A Versatile Crack Inspection Portable System based on Classifier Ensemble and Controlled Illumination

Milind G. Padalkar1, Carlos Beltrán-González1, Matteo Bustreo1,5, Alessio Del Bue2* and Vittorio Murino1,3,4*

{milind.padalkar, carlos.beltran, matteo.bustreo, alessio.delbue, vittorio.murino}@iit.it
1 Pattern Analysis and Computer Vision (PAVIS), Istituto Italiano di Tecnologia, Genova, Italy
2 Visual Geometry and Modelling (VGM), Istituto Italiano di Tecnologia, Genova, Italy
3 Ireland Research Center, Huawei Technologies Co., Ltd., Dublin, Ireland
4 Dipartimento di Informatica, University of Verona, Verona, Italy
5 Dipartimento di Ingegneria Navale, Elettrica, Elettronica e delle Telecomunicazioni, University of Genova, Italy

Abstract—This paper presents a novel setup for automatic visual inspection of cracks in ceramic tile as well as studies the effect of various classifiers and height-varying illumination conditions for this task. The intuition behind this setup is that cracks can be better visualized under specific lighting conditions than others. Our setup, which is designed for field work with constraints in its maximum dimensions, can acquire images for crack detection with multiple lighting conditions using the illumination sources placed at multiple heights. Crack detection is then performed by classifying patches extracted from the acquired images in a sliding window fashion. We study the effect of lights placed at various heights by training classifiers both on customized as well as state-of-the-art architectures and evaluate their performance both at patch-level and image-level, demonstrating the effectiveness of our setup. More importantly, ours is the first study that demonstrates how height-varying illumination conditions can affect crack detection with the use of existing state-of-the-art classifiers. We provide an insight about the illumination conditions that can help in improving crack detection in a challenging real-world industrial environment.

I. INTRODUCTION

Visual inspection is an important step for ensuring quality of several industrial components. This is especially necessary in the case where a component requires maintenance and its failure could be catastrophic. In such a case, visual inspection is often performed by a human expert that is responsible for identifying the defective parts and suggesting their replacement, which they carry out based on their training and experience. Such a visual inspection process involves careful assessment of large number of parts, only few of which could present defects. Automating the visual inspection process can help in the assessment of large number of trivial cases, while the experts can be referred to only the non-trivial ones. Examples of such applications include identifying defects in casted steel [1], inspection of metallic components in nuclear power plants [2], detecting cracks in tiles that could be part of a leak-proof compartment, examining flaws in concrete structures [3], etc. Automating visual inspection in such applications involves acquiring digital images (or videos) of the areas to be inspected followed by defect detection using computer vision based algorithms.

Historically, the algorithms designed to detect defects like cracks followed a pipeline which typically involved contrast enhancement, edge linking and refinement [4], [5], [6], [7]. The method proposed in [8] also follows this approach wherein morphological features are used for edge extraction followed by refinement based on curvature. Similarly, the method in [9] uses pixel-neighbourhood statistics to identify crack pixels and tensor voting [10] to connect them.

Over the past decade, deep learning based methods have
The paper is organized as follows. In Section II we provide the ad-hoc hardware development for this work. Our proposed setup consists of:

- 3D printed illumination frame;
- addressable LED strips;
- a machine vision camera;
- optically rectified lens.

The illumination frame, depicted in Fig. 1a, hosts four lines of LED’s strips which illuminate the object (i.e., a ceramic tile in our case) from all the four sides at different levels of height. With the press of a button, the setup acquires 5 images of the tile in the following lighting configurations:

- all lights switched on,
- only level 1 (LED level closest to the tile) switched on,
- only level 2 (LED level above level 1) switched on,
- only level 3 (LED level above level 2) switched on, and
- only level 4 (LED level above level 3 and furthest from the tile) switched on.

Our design is inspired by the so-called dark field illumination [19] which tries to increase the contrast between the background and foreground regions using oblique lighting. The intuition behind this design is that cracks can be better visualized under specific lighting conditions based on the illumination angle. The four illumination levels provide different angles of illumination with a total of 189 white colored LEDs that can be activated individually or in pre-programmed multi-led illumination patterns. The goal is that of improving the visibility of defects on the tile and avoiding specularity.

