https://doi.org/10.2352/ISSN.2470-1173.2021.8.IMAWM-281

Industrial defect detection by comparison with reference 3D CAD model

Deangeli G. Neves; Eldorado Research Institute; Campinas, SP, Brazil
Guilherme A. S. Megeto; Eldorado Research Institute; Campinas, SP, Brazil
Augusto C. Valente; Eldorado Research Institute; Campinas, SP, Brazil
Qian Lin; HP Inc; Palo Alto, California, USA

Abstract

In this work, we propose a method that detects and segments manufacturing defects in objects using only RGB images. The method can be divided into three different integrated modules: object detection, pose estimation and defect segmentation. The first two modules are deep learning-based approaches and were trained exclusively with synthetic data generated with a 3D rendering engine. The first module, object detector, is based on the Mask R-CNN method and provides the classification and segmentation of the object of interest as the output. The second module, pose estimator, uses the category of the object and the coordinates of the detection as input to estimate the pose with 6 degrees-of-freedom with an autoencoder-based approach. Thereafter it is possible to render the reference 3D CAD model with the estimated pose over the detected object and compare the real object with its virtual model. The third and last step uses only image processing techniques, such as morphology operations and dense alignment, to compare the segmentation of the detected object from the first step, and the mask of the rendered object of the second step. The output is an image with the shape defects highlighted. We evaluate our method on a custom test set with the intersection over union metric, and our results indicate the method is robust to small imprecision from each module.

Introduction

Vision-based automatic manufacturing defect inspection is a task that remains a challenge nowadays due to the complexity of detecting many defect types on a variety of materials and shapes. We can frame the defect detection in different categories of problems: classification, detection, and segmentation. For the first one, we input an image (RGB, greyscale, depth map) and then a classification model can be employed to assign a label (one for each defect) for the image. There is also the possibility of creating a pure imaging processing approach that computes features from the image and then checks if some of these features are correlated with manufacturing-defect. However, the defect location is important information that may be used as a clue to find problems during the manufacturing process, that is why the majority of approaches focus on detection and/or segmentation of the defect [1, 2, 3, 4]. Defect detection is an improvement over the classification-only approach because it provides both: defect label and defect bounding-box location. In addition, it can detect multiple defects in the same image since a label is assigned for each bounding box. The target of a segmentation approach is to assign a label for each defect pixel. Hence we can delimiter more precisely the defect and extract more detailed information regarding defects, such as shape, size and other.

There are some public datasets available for defect detection. The majority of them [5, 6] only contain 2D images (greyscale, RGB, depth) with a limited number of views for each type of material. We are assuming that this type of restriction is undesirable for an automated defect detection system because it must perform the detection robustly, regardless of the orientation of the object or camera. Also, create or expand a dataset is time-consuming and might be impractical in some scenarios. Our method proposes a solution that uses only synthetic data which is generated from 3D meshes avoiding human effort. To achieve that, we divide the defect inspection problem into three different steps: object detection, pose estimation and defect segmentation. The first two are deep learning-based approaches while the last one uses image processing techniques. To train the deep learning models we used synthetic data generated from 3D CAD models. Thus, our method can be fully extended to detect manufacturing defects in any manufacturing part. The results indicate that our method can perform manufacturing defect segmentation in near real-time with promising results.

Related work

Classical systems for defect inspection require a complex setup to work properly. This setup may be composed, for example, of depth sensors, lighting systems, X-ray, high-resolution RGB cameras, among others. To reach satisfactory results in these systems it is required a contrasting background or a careful calibration process that demands a significant amount of time and effort. For example, [7] created a stand-alone system which enclosure the object of interest in ”black box” with controlled lighting condition. In the black box, there is a mechanism used to detect the object inside the box and then an RGB image is taken by a high-resolution camera. Next, features are extracted from this image and a reference image (for the same object of interest) and after that, these features are matched to align the images. Since the images are aligned, the RGB images are binarized and then morphological operations are applied to segment the contour defect. However, the proposed method is not robust to the object 6-DoF (Degrees of Freedom) pose, since depends on a reference image to compute a linear transformation to align the input image to the reference one.

