2020 IEEE International Conference on Robotics and Automation (ICRA)
31 May - 31 August, 2020. Paris, France

DirtNet: Visual Dirt Detection for Autonomous Cleaning Robots

Richard Bormann, Xinjie Wang, Jiawen Xu, Joel Schmidt

Abstract— **Visual dirt detection is becoming an important capability of modern professional cleaning robots both for optimizing their wet cleaning results and for facilitating demand-oriented daily vacuum cleaning. This paper presents a robust, fast, and reliable dirt and office item detection system for these tasks based on an adapted YOLOv3 framework. Its superiority over state-of-the-art dirt detection systems is demonstrated in several experiments. The paper furthermore features a dataset generator for creating any number of realistic training images from a small set of real scene, dirt, and object examples.**

I. Introduction

With the development of more advanced cleaning robots dirt detection becomes a research topic of increasing importance. On the one hand, the detection of local dirt levels can be used for achieving a desired level of cleanliness with consumer robots [1]. On the other hand, professional cleaning machines start to enter less structured environments which require more powerful and intelligent algorithms. Professional office floor cleaning is often conducted as daily cleaning on demand nowadays. A cleaning robot must hence be able to focus on finding and cleaning only dirty spots instead of cleaning the whole ground each day, wasting energy and time [2]. Professional cleaning robots are also supposed to verify their cleaning results for tuning local cleaning efforts and documentation purposes [2], [3].

Cleaning companies strive to incorporate robots into their services due to increasing shortages on qualified personnel. Several publicly funded research projects like AutoPnP [2], Flobot [3], or Baker [4] underline the importance of robotic cleaning machines. The work demonstrated in this paper originates from the BakeR project which is concerned with developing a modular office cleaning robot for vacuum cleaning, wet cleaning, and trash bin clearing. The visual dirt detection system is supposed to guide the robot to pollutions during room inspection for cleaning dirt spots but it should also detect typical office items lying on the ground without raising a false alarm or trying to clean those items.

A basic, learning-free dirt detection system is already available from previous work [2], [5], [6], i.e. it can be directly applied in any new environment. Though achieving a high recall rate, it comes with a high false positive rate due to the lack of a learning mechanism. Operators can only mark certain locations in the environment as false alarms. Moreover, the system cannot distinguish between useful objects and dirt items [2]. This paper introduces a learning-based dirt and object detection system which is significantly superior to the baseline system [6] both in performance and functionality while still being applicable to new environments without any retraining. Fig. 1 provides some exemplary classification results of the new system.

This project has received funding from the German Federal Ministry of Economic Affairs and Energy (BMWi) through grant 01MT16002A (BakeR), and from the Ministry of Economic Affairs of the state Baden-Württemberg (Zentrum für Cyber Cognitive Intelligence – Grant No. 017-192996).

The authors are with the Robot and Assistive Systems Department, Fraunhofer IPA, 70569 Stuttgart, Germany. `<first>.<last name>@ipa.fraunhofer.de`

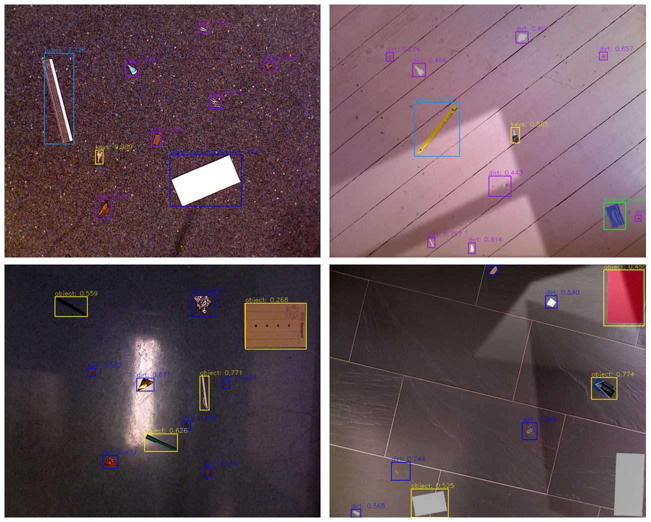


Fig. 1. Dirt and object detection on real (left) and synthesized data (right).

Collecting and labeling sufficiently large datasets for the initial training is a common problem in robotics. To overcome dataset size limitations due to manually arranged, captured, and labeled scenes, we developed a dataset creation tool which combines all necessary ingredients for synthesizing realistic training images from a small set of real data. For obtaining dirt and object detection training samples, the dataset generator blends randomly modified patches of dirt and office objects with clean ground floor images and optionally adds simulated light sources and shadows to the scene. This way, all kinds of realistic scene images can be created in any desired amount. We believe that the image synthesizer might be useful beyond our application for creating other kinds of datasets as well. The dataset and software are online available[1].

