponding to objects 0, 1 and 2, while in the second row there are samples of synthetic images with a random background.

The first one deals with object detection by generating proposals about the regions that have a high probability of containing an object. In this procedure, Mask R-CNN creates several Regions of Interest (RoIs) and incorporates an attention mechanism using a Region Proposal Network (RPN). The input for RPN is a feature map provided by a ResNet [17] or ResNeXt [18] optionally combined with Feature Pyramid Network (FPN) [19]. The output of this procedure is a set of candidate object bounding boxes.

The second procedure copes with semantic segmentation. RoI proposals are combined with the feature map obtaining RoIs of feature map. This result is fed into RoI align, which evolves from RoI pooling (used in Faster R-CNN [20]) and provides higher accuracy by avoiding quantization and hence the loss of information. From applying RoI Align, we obtain warped RoIs, which follows two paths. In the first one, warped RoIs are passed to Fully Connected Layers (FC) to obtain a class vector and a bounding box for each RoIs. In the second one, warped RoIs are fed into a set of sequential convolutions, whose structure is similar to Fully Convolutional Network (FCN) [21]. As a result, we obtain a binary mask in pixel level for each warped RoI. Fig. 4 illustrates the procedure described above. The training process is performed over the synthetic data and the testing over real data, as shown in Fig. 5.

IV. Experiments and Results

Our segmentation problem is restricted to three distinct metal pieces of machinery. Each of these three pieces (objects) has 108 real images; thus, our real dataset contains 324 real images, whose size is 1024 × 576. For the generation of synthetic data and their labels, we collected a set of 93 distinct backgrounds, which are different from the those of the real images, with which we generated 11 random 6D poses for each piece-background pair. Then, 3 pieces × 93 backgrounds × 11 poses results in 3069 synthetic images. For each of them, we generate its corresponding mask image, as well.

For object segmentation, we used the Mask R-CNN network of Detectron2 Model Zoo [22]. This network employs a 50-layer ResNet [17], along with a Feature Pyramid Network [19], as feature extractor or backbone. In Detectron2, the authors pre-trained the network with 37 epochs using COCO dataset [23]. For conducting our experiments, we trained the network with the synthetic images of machinery pieces and their labels over the pre-trained network provided by Detectron2.

In particular, we executed eight experiments: 1) the first one consists in training the network with synthetic images without Gaussian Blur; 2-7) from the second to the seventh experiment, we trained the network with synthetic images using Gaussian Blur with kernel sizes of 51, 31, 23, 15, 7, and 3, respectively; 8) in the last experiment, we only used real data images for training. We used 108 real images out of 324 real images for testing for all the experiments from 1 to 8; likewise, we used the remaining 216 images for training, only in Experiment 8.

Besides, in all experiments (1 - 8), we tested over real data without Gaussian Blur and over real data with Gaussian Blur with a kernel size of 3, since this size gave the best results. The results of the experiments in the detection of objects are observed in Table II (for testing with real data without Gaussian Blur) and III (for testing with real data with Gaussian Blur), while the results of the segmentation of objects are presented in Table IV (for testing with real data without Gaussian Blur) and V (for testing with real data with Gaussian Blur). In Fig. 6 we can see some predictions of detection and segmentation with Mask R-CNN trained over synthetic data with Gaussian Blur of 7, and tested/predicted over real data without blur. These experiments were carried out on an Intel Core i7-8700, 3.2GHz computer. The network model was trained using a NVIDIA GeForce RTX 2060 GPU.

As observed in Table II, which shows the results of object detection tested over real data without Gaussian Blur, the training with real data without blur provides a mAP of 91.7%, while training with synthetic data with a blur of 7 gives the

