Received November 4, 2020, accepted November 27, 2020, date of publication December 8, 2020, date of current version December 18, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3043256

Fast Synthetic Dataset for Kitchen Object Segmentation in Deep Learning

RUBEN SAGUES-TANCO[1,2], (Graduate Student Member, IEEE), LUIS BENAGES-PARDO[1,2], GONZALO LÓPEZ-NICOLÁS[1,2], (Senior Member, IEEE), AND SERGIO LLORENTE[3]

[1]Departamento de Informática e Ingeniería de Sistemas, Universidad de Zaragoza, 50018 Zaragoza, Spain
[2]Instituto de Investigación en Ingeniería de Aragón, Universidad de Zaragoza, 50018 Zaragoza, Spain
[3]Research and Development Department, Induction Technology, Product Division Cookers, BSH Home Appliances Group, 50016 Zaragoza, Spain

Corresponding author: Ruben Sagues-Tanco (rubensagues@unizar.es)

This work was supported by the Ministerio de Ciencia, Innovación y Universidades, Gobierno de España-European Union under Project RTC-2017-5965-6.

ABSTRACT Object recognition has been widely investigated in computer vision for many years. Currently, this process is carried out through neural networks, but there are very few public datasets available with mask and class labels of the objects for the training process in usual applications. In this paper, we address the problem of fast generation of synthetic datasets to train neural models because creating a handcraft labeled dataset with object segmentation is a very tedious and time-consuming task. We propose an efficient method to generate a synthetic labeled dataset that adequately combines background images with foreground segmented objects. The synthetic images can be created automatically with random positioning of the objects or, alternatively, the method can produce realistic images by keeping the realism in the scales and positions of the objects. Then, we employ Mask-RCNN deep learning model, to detect and segment classes of kitchen objects using images. In the experimental evaluation, we study both synthetic datasets, automatic or realistic, and we compare the results. We analyze the performance with the most widely used indexes and check that the realistic synthetic dataset, quickly created through our method, can provide competitive results and accurately classify the different objects.

INDEX TERMS Kitchen object detection, computer vision, machine vision, deep learning, object segmentation, image databases.

I. INTRODUCTION

Object detection and recognition through neural networks and deep learning is a hot topic in computer vision. This is a useful ability required in automation, for robotics or smart applications. The objects to be recognized can be quite different, from pets such as dogs or cats, to boxes of cereals or cookies, between others. To achieve this objective and robustly recognize objects of different classes, it is necessary the system to be trained with a large amount of images with objects of each class in different situations.

Object recognition can be addressed by using different techniques or methods. The use of convolutional neural networks (CNN) is currently broadly assumed. In [1], [2] a study and analysis of CNN is carried out, discussing their performance in this type of tasks. Also in [3], the authors provide view of how deep learning is currently an extremely active research area. Other researchers focus on a thorough evaluation of networks, of increasing depth, using an architecture with very small (3×3) convolutional filters. With these approaches, current methods achieve a significant improvement on the configuration of the net and this is in part gained by pushing the depth to 16-19 weight layers [4]. Some researchers focus on deep learning and its development for the reconstruction and analysis of images. They make image combinations so that the neural network has a better learning [5]. Alvaro Collet *et al.* in [6] present MOPED, a framework for Multiple Object Pose Estimation and Detection. They use a very robust algorithm to detect objects in different positions within a complex scene. Other authors focus on the recognition of text within images. They use convolutional networks to locate and analyze the position of

The associate editor coordinating the review of this manuscript and approving it for publication was Paolo Remagnino.

the text [7]. Another method used for the detection of objects is shown in [8], where the authors describe an object detection system based on mixtures of multi-scale deformable part models. Their proposal relies on new methods for discriminative training with partially labeled data. Experimentally, some authors focus on the implementation of object recognition for robotics. To facilitate this task, the robot is able to obtain a 3D map of the environment and locate the new objects [9]. Kaiming He *et al.* in [10] present Mask R-CNN, successor of the Faster R-CNN [11]. This network is widely used to recognize and track objects. They present the use of objects with masks, improving the use of bounding boxes.

Deep CNNs belong to the best existing methods for recognizing objects, but they need a large amount of labeled data to learn. There are some public datasets, such as COCO [12] and PASCAL VOC2012 [13], in which a great variety of object classes are labeled. These datasets are hand tagged and in particular by marking one by one the object mask and the corresponding label. One of the drawbacks of using general datasets is the low precision that is achieved if the goal is the detection of objects in specific environments different than those of the training dataset. Other datasets such as [14] focus on a specific environment, but each object is only labeled with its bounding box. In addition, to achieve a robust model, images with the objects to be recognized and with different backgrounds are also very convenient. Finding scenarios in public datasets with these specific features is quite complicated. The creation of handcraft useful datasets requires a large amount of time and effort of many people, which is a very tedious task. Georgakis *et al.* in [15] propose an alternative to manual labeling. They present a method, working with the Faster R-CNN network, in which the objects are pasted on background images. The objects are selected from the BigBird dataset [16], and placed on the support surfaces obtained from the background. Due to the importance of data during neural network training, some researchers have developed some new techniques for fast data augmentation. These techniques are known as smart augmentation and they work by creating a network that learns how to generate augmented data during the training process of a target network [17]. Dwibedi *et al.* [18] create a synthetic dataset in a similar way, pasting the objects on the background in a random way, although their dataset does not focus on a specific environment. In addition, objects are placed on backgrounds of any kind without taking into account a specific location. This makes object recognition worse due to the variety of scenarios. For example, a specific dataset for the automatization of the grape selection process is created in [19].

Other researchers, use 3D models to create the dataset to train neural networks. Su *et al.* [20] present a dataset through 3D models which may give many views of the objects to be labeled. Others extract information from 3D models which can be used to reduce noise and convert 2D images into 3D models [21]. 3D object recognition from arbitrary viewpoints is one of the most challenging problems in computer vision. Some authors use 3D objects models as the only source of information for building a multi-view object class detector [22]. They get as much information as possible from the parts of the 3D model, and this helps to recognize objects later by grouping matches. However, the creation and representation of 3D models requires very powerful software. The training of neural networks through dataset obtained with 3D models can have disadvantages due to the final quality of the model. In deep CNN trained with images obtained through 3D models, photorealistic aspects such as the texture, position or background are not required [23], [24]. Movshovitz-Attias *et al.* [25] focus on semi-automatic dataset creation through the use of synthetic data. They generate a large labeled car dataset densely rendered in viewpoint space and they investigate the effect of rendering parameters on estimation performance to show realism. Some authors propose to model complex visual scenes using a non-parametric Bayesian model to learn from weakly labeled images [26]. The model learns the appearance of objects and attribute classes thanks to the weak image annotations of objects and attributes. Then, the model allows to describe and to recognize objects in the scene using their attributes. In [27] the authors propose FlickNet, which is able to explore different combinations of locations on feature maps and randomly select hidden units to classify images. FlickNet learns of each location in the feature maps and generates a localization map which identifies the object. The authors of [28] propose a Constrained CNN, which uses a different loss function and can be easily implemented into the gradient descent optimization.