The LEDs control is provided by an Arduino Nano board running a dedicated software routine. This routine runs in the Arduino microcontroller and executes the messages corresponding to the control actions given by the user. Here, a computing device connected through the USB serial connection is used for receiving the control actions from the user and sending the corresponding messages to the Arduino microcontroller for execution.

This setup has been designed keeping in mind the dimension-constraints for deployment in a real industrial scenario for the inspection of ceramic tiles. For this reason the main goal of the design has been that of creating a compact device that can be easily transported by a human operator. The main problem in such situation comes from the optics geometry. In particular, we need to capture a surface of 20x20 cm (the dimensions of the surface) from a short distance. In the presented system, this translates into lens with few millimeters of focal lengths entering in the realm of ultra-wide lens. We achieve the desired compactness with a combination of a machine vision camera and an optically rectified lens. This helps us to have an extreme diagonal field of view (≈ 135 degrees) while having a very short distance (just 6cm) between the tile and the lens tip.

III. EXPERIMENTAL PIPELINE

The pipeline used for our experimental procedure is shown in Fig. 2 and the components are discussed below.

A. Acquisition

The portable setup is placed over a tile kept on an acquisition table for the acquisition to be performed in a controlled illumination manner. It then acquires 5 images of the tile in...
various lighting configurations as discussed in Section II. The setup is then moved and placed over the next tile and this process is repeated to acquire images of all the tiles. For every tile, all the 5 images are acquired without moving the setup. Therefore, there is no relative motion and these are automatically registered with respect to each other. For these tiles, ground truth is provided by an expert (from our industrial partner) in the form of digital annotations (as shown in Fig. 2) which includes crack contours and locations of the tile corners. With the acquired images and their respective ground truth, we are now ready to generate the dataset (of patches) to be used for training and evaluation. The following subsection explains the extraction of labeled patches from the acquired images and their organization to create the dataset.

B. Dataset Generation

The registered images and the binary ground truth image containing the crack contours are used to extract labeled patches in a sliding window fashion. During extraction, the image patches are labeled as either positive (patches containing cracks), negative (patches that do not have cracks) or ambiguous based on the proportion of crack pixels in the corresponding ground truth patch. The proportion of crack pixels \( p \) in a ground truth patch \( \psi \) of size \( m \times n \) is defined as:

\[
p = \frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} I(\psi(i,j) > 0),
\]

where \( I(\bullet) \) is an indicator function such that \( I(True) = 1 \), \( I(False) = 0 \) and \( \psi(i,j) \) is the value of pixel at location \((i,j)\) in the ground truth patch \( \psi \). Using this definition of proportion of crack pixels \( p \), patches in the registered images are labeled as follows:

- Negative patches: \( p < 0.1 \),
- Ambiguous patches: \( 0.1 \leq p < 0.2 \),
- Positive patches: \( p \geq 0.2 \).

Intuitively, only those patches that contain no crack pixels (i.e., \( p = 0.0 \)) should be labeled negative. Thus, setting \( p = 0.0 \) should suffice in labelling the patches as either negative or positive. However, in practice, the annotations are a few pixels wider than the actual crack-width, due to which some patches are incorrectly labelled as positive. To correctly label such patches as negative, we set a lower threshold to 0.1. Secondly, patches along the crack boundaries that contain few crack pixels have a similar appearance to that of the negative patches. Such ambiguous patches increase the number of false positives when labelled as positive. To avoid this, we set a higher threshold to 0.2. Only the generated positive and negative patches are used for training and evaluation of the classifiers.

**Data Balancing:** Since every tile has only few crack patches, the number of negative examples is substantially larger than the number of positive examples. Thus, we get an imbalanced dataset by considering all extracted positive and negative patches. If an imbalanced dataset is used for training, the trained classifier can be biased towards the majority class, which in our case is the negative class. To avoid this problem, balancing the training data with random under-sampling has been considered as one of the effective, easy and widely accepted methods [20]. Alternate approaches that use cost sensitive loss functions [21], [22] with imbalanced training data did not show a significant change in the results in our case, but notably increased the training time. We have therefore adopted the former method and artificially balanced the patches extracted from every image by random undersampling, i.e., we consider all the positive patches and randomly select an equal number of negative patches for balancing the dataset.