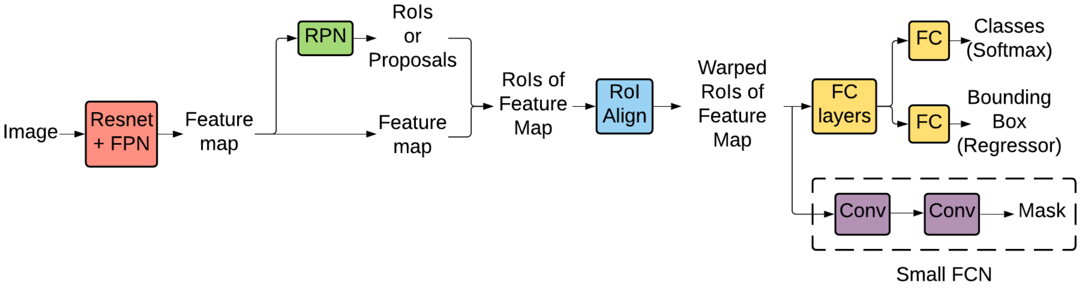

Fig. 4. Architecture of Mask R-CNN.

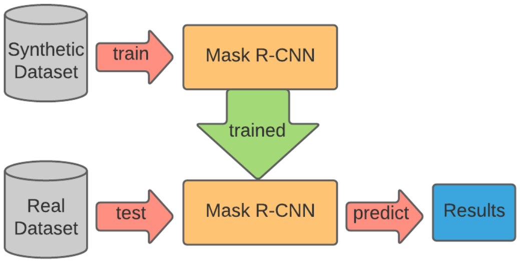

Fig. 5. Module: Training of the Mask R-CNN Network.

best results, i.e., we get a mAP of 85.7%. Between these two mAPs, there is a difference of only 6.0%. Moreover, comparing the mAP of training over synthetic data without blur (28.8%) and the best mAP of training over synthetic data with a kernel size of 7 (85.7%), we gain a mAP of 56.9% by applying Gaussian Blur.

Analyzing Table III, which shows the results of object detection tested over real data with Gaussian Blur, the training with real data without blur provides a mAP of 91.7%, but training over synthetic data using a kernel size of 7 gives the best results, i.e., a mAP of 85.6%. The difference between these two mAPs is 6.1%. Besides, comparing the mAP of training over synthetic data without blur (i.e., 0.0%) and the best mAP of training over synthetic data with a kernel of 7 (85.6%), we gain a mAP of 85.6% when we apply blurring.

Table IV, which presents the results of object segmentation tested over real data without Gaussian Blur, shows that the training over real data without blur provides a mAP of 92.6%, while training over synthetic data with a kernel size of 7 gives the best results, i.e., a mAP of 83.8%. Between these two mAPs, there is a difference of 8.8%. In addition, comparing the mAP of training using synthetic data without blur (26.7%) and the best mAP of training using synthetic data with a kernel size of 7 (83.8%), we gain a mAP of 57.1% by applying Gaussian Blur.

Note in Table V, which shows the results of object segmentation tested over real data with Gaussian Blur, that the training over real data without blur provides a mAP of 92.7%, while training over synthetic data with a kernel size of 7 provides the best synthetic results, i.e., we get a mAP of 84.8%. The difference between both mAPs reaches 7.9%. On the other side, comparing the mAP of training over synthetic data without blur (0.0%) and the best mAP of training over synthetic data with a kernel size of 7 (84.8%), we gain a mAP of 84.8% when we apply blurring.

From Tables II, III, IV, and V, we observe that training with synthetic data gives similar results to training over real data since the difference (in terms of mAP) between the best synthetic training and real training is on average 7.2%. Also, observe that the average gain of the best case of training using synthetic data with blur compared to training using synthetic data without blur is 71.1%. This observation demonstrates that, for our dataset, applying Gaussian Blur to the synthetic training data is significant. Furthermore, we note that our dataset's best synthetic result is achieved training with synthetic data with a kernel of size 7. In the ”Average” column of all the tables, it is shown that the mAP gets the highest values using a kernel size of 7, and when the kernel grows to 15, 23, 31 and 51, the mAP decays. The same behavior is observed when the kernel shrinks to a size of 3; i.e., the mAP decreases. We think this happens not only because the objects inside the images are medium-sized compared to the size of the images that contain them, but also because the images' resolution is the same for all images of the dataset (i.e., 1024 × 576).