In summary, the main contributions of this paper are:

- a dataset generator for synthesizing image data based on a few floor, dirt, and object samples,
- a learning-based dirt and object detection system, directly applicable to new floor types, and
- a comprehensive evaluation of the proposed methods.

[1]Instructions: `http://wiki.ros.org/ipa_dirt_detection`

II. Related Work

Cleaning machines become smarter and more automated these days. A good overview on smart cleaning machines and their components can be found in [1], [2], [7] and with special focus on the navigation capabilities also in [8], [9].

Dirt detection is relevant to wet and dry cleaning [2], [3], [7]. So far, levels of cleanliness have been evaluated by consumer robots with optical and piezo-electronic sensors inside the vacuum bin sensing the degree of pollution and adapting the cleaning process. However, these sensors cannot detect all types of dirt, e.g. stains on a carpet, and they can only sense cleanliness after cleaning a location [3]. Visual dirt detection in contrast has a larger receptive field and can run without an activated cleaning device, thus saving energy and machine lifetime.

There are only two approaches that directly deal with visual dirt detection tasks [3], [6]. Our previous approach [6] is based on a saliency detection algorithm [10] for finding outstanding regions at the floor. [3] proposes the unsupervised online learning of a Gaussian Mixture Model representing the floor pattern with high success rates.

In both approaches, the authors developed their own datasets. The *IPA dataset* contains 9 categories of dirt [6], whereas the *TU Vienna dataset* provides 3 dirt types [3]. However, none of these datasets collected 'non-dirt' objects such as pens and notebooks. Further, the total amount of dirt and types of ground patterns is limited in these datasets and extensions are very time consuming. To reproduce training data in sufficient amounts and variety for the use in real applications, this work introduces a synthetic dataset with expanded amounts of dirt, office objects, and ground pattern samples, which can easily be extended with further data of any kind. The approach is comparable to image synthesizers for object detection, see [11] for a good overview.

Since learning-based methods were not powerful enough to cover the variety of ground patterns and dirt types in the past, learning-free methods were preferred [3], [6]. The advent of Convolutional Neural Networks opened new possibilities for treating the dirt detection problem as an object detection task leveraging domain-specific feature learning for coping with the vast variety of dirt types. The current most main-stream object detection methods are region proposal approaches [12], single shot methods [13], and YOLO [14]. A core requirement on a combined dirt and object detection system is good detection performance over several scales since dirt may be captured as small as a few pixels whereas large objects like books may occupy several ten thousands of pixels. Single shot methods are said to degrade significantly when dealing with small objects [13]. Though, recorded images need to be presented to the network at high resolution, which requires an efficient architecture. We evaluated the region proposal method Faster RCNN [12] beforehand, but inference was too slow (max. 2 Hz) when the network was tuned for high detection performance [15]. Consequently, the potential of YOLO is leveraged in this work to design a robust, fast, and accurate dirt detection system.

III. Methods

The dirt and object detection system proposed in this work is based on YOLOv3 [14], which offers three branches for small, medium, and large object detection as needed for detecting small dirt spots and large objects simultaneously. YOLO can process high resolution images at sufficient speed for the online, onboard operation on a fast moving cleaning robot, inspecting its environment for dirt. In this work, the utilized baseline YOLOv3 system with Darknet53 backend originates from [16], which achieves the same performance on COCO as the original YOLOv3, and is implemented in PyTorch [17]. This section explains several fine-tuning measures to the baseline implementation for optimized performance in the dirt and object detection setting.

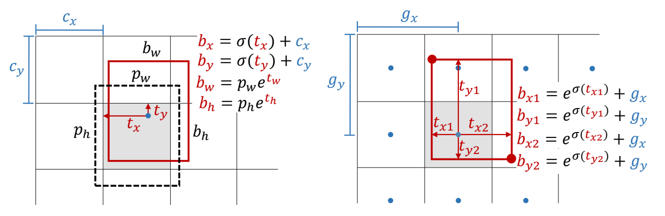
a) Rectangle training and inference: Instead of keeping the square aspect ratio the network input is redesigned as rectangular shape of size 1024×832 pixels. Thus, no image padding is necessary as proposed in original YOLOv3, and the images are resized to the desired network input resolution without padding. This saves 19% FLOPs (floating point operations) compared to a square resolution of 1280×1024, and hence accelerates the training and inference phases and even comes with a slightly higher mAP (see Sec. V-B).