Other approaches employ a deep learning model to detect or segment defects from RGB images [1] or x-ray images [2]. However, those approaches require the creation of a training set that demands time, especially if it is desired to detect defects in many types of manufacturing parts. Our method is fully trained

with synthetic data that is generated from 3D meshes files for each object of interest.

In [8], the authors trained an autoencoder that given an RGB image of the object of interest, with or without defects, the model reconstructs the input with no defects. The authors were able to highlight defects in the input image (if they exist) analyzing the differences between the original one and its reconstruction. However, the proposed solution depends on the quality of reconstruction and a poor autoencoder reconstruction can result in many false positives. Thus, the solution may not generalize well in a scenario where many types of materials or when the input data mismatch a bit from the distribution of the training data. Our method can be extended for many scenarios since we use robust object detection to detect the object of interest and then we compute its 6-DoF.

Our method

As mentioned before, our method is separated into three modules: the first one is an object detector, the second one is a 6-DoF object pose estimator, and the last one is an image processing technique to detect discrepancies between two binary images. The dataset used to validate our method was the T-LESS [9]. We decided to select this dataset because of the following aspects:

- It contains textureless manufacturing parts
- High similarity among the manufacturing parts
- High degree of symmetry

Most of these aspects make this dataset particularly tough to train a deep learning model, that is why this dataset has not become so popular [10]. The following subsections explain our method. Firstly, we discuss about the synthetic data generation and how we trained the object detection/segmentation model using generated data; after that, we detail how we estimated the 6-DoF object pose from the detection output; then we discuss how using image processing to detect manufacturing defect in the input image; finally, we describe how to combine the three modules to perform the defect detection.

Synthetic data generation and object detector

In order to mitigate the domain shift [11] between the synthetic data and the real one, we must generate synthetic samples as realistic as possible. This means the synthetic data generation must take into account key aspects for satisfactory realism:

- high-quality textured objects
- variety of 3D environments (to generate a variety of backgrounds)
- photorealism
- physics (collision, gravity, etc)

We tested both OpenGL and VTK [12] rendering library; however, using Unreal Engine 4 (UE4) [13] we were able to achieve the necessary level of realism. Also, UnrealCV [14] was employed to automate the process of moving the camera and the objects of interest through the 3D environment. Two 3D environments were created, an office environment and a room, and for each one of them, sets of textures were created for all 3D objects in the environment. The transition between 3D environments was made manually while the entire process of acquisition of the RGB and label (mask) images was fully automated.

Once a 3D environment is selected, the training data is generated as follows:

1. Read list of objects of interest on 3D environment ($objs_{interest}$)
2. Read the list of surfaces (may be a table, chair, desktop, etc) where the objects from $objs_{interest}$ will be spawn on it ($surfaces_{spawn}$)
3. Define a pose set ($pose_{set}$)
4. Move all objects in $obj_{interest}$ outside of sigh from camera's point of view
5. For each obj in $objs_{interest}$ do
   (a) Move obj to the 3D environment
   (b) For each obj_{pose} in $pose_{set}$
      i. Select a random surface from $surfaces_{spawn}$
      ii. Spawn the object with the pose $object_{pose}$ on the selected surface
      iii. Generated a random displacement (dx, dy, dz) between the camera and the object (displacement) and update the camera position ($camera_{pos}$) from object position ($object_{pos}$). $camera_{pos} = object_{pos} + displacement$
      iv. Compute the camera rotation components to guarantee the camera points out to the $object_{pos}$
      v. Apply the camera translation and rotation components computed in (iii) and (iv) respectively
      vi. Save the RGB image and mask (labeled image) from current camera visualization
   (c) Move obj "far away" from 3D environment

In our approach we generated only one object per image, but we can change the algorithm to spawn multiple objects on the same surface. This way, we can train a model that is more robust to clutter and occlusion. In our training dataset we generated almost 31k pairs of images and masks. The Figure 1 shows some images from our synthetic dataset used to train a detection/segmentation model. We selected the Mask R-CNN model as the detector model. To train this DNN (Deep Neural Network) model, we used an open-source framework from Facebook Research [15].