Another approach for training neural networks utilize directly a virtual environment. In [29], [30] a virtual pedestrian is created and the network robustness is checked in the real world, obtaining satisfactory results. There are also methods capable of cloning real to virtual worlds to create a fully tagged dataset. Within this virtual dataset, objects can be created and tracked [31].

Many applications of deep learning techniques have appeared in the last decade. Some of them focus on general applications and others on specific ones. They need a pre-training of the neural network. Here, we focus on the generation of a dataset for object recognition in general scenes and particularly in kitchens. It is a common place in every home where countless actions are carried out and where the development and improvement of applications can help the users. Our method to generate the dataset allows the labeling of the masks of the objects with good results. It produces precise segmentations, unlike others which only include the label with bounding box [14].

Our main goal is to achieve a method to quickly create reliable synthetic datasets. They are implemented on the Mask-RCNN neural network model to perform instance segmentation for any application and particularly for kitchen objects. The synthetic datasets are based on the segmentation and labeling of real objects and their overlaying in real kitchen backgrounds. We address this task with two different methods: the first one creates the realistic dataset manually mixing images of foreground objects with background images, and the second one automatically places the objects on the backgrounds. We also use different processing techniques (geometry operations, filtering) over both datasets. This allows to obtain different scenarios by changing conditions such as the luminosity or movement distortion. So the detection of objects on videos with different environments is more robust. Both synthetic datasets are tested and compared to each other and validated with a handcraft labeled dataset. Most of the datasets currently available are labeled manually, thus achieving total reliability in training, but this requires a lot of time. In summary, the interest of the paper is to bring to light a procedure to create synthetic datasets with images of typical general scenes, and particularly kitchens with objects involved in cooking processes. The main contribution of the proposal is a method to generate extensive and versatile datasets with a precise object segmentation, not only with the bounding box but also the object mask. It will allow to reduce the time to create useful specific datasets, and it can be a first step in the automation of learning in application focused environments.

II. DEEP LEARNING BACKGROUND

Convolutional neural networks can be used for different tasks such as data analysis, object recognition or object tracking. We are involved in the recognition of objects or actions related with cooking in the kitchen environments. In the first stage, it is necessary to perform a training of these networks using datasets with tagged images. Our objective is to analyze the results when synthetic datasets are used. These are validated against datasets generated with hand tagged images which are considered as the ground-truth. Our work is based on the application of the deep learning model called Mask R-CNN [10] for the detection and segmentation of the objects present in the kitchen. The goal is to locate the presence of objects in a specific environment with their corresponding bounding box and mask, as well as the ability to detect the classes of the located objects in the image.

We have analyzed two different families of neural networks to do object detection, the R-CNN Model Family and the Yolo Model Family [32]. Comparing the network of the YOLO family to the R-CNN family, performance differences can be noted. The YOLO network operates at 45 frames per second and up to 155 frames per second for a speed-optimized version of the model. In addition, the best versions of this network also detect the object bounding box as does the Faster R-CNN network. Because our work is motivated by applications that need action detection reliably, we are interested in real-time detection of the object with its specific shape, i.e. with its mask. Mask-RCNN is able to produce the object's mask using an additional branch to Faster R-CNN that produces a binary mask representing whether or not a given pixel is part of an object. The branch is just a Fully Convolutional Network on top of a CNN based feature map. This implementation comes up of the framework made by Matterport Inc., [33] in a previous work, which is provided as OpenSource under the MIT license. The Mask-RCNN allows to carry out all the tasks and it is fast enough to recognize the class and the mask of the objects.

The architecture and operation of Mask-RCNN is detailed in [10], [34]. The first block of this architecture is made up of three sub-blocks: Backbone that extracts feature maps of images with convolutional layers (ResNet50 or ResNet101), Feature Pyramid Network (FPN) [35], which controls that the extracted high-level features are at different scales and with information of different levels, and the Region Proposal Network (RPN) introduced in [11]. It gives proposed regions in which it is possible to find an object of one of the trained classes.

Once the proposed regions have been obtained, ROI-Align introduced in [10] is applied to fix the size of the proposed regions of the classifier input. This is done by a bilinear interpolation method. Each proposed region is entered into different headers: a classifier to predict the class to which they belong, another of regression to predict the bounding box of the object, and another formed by a Fully Convolutional Network (FCN) [36] to predict the mask of the segmented object. The cost function used (1) is the sum of the errors from the class (L_{cls}), the bounding box (L_{bbox}) and the mask (L_{mask}) [37].

$$L = L_{cls} + L_{bbox} + L_{mask} \qquad (1)$$

III. METHODOLOGY

We have fully analyzed the state of the art to find existing datasets to train neural networks. The dataset necessary to perform a specific task requires the labeled object and its referenced mask. In this way it is possible to locate the object on the image. There are not many public useful datasets with this kind of characteristics.

The COCO dataset [12] is one of the most popular in object segmentation. This has more than 200k labeled images and 80 different object classes. However, few of these classes are usually useful for specific applications, particularly for the environments considered here (kitchens). To create a specific dataset with the images labeled in the desired format, different methods have been used. Firstly we start by hand-labeling real images using the VGG Image Annotator tool (VIA) [38] developed by the Visual Geometry Group of the University of Oxford. The creation of the handcraft labeled dataset consumes a lot of time and motivates the development of an application to create synthetic datasets quickly. We use the handcraft dataset to validate the synthetic datasets and to show the robustness of our proposal. We can develop two different datasets with a tradeoff between realism and speed generation. Later, we compare the results with both datasets checking if the network classifies the objects correctly.

A. HANDCRAFT LABELED DATASET

To create the manual labeled dataset, images of real kitchens with typical objects used in the cooking process are acquired.

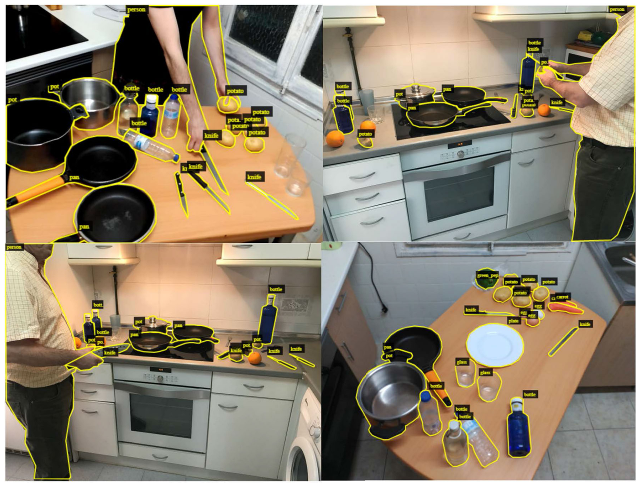


FIGURE 1. **Examples of images labeled with the VIA tool. A similar process is used for all the images of the handcraft labeled dataset.**

Given the images, the objects belonging to each of the desired classes are labeled. Each object is segmented with its mask and the corresponding label of its class. Fig. 1 shows examples of the process.

