Closing the Simulation-to-Reality Gap using Generative Neural Networks: Training Object Detectors for Soccer Robotics in Simulation as a Case Study

Nicols Cruz[1] and Javier Ruiz-del-Solar[2]

***Abstract*— In order to address the simulation-to-reality-gap, in this paper a methodology for the real-time generation of realistic images in robotic simulation environments is proposed. The images rendered by the simulator are first segmented, and then a generative neural network transforms them into realistic images. This allows training object recognition methods in situations dynamically generated by the simulator, but using realistic images. The generative neural network is trained using a database obtained using an instance segmentation network (Mask R-CNN). The whole methodology is validated in the soccer robotics domain. The reported experiments show that CNN based object detectors trained in simulation, using the generated realistic images, can be directly transferred to reality, where state-of-the art results are obtained. Moreover, we show how the training process of these detectors is fast, easy, and does not require the repetitive use of robots, which is time consuming for humans.**

I. INTRODUCTION

One of the main challenges for the training of machine learning algorithms used in robotics is the need of large amounts of training data. This is especially true in the case of robot vision systems that need to be trained using images that consider the large variability of the target real-world application. Simulations can be used to address this challenge, but this approach suffers from the simulation-to-reality gap, due to the fact that there is a visual mismatch between the images generated/rendered by the simulators and the real images. Several approaches have been proposed to deal with the simulation-to-reality gap. The standard approach consists of training in simulation and then fine-tuning in reality. New approaches attempt to make images in simulation and in reality look the same [1][2][3], either by transforming simulated images into realistic ones during simulation, by processing the real images with the goal that they look like simulated ones, or by transforming both kinds of images into a unified domain.

Following these ideas, in this paper a methodology for the real-time generation of realistic images in robotic simulation environments is proposed. In this methodology, a generative neural network is used to transform the images normally rendered by the simulator into realistic images, i.e. images with realistic real-world characteristics.

*This work was partially funded by FONDECYT project 1161500 and CONICYT/PIA Project AFB180004.

[1]Nicolas Cruz is with the Dept. of Electrical Eng, Universidad de Chile nicolascruz2187@gmail.com

[2]Javier Ruiz-del-Solar is with the Dept. of Electrical Eng. and the Advanced Mining Technology Center, Universidad de Chile jruizd@ing.uchile.cl

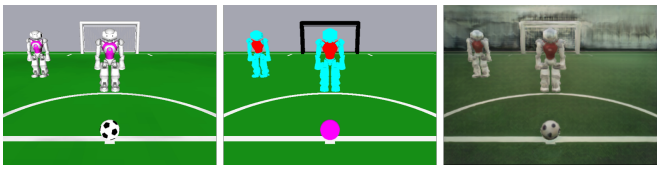

Fig. 1. Example of a rendered simulated image (left), a segmented simulated image (center), and a generated realistic image (right).

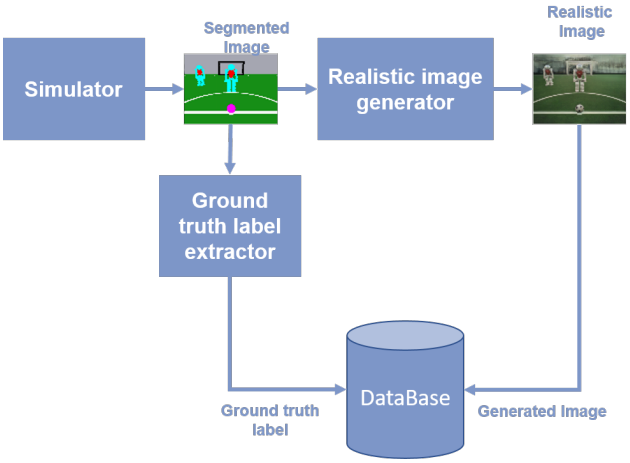

Fig. 2. Pipeline used to generate a database containing realistic images and the ground truth (obtained from the segmented images) for the training of object recognition methods.

More specifically, rendered images are first segmented, and then transformed into realistic ones (see Fig 1). This allows training object recognition methods using realistic images in situations dynamically generated by the simulator. Fig. 2 shows how the database used for this training is generated.

A main challenge of the proposed methodology is the training (construction) of the generative neural network in charge of generating realistic images. This requires a massive amount of training data, i.e. pairs of aligned segmented/simulated-real images, which are difficult and time consuming to obtain. In the here-proposed approach, these image pairs are obtained using the following procedure, which creates segmented images from real ones: First, a large amount of real images of the target environment is acquired. Then, the relevant objects in each real image are segmented using an instance object segmentation neural network (e.g.