Object 6-dof pose estimation

We used a method called "Augmented Autoencoder" (AAE) [16] to estimate the pose. This method consists of training an autoencoder to reconstruct only the object of interest from the input image. This autoencoder shall be robust to translation, rotation, occlusion, noise and other augmentations that can be applied on data.

After training the autoencoder model, we generate a set of predefined views for the object. To generate these predefined views we used the same approach described in the original paper which consists of centering the object and moving the camera across a spherical surface (Figure 2). We then compute the camera rotation to guarantee the camera is pointing to the object and take the screenshot.

Finally, it is required to generate the codebook which embeds

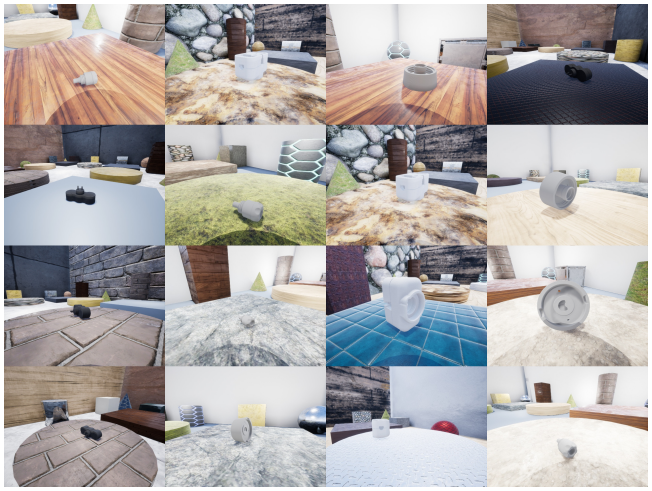


Figure 1. *Training images generated from 3D rendering. Best viewed in color. All 3D scenes was rendered using Unreal Engine 4 [13]*

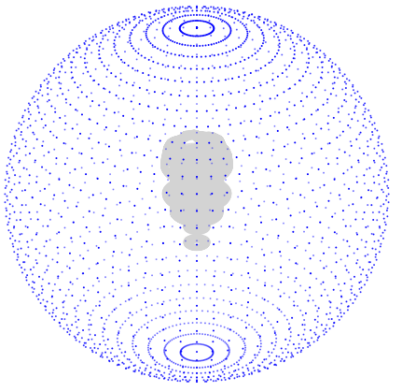


Figure 2. *Generating the predefined views for a given object. Each dot represents a possible camera location. source*

a triplet for each view: a latent code (encoder output), a rotation matrix and the original object bounding box (bounding box before the crop and resize). To estimate the pose for a new entry, we first compute the cosine distance between it and the other entries in the codebook. The pose from the closest entry in the codebook is assigned to the new entry.

To estimate the translation components, we used the same procedure described in [16]. It consists of first estimating depth from the scale ratio between the detected bounding box and the codebook scale and then using the bounding box center to estimate the translation in X and Y.

Detecting shape defect on manufacturing parts

In this work, we focus only on shape defect detection because, in our judgment, it is more crucial to industrial parts to have the correct shape when compared to surface defects such as scratches. The main idea of the method is to compare the silhouettes (masks) of the detection and the rendered object to highlight the differences that are possible defects (see Figure 3). The masks are binary images, with black pixels that represent the background and white pixels to represent all pixels that compound the object of interest.