The process of image labeling is very hard and time-consuming, even when only a class label per image is required. Each object is labeled and segmented with a mask to define its location. The labeling of images similar to Fig. 1 takes about 15-20 minutes. We tagged 81 images with 10-12 objects per image. Once the labeling of the images is done, the training begins applying transfer learning in the Mask-RCNN model from the weights trained with the COCO dataset and is validated by 20 real images that have been tagged in the same way. In this way and despite the small number of training images, the results are quite good. However, one of the biggest problem in this dataset is that the real images used only can have the selected classes to train the network. In other case the network may mistake the not-labeled classes with the background and those classes can not be identified. Additionally, to improve the results or to include an additional class to the dataset, it is necessary to invest a lot of time in the image tagging.

B. SYNTHETIC DATASET

Once evidenced that the manual labeling of real images is very time consuming and unfeasible in real situations, an application to create a synthetic dataset is proposed. We developed this application to create synthetic scenes in Python. The application attaches segmented objects, such as knifes, pots, potatoes, etc. over backgrounds that can be countertops, glass-ceramic hobs or even whole kitchens. The addition of objects can be fully automatic (generating unrealistic images), or realistic (attaching the objects realistically). In the first case, the dataset is generated automatically. It is only necessary to mark the classes and number of objects. The application also scales and orients them. In the realistic

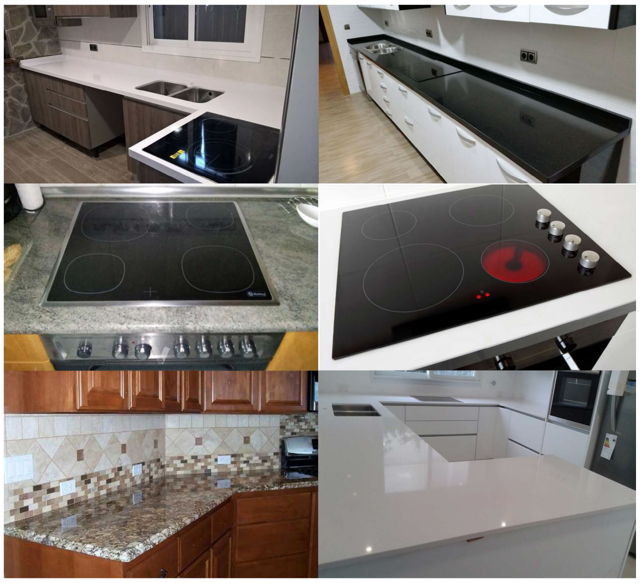


FIGURE 2. **Example of backgrounds used for the development of the synthetic dataset. With these backgrounds we generate the automatic synthetic dataset and the realistic synthetic dataset.**

synthetic dataset this is controlled manually, while the objects are placed on the backgrounds. But in both datasets the conditions are similar. Also, in both cases the class labels and the masks of the objects are directly stored. During the insertion of the objects, the necessary parameters, i.e. the scale of the object and its position, are saved to generate the masks. These parameters and the class labels are stored in a JSON (JavaScript Object Notation) format file. Therefore the image annotations occupy very little space. Fig. 2 shows some examples of backgrounds used for the creation of synthetic images. All background images and the foreground objects, have been obtained from Google images (www.google.com).

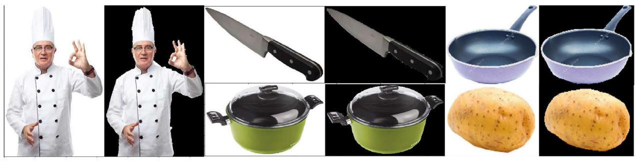


FIGURE 3. **Example of objects and their masks used in the developed application to generate synthetic datasets.**

In Fig. 3 we show some objects extracted from the original images with white background (left) and the masks of objects placed on the black background (right). These images of foreground objects are searched in public image datasets. The object mask is automatically separated from the background by thresholding. All the object classes selected usually appear in the kitchen. The classes of objects are *knife, person, potato, bottle, pan* and *pot*. Objects like *knife, person* or *bottle* have quite clear geometric differences, but other objects such as *pan* and *pot* are really similar in geometry, color and size. Also, we take images where the objects are in a great variety of positions. This helps the network because a better learning

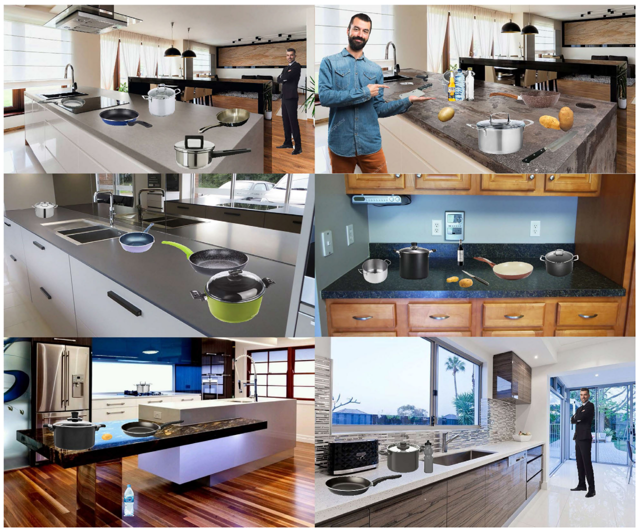


FIGURE 4. Example of synthetic realistic images created through the application. Note that the objects are placed in appropriate locations and naturally introduced in the scene. The parameters and labels of the class of each object seen on the image are stored in a JSON file.

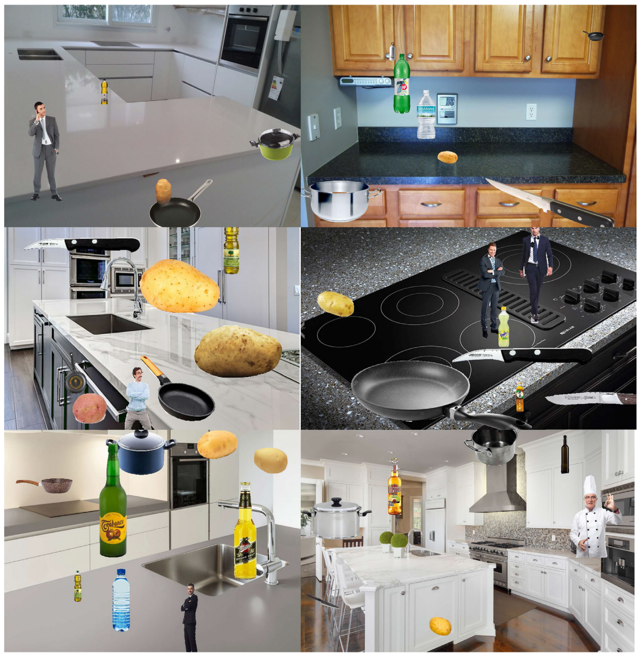


FIGURE 5. Example of synthetic automatic training images created through the application. Note that the induced objects appear in unnatural places with inappropriate scale. The parameters and class labels of of each object seen in the image are stored in a JSON file.