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

a Mask R-CNN in our case). The output of this network are segmented images, which are of similar nature than the segmented images generated in simulation (the segment's identifiers just need to be the same in the segmentation network and in the simulator).

However, the training of the instance object segmentation network also requires a large amount of training data of the target domain. In this work, this training process is addressed by using a small number of labeled data together with transfer learning and active learning mechanisms, which avoids the use of a massive number of images with labeled objects. It must be stressed that this training is made just once for each application domain.

The whole framework is validated in the soccer robotics domain. We show how state-of-the-art detectors of relevant soccer objects (humanoid robot players and the ball) can be trained in simulations and then directly used in reality. These detectors show the same performance than state-of-the-art detectors directly trained in reality, but the time to generate the necessary databases is largely decreased.

In summary, the three main contributions of this work are the following: (i) The use of a generative neural network for generating realistic images from the images rendered by the simulation. (ii) A novel method for generating the training set of the generative neural network, based on the use of a state-of-the art instance segmentation network. (iii) The application of the proposed framework in a specific application domain, robotics soccer, where object detectors are trained in simulation and directly transfer to reality, where state-of-the art results are obtained.

This paper is organized as follows: Section II presents the related work. The proposed methodology is described in Section III, followed by the experimental validation in Section IV. Finally, conclusions are drawn in Section V.

II. Related work

The use of simulators for learning tasks and then transferring them to real robots (sim-to-real) is a promising approach [3][4][5][1][6][7][8]. However, training an algorithm in simulation is not always a feasible approach given the mismatch between the simulated and the real environments. To address this issue, a first approach consists of pre-processing the simulated and/or the real data to reduce their mismatch and to make them look alike. In [3], DRL based visual navigation for humanoid, biped robots in robotic soccer is proposed. Images are segmented and down-sampled, both in simulation and in reality, in order to make them look similar, thus reducing the simulation-reality gap. The system is trained using the down-sampled, segmented images generated in the simulator, and then directly deployed in reality, where the robot is able to navigate between static and moving obstacles (robots). In [4], a collision avoidance system for indoor service robots based on multimodal DRL is proposed. In this case, the reality-gap is reduced by corrupting and post-processing the simulated depth images so they resemble their real-world counterparts. In addition, both real and simulated images are decimated (and subjected to an anti-aliasing filter) to reduce their dimensionality to 60x80 pixels. The collision avoidance system is trained in simulation and then successfully transferred to reality.

A second approach for addressing the reality gap is using adversarial training techniques for learning a latent-space/representation where simulated and real data look alike. In [5], a data-efficient framework for sim-to-real transfer of navigation policies is presented. In [1], an adversarial discriminative sim-to-real transfer approach is presented. The system is first trained with simulated data, using a supervised loss, and then adapted using both simulated and real data, by adding an extra adversarial loss term. The system is able to achieve a comparable accuracy when using 50% fewer labeled real data, and a slightly worse accuracy with 75% fewer labeled real data (compared to supervised adaptation). In [2], an adversarial network is used to transfer real images into the simulation domain. This allows to train algorithms using vast amounts of simulated data. It also reports an improvement in the decision making of autonomous vehicles as the result of a less stochastic environment. In [7], a Simulated+Supervised training approach is presented. The system is trained by using an adversarial network, but with synthetic images as inputs instead of random vectors. Several modifications are introduced for preserving annotations, avoiding artifacts and stabilizing training.

A third approach is domain randomization [9], where rich random variations of the simulated data are incorporated during training, with the goal that the real data distribution will be within the distribution of the training data; domain randomization involves randomizing non-essential aspects of the training distribution in order to better generalize to a difficult-to-model test distribution [10]. One of the main challenges of this approach is to select which parameters to randomize and to what degree [5]. Domain randomization is a promising research direction where different robotic problems are being addressed [10][11][12][13].

III. Proposed Methodology

As already mentioned, the proposed methodology consists of two main stages. The first one allows the automatic generation of segmented/simulated-real image pairs using a state-of-the-art instance object segmentation network. The second stage implements the translation of segmented/simulated images onto realistic ones using a generative neural network. This network is trained using the image pairs generated in the first stage.