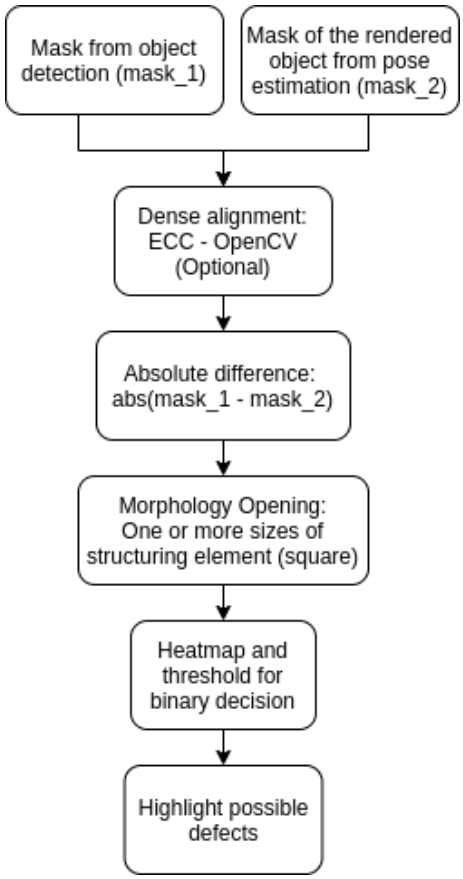


Figure 3. *Diagram of the pipeline for defect detection module*

One input of this module is the mask of the segmented object from the detection module. The other input is the mask of the rendered object from the pose estimation module. Both masks are binary images of the same size. To highlight the differences, the main idea is to take the absolute difference between the masks. The expected result should highlight shape defects (missing parts or additive material); however, misalignment between the masks (e.g. imprecision from the pose estimation) and imperfections from the segmentation (mainly in finer details) also may appear as possible defects.

It is expected that the estimated pose should be very close to the correct one. However, to deal with some imprecision and imperfections from the previous modules, we applied dense alignment to the masks of the objects. We used the ECC-based iterative image alignment algorithm [17] implemented in OpenCV [18] with a predefined low number of iterations (e.g. 10) to keep near real-time results. Note that this works as a refinement step, and it is not necessary if we are able to guarantee a nearly perfect pose estimate, which is usually not the case.

After computing the alignment, we compute the absolute difference between masks and apply the mathematical morphology opening operation over the resulting binary image. It is possible to specify one or more sizes (in pixels) for the structuring element (a square) to remove noise and small systematic imperfections keeping only the most significant differences that are considered

possible defects. When more than one size of the structuring element is specified, the result from the morphology opening for each size is accumulated like a heatmap. The maximum value in the heatmap is the total number of sizes for the structuring elements. A threshold is specified to create a binary solution, keeping only the greater differences and discarding the small ones. In our experiments, we empirically set the threshold as 70% of the maximum value of the heatmap.

Evaluation of the shape defect detection is not trivial because the results depend on the outputs of the previous modules. The following example (Fig. 4) was taken when the estimated pose was near to the correct pose of the object, and, after that, the defect (additive material) was annotated manually. The evaluation was based on one of the most used metrics for segmentation, the Jaccard index also called intersection over union (IoU).

(a) Original image (b) Annotated defect

Figure 4. *Original image and defect annotation. Best viewed in color.*

The results in Figure 6 considered the structuring elements of sizes 3, 5, 7, 9, 11, 13 and 15, and a threshold of 70% of the maximum value from the heatmap to create a binary solution for defect/no-defect.

Combining the modules

The full pipeline of the method can be summarized in the following steps:

1. The trained object detector detects all objects of interest given an input image.
2. Each detected object is cropped and resized to a 128×128 RGB image.
3. For each crop, select the suitable codebook (a codebook per object of interest) to be used on pose estimator.
4. Given the estimated pose, render the 3D mesh at that pose; and then generate the binary mask from it.
5. Take the segmentation mask from the detector (step 1) and the mask from the rendered object (step 4) and use them to compute the manufacturing defect.

The Figure 5 illustrates the combination of all modules.

Evaluation

In our proposed pipeline, the quality of the defect detection depends on the first two modules, object detection and pose estimation. For this reason, we decided to evaluate each module separately.

Object detection

We trained a Mask R-CNN model using only synthetic data generated by the Unreal Engine 4 [13] 3D rendering engine.