Comparing Tables II and III, which show object detection results, we noted that the best mAP in Table II is 85.7%, and the best mAP in Table III is 85.6%. Note that both mAPs correspond to a kernel size of 7. So, testing over real images without Gaussian Blur (Table II) yields better results for object detection over our dataset. Similarly, comparing Tables IV and V, which present segmentation results, we note that the best mAP without blur is 83.8%, while the best mAP with blur is 84.8%. Similar to object detection, both mAPs correspond to a kernel size of 7. Therefore, testing with real images over Gaussian Blur (Table V) gives better results for object segmentation over our dataset.

Finally, the time required for labeling the 324 real images used was 0.5 minutes per image, for a total of 162 minutes (2.7 hours). In contrast, the synthetic generation of 3,069 synthetic images used for training, along with their corresponding mask images, took a total of 20 minutes. We do not consider the time to generate a model since we obtain the 3D CAD models from manufacturers, like carried out in Hinterstoisser et al. [11].

TABLE II
OBJECT DETECTION mAP (PERCENTAGE) TESTED IN THE REAL DATASET WITHOUT GAUSSIAN BLUR

Trained with	Object 0	Object 1	Object 2	Average
Synth. w/o blur	48.7	26.4	11.2	28.8
Synth. w blur 51 x 51	51.4	2.0	76.3	43.2
Synth. w blur 31 x 31	83.4	27.5	**81.0**	64.0
Synth. w blur 23 x 23	74.5	14.6	78.4	55.8
Synth. w blur 15 x 15	89.1	76.4	60.6	75.4
Synth. w blur 7 x 7	**92.0**	**87.6**	77.5	**85.7**
Synth. w blur 3 x 3	86.4	82.9	48.3	72.5
Real w/o blur	**92.0**	**95.1**	**88.1**	**91.7**

TABLE III
OBJECT DETECTION mAP (PERCENTAGE) TESTED IN THE REAL DATASET WITH GAUSSIAN BLUR

Trained with	Object 0	Object 1	Object 2	Average
Synth. w/o blur	0.0	0.0	0.0	0.0
Synth. w blur 51 x 51	56.6	2.7	**81.0**	46.8
Synth. w blur 31 x 31	84.1	30.2	80.4	64.9
Synth. w blur 23 x 23	77.9	19.7	73.9	57.2
Synth. w blur 15 x 15	87.2	76.7	45.1	69.6
Synth. w blur 7 x 7	**92.3**	**88.8**	75.7	**85.6**
Synth. w blur 3 x 3	76.7	75.4	38.1	63.4
Real w/o blur	**91.6**	**94.9**	**88.5**	**91.7**

The blur of the test real data for all experiments has a kernel size of 3.

TABLE IV
OBJECT SEGMENTATION mAP (PERCENTAGE) TESTED IN THE REAL DATASET WITHOUT GAUSSIAN BLUR

Trained with	Object 0	Object 1	Object 2	Average
Synth. w/o blur	48.9	26.7	4.6	26.7
Synth. w blur 51 x 51	54.6	1.9	57.8	38.1
Synth. w blur 31 x 31	93.2	30.1	**66.8**	63.3
Synth. w blur 23 x 23	80.3	15.2	65.0	53.5
Synth. w blur 15 x 15	98.3	79.2	49.7	75.7
Synth. w blur 7 x 7	**98.7**	**89.1**	63.5	**83.8**
Synth. w blur 3 x 3	93.3	87.2	39.0	73.2
Real w/o blur	**98.1**	**96.7**	**83.1**	**92.6**

TABLE V
OBJECT SEGMENTATION mAP (PERCENTAGE) TESTED IN THE REAL DATASET WITH GAUSSIAN BLUR

Trained with	Object 0	Object 1	Object 2	Average
Synth. w/o blur	0.0	0.0	0.0	0.0
Synth. w blur 51 x 51	60.3	2.6	59.9	41.0
Synth. w blur 31 x 31	93.7	32.8	**68.1**	64.9
Synth. w blur 23 x 23	82.3	21.1	63.6	55.7
Synth. w blur 15 x 15	95.7	78.9	37.0	70.5
Synth. w blur 7 x 7	**98.7**	**92.1**	63.6	**84.8**
Synth. w blur 3 x 3	81.7	78.9	29.5	63.4
Real w/o blur	**98.3**	**97.4**	**82.5**	**92.7**

The blur of the test real data for all experiments has a kernel size of 3.