can be achieved. This increases the robustness of the neural network because in real cases the position of the objects may change a lot. Background images and all the objects positioned on them are common to both datasets. With all these objects and background images, the application begins to create useful synthetic images to train neural networks that can work with the mask of the objects. The quality of the images is given by the scale and position of the objects with respect to the background. This positioning and scaling of the objects is what makes one dataset take longer to generate than the other. Some of the images created with the described application in a realistic way are shown in Fig. 4. In these images, the scales and orientations of the objects on the background are maintained, in order to obtain more realistic results. In Fig. 5 some images are shown with the objects automatically fitted and giving results which do not look at all as in real images. As it can be seen in Fig. 4 and Fig. 5 the quality of the images in the realistic dataset is much better than in the automatic dataset.

Using the described application, the time to generate realistic images takes approximately 10-15 seconds per image and is easy to do. The automatic dataset, which combines the background images with the objects randomly placed and scaled is much faster. The application works automatically and the user only has to select the number of images, the number objects per image and the allowed percentage of occlusion between objects. The process to generate the automatic dataset is approximately 1 minute for 350 labeled images (0.15 seconds/image). Both synthetic datasets (realistic or automatic) are a very good improvement over handcraft labeled dataset, that takes up to 1200 seconds per image, because they reduce their generation time considerably.

The synthetic images are of all kinds to achieve richer and more varied datasets. We show some images of occlusions

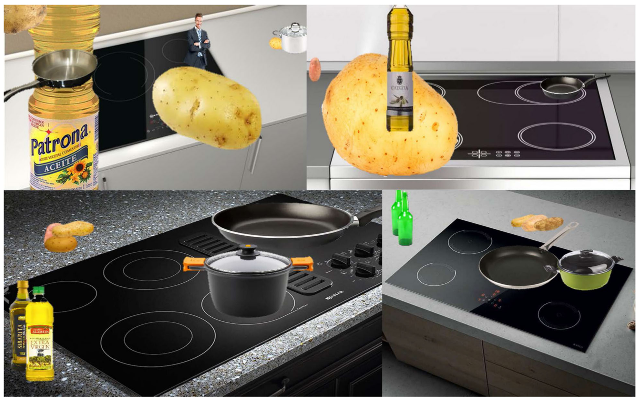


FIGURE 6. Example of some of the occlusions found in the automatic synthetic dataset (top) and realistic synthetic dataset (bottom). Top: Since the objects are randomly distributed, these occlusions are also random. The maximum allowed level of occlusion can be modified with a parameter. Bottom: These occlusions appear similar to those we can find in reality.

in the automatic synthetic dataset (Fig. 6 - top), and some images of the realistic synthetic dataset (Fig. 6 - bottom). We validate these datasets with the handcraft labeled dataset and we test the model with real images so we can verify how it works in real situations. These images have objects used to train the network and others that have not been used in the training. This allows to check the robustness against unknown objects that look similar. Table 1 contains the information of the synthetic training datasets. It shows the number of images and object instances for each object class. Information about

TABLE 1. Information of the objects included in the Synthetic dataset.

Class	Automatic (319 images)		Realistic (319 images)		Validation (81 images)	
	Images	Object instances	Images	Object instances	Images	Object instances
Bottle	245	444	135	344	81	291
Pan	233	390	212	346	78	143
Pot	192	288	219	369	78	131
Knife	232	366	136	271	81	277
Potato	229	383	129	412	81	427
Person	183	247	104	115	58	59

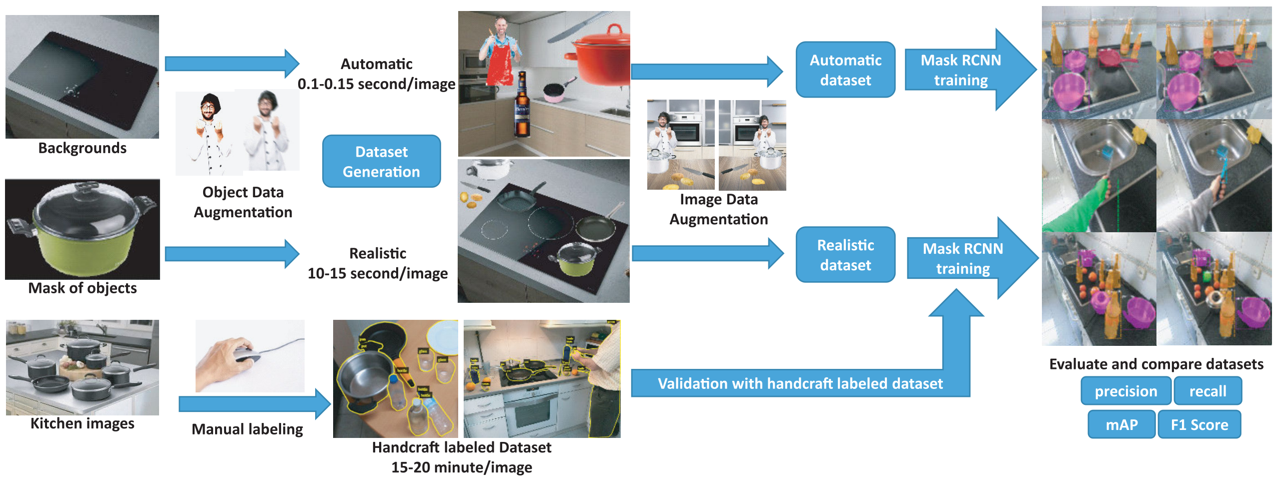


FIGURE 7. **Flow diagram of the proposed process to develop the realistic synthetic dataset and the automatic synthetic dataset. We show the process to create the datasets. The validation and the comparison with the handcraft labeled dataset allows to test the performance.**

the images and objects instances used for validation are also shown.

IV. EXPERIMENTAL VALIDATION

Once the handcraft labeled dataset and the synthetic datasets have been created, we tested their performance. We start training our neural network with the datasets independently. The three datasets to train the network are: (i) the handcraft labeled dataset (81 images), which is created from the images manually labeled, and by marking each object one by one with its mask in the image, (ii) the automatic synthetic dataset (319 images) which is created with foreground objects that are placed randomly on the background image, (iii) the realistic synthetic dataset (319 images) which is made by positioning the objects in the background image in a realistic way. Once the network is trained independently with each of them, the results are examined with respect to the same images and the results are checked. The generic configuration of the network with all the available parameters have been analyzed and the configuration to obtain better results has been proposed. In Fig. 7, we present a flow diagram with the process to generate all datasets. We depict the initial images of the objects and the backgrounds, and how we create the datasets until the final comparison.