b) Individual detection thresholds for each branch: Detectors like YOLOv3 [14] or Feature Pyramid Networks [18] detect small objects in shallow layers, which contain more details and edge information, and large objects in deep layers which contain more abstract semantic representations. We observed that the larger objects with deeper feature representation tend to have higher confidence. In the dirt detection task, some dirt samples are smaller than 15×15 pixels whereas some objects like books or rulers are larger than 200×200 pixels. It is necessary to reduce the confidence threshold for detecting more small dirt, but this increases false positive detections from higher layers especially on sophisticated floor patterns, resulting in decreasing precision. Therefore, individual confidence thresholds were implemented for each branch: a low confidence threshold in the small object branch, and higher confidence thresholds in the middle and large object detection branches. The resulting bounding boxes from all 3 layers are then filtered by Non Maximum Suppression (NMS).

c) GIOU loss and focal loss: Generalized Intersection over Union (GIOU) loss [19] is used for bounding box regression since the mAP evaluation is mainly based on maximizing IoU. The GIOU loss is directly related to this criterion in contrast to the commonly used indirect distance losses. Focal loss [20] is applied on top of the binary cross entropy loss for foreground and background classification to cope with foreground-background class imbalances.

d) Warm up and cosine learning rate: Inspired by SGDR [21], the learning rate is raised from nearly 0 at the beginning to l_i at the end of the second epoch during a warm up phase and then continuously decreases according to a cosine learning rate schedule to l_e at the end of training.

e) Mixup: Mixup [22] is introduced into the training as it increases robustness of the detection system by blending



c_x
c_y
b_w
p_w
$b_x = \sigma(t_x) + c_x$
$b_y = \sigma(t_y) + c_y$
$b_w = p_w e^{t_w}$
$b_h = p_h e^{t_h}$
t_x
t_y
p_h
b_h
g_x
g_y
t_{y1}
t_{x1}
t_{x2}
t_{y2}
$b_{x1} = e^{\sigma(t_{x1})} + g_x$
$b_{y1} = e^{\sigma(t_{y1})} + g_y$
$b_{x2} = e^{\sigma(t_{x2})} + g_x$
$b_{y2} = e^{\sigma(t_{y2})} + g_y$

Fig. 2. Visualization of the original YOLO-based predefined anchors (left) and corner offset definition used for bounding box regression throughout this work (right).

multiple ground truth images and their labels in addition to the original data.

f) Label smoothing: Inspired by Inception v2 [23], label smoothing is employed during training for category classification.

g) MobilNetV2 backbone: While the previous steps focused on increasing robustness and accuracy, replacing the Darknet53 backbone of YOLO by MobileNetV2 [24] majorly aims for gains in processing efficiency.

h) Separable convolution: Inspired by Xception [25], the convolutional layers in the detection branches are replaced with depthwise separable convolutional layers keeping the same parameters, likewise for efficiency reasons.

i) Redesign layers: As discussed in [26] the receptive field also affects the detection performance. Larger receptive fields hurt the detection performance of small dirt spots since more context information of the floor becomes included. Further, features of small dirt easily get lost through reduced feature map resolution in deeper convolutional architectures. Therefore, one 1×1 and one 3×3 convolutional layer is removed from the small object detection branch. Since the receptive field from the last layer of MobileNetV2 is not large enough for large object detection with input images of 1024×832 pixels resolution, one 3×3 convolutional layer is added to the large object detection branch. Finally, the last layer of MobileNetV2 is removed because we observed that the features are already sufficient for succeeding in the dirt detection task at hand.

j) Remove predefined anchors: YOLOv2 [27] proposes kmeans for determining bounding box priors, which can be sensitive to the data distribution and initial value. Here we remove the predefined anchors, which introduce many hyperparameters for the training phase. Instead, the two corners' offsets $t_{x1}, t_{y1}, t_{x2}, t_{y2}$ from the center of the feature map grid are regressed directly. Fig. 2 illustrates this approach, where $b_{x1}, b_{y1}, b_{x2}, b_{y2}$ are the coordinates of the bounding box, σ stands for the sigmoid function, and g_x, g_y are the center coordinates of the grid. As there is limited scale variance in the appearance of the target objects in the dirt detection task and since there are already three branches for differently scaled objects, the number of regressed anchors per grid is reduced from 3 to 1 for efficiency reasons.

IV. Dataset

This section explains the dataset and the synthesizing tool for creating realistic training data from real image data of clean floors, various kinds of dirt, and typical office objects.

All images were recorded with an Asus Xtion Pro Live camera at 1280×1024 pixels resolution from a top view perspective at 1 m distance to the ground since in earlier work we found that working from a normalized perspective is advantageous [6]. This way we recorded images of clean floor areas as well as objects and dirt samples placed individually on unicolored sheets of paper with high contrast to the respective samples. Dirt and objects were segmented from the background by Mean Shift clustering in L*a*b* color space, yielding segmentation masks. These masks encode opacity levels at fine grained structures and object boundaries for more realistic blending effects.

