A DCNN-based Arbitrarily-Oriented Object Detector for a Quality-Control Application

Kai Yao, Alberto Ortiz, Francisco Bonnin-Pascual
Department of Mathematics and Computer Science, University of the Balearic Islands, Spain
yaokaimallorca@gmail.com, {alberto.ortiz, xisco.bonnin}@uib.es

***Abstract*—Following the success of machine vision systems for on-line automated quality and process control, in this paper we describe an object recognition solution aiming at detecting the presence of quality control elements in surgery toolboxes prepared by the Sterilization Unit of a hospital. Our solution actually consists in a two-stage arbitrarily-oriented object detection method making use of indirect regression of oriented bounding boxes parameters. The paper describes the design process and reports on the results obtained up to date.**

***Index Terms*—Quality Control, Object Recognition, DCNN**

I. Introduction

Machine vision systems are emerging as more and more popular solutions for on-line automated quality and process control applications. Enabling non-contact, thus non-destructive inspection, optical techniques are especially well suited when the correct manipulation of the object under inspection is crucial. This is precisely the inspection problem that we deal with in this paper: it consists in the detection of a number of control elements that are placed in boxes and bags containing surgical tools that surgeons and nurses have to be supplied with prior to starting surgery by the sterilization unit of the hospital. These elements provide evidence that the tools have been properly submitted to the required cleaning processes. Figure 1 illustrates, from left to right and top to bottom, the four kinds of elements to be detected: the label/bar code used to track a box/bag of tools, the yellowish seal, the paper tape which changes to the stripped appearance when the box/bag has been inside the autoclave, and an internal filter which is placed inside some boxes and creates the white-dotted texture that can be observed (instead of black-dotted).

In this work, we adopt Deep Convolutional Neural Networks (DCNN)-based methodologies as highly robust machine-learning approaches to face different lighting and inspection conditions. As already known, DCNNs have already shown good results for object recognition in images [1], although what is most interesting is the fact that they have shown highly promising performance for inspection applications [2]. In contrast to manually designed image processing solutions, DCNNs automatically generate powerful features, i.e. *learn the representation*, from training data by means of hierarchical learning strategies with a minimum of human interaction or expert process knowledge.

This work is partially supported by projects ROBINS (EU-H2020, GA 779776), PGC2018-095709-B-C21 (MCIU/AEI/FEDER, UE), and PROCOE/4/2017 (Govern Balear, 50% P.O. FEDER 2014-2020 Illes Balears)

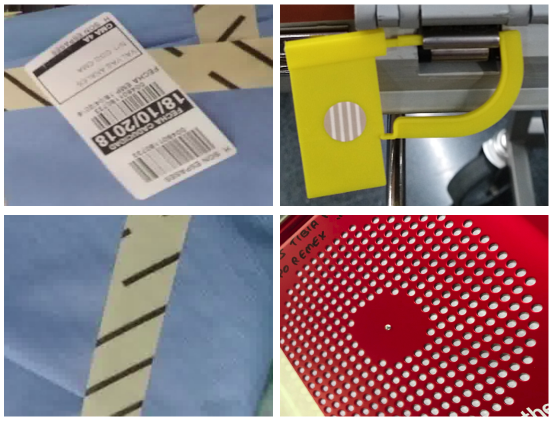
Figure 1. Objects to be detected: label, seal, paper tape and internal filter. In the same image, one can find several of these items, e.g. top-left case.

In more detail, this paper proposes a two-stage arbitrarily-oriented detector based on SSD (Single Shot MultiBox Detector) [3]. The main contributions are as follows: (1) we design a two-stage arbitrarily-oriented multi-category object detector, which can successfully operate in a dense and complicated scene; (2) unlike SSD, we select default bounding boxes by means of an automatic clustering procedure to obtain high-quality priors and improve object localization accuracy; and (3) we propose a new method for oriented bounding boxes regression.

II. Detector Overview

The detector that we propose in this section comprises two stages: in the first stage, we make use of a fine-tuned version of SSD [3] to regress straight bounding boxes containing the objects of interest. This version of SSD employs a set of prior boxes configurations determined after a clustering analysis on the training dataset (unlike the original algorithm), to focus the search on relevant bounding boxes and improve on object localization performance; in the second stage, from the output of the first stage, we regress the parameters of a rotated bounding box maximally contained into one of the straight bounding boxes. For a start, we first describe the parameterization we employ for the two kinds of bounding boxes which are handled. The two stages are described next.