A. EXPERIMENTAL CONFIGURATION

Our model uses the ResNet101 network as Backbone to carry out the feature extraction. As starting point, we have used transfer learning with the pre-trained weights on the COCO dataset [12]. The most relevant hyperparameters chosen for the training were: learning rate 0.001, weight decay 0.9 and a batch size of two images on an NVIDIA Titan XP GPU. The network has been trained in three different stages: the first, training only the headers during 60 epoch, the second, training from stage 4 and up in the ResNet101 model for 30 epoch, and in the third, training the complete network with a learning rate reduced by one factor of 10, for fine-tune of weights during 15 epoch. For additional details on the Mask-RCNN network, we refer the reader to [10], where the authors explain how the different layers work, all the parameters that make up its configuration and the process followed to locate the object's masks.

In the training, data augmentation has been used to avoid the overfitting of the network and to give variability to the training examples. Data augmentation was done using Python library *imgaug* [39]. With this library the number of images to train the network can be easily increased. This augmentation converts the set of images in a larger set adding images slightly altered. This modification of the images improves the recognition of objects since it distorts and varies the luminosity and position of the objects. We have also performed some global operations on the image and some additional individual modifications on each object. This makes the synthetic dataset to have even more variability during the training.

The network has been trained using the following modes of data augmentation to the objects: Gaussian filter, Blur

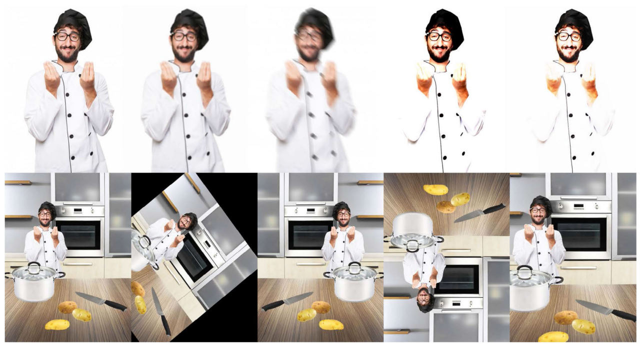


FIGURE 8. Example of modes of data augmentation applied to the objects in the synthetic dataset (top), and example of modes of data augmentation applied in both datasets (bottom).

filter, Linear Contrast filter and Sigmoid Contrast filter (Fig. 8 - top) and the following modes of data augmentation to the complete images: Affine, Flip left or right, Flip up or down and Cropped (Fig. 8 - bottom). We only make object augmentation that modify its color or sharpness. On the other hand, another type of augmentations is made on the image, such as the rotation or the flip of the complete image. All the used augmentations techniques create a more robust model in any situation. For example, *pans* and *pots* are placed face down after washing to allow the water to dry or, perhaps, the position of the camera is not always the same.

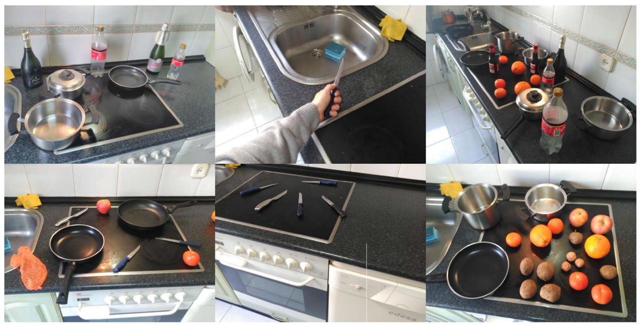


FIGURE 9. Example of images used for testing the training of the neural network with both datasets.

Once the network has been trained and using data augmentation on the images, we obtain the trained weights of both synthetic datasets. These weights are tested on a set of images obtained from real kitchens. These are new images that have not been used in the training or in the validation of the network. The test images have the same classes of objects, *pan, pot, bottle, knife, person and potato*. Also, other objects *orange, fork, nut...* are introduced on these images, to check false positives. In Fig. 9 some examples of the images used to perform the test weights during the training of the network can be seen. This allows to visually compare both datasets and check the efficiency as shown in the following section.

B. ANALYSIS OF THE EXPERIMENTAL RESULTS

A comparison between the results obtained with the model trained with realistic images and the model trained with automatically generated images is presented. Both datasets to train the Mask RCNN have the same number of images and have been validated in the same conditions. Furthermore, the variation in the total number of objects appearing in the datasets does not exceed 13%. We can observe in Table 1, that the number of object instances is greater for some classes in the automatic dataset and for other in the realistic dataset, which makes the comparison fair enough. This comparison has been performed by calculating the mean Average Precision metric (*mAP*) defined in PASCAL VOC 2012 competition [13] on the validation set. This metric uses a threshold value of Intersection Over Union (IoU) of 0.5 (50%). As it can be seen in Table 3, the Average Precision (*AP*) of each class has been calculated and the average of all of them is shown as the *mAP*. Also, the comparison is made using the same Mask RCNN configuration during the training of the two datasets (automatic and realistic dataset).

The handcraft labeled dataset (81 images) is created as a ground-truth for the validation of the synthetic datasets and it is done using the same labeled class objects. The results obtained with the realistic dataset are better than the results obtained with the automatic one. This difference can be clearly seen in the extreme case of the *potato* class. The automatic dataset has a value of 0.07 AP while the realistic dataset achieves a value of 0.91 AP (Table 3). This shows that the Mask R-CNN network learns from the realism of the images, i.e. the position and scale of the objects in the environment. By analyzing the realistic dataset we can see how some classes are more difficult to be detected than others, for example *knife* which may have different shapes, or *pot* and *pan* which may be very similar. In the Table 2 the false positives and the true positives obtained with both models can be observed for all the analyzed classes. The true positives are those in which the neural network hits the class of the object it predicts. Instead, false positives are those where the neural network is confused in the class of the object. There are two different false positives, the first type is when the neural network does not succeed in the class of an object that has been trained and, for example, it confuses *pan* with *pot*. The second type of false positives appears because of a confusion of a new object (*orange*) with a learned object (*potato*).

Other well-known indexes used to evaluate how well network classifies the results are *precision* and *recall*. These two indexes can be calculated through four prediction results: true positives (*tp*), false positives (*fp*), true negatives (*tn*) and false negatives (*fn*). *Precision* indicates how many of the classified objects in a class are correct. This index is computed as follows:

$$\text{precision} = \frac{|\{\text{relevant objects}\} \cap \{\text{retrieved objects}\}|}{|\{\text{retrieved objects}\}|} \quad (2)$$

$$precision = \frac{tp}{tp + fp} \quad (3)$$

The index denoted with *recall* refers to the objects that have been selected over the true set of the objects that should have been selected. The *recall* is given by the

TABLE 2. Information of the false positives, true negatives, *precision*, *recall* and *F1-score* of each object and of each dataset.