V. CONCLUSIONS

We propose a methodology to generate synthetic data, which is used for the training of Mask R-CNN model specialized in object detection and segmentation tasks. In our work, the objects used are pieces of mining machinery. The experiments confirmed our hypothesis that training with synthetic data allows us to obtain similar results to training with few real data. Furthermore, we show that for the images employed in this research, the generation of synthetic data and its labels requires less time than obtaining and manually labeling real images. Finally, for our dataset, the best results were obtained by training over synthetic data and applying a kernel size of 7 for blurring the images. More precisely, we got a mAP of 85.7% in object detection and a mAP of 84.8% for object segmentation. Those results lead us to affirm our method for generating synthetic data reduces the reality gap both in object detection and in the segmentation task when applied in our dataset.

ACKNOWLEDGMENT

M. E. Loaiza acknowledges the financial support of the ”Proyecto Concytec - Banco Mundial”, through its executing unit ”Fondo Nacional de Desarrollo Científico, Tecnológico y de Innovación Tecnológica (Fondecyt)”, for his research work entitled ”Reconstrucción y modelado 3D de las superficies de componentes y piezas de maquinaria pesada usada en Minería, con nivel de precisión milimétrica, para su aplicación en un nuevo proceso optimizado de manutención especializada”.

REFERENCES

[1] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, “Domain randomization for transferring deep neural networks from simulation to the real world,” in *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2017, pp. 23–30.

[2] J. Tremblay, T. To, B. Sundaralingam, Y. Xiang, D. Fox, and S. Birchfield, “Deep object pose estimation for semantic robotic grasping of household objects,” *arXiv preprint arXiv:1809.10790*, 2018.

[3] A. Gaidon, Q. Wang, Y. Cabon, and E. Vig, “Virtual worlds as proxy for multi-object tracking analysis,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 4340–4349.

[4] B. T. Phong, “Illumination for computer generated pictures,” *Communications of the ACM*, vol. 18, no. 6, pp. 311–317, 1975.

[5] K. He, G. Gkioxari, P. Dollár, and R. Girshick, “Mask r-cnn,” in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.

[6] M. Ruiz, J. Fontinele, R. Perrone, M. Santos, and L. Oliveira, “A tool for building multi-purpose and multi-pose synthetic data sets,” in *ECCOMAS Thematic Conference on Computational Vision and Medical Image Processing*. Springer, 2019, pp. 401–410.

[7] J. Tremblay, T. To, and S. Birchfield, “Falling things: A synthetic dataset for 3d object detection and pose estimation,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 2038–2041.

[8] J. Tremblay, T. To, A. Molchanov, S. Tyree, J. Kautz, and S. Birchfield, “Synthetically trained neural networks for learning human-readable plans from real-world demonstrations,” in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1–5.

[9] C. Chen, X. Jiang, W. Zhou, and Y.-H. Liu, “Pose estimation for texture-less shiny objects in a single rgb image using synthetic training data,” *arXiv preprint arXiv:1909.10270*, 2019.

[10] S. Hartwig and T. Ropinski, “Training object detectors on synthetic images containing reflecting materials,” *arXiv preprint arXiv:1904.00824*, 2019.

[11] S. Hinterstoisser, V. Lepetit, P. Wohlhart, and K. Konolige, “On pre-trained image features and synthetic images for deep learning,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 0–0.