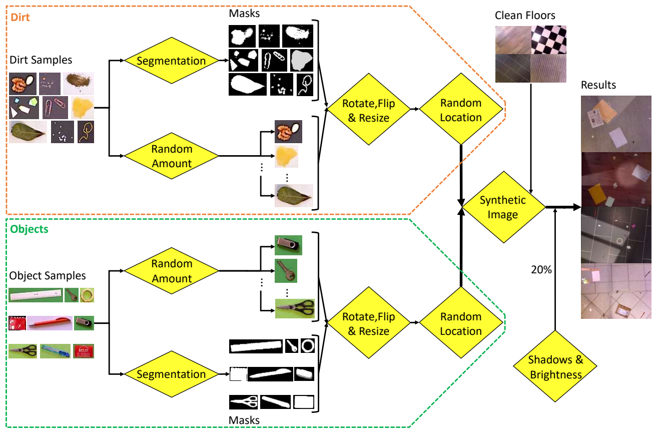


Fig. 3. Overview of the pipeline for synthesizing the dirt dataset.

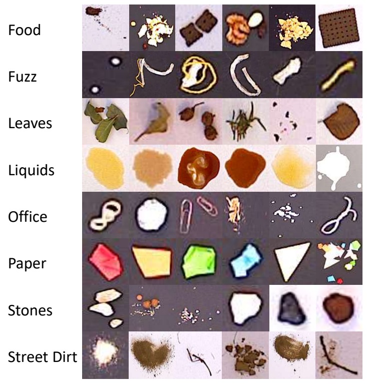


Fig. 4. Examples of the collected real world dirt samples.

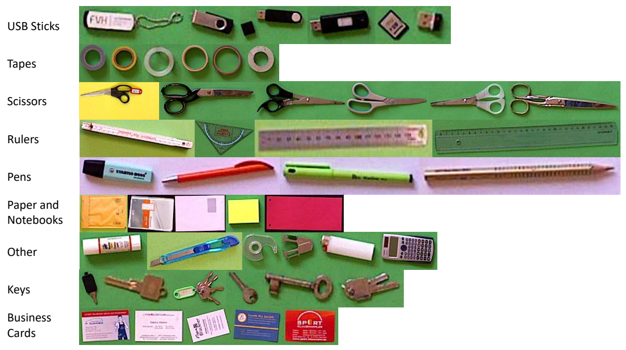


Fig. 5. Examples of the collected real world office objects.

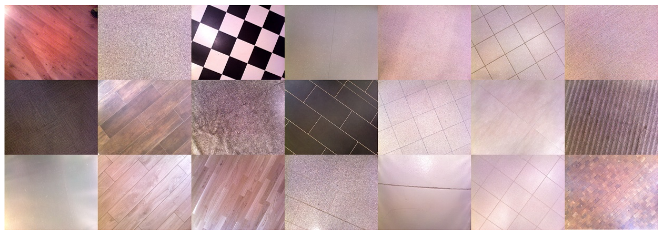


Fig. 6. Exemplary clean ground patterns of the 21 training set floor types.

For image synthesis random numbers of segmented dirt and object samples are selected and randomly rotated, flipped and resized (resize factor between 0.8 and 1.2) before being blended into the clean ground images. The corresponding object bounding boxes are computed and saved according to the masks. The synthesized images are finally post-processed by adding artificial shadows and brightness of random shape and intensity at random positions to better mimic real scene variance. Shadows and brightness masks are modeled as random polygons with blurry edges and affect all rendered dirt spots and objects as well. Shadow masks are subtracted with some opacity factor, brightness masks are added to the given image. The whole pipeline is illustrated in Fig. 3. The image synthesis tool can create infinite numbers of distinctive images. Some exemplary dirt, object, and floor images are shown in Fig. 4, 5, and 6. Examples of synthesized images are provided in Fig. 1 (right column), in Fig. 10, an in the accompanying video. We generated a training set and two test sets with this technique with the following properties.

a) Synthesized training data: The training set contains 9248 synthesized images which were created with the following real world samples: (i) 21 floor types with 1156 images altogether, (ii) 358 different dirt samples, (iii) 438 different object samples. Each image is blended with 5 to 8 dirt spots and 3 to 6 office objects.

b) Real test data (Test): The Test set contains 80 images of manually arranged and labeled real scenes, recorded on 8 novel floor types which are not contained in the training set. Dirt and objects are likewise new and not contained in the training set. This dataset is quite small since recording was extremely time-consuming (2 days).

c) Synthesized test data (TestSynth): The TestSynth set contains 1360 synthesized images which were created with the following real world samples: (i) 8 floor types with 170 images altogether (new floor types, same as in Test), (ii) 70 different dirt samples, (iii) 38 different object samples. All dirt and object samples coincide with those in set Test. They are new and not contained in the training set. The advantage of a synthesized test set is the potentially unlimited amount of test data with novel floors, dirt, and object samples at limited recording effort. It is meant to compensate for the small real world test set (Test).

d) Synthesized test data with training floor types (TestSynthTF): The TestSynthTF set contains 3448 synthesized images which were created with the following real world samples: (i) 21 floor types with 431 images altogether (same floor types as in training but new images), (ii) 70 different dirt samples, (iii) 38 different object samples. All dirt and object samples coincide with those in set Test. They are new and not contained in the training set. This set is very similar to TestSynth except for the training floor types. It is meant as a verification dataset for evaluating the generalization performance to new floor types in TestSynth.