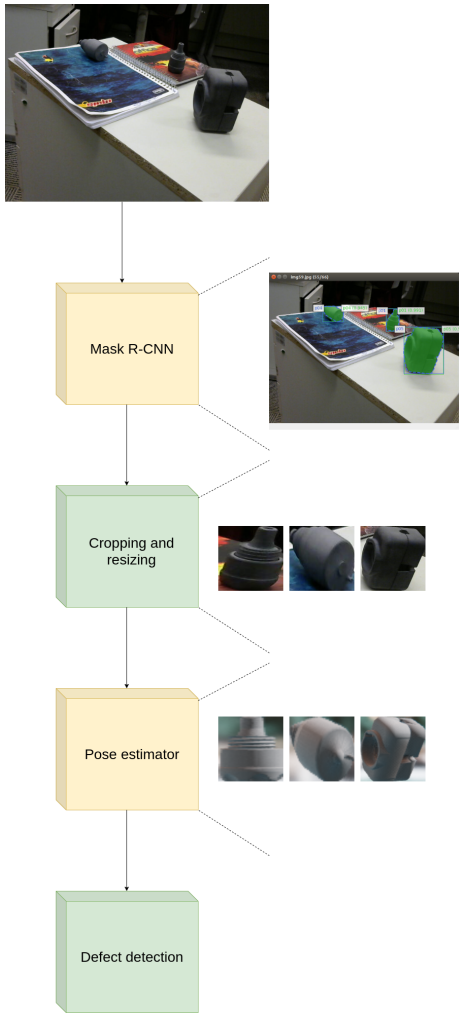


Figure 5. *An overview of the proposed method to detect shape defects. Best viewed in color.*

The chosen backbone for the network was a ResNeXt-101 [19] with Feature Pyramid Networks [20]. The generated training set contains 40380 images (1346 per object), and each image contains only one object. The batch size was set as 2 and the maximum number of iterations was 720000. We used Stochastic Gradient Descent with base learning rate as 1.25e−3 and weight decay of 1e−4. After all iterations, the best model was selected based on validation set performance.

The validation and test sets contain images of objects in the real world (i.e. no synthetic data). The test set contains 67 images with more than 100 annotated object instances, so there are images with more than one object per image in this set. The model achieved a score of 81% mean average precision for bounding box detection and instance segmentation.

Pose estimation

For the Augmented Autoencoder approach, we trained one pose estimation model per object. The metric used to evaluate the models was the ADI (Average Distance for Indistinguishable views) [21].

$$e_{ADI}(R,T,\hat{R},\hat{T},M) = \underset{x_1 \in M}{\text{avg}} \min_{x_2 \in M} \left\| (Rx_1 + T) - (\hat{R}x_2 + \hat{T}) \right\| \quad (1)$$

where x_1, x_2 are views from a set of indistinguishable views M. The matrices R and T are the expected rotation matrix and translation vector respectively, while $\hat{R}$ and $\hat{T}$ are the predicted ones. In summary, the equation 1 computes the difference between the closest 3D points from the expected transformation (R,T) and predicted one $(\hat{R},\hat{T})$. Thus low values of e_{ADI} indicate that the pose prediction is close to the expected one. As meshes can be quite distinct from each other, the scores were normalized by mesh diameter (distance between the two farthest points on mesh). The test set contains 1 265 annotated images with a similar process used to annotate the training set for T-LESS dataset. The test set contains 23 out of 30 from T-LESS meshes. The average score computed for the test set was 0.0737, which means the misalignment between two points is, in average, 7% of the object diameter. A score below 10% is usually considered a successful alignment [22]. The misalignment score for each object varies from 3% to 13%, which means the pose estimator did not perform satisfactorily for some objects.

Shape defect detection

We prepared a small test set with 12 images with ground truth to evaluate the shape defect detection module. We selected 7 images to show more representative results, but the results were similar for the entire test set. The images were annotated as in Fig. 4. Two aspects of this module were evaluated: i) the size of the structuring element, in pixels, of the morphology operations, and ii) the presence or not of dense alignment. The intersection over union (IoU) was the selected metric to evaluate the shape defect detection module.