A. Generation of segmented-real image pairs using an instance object segmentation network

In the proposed methodology the aligned segmented-real image pairs are generated by first obtaining real images from the target environment, and then by segmenting them using an instance object segmentation neural network. In this work we select the Mask R-CNN [14] for this task. Mask R-CNN is a CNN (Convolutional Neural Network) architecture designed to perform instance segmentation. The network outputs a mask for each object instance, alongside

a bounding box, a label class and a confidence score. While several architectures can be used as a backbone for this model, we use Resnet-50 [15] due to its state-of-the-art performance.

The training of the Mask R-CNN for a new application domain requires a large amount of labeled data. We avoid this by training the network using a three-step procedure.

First, we train the Mask R-CNN using the COCO dataset [16], which is a very large database of labeled images, but from a different domain.

Second, once the model is trained we perform transfer learning to our application domain. To implement this, a database of real images of the environment in which the autonomous robot will operate must be obtained. We call this database D_{real}. For the robot soccer case described in Section IV, the database is composed of 236 real images. Then, a small subset of around five images in this database is manually segmented by a human operator. We name the annotated database of real-segmented image pairs as $D_{RS-subset}$. Afterwards, aggressive data augmentation using the manual segmented subset is carried out. This is implemented by randomly scaling, warping and then pasting the labeled objects onto background images. The obtained images are then processed to change its exposure, saturation and hue values. The process outputs a large number of images with known labels. The database of augmented-segmented image pairs is called $D_{RS-augmented}$. Then, fine-tuning of the Mask R-CNN is carried out using the $D_{RS-augmented}$ database. Given the excellent generalization capabilities of Mask R-CNN, good results can be achieved with a small amount of training iterations.

Third, active learning is used to improve the segmentation results: in each iteration, the trained Mask R-CNN is used to perform automatic labeling of the unlabeled images from a subset $D_{real-subset}$ of the D_{real} database. The predictions of the network are corrected by a human, and added to the $D_{RS-subset}$ database. Then, data augmentation is performed as previously described and the resulting samples are added to the $D_{RS-augmented}$ dataset. The Mask R-CNN network is then retrained using this dataset. Thus, by following this active learning procedure (see Algorithm 1), we are able to rapidly increase the performance of the Mask R-CNN model, while requiring minimal human interaction. We repeat this process until the Mask R-CNN achieves a satisfactory accuracy at labeling the D_{real} database.

Once this condition is met, we infer over all the images in the D_{real} database in order to obtain a large number of real-segmented pairs in the $D_{RS-subset}$ database. After discarding badly segmented images, we extract the relevant objects of this database (for the test case described in Section IV, 125 images of robots and 56 images of balls), and we perform a final data augmentation step to obtain the final the $D_{AS-pairs}$ database, composed of 2,000 images.

Algorithm 1 Active learning training process

$obtain\ D_{real}$
$D_{RS-subset} = \emptyset$
$D_{RS-augmented} = \emptyset$
while $Mask\ R-CNN accuracy \leqslant min\_accuracy$ **do**
  //choose a small number of new samples
  $D_{real-subset}(j) \subseteq D_{real}$
  for $i \in Size(D_{real-subset})$ **do**
    $Seg_i = Mask\ R-CNN_{infer}(Image_i)$
    $SegCorr_i = ManualCorrection(Seg_i)$
    $D_{RS-subset}.add(Image_i,\ SegCorr_i)$
  end for
  $D_{RS-augmented} = Data\ Augmentation(D_{RS-subset})$
  $Mask\ R-CNN_{train}(D_{RS-augmented})$
end while

B. Realistic image generation using a generative neural model

Generative models trained on a dataset aim to generate new samples following the distribution of such dataset. Two of the most popular approaches involve using GANs (Generative Adversarial Networks) [17] and VAE (Variational Auto Encoders) [18]. However, these approaches suffer from unstable training processes, problems such as mode collapse and high computational costs during training. Given this, we choose a supervised approach based on a cascade refinement network to achieve the desired results.