V. Evaluation

A. Experimental setup

The dirt detection task at hand considers the detection of foreground objects on ground floors and their classification into dirt or (office) object samples (*2 class task*). A cleaning robot needs to clean the dirt occurrences while avoiding useful objects. If required, the detection system can also be trained to distinguish dirt and the individual object categories of the dataset (*10 class task*).

The database images of size 1280×1024 are downsampled to the network input of size 1024×832. The Darknet53 and MobileNetV2 backends are pre-trained on ImageNet. A representative validation set (last 5 of the 21 training set floors) is split off the training set. During fine-tuning training all layers are trained for 15 epochs. The validation error always settled before 15 epochs of training and the best model according to validation loss was always found between epoch 10 and 15.

The network was trained with typical parameters for training; there was no special tuning on the parameters w.r.t. mAP optimization. The most important settings are: IoU=0.5; small/medium/large object detection thresholds = (0.25, 0.3, 0.3); NMS-IoU=0.1; learning rate: warm up $3.33\text{e}{-7} \rightarrow l_i = 3.33\text{e}{-4}$ (2 epochs), then $l_i \rightarrow l_e = 3.33\text{e}{-7}$ at the 15th epoch; batch size: 2 (Darknet), 3 (MobileNetV2). The utilized computer system consists of an Nvidia Geforce GTX 1080Ti GPU with 11 GB memory and an Intel i7-7700K eight-core CPU.

B. DirtNet Adaptations

The first experiment demonstrates the impact of all improvement steps as introduced in Sec. III. Results on the 10 class detection task in the real test set are summarized in Tab. I. The rectangular network input speeds up inference time while all other enhancements on the Darknet53 backbone increase mAP incrementally adding up to a total mAP improvement of 7.2%. The network obtained from all optimizations until here is termed DirtNet-D. The introduction of the significantly smaller MobileNetV2 and separable convolutions reduces the computational complexity further by more than factor 12 at only 4.8% lower mAP. Some of the lost mAP performance is recovered with the remaining 2 optimizations. The resulting network is called DirtNet-M.

Tab. II shows that the series of smaller improvements is not specific to the dirt detection setting but also leads to an increased performance on the Pascal VOC 2007 test set for object detection surpassing the baseline YOLOv3 implementation by 4.5% in mAP. All reported results originate from training with the 2007 and 2012 training sets.

TABLE I

Improvement steps over the basic YOLOv3 implementation measured on test set of the 10-class dirt detection problem.

Improvement	mAP	Δ	∑ Δ	BFLOPs
baseline [16] (with Darknet53)	73.5	0.0	0.0	396.66
+ rectangle training and infer.	73.7	+ 0.2	+ 0.2	322.29
+ thresholds for diff. layers	74.2	+ 0.5	+ 0.7	322.29
+ GIOU loss [19]	75.0	+ 0.8	+ 1.5	322.29
+ focal loss [20]	76.5	+ 1.5	+ 3.0	322.29
+ warm up, cos learn rate [21]	77.9	+ 1.4	+ 4.4	322.29
+ mixup [22]	79.5	+ 1.6	+ 6.0	322.29
+ label smoothing [23]	80.7	+ 1.2	+ 7.2	322.29
+ backbone MobileNetV2 [24]	76.0	- 4.7	+ 2.5	89.35
+ separable convolution [25]	75.9	- 0.1	+ 2.4	26.68
+ redesign layers	77.0	+ 1.1	+ 3.5	21.49
+ remove predefined anchors	77.1	+ 0.1	+ 3.6	21.45

TABLE II

Performance comparison on Pascal VOC 2007 test set, trained on Pascal VOC 2007 and 2012, mAP@IoU=0.5.

Detector	Faster R-CNN [12]	SSD [13]	YOLOv3 [14], [28]	DirtNet-D	DirtNet-M
mAP	73.2	76.8	79.6	**84.1**	79.0

C. *Influence of synthetic training set size*

Since the dataset blender introduced in Sec. IV may deliver any desired amount of training data it needs to be examined where the sweet spot between dataset size, and hence training time, and achievable test set performance is located. In this experiment, 5 differently sized training datasets were synthesized and evaluated on all three test sets on the 2 class task. For each training set the maximal number of training epochs was adapted accordingly to yield a constant total number of images presented to the learning algorithm.