For the sample in Fig. 4, the results in Fig. 6 considered the structuring elements of sizes 3, 9, 15, and the cumulative results for the series 3, 5, 7, 9, 11, 13, 15 with a threshold of 70% of the maximum value from the heatmap to create a binary solution for defect/no-defect.

The Fig. 7 shows the best result for each sample. The Table 1 shows the average results for the selected seven images, samples of Fig. 6 and Fig. 7. The best result is with alignment and the structuring element of size 9.

Average of intersection over union considering seven examples from Figures 4 and 7

	with alignment	no alignment
size 3	0.359	0.189
size 5	0.408	0.245
size 7	0.214	0.244
size 3-15	0.378	0.271

The results show that the dense alignment is crucial to have good results in most situations. As expected, as the size of the structuring element increases, finer details of the difference masks are lost, and this could result in failure to detect small defects. On the other hand, greater structuring element sizes could ignore all the differences. Therefore, a good trade-off is a medium

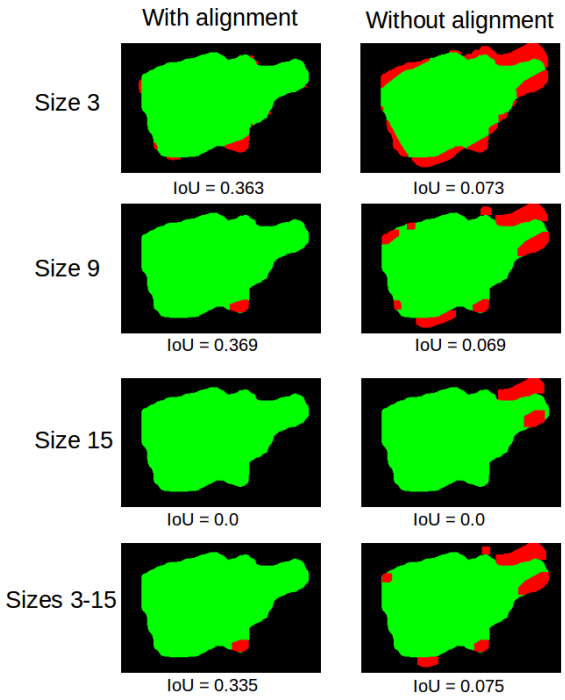


Figure 6. *Intersection over union for different sizes of structuring element of morphology operations and with and without the dense alignment for object of Fig. 4. Best viewed in color.*

structuring element size, which can suppress some misalignment noise and keep only possible defects. Furthermore, the proposed technique involving cumulative results and heatmap thresholding could smooth the output and provide good results for more diverse situations.

Conclusion

This work presented a method to automatically detect and segment shape defects on objects from an industrial context. The system is composed of three different modules: object detection, pose estimation and shape defect detection. Deep learning models were used in the object detection and pose estimation modules, and all of them were trained using only synthetic data generated by a 3D graphics engine. Our results indicate that the object detector successfully identifies the objects of interest in RGB images. The pose estimator, however, did not perform well enough for some objects, which indicates some improvements are required to better handle the symmetrical and texture-less objects. The shape defect detection module compares the binary masks from the segmented object and the reference 3D mesh to estimate the possible regions with defects. Our results indicate the method is robust enough to detect shape defects in manufacturing parts. Possible future work include extending the defect detection module to handle other types of imperfections such as surface scratches, tearing, and wrinkling.