We based our work on [19] with some minor modifications such as using batch normalization and better initial weights. The network works by using refinement modules (see Fig. 4) to analyze an image pyramid based on the original segmented input image. An image pyramid consists on a set of images with different sizes, all generated by downsampling the same original image by a factor of 2^l where l is the layer of the feature pyramid. Given an original image of size (W, H, c_0), a feature pyramid of L layers is constructed. Then, each refinement module Fig. 4 takes as input the concatenation of the features produced by the previous refinement module with size $(w/2, h/2, c_{k-1})$ and a one hot encoded vector of the segmented image of size $(w/2, h/2, c_0)$ taken from the image pyramid. The resulting feature tensor is then processed by applying a series of $3x3$ convolutions. The features obtained by this process are then upsampled such that the output tensor shape is (w, h, c_k). These output features are the concatenated with the next image in the image pyramid, with size (w, h, c_0) to form the input to the next refinement module. Given that the first refinement block has no access to any previous features, in that case only the one hot encoded vector of the segmented image of the last layer of the image pyramid is used (i.e. the image of size $(W/2^L, W/2^L, c_0)$. By doing this, each refinement module computes increasingly higher granularity features. This in turn allows the network to work with both local and global spatially distributed patterns. Once the desired output resolution is achieved, a 1x1 convolution

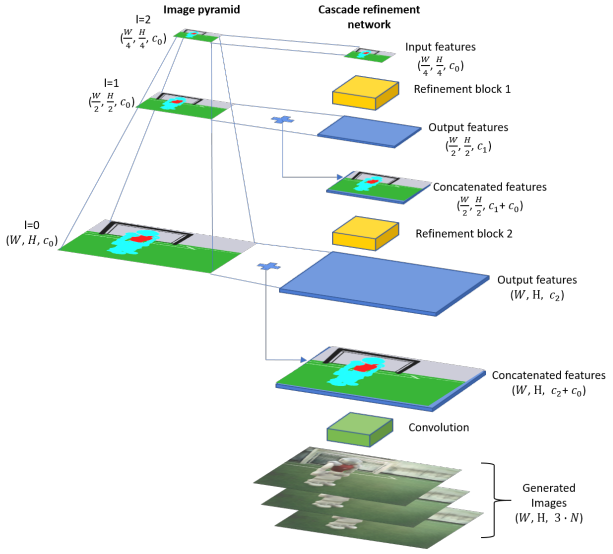

Fig. 3. Cascade refinement network architecture with two refinement blocks.

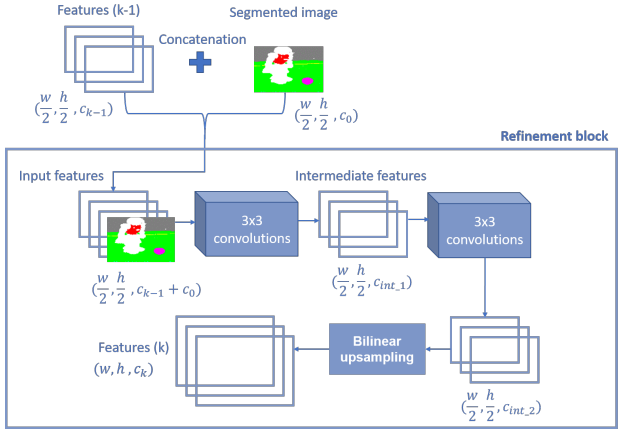

Fig. 4. Refinement block architecture.

is performed over the feature maps to generate N RGB images. Each of these N images will be slightly different in order to generate a diverse distribution of outputs and approximate domains with high class texture variance. The general structure of a cascade refinement network composed of two refinement modules and using an original image with shape (W, H, c_0) is presented in Fig. 3.

The loss function required for training the network is determined as follows. First, all the generated images $\hat{y_f}$, with $f = 0, .., N$, are fed to a VGG-19 neural network [20], which was previously trained on Imagenet. The activation maps of selected layers of the VGG-19 network are recorded as $M(\hat{y_f})$. Then, the ground truth image y is fed to the same VGG-19 netowrk and the activation maps of the same layers are recorded as $M(y)$. Afterwards, we compute the mean difference between the activation maps of each one of the generated images and the activation maps resulting from the real image as $E_f = mean(abs(M(\hat{y_f}) - M(y)))$. Finally, the loss is calculated as:

$$\alpha \cdot min(E_0, ..., E_N) + (1 - \alpha) \cdot 1/f \cdot \sum_{i=0}^{N}(E_i) \qquad (1)$$

where α is a user tunable parameter.

We first train the generative model using the $D_{RS-augmented}$ database without using any saturation, hue or exposure data augmentation. This allows the model to be more temporally consistent in terms of lighting. Then, once the model has achieved a high enough accuracy on the $D_{RS-augmented}$ database, we fine tune the model by using the $D_{RS-subset}$ database in order to be able to learn complex image features such as shadows and reflections.