The results in Tab. III indicate that a training set size of only 9248 images is the optimal or close-to-optimal choice over all test sets and backbones and was hence chosen throughout this work. Larger datasets do not improve mAP anymore since the visual variance of the provided dirt and object samples seems to be mostly covered by 9248 images already: each image contains 11 samples of dirt or objects on average, yielding about 100,000 training samples in 9248 images. As there are almost 800 dirt and object samples in the training set, each item is represented 128 times in the dataset on average (i.e. 6 times per floor type). We assume that an increased performance can only be achieved by adding more dirt and object samples to the training set.

Another observation is that the results are very consistent between the small Test (real images) and the much larger TestSynth (synthetic images) sets, which contain different images of the same samples and floors. This supports the validity of training both DirtNets only with sufficiently many synthesized images and testing with a relatively small real world test set. Thus, data collection for dirt detection tasks can be significantly simplified and sped up. Likewise, the results obtained on the TestSynthTF (synthetic images) are not better than for the other test sets although this test set contains images of known floor types. This finding proves the

TABLE III

Influence of training set size on the performance in the three test sets on the 2-class dirt detection task.

Backbone	Dataset size (# images)	TestSynthTF mAP	TestSynth mAP	Test mAP
Darknet53	1156	81.4	81.6	83.7
	4624	82.4	83.2	85.1
	9248	83.9	**84.6**	**85.5**
	23120	**84.5**	**84.6**	84.1
	46240	82.1	83.0	83.2
MobileNetV2	1156	78.1	78.9	79.6
	4624	79.7	**82.3**	82.2
	9248	79.4	81.6	**83.8**
	23120	**80.7**	81.9	82.0
	46240	79.9	81.3	82.1

generalization ability of the DirtNets from a representative selection of floors to new and untrained floor types.

D. *Dirt Detection Performance*

Having motivated the design of the method and the training procedure in the previous experiments, here we summarize the results over all evaluated conditions as shown in Tab. IV, Fig. 7 and Fig. 8. The performance metrics on classifying 10 classes are lower than for 2 classes as to be expected. Moreover, while all metrics are quite consistent over the three test sets in the 2 class setting, the real test set has significantly better results than the others for the 10 class task. This effect is caused by the small size of set Test and the larger variance in the 10-class problem because some object classes like *other* and *keys* have a high variance in appearance but too few samples in the training set for modeling this variance well. This is especially visible for class *other* in Fig. 8. The performance on other classes remains quite stable over all test sets, though. For confirming this claim, the reader may have a look at the detailed performance tables which are available online[2].

The inference time on an Nvidia Geforce GTX 1080Ti is 70.6 ms for DirtNet-D and 33.1 ms for DirtNet-M which is both fast enough for online inference on a moving cleaning machine. If no GPU is available on the mobile robot, DirtNet-M can be converted to a CPU model using OpenVino [29] which conducts inference within 38.8 ms on an Intel Core i7-7700K CPU @ 4.20GHz with 8 cores. Further experiments showed that an Nvidia Jetson could run DirtNet-D with ca. 800 ms inference time and DirtNet-M with ca. 300 ms when operating on images of size 640×480 pixels. It is noteworthy that DirtNet-D has a model size of 235 MB whereas DirtNet-M requires only 26 MB. Exemplary detection results are visualized in Fig. 1, Fig. 10, and in the accompanying video.

E. *Comparison with State-of-the-Art Dirt Detection Methods*

The dirt detection performance is compared to our previous spectral residual method [6] and to the GMM1-16 approach of [3] using another synthesized test set which only contains dirt samples as both previous methods cannot

[2] `http://wiki.ros.org/ipa_dirt_detection`

TABLE IV

DIRTNET PERFORMANCE FOR THE 2 AND 10 CLASSES TASKS ON THE THREE TEST SETS USING DARKNET53 OR MOBILENETV2 AS BACKBONE.

		TestSynthTF				TestSynth				Test				time
Classes	Backbone	mAP	recall	precision	f1	mAP	recall	precision	f1	mAP	recall	precision	f1	(GPU)
2 classes	DirtNet-D	83.9	86.6	83.3	84.9	84.6	87.3	77.3	81.7	85.5	88.8	78.3	83.2	70.6 ms
	DirtNet-M	79.4	83.0	83.4	83.2	81.6	84.8	82.2	83.5	83.8	86.3	82.8	84.5	33.1 ms
10 classes	DirtNet-D	73.9	79.7	75.2	76.0	74.4	80.2	73.4	75.3	80.7	84.0	78.6	79.4	70.6 ms
	DirtNet-M	67.5	75.3	71.0	71.8	68.4	76.8	70.5	72.5	77.1	80.9	75.9	77.5	33.1 ms