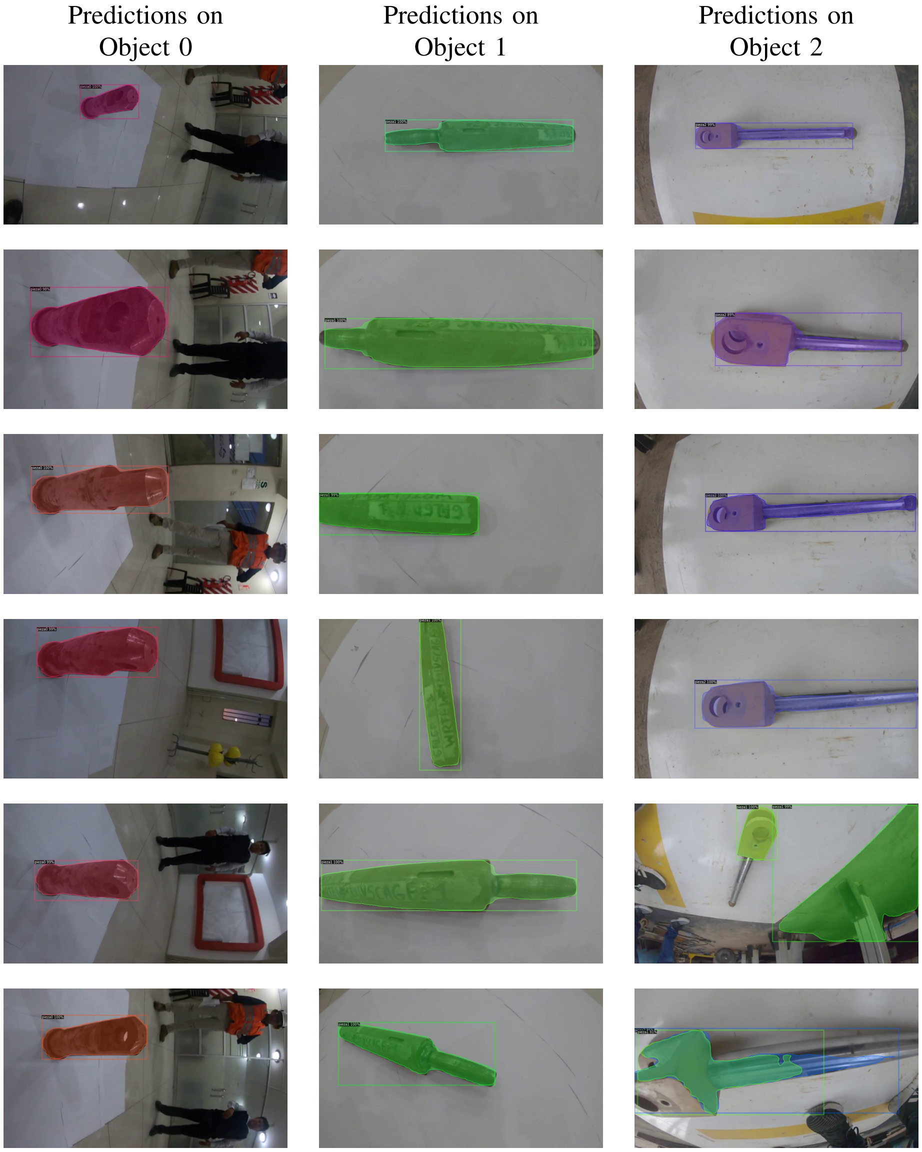

Fig. 6. Predictions on real data without Gaussian Blur of a model trained with synthetic data with Gaussian Blur (7 x 7). The last 2 rows of predictions on object 2 show some cases where the detection and segmentation fail.

[12] T. To, J. Tremblay, D. McKay, Y. Yamaguchi, K. Leung, A. Balanon, J. Cheng, W. Hodge, and S. Birchfield, "NDDS: NVIDIA deep learning dataset synthesizer," 2018, https://github.com/NVIDIA/Dataset_Synthesizer.

[13] B. Calli, A. Walsman, A. Singh, S. Srinivasa, P. Abbeel, and A. M. Dollar, "Benchmarking in manipulation research: The ycb object and model set and benchmarking protocols," *arXiv preprint arXiv:1502.03143*, 2015.

[14] A. Newell, K. Yang, and J. Deng, "Stacked hourglass networks for human pose estimation," in *European conference on computer vision*. Springer, 2016, pp. 483–499.

[15] J. F. Blinn, "Models of light reflection for computer synthesized pictures," in *Proceedings of the 4th annual conference on Computer graphics and interactive techniques*, 1977, pp. 192–198.

[16] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3d surface construction algorithm," *ACM siggraph computer graphics*, vol. 21, no. 4, pp. 163–169, 1987.

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[18] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1492–1500.

[19] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2117–2125.

[20] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[21] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[22] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, "Detectron2," https://github.com/facebookresearch/detectron2, 2019.

[23] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*. Springer, 2014, pp. 740–755.