Human labeling is only sparsely necessary to create the database $D_{AS-subset}$ which is used to train the generator network. Once the network is trained, no further human interaction is needed and the network can be used to generate realistic images. This is done by segmenting the simulated image and then feeding the segmented image to the generative neural network that then outputs a realistic image. Given that all the generated realistic images have a corresponding aligned segmented image, and therefore the objects in that images are known, a database composed of realistic images with ground truth can be quickly obtained without any human labeling in order to train machine learning based algorithms (e.g. CNN based detectors).

IV. Experimental Results

A. Case Study: Playing soccer in the RoboCup SPL League

As a test case, we use the B-Human simulator [21] to apply the here-proposed methodology. This is the most popular simulator available in the Standard Platform League (SPL) of the RoboCup World federation, and it is used by a vast majority of the participant teams. The simulator is able to simulate soccer games with several humanoid players, and it is commonly used to test self-localization, obstacle modeling, high-level soccer behaviours and other relevant functionalities required to play a soccer game. In order to render an image in real-time, the B-Human simulator uses opengl, an API that is cross platform compatible and is highly efficient. While the simulator is realistic in terms of object shapes and proportions, the textures are not representative of reality. This was a minor drawback in early years of the RoboCup competition as colored objects, such as an orange ball, yellow posts and field markers, were used on the field. This allowed the use of color-based vision systems. As the league progressed, increasingly realistic conditions were added such as a realistic ball, white goal-posts, natural illumination and no color markers, which meant that algorithms developed in simulation using simple rendered images did not translate well into reality. Most notably, as the league transitioned from heuristic color-based vision system to machine learning based vision systems, it has become increasingly difficult to develop and test algorithms using this simulator, because of the simulation-to-reality gap. This makes soccer robotics an ideal case study for the proposed methodology.

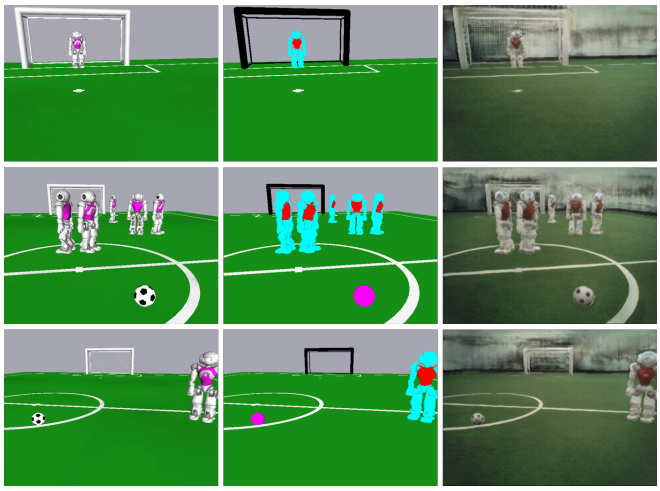

Fig. 5. First column: Simulated rendered images. Second column: Corresponding segmented images. Third column: Corresponding realistic generated images

We use Pytorch [22] as framework for all the implemented neural networks including Mask R-CNN and the generative refinement network. The generative model is trained in python and then exported to a C++ compatible model by using Pytorch JIT (Just In Time) compiler options. Then, the model can be used in simulation just by compiling the simulator and including the torch library. In order to generate the segmented images in simulation, a unique color is assigned to each class object in the form of a particular texture. Then, the image is segmented by using simple RGB bounds, and fed to the generative refinement network in order to generate realistic images in simulation. Examples of the rendered, segmented and resulting realistic images can be seen in Fig. 5. Further examples of real time simulation can be found in https://youtu.be/WVPgr_xxd7I.

In the next sub section, we analyze the complexity of building a training database for object recognition methods using the classical approach (i.e. using the robot players), and using our realistic simulation. We show how this process is largely reduced in this last case.

Then, we compare the performance of object recognition algorithms trained using real samples collected directly from the robot, and of algorithms trained using samples generated by the here-proposed methodology. This comparison is based on the experience of our soccer robotics team, UChile robotics, which has developed a series of state-of-the-art CNN based robot vision algorithms for detecting robots [23], and for detecting the ball [24].

B. Generation of the training database

Machine learning models used in robotics soccer require large amounts of training data, which should consider different game situations. One of the main challenges that we address in this paper is the difficulty of obtaining such data. While standard approaches are based on the capture and labeling of real samples using a robot and realistic soccer situations, the here-proposed proposed approach is able to generate realistic labeled samples in a simulated environment.

Building a database in robotics soccer and training CNN based detectors require several steps. We analyze these different steps and conclude that developing a vision system using a realistic simulator is less time consuming, easy, and better to avoid damaging the robots. We then compare the performance of detectors trained in different environments when deployed in real conditions and prove that our simulated environment can be reliably used to test and prototype other algorithms.