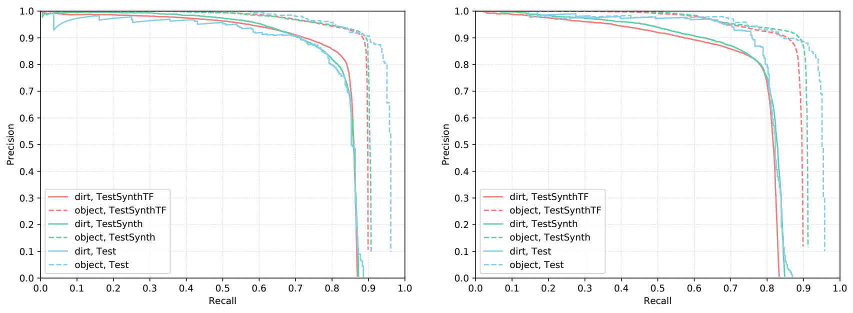


Fig. 7. Precision recall diagrams on all three test sets in the 2 class task for Darknet53 (left) and MobileNetV2 (right) as backbone.

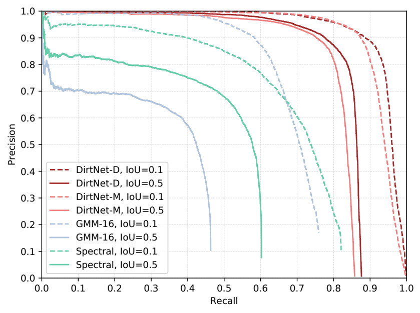


Fig. 9. Comparison of different dirt detection approaches evaluated for IoU=0.1 and IoU=0.5.

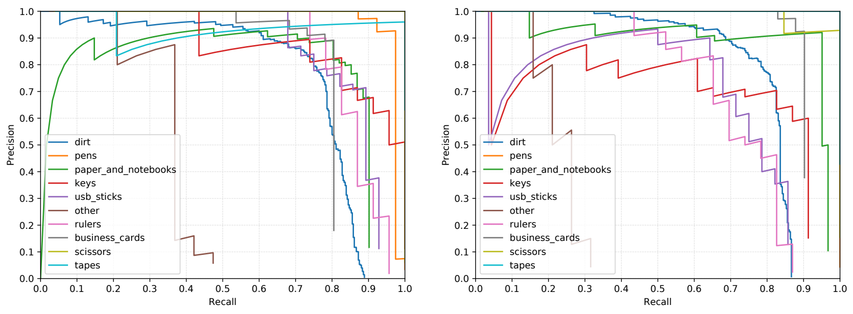


Fig. 8. Precision recall diagrams on the real test set in the 10 class task for Darknet53 (left) and MobileNetV2 (right) as backbone. It becomes visible that class *other* is obviously insufficiently represented in the training set.

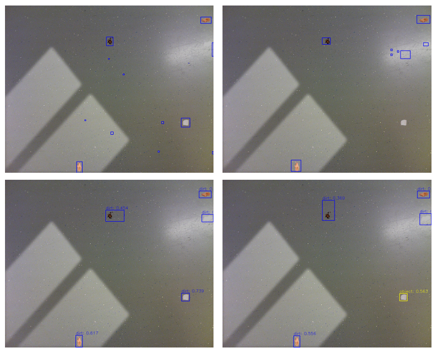


Fig. 10. Exemplary dirt detection results for the spectral and GMM method (upper row) and DirtNet-D and DirtNet-M (lower row).

TABLE V

COMPARISON WITH PREVIOUS DIRT DETECTION SYSTEMS W.R.T MAP DETERMINED FOR IOU=0.1 AND IOU=0.5, AND COMPUTATION TIME.

mAP for	Spectral [6]	GMM [3]	DirtNet-D	DirtNet-M
IoU=0.1	66.3	68.2	**94.5**	92.8
IoU=0.5	45.4	30.2	**83.7**	80.9
comp. time	**8.2 ms**	161.5 ms	70.6 ms	33.1 ms

distinguish objects from dirt. The results are summarized in Tab. V and precision recall graphs are provided in Fig. 9. The DirtNets are those trained on the 2 class task, i.e. the reported performance could even be higher if the DirtNets would have been trained solely for detecting dirt spots and no objects. Still, DirtNet-D and DirtNet-M achieve much better mAP and are up to 5 times faster compared to GMM. Spectral is the fastest method because of its simplicity, but both spectral and GMM degrade quickly with increasing IoU and suffer from lower achievable recall and higher numbers of false positives. Fig. 10 visualizes typical results of all four methods on a synthesized test image. Typical false positives are visible for the spectral and GMM approaches.