A. Bounding Boxes Parameterization

Figure 2[top,left] illustrates the way how bounding boxes are parameterized for this application. On the one hand, the

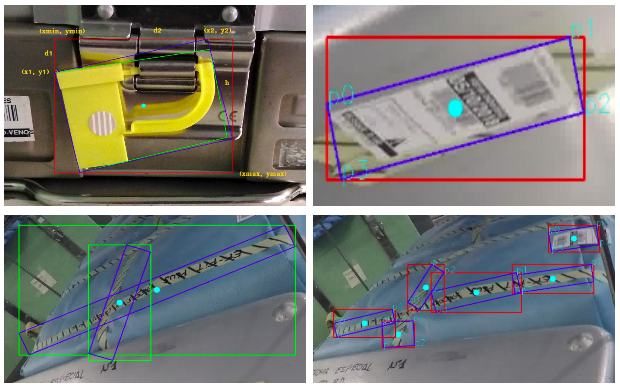

Figure 2. (top) Parameterization of bounding boxes: [left] straight and oriented boxes, [right] order of vertices for rotated boxes. (bottom) Ground truth for dataset A [left] and B [right]. (Better seen in colour and zooming.)

yellow lines describe a 4-side polygon minimally enclosing the object, from which the minimal rotated rectangle/bounding box is generated, indicated by the violet lines in the figure. A minimal straight bounding box is finally obtained from the rotated bounding box, depicted through the red lines. The latter is parameterized by the anchor point coordinates (c_x, c_y) and the box size (w_b, h_b), as usual, while rotated boxes are described by the intersects (d_1, d_2) of the upper side of the rotated rectangle with the sides of the unrotated rectangle. Optionally we also add parameter h to choose one of the two possible rectangles which may arise from tuple (d_1, d_2). Besides, we also define a clockwise order onto the four corners of the rotated box (Fig. 2[top,right]). The aforementioned has been used to generate the ground truth necessary during training.

Also, as part of the ground truth, and for testing purposes, we have defined two different datasets as for the boxes associated to the objects of every image. This is illustrated through Fig. 2[bottom]. On the one hand, *dataset A* [left] defines one box for every object of the training image. Although this seems natural for relatively squared objects, such as the label and the seal, it is not as straightforward for the paper tape, because of its elongated shape, and hence we define *dataset B* [right] which splits the object in several parts to favor a better training and latter detection of this kind of objects.

B. Straight Bounding Boxes Regression

1) Base method: For regressing the unrotated bounding boxes containing the objects of interest, we make use of SSD. This algorithm predicts category scores and box offsets for a fixed set of default bounding boxes using a set of convolutional filters applied to feature maps. As a base network, it makes use of a standard VGG16 network [1] whose fully connected layers have been replaced by a set of auxiliary convolutional layers progressively decreasing in size, thus enabling to perform predictions at multiple scales.

For a feature layer of size $m \times n$ with p channels, the basic element for predicting parameters of a potential detection is a $3 \times 3 \times p$ small kernel that produces either a score for a category, or a shape offset relative to the default box coordinates (c_x, c_y, w, h). At each of the $m \times n$ locations where the kernel is applied for multiple feature maps, SSD produces an output value, i.e. predicts offsets relative to the default box shapes in the cell $\Delta(c_x, c_y, w, h)$, as well as the per-class scores that indicate the presence of a class instance in each of those boxes. Specifically, for each box out of k at a given location, SSD computes c class scores and the 4 offsets relative to the original default box shape. This results in a total of $(c+4)k$ filters that are applied around each location in the feature map, yielding $(c+4)kmn$ outputs for an $m \times n$ feature map. Given the large number of boxes generated during a forward pass of SSD at inference time, it prunes most of the bounding boxes by applying *non-maximum suppression* to keep only the N top predictions.

To finish, the overall loss function is a weighted sum of the *localization loss* L_{loc} (implemented as a Smooth$_{\text{L1}}$ loss [4] between the predicted box l and the ground truth box parameters g for offset regression) and the *confidence loss* L_{conf} (implemented as a multi-category Softmax loss):

$$L(x, c, l, g) = \frac{1}{N}\left(L_{\text{conf}}(x, c) + \alpha L_{\text{loc}}(x, l, g)\right) \tag{1}$$

where x denotes the N final predictions as $x_{ij}^p = \{0, 1\}$ indicator functions for matching the i-th default box to the j-th ground truth box of category p. Parameter α is used to balance the impact of the classification and the bounding box regression loss values on the final loss value (α is finally set to 1 by cross validation). SSD thus starts with the priors as predictions and attempt to regress closer to the ground truth bounding boxes. The reader is referred to [3] for more details.