1) Object detector development: In the RoboCup SPL league the robots are required to detect different objects in real-time while playing. However, these robots possses limited computational resources in the form of a single core, 1.6 Ghz Intel Atom processor and 1 GB of RAM memory. In order to achieve real time operation, most teams implement detectors using a two-step approach consisting of an heuristic-based region proposal generation followed by a CNN classification of the proposal regions. In the classical approach, the development of the detector is first carried out in simulation. Once a working prototype has been obtained, the development shifts to the real robot in order to adapt the detector to the real-world conditions. This means fine-tuning the region extractor parameters and then performing sample collection to re-train the network with real samples in order to achieve proper operation in the real environment. If this is not feasible, the algorithm is re-designed.

In contrast, by using our realistic simulator the complete process can be performed in simulation. This means that both the architecture of the region extractor as well as it parameters' tuning can be developed in conjunction, which streamlines the process. This also results in faster development, as iterating over the system can be done exclusively in simulation, without the need of the real robot. Finally the reality gap is low enough that the detector which was developed in simulation can be directly deployed in the real robot with minimal performance penalties.

2) sample collection: Under the classical approach, this process consists in using a real robot to obtain positive and negative samples of the objects to be detected (e.g. robots or the ball) given by the region proposal algorithm. Normally, sample collection is a long process, which can take several hours and involves preparing the robot and the environment, and then manually modifying the position of the robot in the environment to avoid potential collisions, as well modifying the position as the different objects within the environment that need to be detected.

When using our proposed methodology, sample collection is heavily simplified as all samples can be collected in simulation. Since this is the case, no robot preparation is needed and samples are collected by using a simulated robot to traverse the field, saving the samples generated by the region proposal algorithm. To achieve the same variance of the field as in the classical case, objects on the field are randomly moved on the environment automatically (i.e. by the simulator). If the robot sample collector collides with an

object, the simulation is simply restarted. This means that when using the realistic simulator, no human operators are needed, sample collection is faster given the automatic nature of the sample collection process, and finally, more samples can be collected with no cost for the hardware, since the simulation can be run for as long as needed.

3) sample labeling: The last step to train the supervised CNN based classifiers is to perform sample labeling. In a classical scenario, with samples collected from the real robot, humans need to manually label the collected samples. While this may sound trivial, the process becomes harder with large databases and can take several hours for databases of thousands of images. Furthermore, in the robot soccer application where real-time operation is need, samples are usually low-resolution and very small. This means that humans are prone to making errors in the labeling process.

On the other hand, samples that are obtained using the proposed visual realistic simulation are labelled automatically without the need of human intervention. This can be easily done since the corresponding segmented image is available in simulation, which indicates the labels of all the objects present in a region proposal. This means of course that no humans are needed, the process is instantaneous, and there are no samples with incorrect labels.

C. Performance in reality

In this section, we train two CNN based classifiers: a classifier of robots players and a ball classifier. These CNN models are used in conjunction with a region proposal algorithm to detect robots and balls in real time. The robot detector uses a modified version of the CNN architecture described in [23], while the ball detector is a modified version of the CNN architecture described in [24], with the only variation being the number of filters in each layer. We train these models with three different datasets: (1) training using a database of real images (see Fig. 6), (2) training using realistic images generated in simulation (see Fig. 7), and (3) training using the standard images rendered in simulation (see Fig. 8). For the real dataset, samples were manually labeled, whereas for the simulated datasets, samples were automatically labeled by the simulator. In order to make a fair comparison, in the first three tests cases all three methods use 2,900 samples for the ball detector and 4,100 samples for the robot detector with the same class proportion. Since the methods that use simulated images can use a much larger number of images with no penalty to the hardware and very little database generation time, our last test, named large dataset (see Table I), uses a higher number of samples, with 6,900 samples for the robot detector and 5,000 for the ball detector.

The performances of the robot and ball classifiers trained with the real, realistic and rendered datasets are compared by testing them using real data. The ball testing dataset is composed of 760 manually labeled samples, while the robot testing dataset is composed of 959 manually labeled samples. The obtained results are shown in Table I.

Fig. 6. Top row: Real ball samples. Bottom row: Real robot samples.

Fig. 7. Top row: Generated realistic ball samples. Bottom row: Generated realistic robot samples.