Class	Automatic dataset					
	Num. Objects	False Positives (fp)	True Positives (tp)	precision	recall	F1-score
Bottle	444	12	233	0.95102	0.4192	0.58192
Pan	390	31	112	0.78322	0.4457	0.56814
Pot	288	14	99	0.87611	0.4349	0.58131
Knife	366	7	131	0.94927	0.2484	0.39377
Potato	383	38	46	0.54762	0.0548	0.09956
Person	247	10	54	0.84375	0.5416	0.65969
Class	**Realistic Dataset**					
	Num. Objects	False Positives (fp)	True Positives (tp)	precision	recall	F1-score
Bottle	344	16	256	0.94118	0.45404	0.61257
Pan	346	17	127	0.88194	0.50951	0.64589
Pot	369	21	89	0.80909	0.40436	0.53923
Knife	271	13	189	0.93564	0.35688	0.51668
Potato	412	58	396	0.87225	0.50922	0.64303
Person	115	6	55	0.90164	0.53705	0.67315

TABLE 3. Evaluation of the model trained in both synthetic datasets with *mAP* metric.

Dataset	AP						mAP
	Bottle	Pan	Pot	Knife	Potato	Person	
Automatic	0.8	0.73	**0.78**	0.47	0.07	0.93	0.6306
Realistic	**0.86**	**0.91**	0.68	**0.68**	**0.91**	**0.96**	**0.8341**

following expression:

$$recall = \frac{|\{\text{relevant objects}\} \cap \{\text{retrieved objects}\}|}{|\{\text{relevant objects}\}|} \quad (4)$$

$$recall = \frac{tp}{tp + fn} \quad (5)$$

The *precision* and *recall* values for the automatic and realistic datasets are shown in Table 2. With these indexes, *precision* and *recall*, the value of the average precision, *AP* can be calculated, as shown in the following equation (6):

$$AP = \int_0^1 p(r)dr \quad (6)$$

being p and r the *precision* and *recall* respectively. After performing the calculation of the *AP*, the average is computed of all *AP* dividing by the total number of classes (Q). This is known as mean average precision (*mAP*), which can be computed as:

$$mAP = \frac{1}{Q}\sum_{q=1}^{Q} AP_q \quad (7)$$

with $q = 1 \ldots Q$ and Q the number of classes. Also we evaluate the *F1-score* for each class which depends on the value of the *recall* and the *precision*. This index is computed with the following equation:

$$F1{-}score = \frac{2}{r^{-1} + p^{-1}} = 2 \cdot \frac{p \cdot r}{p + r} \quad (8)$$

The *macro-averaged F1-score* is computed as:

$$macro{-}averaged\ F1{-}score = \frac{1}{Q}\sum_{q=1}^{Q} F1{-}score \quad (9)$$

This index is an excellent measure because a good value close to 1 can be only obtained if the model achieves good values in both *recall* and *precision*. The minimum value is zero. With this equation we can calculate the value of the *F1-score* for the total of both datasets. The realistic dataset has a *macro- averaged F1-score* value of 0.6051 and the automatic dataset of 0.4807. We can see again a better result with the realistic dataset.

It can be observed how the value of *F1-score* for each class is generally higher in the realistic dataset, having great relevance in the detection of the *potato* class, where the realistic dataset gets a value of 0.64303 while the automatic dataset gets a value of 0.09956 (Table 2). These results also agree on those obtained from *AP* and *mAP* (Table 3) and *tp* (true positives) (Table 2). On the one hand the automatic dataset show very few *tp* for the *potato* class and a very low *mAP*. On the other hand, the realistic dataset obtain a value of *tp* and *mAP* according to the other classes. This great variation in the case of the *potato* is due to the fact that is an object with great variability in shape and size, which makes their recognition very difficult. The realism helps the objects in the images to have real scales and therefore better results can be obtained with the realistic synthetic dataset. In Table 3 we show the values of the *mAP* with the best configuration. Different trials of the experiments would be necessary only if some configuration changes are introduced. For example, we have carried out several tests creating different automatic synthetic datasets and we have trained the network with the same configuration as before. The automatic synthetic datasets are created with the same backgrounds and classes of objects, only changing the random selection to generate the final images. These experiments allow to check the performance of the automatic synthetic dataset and the influence of the randomness in the position and scale of the objects. In Table 4 we show the value of AP for each class and each test. In this Table 4, it can be checked that the difference between the best and the worst value of mAP does not exceed 5%. So, the randomness of the automatic synthetic dataset does not impair the performance of proposed method.

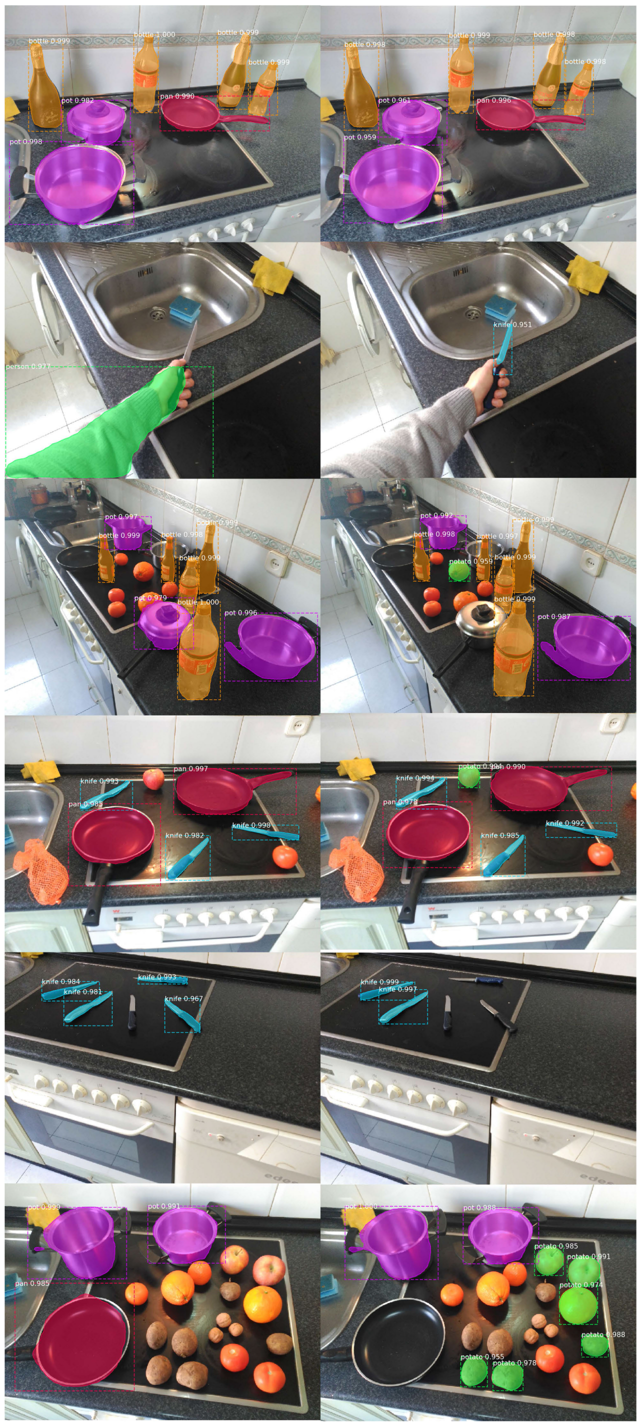


FIGURE 10. Examples of results obtained in the inference of the models. The left column shows the results obtained for the model trained with the realistic synthetic dataset. The results obtained for the model trained with the automatic synthetic dataset are shown in the right column.