2) Prior boxes selection: SSD predefines a total of 6 default boxes per feature map location, i.e. scale, by imposing different size combinations (w_k, h_k) manually picked. Since, on the one hand, the shape of the correct bounding boxes can vary significantly and, on the other hand, SSD regresses the predicted bounding boxes from the prior boxes, a proper selection of default boxes becomes crucial for achieving a high detection success; as already noted in [5], such a proper selection contributes to the stability of the underlying optimization process, converges faster and improves effectively the *Intersection over Union* (IoU) between predicted and correct boxes. Hence, our object detector makes use of default boxes selected automatically in accordance to the available data.

In more detail, we run the well-known K-means algorithm over the bounding boxes belonging to the ground truth, using box width and height as the clustering features. Instead of the Euclidean distance, typically used by K-means implementations, we define IoU as a distance metric because the former tends to miss large bounding boxes. The distance between a sample box b_i and the cluster centroid c_j is hence defined as:

$$d(b_i, c_j) = 1 - \text{IoU}(b_i, c_j) = 1 - \frac{o(b_i, c_j)}{a(b_i) + a(c_j) - o(b_i, c_j)} \tag{2}$$

where $o(\cdot, \cdot)$ denotes area overlap and $a(\cdot)$ denotes area.

Table I shows averages of the IoU metric for hand-picked default boxes and automatically selected boxes by clustering,

Table I
AVERAGE IOU (AVIOU) OF DEFAULT BOXES VS SELECTION APPROACH

Dataset	Approach	# def. boxes	AvIoU (%)
	Hand-Picked	4	35.39
	Hand-Picked	5	38.53
	Hand-Picked	6	51.81
A	Hand-Picked	10	60.24
	Clustering	4	62.81
	Clustering	5	67.75
	Clustering	6	68.90
	Hand-Picked	4	30.03
	Hand-Picked	5	32.09
	Hand-Picked	6	51.55
B	Hand-Picked	10	61.36
	Clustering	4	65.64
	Clustering	5	66.46
	Clustering	6	67.25

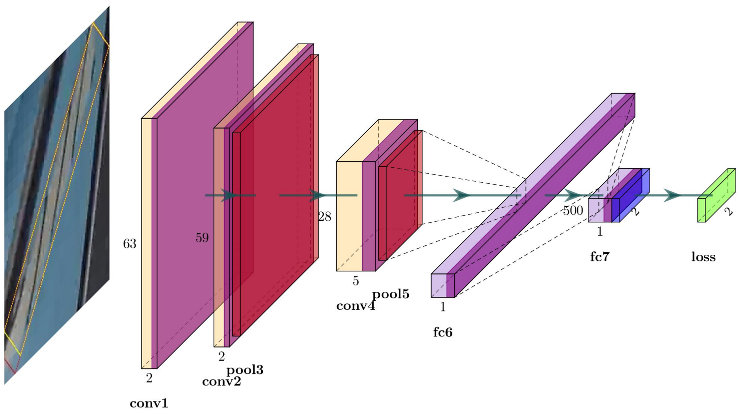

Figure 3. Architecture of the network for rotated bounding boxes regression.

for both datasets A and B and a different number of default boxes (for the hand-picked cases, we predefine the boxes similarly to SSD). We can see that 4 clusters automatically selected yields better performance than 10 hand-picked default boxes. This means that we propose high-quality and better parameterized default boxes. As could be expected, the more clusters, the better is the performance (the trend can be observed to continue for # def. boxes $\geq$ 7), although the number of clusters should not be high to keep reasonable the running time.

C. Arbitrarily-oriented Bounding Boxes Regression

For the second stage of the detector, i.e. regression of oriented bounding boxes parameters, we consider a specifically designed lightweight CNN, since the performance resulting from existing pre-trained networks, e.g. ResNet [6], VGG16 [1], etc. did not reach the desired level.

This network (see Fig. 3), although inspired by a LeNet architecture [7], presents several differences: (1) the input size is 63×63 after incorporating an additional convolutional layer at the beginning of the network, in order to avoid reducing the image to LeNet's 28×28 pixels, which means loosing too much information; (2) a combination of batch normalization and scaling layers has been added after each convolutional and innerproduct layer, before the ReLU activation layers, in order to decrease the effect of covariate shift from hidden layers [8]; (3) since the bounding box parameters (d_1, d_2, h) are values between 0 and 1, a sigmoid layer lies between the last fully connected layer and the loss layer; and (4) the final layer is an Euclidean loss layer:

$$L(d,g) = \frac{1}{2N} \sum_{i \in N} (\|d_1^i - g_1^i\|_2^2 + \|d_2^i - g_2^i\|_2^2 + \|d_h^i - g_h^i\|_2^2) \quad (3)$$

where d denotes the predicted offsets and height, g denotes the ground truth, and N is the size of the minibatch.