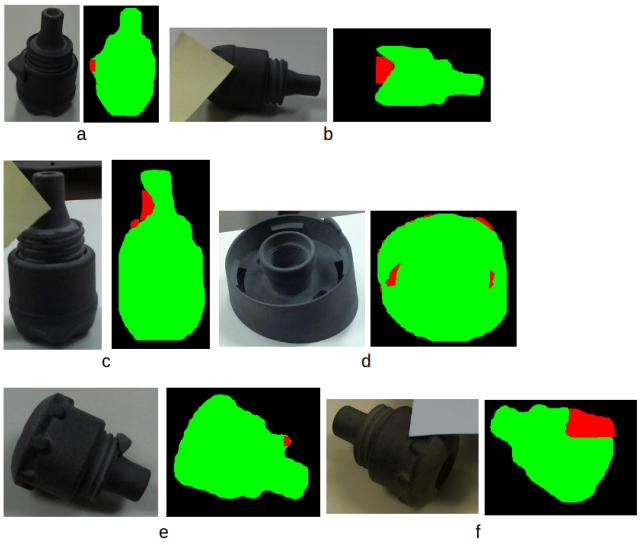


Figure 7. *Best results of IoU for some examples. a) With alignment, sizes 3–15, IoU=0.378. b) No alignment, size 15, IoU=0.553. c) With alignment, size 9, IoU=0.590. d) With alignment, sizes 3–15, IoU=0.308. e) No alignment, size 15, IoU=0.825. f) With alignment, size 3, IoU=0.203. Best viewed in color.*

References

[1] O. Essid, C. Samir, and L. Hamid, "Automatic detection and classification of manufacturing defects in metal boxes," *PLOS ONE*, vol. 13, October 2018.

[2] M. Ferguson, R. ak, Y.-T. Lee, and K. Law, "Detection and segmentation of manufacturing defects with convolutional neural networks and transfer learning," *Smart and Sustainable Manufacturing Systems*, vol. 2, pp. 137–164, 09 2018.

[3] D. Yapi, M. Mejri, M. S. Allili, and N. Baaziz, "A learning-based approach for automatic defect detection in textile images," *15th IFAC Symposium on Information Control Problems in Manufacturing*, vol. 48, no. 3, pp. 2423 – 2428, 2015.

[4] F. Zhou, G. Liu, F. Xu, and H. Deng, "A generic automated surface defect detection based on a bilinear model," *Applied Sciences*, vol. 9, p. 3159, August 2019.

[5] P. Bergmann, M. Fauser, D. Sattlegger, and C. Steger, "MVTec AD - A comprehensive real-world dataset for unsupervised anomaly detection," in *CVPR*, June 2019.

[6] "DAGM 2007 datasets," https://hci.iwr.uni-heidelberg.de/node/3616, accessed: 2019-12-17.

[7] H. Chen, Y. Cui, R. Qiu, P. Chen, W. Liu, and K. Liu, "Image-Alignment Based Matching for Irregular Contour Defects Detection," *IEEE Access*, vol. 6, pp. 68 749–68 759, 2018.

[8] P. Bergmann, S. Löwe, M. Fauser, D. Sattlegger, and C. Steger, "Improving unsupervised defect segmentation by applying structural similarity to autoencoders," *VISIGRAPP*, vol. 5, pp. 372–380, 2019.

[9] T. Hodaň, P. Haluza, Š. Obdržálek, J. Matas, M. Lourakis, and X. Zabulis, "T-LESS: An RGB-D dataset for 6D pose estimation of texture-less objects," in *WACV*, 2017.

[10] R. Kaskman, S. Zakharov, I. Shugurov, and S. Ilic, "HomebrewedDB: RGB-D dataset for 6D pose estimation of 3D objects," in *ICCVW*, April 2019.

[11] Y. Luo, L. Zheng, T. Guan, J. Yu, and Y. Yang, "Taking A closer look at domain shift: Category-level adversaries for semantics consistent domain adaptation," *CoRR*, vol. abs/1809.09478, 2018.

[12] W. Schroeder, K. Martin, and B. Lorensen, *The Visualization Toolkit*, 4th ed. Kitware, 2006.

[13] Epic Games, "Unreal engine." [Online]. Available: https://www.unrealengine.com

[14] W. Qiu, F. Zhong, Y. Zhang, S. Qiao, Z. Xiao, T. S. Kim, and A. Y. Yizhou Wang, "Unrealcv: Virtual worlds for computer vision," *ACM Multimedia Open Source Software Competition*, 2017.