Finally, in table 5 we compare the results of the realistic synthetic dataset (319 training images and 81 validation images) and the handcraft labeled dataset (81 training images and 20 validation images). Looking at the table it can appreciated how the handcraft labeled dataset is the one that gives the best performance. The differences between both

TABLE 4. Evaluation of different automatic synthetic datasets with *mAP* metric.

Dataset	AP						mAP
Automatic	Bottle	Pan	Pot	Knife	Potato	Person	
Trial 1	0.8	0.73	0.78	0.47	0.07	0.93	0.63
Trial 2	0.82	0.52	0.57	0.62	0.39	0.92	0.64
Trial 3	0.79	0.38	0.59	0.57	0.26	0.94	0.59
Trial 4	0.79	0.41	0.76	0.51	0.23	0.92	0.603
Trial 5	0.81	0.54	0.6	0.61	0.15	0.94	0.61

TABLE 5. Comparison between the realistic synthetic dataset and the handfcraft labeled dataset with *mAP* metric.

Dataset	AP						mAP
	Bottle	Pan	Pot	Knife	Potato	Person	
Handcraft labeled	0.96	0.91	0.96	0.82	0.96	0.96	0.928
Realistic Synthetic	0.86	0.91	0.68	0.68	0.91	0.96	0.8341

are not relevant (around 10%) but if we analyze the results with respect to the time to create them, the realistic synthetic dataset would be a better option. In addition, we must take into account the variability of images that can be made with the synthetic datasets. The handcraft labeled dataset has the disadvantage of needing real images which in our case, includes the necessity of having a variety of different kitchens.

In Fig. 10 some examples of the segmentation obtained by the two trained models are shown. The left column shows the results obtained with the model trained with the realistic synthetic dataset and the right column shows the results obtained with the automatic synthetic dataset. For instance, it can be seen how in the third image of the left column the *pot* is found and no false positives are produced. However in the right image the network says that an *orange* is a *potato*, thus producing a false positive. The fifth image shows how the recognition of objects obtained using the realistic synthetic dataset is better than the obtained from using the automatic dataset. Finally in the last images we can see that the realistic dataset does not produce any false positives. Note that the alternative of automatic dataset performs correctly except for a couple of classes that yields poor values of the *mAP* (Table 3).

V. CONCLUSION

In this paper, we have proposed new ideas to facilitate the training of models for the recognition of different objects that are present in a kitchen during the cooking process. Eventually, our objective is the recognition of these objects in real situations. We have adopted the neural model Mask R-CNN which has been widely used in the last few years for instance segmentation. Due to the scarce or null availability of public datasets for some applications and particularly with kitchen objects, our own datasets have been developed. Labeling images by hand is the first choice, but this needs to invest an unfeasible amount of time. For this reason, we have developed two approaches for synthetic dataset generation to train the model, and we use real images handcraft labeled to validate it. We have developed an application that generates realistic data images that greatly reduces the cost of labeling, and we have generated images with automatic placement of kitchen objects which takes negligible labeling cost. We have compared and analyzed the results obtained with the generated datasets.Note that a comparison between the synthetic dataset and the handcraft labeled dataset cannot be fair enough because the time necessary to create a handcraft labeled dataset, with the same characteristics as synthetic dataset, is unaffordable when particular applications are involved. However, the results show that spending less time to generate datasets we can get competitive performance to train neural networks. The results are promising, achieving good performance with a small amount of data. The application developed to create the synthetic dataset opens up a wide range of possibilities in this field. It also allows us to introduce new object classes on the dataset with little additional cost.

As future work, we propose to test the behavior of both synthetic datasets in other networks like YOLO. We have to change the configuration of both networks to further analyze the advantages and disadvantages. Also, we consider the possibility of creating training images from 3D models. This allows to automate and collect a large number of views of the same object, since an important problem in learning is to find images of objects with different views to robustly train the neural model. We are also studying the detection different kind of actions inside the kitchen, such as *putting the pan*, *salt the meat* or *turn on the countertop*, etc.

ACKNOWLEDGMENT

The Titan Xp used for this research was donated by the NVIDIA Corporation.

REFERENCES

[1] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. Lecun, "Overfeat: Integrated recognition, localization and detection using convolutional networks," in *Proc. Int. Conf. Learn. Represent.*, 2014, pp. 1–16.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Red Hook, NY, USA: Curran Associates, 2012, pp. 1097–1105.

[3] X.-W. Chen and X. Lin, "Big data deep learning: Challenges and perspectives," *IEEE Access*, vol. 2, pp. 514–525, 2014.

[4] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. Int. Conf. Learn. Represent.*, 2015, pp. 1–14.

[5] G. Wang, "A perspective on deep imaging," *IEEE Access*, vol. 4, pp. 8914–8924, 2016.

[6] A. Collet, M. Martinez, and S. S. Srinivasa, "The MOPED framework: Object recognition and pose estimation for manipulation," *Int. J. Robot. Res.*, vol. 30, no. 10, pp. 1284–1306, Sep. 2011.

[7] A. Gupta, A. Vedaldi, and A. Zisserman, "Synthetic data for text localisation in natural images," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Las Vegas, NV, USA, Jun. 2016, pp. 2315–2324.

[8] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 9, pp. 1627–1645, Sep. 2010.

[9] S. Song, L. Zhang, and J. Xiao, "Robot in a room: Toward perfect object recognition in closed environments," *CoRR*, vol. abs/1507.02703, 2015.

[10] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Venice, Italy, 2017, pp. 2980–2988.

[11] 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 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds. Red Hook, NY, USA: Curran Associates, 2015, pp. 91–99.

[12] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context," *CoRR*, vol. abs/1405.0312, 2014.

[13] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal visual object classes (VOC) challenge," *Int. J. Comput. Vis.*, vol. 88, no. 2, pp. 303–338, Jun. 2010.

[14] D. Damen, H. Doughty, G. M. Farinella, S. Fidler, A. Furnari, E. Kazakos, D. Moltisanti, J. Munro, T. Perrett, W. Price, and M. Wray, "Scaling egocentric vision: The epic-kitchens dataset," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 720–736.

[15] G. Georgakis, A. Mousavian, A. C. Berg, and J. Kosecka, "Synthesizing training data for object detection in indoor scenes," *CoRR*, vol. abs/1702.07836, 2017.