The results show that the detectors obtain a similar performance when trained using real samples and when trained realistic images produced in simulation with the generative neural network. In the case that the same number of training images is used in both cases, the differences in accuracy are about 4% for the robot detector and 3% for the ball detector. This difference is less than 2% when more training images are used in simulation with realistic images. It should be remembered that the generation of realistic images is straightforward with the proposed methodology. This means that the proposed approach is able to bridge the reality gap between simulation and reality, allowing to train models in simulation and deploy them in real situations without the need of fine-tuning.

On the contrary, the robot detector model trained using standard rendered simulation images shows a vastly inferior accuracy than those obtained by training the model using real and realistic simulated images (see Table I). This can be explained by the reality-simulation gap, which means that

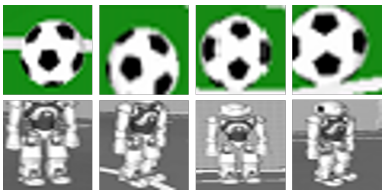

Fig. 8. Top row: Standard simulated ball samples. Bottom row: Standard simulated robot samples.

TABLE I

ROBOT AND BALL MODELS TRAINED IN DIFFERENT DATASETS AND EVALUATED IN REALITY (USING REAL DATA).

training data type	Robot detector accuracy %	Ball detector accuracy %
real images	90.7	97.2
realistic images (simulation)	86.5	94.3
rendered images (simulation)	63.3	95.6
realistic images, large dataset (simulation)	89.2	95.7

TABLE II

ROBOT AND BALL MODELS TRAINED IN REALITY (USING REAL DATA) AND EVALUATED IN DIFFERENT DATASETS.

testing data type	Robot detector accuracy %	Ball detector accuracy %
real images	90.7	97.2
realistic images (simulation)	94.2	91.7
rendered images (simulation)	85.6	83.7

filters learned by the classifier on the rendered simulated images are not valid when used in reality.

On the other hand, the accuracy of the ball detector does not change by a significant amount, between the real, realistic and rendered domains. This is probably the result of the relative simplicity of the ball detection problem. Indeed, the ball pattern of black pentagons over a white ball is a very unique feature on the field, so it is difficult to find adversarial examples to produce false positives and false negatives. Furthermore, the high contrast of the pattern and very low sample variance makes the ball classification problem a rather simple task for CNN based classifiers. Lastly, the main features of the ball, clearly defined geometric shapes and very high contrast, are shared between the real and simulated domains, so there is not really much of a reality gap, which explains the similar results between domains.

D. Model adaptation to simulation

When a classical approach is used to built an object detector, the machine learning based model is trained to work in the real environment. However, this model will perform vastly different when used in a standard simulator given the reality gap. This means that testing other tasks in simulation which are vision-dependant such as self-localization or obstacle avoidance becomes non-representative or impossible. To solve this, a second model (e.g., CNN based robot detector) is commonly trained using simulated samples to work in the standard simulated environment, while trying to achieve similar recall and precision to those of the real model. By constructing a realistic simulator, we hypothesise that this step is no longer necessary since the reality gap is low enough to have similar behavior in reality and in the realistic simulation, using the same model in both.

We prove this by first training the same ball and robot classifiers using the real ball and real robot datasets, composed of 2,900 and 4,100 manually labeled samples respectively. We then test the performance of these models, which were trained using real data in three different environments: (1) testing using a database of real images, (2) testing using realistic images generated in simulation, and (3) testing using the standard images rendered in simulation. The real, realistic and rendered test datasets are composed of 760 images for the ball dataset and 959 samples for the robot dataset. Samples from the real domain were manually labeled, whereas samples from simulated environments were labeled by the simulator itself. Results for these experiments are presented in Table II.

While the detectors trained using real samples achieve similar accuracy when tested on real samples and in our realistic simulator, the same is not true for the standard simulator case, where the model achieves an inferior accuracy. This means that models trained using real data can be evaluated on the realistic simulator, but the standard simulator is not a good tool to evaluate the performance of models trained on real samples. This also means that high-level modules that depend on visual systems, such as behavior or world-modeling, will perform closer to their expected real behaviour in the realistic simulator.

V. CONCLUSIONS

In this work we presented a methodology to generate realistic images from simulated ones. This allows training object recognition methods in situations dynamically generated by the simulator, but using realistic images. The proposed methodology is implemented using a generative neural network which the generates the realistic images, and using an instance segmentation network (Mask R-CNN) to generate the training data for the generation network. Naturally, this second network is used just once, when the generative network is trained.