VI. CONCLUSIONS AND FUTURE WORK

This paper has introduced a useful dataset blending tool for generating unlimited labeled training data from limited amounts of real example footage as well as DirtNet, an adapted YOLOv3 detector for dirt and objects on arbitrary floor types. The presented system surpasses the previously utilized approach [6] and state-of-the-art [3] significantly in detection performance and speed, and can be well-utilized as onboard detection system for real cleaning machines. We presented experiments that proved the applicability and validity of training our system solely on the basis of synthesized images.

Future work should aim at collecting an even larger dataset than the one presented in this work for boosting the detection performance and robustness further. Additionally, an online learning framework could be developed which collects data from numerous cleaning machines operating in the field to enhance recognition models with new types of dirt and objects automatically. Moreover, the detector could be enhanced with motion-based cues which should provide notably different features that the smoother background.

References

[1] J. Hess, M. Beinhofer, and W. Burgard, “A probabilistic approach to high-confidence cleaning guarantees for low-cost cleaning robots,” in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[2] R. Bormann, J. Hampp, and M. Hägele, “New Brooms Sweep Clean - An Autonomous Robotic Cleaning Assistant for Professional Office Cleaning,” in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, May 2015.

[3] A. Grünauer, G. Halmetschlager-Funek, J. Prankl, and M. Vincze, “The Power of GMMs: Unsupervised Dirt Spot Detection for Industrial Floor Cleaning Robots,” in *Proceedings of the Conference Towards Autonomous Robotic Systems*. Springer, 2017, pp. 436–449.

[4] “Baker project homepage,” https://www.baker-projekt.de.

[5] R. Bormann, J. Fischer, G. Arbeiter, F. Weißhardt, and A. Verl, “A visual dirt detection system for mobile service robots,” in *Proceedings of the 7th German Conference on Robotics (ROBOTIK 2012)*, Munich, Germany, May 2012.

[6] R. Bormann, F. Weisshardt, G. Arbeiter, and J. Fischer, “Autonomous dirt detection for cleaning in office environments,” in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2013, pp. 1252–1259.

[7] U. Jost and R. Bormann, “Water streak detection with convolutional neural networks for scrubber dryers,” in *Proceedings of the 12th International Conference on Computer Vision Systems*, 2019.

[8] R. Bormann, F. Jordan, W. Li, J. Hampp, and M. Hägele, “Room Segmentation: Survey, Implementation, and Analysis,” in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2016.

[9] R. Bormann, F. Jordan, J. Hampp, and M. Hägele, “Indoor coverage path planning: Survey, implementation, analysis,” in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[10] X. Hou and L. Zhang, “Saliency Detection: A Spectral Residual Approach,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007, pp. 1–8.

[11] M. H. Debidatta Dwibedi, Ishan Misra, “Cut, paste and learn: Surprisingly easy synthesis for instance detection.” in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2017.

[12] 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 (NIPS)*, 2015, pp. 91–99.

[13] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, “Ssd: Single shot multibox detector,” in *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer, 2016, pp. 21–37.

[14] J. Redmon and A. Farhadi, “YOLOv3: An Incremental Improvement,” *arXiv e-prints*, p. arXiv:1804.02767, Apr 2018.

[15] J. Xu, “Visual Dirt Detection for Professional Office Cleaning Robots,” Master’s thesis, Technical University of Munich, 2018.

[16] “Ultralytics YOLOv3 github repository,” https://github.com/ultralytics/yolov3/, accessed: 2019-09-14.

[17] 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,” in *NIPS Autodiff Workshop*, 2017.

[18] T.-Y. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie, “Feature pyramid networks for object detection,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 936–944.

[19] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese, “Generalized intersection over union,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[20] T.-Y. Lin, P. Goyal, R. B. Girshick, K. He, and P. Dollár, “Focal loss for dense object detection,” in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2017, pp. 2999–3007.

[21] I. Loshchilov and F. Hutter, “Sgdr: Stochastic gradient descent with warm restarts,” in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.

[22] H. Zhang, M. Cissé, Y. Dauphin, and D. Lopez-Paz, “mixup: Beyond empirical risk minimization,” in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.

[23] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the inception architecture for computer vision,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 2818–2826.

[24] M. Sandler, A. G. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, “Mobilenetv2: Inverted residuals and linear bottlenecks,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 4510–4520.

[25] F. Chollet, “Xception: Deep learning with depthwise separable convolutions,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 1800–1807.

[26] P. Hu and D. Ramanan, “Finding tiny faces,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 1522–1530.

[27] J. Redmon and A. Farhadi, “Yolo9000: Better, faster, stronger,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 6517–6525.

[28] “Tencent YOLOv3 github repository,” https://github.com/TencentYoutuResearch/ObjectDetection-OneStageDet/tree/master/yolo, accessed: 2019-09-14.

[29] “OpenVino Toolkit,” https://software.intel.com/en-us/openvino-toolkit, accessed: 2019-09-14.