[16] A. Singh, J. Sha, K. S. Narayan, T. Achim, and P. Abbeel, "BigBIRD: A large-scale 3D database of object instances," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2014, pp. 509–516.

[17] J. Lemley, S. Bazrafkan, and P. Corcoran, "Smart augmentation learning an optimal data augmentation strategy," *IEEE Access*, vol. 5, pp. 5858–5869, 2017.

[18] D. Dwibedi, I. Misra, and M. Hebert, "Cut, paste and learn: Surprisingly easy synthesis for instance detection," in *Proc. Int. Conf. Comput. Vis.*, Venece, Italy, Oct. 2017, pp. 1310–1319.

[19] K. P. Seng, L.-M. Ang, L. M. Schmidtke, and S. Y. Rogiers, "Computer vision and machine learning for viticulture technology," *IEEE Access*, vol. 6, pp. 67494–67510, 2018.

[20] H. Su, C. R. Qi, Y. Li, and L. J. Guibas, "Render for CNN: Viewpoint estimation in images using CNNs trained with rendered 3D model views," in *Proc. Int. Conf. Comput. Vis.*, Santiago, Chile, 2015, pp. 2686–2694.

[21] K. Rematas, T. Ritschel, M. Fritz, and T. Tuytelaars, "Image-based synthesis and re-synthesis of viewpoints guided by 3D models," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 3898–3905.

[22] M. Stark, M. Goesele, and B. Schiele, "Back to the future: Learning shape models from 3D CAD data," in *Proc. Brit. Mach. Vis. Conf.*, 2010, p. 106.

[23] X. Peng, B. Sun, K. Ali, and K. Saenko, "Exploring invariances in deep convolutional neural networks using synthetic images," *CoRR*, vol. abs/1412.7122, 2014.

[24] B. Sun and K. Saenko, "From virtual to reality: Fast adaptation of virtual object detectors to real domains," in *Proc. Brit. Mach. Vis. Conf.*, 2014, pp. 1–12.

[25] Y. Movshovitz-Attias, T. Kanade, and Y. Sheikh, "How useful is photo-realistic rendering for visual learning?" in *Proc. Eur. Conf. Comput. Vis. (ECCV) Workshops*, in Lecture Notes in Computer Science, vol. 9915, 2016, pp. 202–217.

[26] Z. Shi, Y. Yang, T. M. Hospedales, and T. Xiang, "Weakly-supervised image annotation and segmentation with objects and attributes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 12, pp. 2525–2538, Dec. 2017.

[27] J. Lee, E. Kim, S. Lee, J. Lee, and S. Yoon, "FickleNet: Weakly and semi-supervised semantic image segmentation using stochastic inference," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 5262–5271.

[28] D. Pathak, P. Krahenbuhl, and T. Darrell, "Constrained convolutional neural networks for weakly supervised segmentation," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1796–1804.

[29] J. Marin, D. Vazquez, D. Geronimo, and A. M. Lopez, "Learning appearance in virtual scenarios for pedestrian detection," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2010, pp. 137–144.

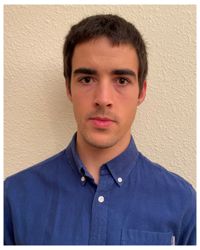
[30] D. Vázquez, A. M. López, and D. Ponsa, "Unsupervised domain adaptation of virtual and real worlds for pedestrian detection," in *Proc. 21st Int. Conf. Pattern Recognit. (ICPR)*, Nov. 2012, pp. 3492–3495.

[31] A. Gaidon, Q. Wang, Y. Cabon, and E. Vig, "Virtual worlds as proxy for multi-object tracking analysis," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Las Vegas, NV, USA, Jun. 2016, pp. 4340–4349.

[32] H.-J. Jeong, K.-S. Park, and Y.-G. Ha, "Image preprocessing for efficient training of YOLO deep learning networks," in *Proc. IEEE Int. Conf. Big Data Smart Comput. (BigComp)*, Jan. 2018, pp. 635–637.

[33] W. Abdulla. (2017). *Mask R-CNN for Object Detection and Instance Segmentation on Keras and Tensorflow*. Accessed: May 9, 2019. [Online]. Available: https://github.com/matterport/Mask_RCNN

[34] R. Anantharaman, M. Velazquez, and Y. Lee, "Utilizing mask R-CNN for detection and segmentation of oral diseases," in *Proc. IEEE Int. Conf. Bioinf. Biomed. (BIBM)*, Dec. 2018, pp. 2197–2204.
[35] T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 936–944.
[36] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 3431–3440.
[37] J. W. Johnson, "Adapting mask-RCNN for automatic nucleus segmentation," *CoRR*, vol. abs/1805.00500, 2018.
[38] A. Dutta and A. Zisserman, "The VIA annotation software for images, audio and video," 2019, *arXiv:1904.10699*. [Online]. Available: http://arxiv.org/abs/1904.10699
[39] A. B. Jung. (Aug. 2018). *Imgaug*. Accessed: May 9, 2019. [Online]. Available: https://github.com/aleju/img


RUBEN SAGUES-TANCO (Graduate Student Member, IEEE) received the B.S. degree in mechanical engineering and the M.S. degree in industrial engineering from the University of Zaragoza, Zaragoza, Spain, in 2016 and 2018, respectively, where he is currently pursuing the Ph.D. degree in electronic engineering. He is a member of the Robotics, Perception, and Real-Time Group. From 2018 to 2019, he has been a Research Assistant. He has been working in deep learning and machine learning.

LUIS BENAGES-PARDO received the B.S. degree in electronic and automatic engineering and the M.S. degree in electronic and automatic engineering from the University of Zaragoza, Zaragoza, Spain, in 2017 and 2018, respectively, where he is currently pursuing the Ph.D. degree in electronic engineering. From 2018 to 2019, he was a Research Assistant with the University of Zaragoza. He was working in deep learning and machine learning.

GONZALO LÓPEZ-NICOLÁS (Senior Member, IEEE) received the Ph.D. degree in systems engineering and computer science from the University of Zaragoza, Zaragoza, Spain, in 2008. He is currently an Associate Professor with the Department of Computer Science and Systems Engineering, University of Zaragoza. He is a member of the Robotics, Perception, and Real-Time Group, and the Aragon Institute of Engineering Research (I3A). His current research interests include visual control, autonomous robot navigation, multirobot systems, and the application of computer vision techniques to robotics.

SERGIO LLORENTE received the M.Sc. and Ph.D. degrees in electronic engineering from the University of Zaragoza, Zaragoza, Spain, in 2001 and 2016, respectively. In 2001, he joined BSH Bosch Siemens Home Appliances, Zaragoza, where he has held different positions in the Research and Development Department of Induction Cooktops. He is currently in charge of several research lines and preprojects, and also an Inventor in more than 200 patents. He has also been an Assistant Professor with the University of Zaragoza since 2004. His research interests include power electronics, simulation and control algorithms for power electronics, and temperature control.