[15] F. Massa and R. Girshick, "maskrcnn-benchmark: Fast, modular reference implementation of Instance Segmentation and Object Detection algorithms in PyTorch," https://github.com/facebookresearch/maskrcnn-benchmark, 2018, accessed: 2019-12-17.

[16] M. Sundermeyer, Z.-C. Marton, M. Durner, M. Brucker, and R. Triebel, "Implicit 3d orientation learning for 6d object detection from rgb images," in *ECCV*, September 2018.

[17] G. D. Evangelidis and E. Z. Psarakis, "Parametric image alignment using enhanced correlation coefficient maximization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 10, pp. 1858–1865, 2008.

[18] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[19] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *CVPR*, July 2017, pp. 5987–5995.

[20] T. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *CVPR*, July 2017, pp. 936–944.

[21] T. Hodaň, J. Matas, and Š. Obdržálek, "On evaluation of 6d object pose estimation," in *ECCVW*, 2016, pp. 606–619.

[22] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige, and N. Navab, "Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes," in *ACCV*, October 2012.

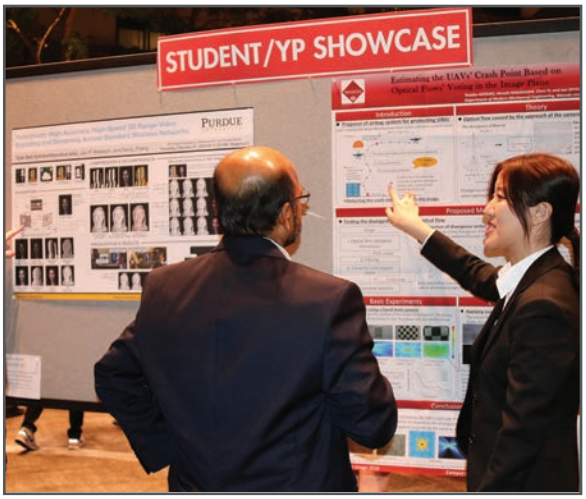
JOIN US AT THE NEXT EI!

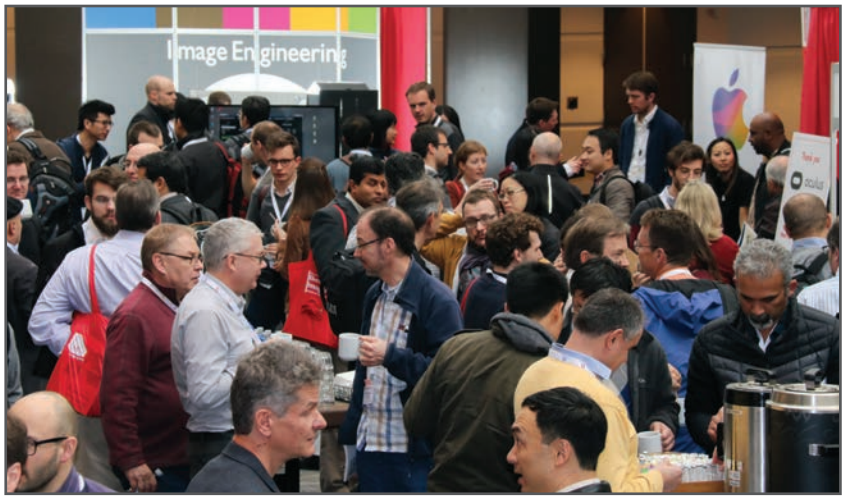
IS&T International Symposium on

Electronic Imaging

SCIENCE AND TECHNOLOGY

Imaging across applications . . . Where industry and academia meet!





• SHORT COURSES • EXHIBITS • DEMONSTRATION SESSION • PLENARY TALKS •
• INTERACTIVE PAPER SESSION • SPECIAL EVENTS • TECHNICAL SESSIONS •

www.electronicimaging.org

IS&T
imaging.org