III. EXPERIMENTAL RESULTS

The dataset we have employed comprises 461 original images in total, which has been augmented, as usual, by means of rotation, scaling and mirroring. The dataset has been next split as 2/3 of the dataset for training and the remaining 1/3 for testing. All experiments have been conducted in Caffe, running in a PC fitted with an NVIDIA GeForce GTX 1080 GPU, a 2.9GHz 12-core CPU with 32 Gb RAM and Ubuntu 64-bit. In general, to distinguish true positives from false positives, a threshold for IoU between ground truth and predicted bounding boxes is set to 0.5, as usual. Standard *recall* (R), *precision* (P), *mean average precision* (mAP) and *average of IoU* (AvIoU) performance metric values [9] are computed for quantitative evaluation and comparison.

On the one hand, we report detection results for SSD using the default boxes automatically defined by the method described in Section II-B2. For efficiency reasons, we have considered clusterings in 4, 5 and 6 clusters. Table II shows the performance data for datasets A and B, highlighting the best results (A or B) in red. Several observations can be made from the previous table: (1) for dataset A, best performance is obtained for 5 default boxes, while for dataset B, performance is better for 6 default boxes; (2) results for dataset B are better not only for paper tape detection, as expected, but in general,

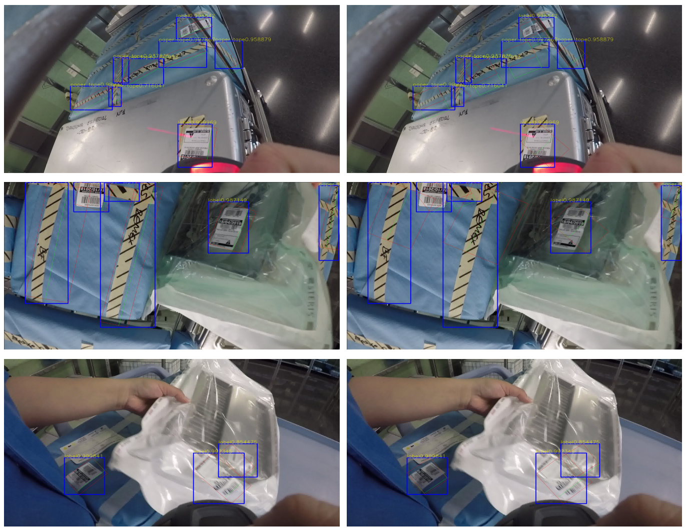
Figure 4. Detection results for (left) our method and (right) TextBoxes++. (Better seen in colour and zooming to see the predicted red/green boxes.)

Table II
PERFORMANCE RESULTS FOR UNORIENTED OBJECT DETECTION

Datas.	#	Class	R	P	mAP	AvIoU
A	4	Label	0.7224	0.9425	0.7185	0.8089
		Seal	0.3018	0.8421	0.2935	0.7299
		Paper tape	0.4800	0.9230	0.4730	0.7789
		Intl. filter	0.8181	0.9000	0.7822	0.7032
		Average	0.5806	0.9019	0.5688	0.7550
	5	Label	0.9030	0.9716	0.8986	0.8308
		Seal	0.9622	0.9444	0.9539	0.7719
		Paper tape	0.6400	0.9411	0.6272	0.7890
		Intl. filter	0.7727	0.7727	0.7349	0.7022
		Average	0.8195	0.9074	0.8036	0.7735
	6	Label	0.9207	0.9631	0.9152	0.8040
		Seal	0.8113	0.8958	0.7829	0.7302
		Paper tape	0.5661	0.8975	0.5496	0.7502
		Intl. filter	0.7727	0.7222	0.6375	0.7004
		Average	0.7677	0.8696	0.7213	0.7462
B	4	Label	0.9637	0.9871	0.9627	0.8781
		Seal	0.9693	0.9813	0.9687	0.8609
		Paper Tape	0.8742	0.9704	0.8804	0.8406
		Intl. Filter	0.8783	1.0000	0.8783	0.8928
	Average		0.9214	0.9847	0.9225	0.8681
	5	Label	0.9306	0.9870	0.9298	0.8460
		Seal	0.8775	0.9772	0.8775	0.8112
		Paper Tape	0.8948	0.9383	0.8893	0.7699
		Intl. Filter	0.9047	0.9500	0.8714	0.8120
	Average		0.9019	0.9631	0.8522	0.8098
	6	Label	0.9721	0.9872	0.9729	0.8861
		Seal	0.9631	0.9751	0.9618	0.8599
		Paper Tape	0.8831	0.9742	0.8879	0.8486
		Intl. Filter	0.9054	1.0000	0.9054	0.8927
	Average		0.9309	0.9841	0.9320	0.8719