The whole methodology is validated in the soccer robotics domain. In this domain, the reported experiments show that the proposed approach is able to bridge the reality gap between simulation and reality; detectors of soccer objects such as robot players and the ball can be trained in simulation and directly transferred to reality, where state-of-the art results are obtained. Furthermore, models tuned to work in real environments can be accurately tested and modified in the simulator. This is not possible using a standard robot simulation environment, and it allows to develop algorithms in a faster way without the need of using the real robot, which is a great advantage when dealing with visual problems in dynamic robotics applications such as robotics soccer.

ACKNOWLEDGMENT

This work was supported by the NVIDIA GPU program, and partially funded by CONICYT-FONDECYT Project 1161500 and CONICYT-PIA Project AFB180004.

REFERENCES

[1] F. Zhang, J. Leitner, M. Milford, and P. Corke, "Sim-to-real transfer of visuo-motor policies for reaching in clutter: Domain randomization and adaptation with modular networks," *CoRR*, vol. abs/1709.05746, 2017.

[2] L. Yang, X. Liang, and E. P. Xing, "Unsupervised real-to-virtual domain unification for end-to-end highway driving," *CoRR*, vol. abs/1801.03458, 2018.

[3] K. Lobos-Tsunekawa, F. Leiva, and J. Ruiz-del-Solar, "Visual navigation for biped humanoid robots using deep reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 3, pp. 3247–3254, Oct 2018.

[4] F. Leiva, K. Lobos-Tsunekawa, and J. Ruiz-del-Solar, "Collision avoidance for indoor service robots through multimodal deep reinforcement learning," *RoboCup Symposium*, 2019.

[5] H. Bharadhwaj, Z. Wang, Y. Bengio, and L. Paull, "A data-efficient framework for training and sim-to-real transfer of navigation policies," *CoRR*, vol. abs/1810.04871, 2018.

[6] A. A. Rusu, M. Vecerík, T. Rothörl, N. Heess, R. Pascanu, and R. Hadsell, "Sim-to-real robot learning from pixels with progressive nets," *CoRR*, vol. abs/1610.04286, 2016.

[7] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb, "Learning from simulated and unsupervised images through adversarial training," *CoRR*, vol. abs/1612.07828, 2016.

[8] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell, "Adversarial discriminative domain adaptation," *CoRR*, vol. abs/1702.05464, 2017.

[9] N. Sünderhauf, O. Brock, W. J. Scheirer, R. Hadsell, D. Fox, J. Leitner, B. Upcroft, P. Abbeel, W. Burgard, M. Milford, and P. Corke, "The limits and potentials of deep learning for robotics," *CoRR*, vol. abs/1804.06557, 2018.

[10] J. Tobin, W. Zaremba, and P. Abbeel, "Domain randomization and generative models for robotic grasping," *CoRR*, vol. abs/1710.06425, 2017.

[11] F. Sadeghi and S. Levine, "(cad)$ˆ2$rl: Real single-image flight without a single real image," *CoRR*, vol. abs/1611.04201, 2016.

[12] 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," *CoRR*, vol. abs/1703.06907, 2017.

[13] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-real transfer of robotic control with dynamics randomization," *CoRR*, vol. abs/1710.06537, 2017.

[14] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick, "Mask R-CNN," *CoRR*, vol. abs/1703.06870, 2017.

[15] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015.

[16] 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.

[17] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems 27* (Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, eds.), pp. 2672–2680, Curran Associates, Inc., 2014.

[18] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *CoRR*, vol. abs/1312.6114, 2013.

[19] Q. Chen and V. Koltun, "Photographic image synthesis with cascaded refinement networks," *CoRR*, vol. abs/1707.09405, 2017.

[20] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.

[21] T. Laue, K. Spiess, and T. Rfer, "Simrobot a general physical robot simulator and its application in robocup," vol. 4020, pp. 173–183, 07 2005.

[22] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.

[23] N. Cruz, K. Lobos-Tsunekawa, and J. Ruiz-del-Solar, "Using convolutional neural networks in robots with limited computational resources: Detecting NAO robots while playing soccer," *Lecture Notes in Computer Science*, vol. 11175, (RoboCup Symposium), 2017, pp. 19-30.

[24] F. Leiva, N. Cruz, I. Bugueño, and J. Ruiz-del-Solar, "Playing soccer without colors in the SPL: A convolutional neural network approach," *Lecture Notes in Computer Science*, vol. 11374 (RoboCup Symposium), pp. 122-134., 2018.