Table III
PERFORMANCE RESULTS FOR ORIENTED OBJECTS DETECTION

Method	Class	R	P	mAP	AvIoU
2 param.	Label	0.8204	0.8445	0.7633	0.6649
	Seal	0.7346	0.8000	0.6254	0.6451
	Paper tape	0.5257	0.6071	0.4080	0.5517
	Intl. Filter	0.5454	0.6667	0.3916	0.5142
	Average	0.6566	0.7295	0.5471	0.5940
3 param.	Label	0.6571	0.6764	0.5505	0.5718
	Seal	0.6326	0.6889	0.4810	0.5480
	Paper tape	0.2422	0.2797	0.1254	0.3898
	Intl. Filter	0.3809	0.4210	0.2208	0.3888
	Average	0.4782	0.5165	0.3444	0.4746
AlexNet	Label	0.5387	0.5546	0.3752	0.5107
	Seal	0.6530	0.7111	0.5064	0.5578
	Paper tape	0.1695	0.1803	0.1063	0.3624
	Intl. Filter	0.5238	0.5789	0.4556	0.5289
	Average	0.4713	0.5062	0.3684	0.4900

for all classes (10% increase). All experiments have been run using Adam as network optimizer in Caffe, with a maximum number of 2×10^5 iterations and a fixed learning rate equal to 10^{-5}.

On the other hand, Table III shows results for oriented object detection, using dataset B and 6 clusters, accordingly to the previous results. In the comparison, we consider the loss function of (3) for both 2 terms, i.e. (d_1, d_2), and 3 terms, i.e. (d_1, d_2, h). Additionally, we include results for AlexNet as a baseline, properly tuned for the detection problem at hand. Observing the table, one can see: (1) regression of 2 terms yields better results in general than regression of 3 terms (around 20% in excess); (2) results for our network with 2 terms in the loss function outperforms the baseline network significatively (also around 20%); (3) in general, the paper tape is the class for which worst detection results are obtained. The Adam optimizer has also been employed in this case, with a learning rate equal to 0.01 and momentum at 0.9 for speeding network convergence up; during training, we decreased the learning rate by a factor 0.8 every 5000 iterations, for a maximum of 40000.

To finish, Fig. 4 shows, for three images, our method outperforming in a qualitative way TextBoxes++ [10], an arbitrarily-oriented text detector which we have also fine-tuned for the quality control application. As can be observed, test images were not taken under controlled conditions. (TextBoxes++ did not reach IoU ≥ 0.5 in our experiments, this is the reason why it is not included in Table III.)

IV. Conclusions and Future Work

A two-stage arbitrarily-oriented object detection method making use of indirect regression of oriented bounding boxes parameters has been described. Promising results have been reported. Future work is planned to focus on achieving pixel-level detection within a hybrid solution making use of the bounding box concept and semantic segmentation.

References

[1] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *arXiv:1409.1556*, 2015. [Online]. Available: http://arxiv.org/abs/1409.1556v6

[2] D. Weimer, B. Scholz-Reiter, and M. Shpitalni, "Design of deep convolutional neural network architectures for automated feature extraction in industrial inspection," *CIRP Annals*, vol. 65, no. 1, pp. 417–420, 2016.

[3] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single Shot MultiBox Detector," in *Proc. European Conference on Computer Vision*, 2016, pp. 21–37.

[4] R. Girshick, "Fast R-CNN," in *Proc. IEEE International Conference on Computer Vision*, 2015, pp. 1440–1448.

[5] J. Redmon and A. Farhadi, "YOLO9000: better, faster, stronger," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 7263–7271.

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *arXiv:1512.03385*, 2015. [Online]. Available: http://arxiv.org/abs/1512.03385

[7] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[8] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv:1502.03167*, 2015. [Online]. Available: http://arxiv.org/abs/1502.03167

[9] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge: A retrospective," *International Journal of Computer Vision*, vol. 111, no. 1, pp. 98–136, 2015.

[10] M. Liao, B. Shi, and X. Bai, "TextBoxes++: A single-shot oriented scene text detector," *IEEE Transactions on Image Processing*, vol. 27, no. 8, pp. 3676–3690, 2018.