**k-Folds:** Patches extracted from the acquired images are divided into 10 folds. Each fold has a fixed set of tiles for training and testing. This ensures that the patches used for testing in a particular fold are not used for training in the same fold. Considering data balancing and folds discussed above, we use the following data during the different phases, viz., train, validation and test.

For fold \( F_K \)
- \( \text{Train}_K \): \{Balanced Positives, Balanced Negatives\} from tiles selected for training in \( F_K \),
- \( \text{Validation}_K \): \{Balanced Positives, Balanced Negatives\} from tiles selected for testing in \( F_K \),
- \( \text{Test}_K \): \{Imbalanced Positives, Imbalanced Negatives\} from tiles selected for testing in \( F_K \).

**Spatial Resolution:** The images acquired using the camera and the corresponding ground truth have a resolution of...
3840 \times 2748$. To have an insight about performance of the different classifiers on VGA-resolution (which is available even with inexpensive cameras), we experiment with low-resolution inputs. This is done by downsampling the acquired images with a factor of $0.1667$ both in height and width so that it has size closer to the VGA-resolution, i.e., $640 \times 480$. Here, the patches of size $50 \times 50$ are extracted with a stride of $10$ pixels. We also experiment with the acquired images in their original size (high-resolution) without any downsampling by extracting patches of size $299 \times 299$ with a stride of $60$ pixels. Thus, the number of patches used in both low and high-resolution experiments is approximately the same.

C. Training

The training of classifiers is done using various architectures. These include two custom architectures and six state-of-the-art classification architectures [23], [24], [25], [26], [27], [28]. For the custom architectures training is done from scratch. For all other architectures, the training is performed by fine-tuning where the pre-trained weights have been learnt on the imagenet classification dataset [29].

Training from scratch: The two custom architectures that we train from scratch, i.e., without using any pre-trained weights, have $3 - 4$ convolutional layers and three fully connected layers. The first of these architectures (TileNet6) is shown in Fig. 3a. The second custom architecture only has one extra convolution layer in comparison to the first architecture. This was done to study the effect of having a slightly deeper architecture. We call this as TileNet7, which is shown in Fig. 3b. Below, we also investigate the effect of having even more deeper architectures.

Fine-tuning: We have considered six state-of-the-art object classification architectures, viz. VGG16 [23], Xception [24], ResNet50 [25], DenseNet121 [26], InceptionResNet-V2 [27] and NASNetLarge [28]. These have outperformed most of the competing methods in the Imagenet Large Scale Visual Recognition Challenge (ILSVRC) [30]. Their pre-trained weights (except for the top fully connected decision layer) have been downloaded from Keras Applications [31].

First, we use these networks as feature extractors, i.e., we feed them with the image patches and run a forward pass using the pre-trained weights. The extracted features and the corresponding labels are then used to train a shallow two-layered fully connected network which performs classification. This architecture is shown in Fig. 4. Note that here the pre-trained weights are fixed and never updated. With the same network architecture we also try to fine-tune the pre-trained weights of certain layers. In this case we add a suffix $\text{finetune}_{<\text{layer name}>}$ onwards to the model name. For example, when we perform transfer learning on ResNet50 by fine-tuning all the layers from “conv$_{5x}$” onwards, the new model is named as “ResNet50$\_\text{finetune}_{conv_{5x}}$ onwards”. Likewise, if all the layers are fine-tuned, the suffix $\text{finetune}_{all}$ is added to the model name. In addition, the models trained using high-resolution (i.e., without any downsampling) patches have the suffix $\_HR$. We use this nomenclature to discuss the results later in Section V. In the next section we describe the metrics used for performance evaluation.

IV. EVALUATION METRICS

Since the training is performed on patches, it is logical to use patch-level metrics. However, since we eventually intend to evaluate the performance of classifiers to detect cracks in the whole tile, the use of image-level metrics is also proposed. The definitions of metrics used for evaluation at patch-level and image-level are as given below.

A. Patch-level Metrics

Denoting true positives, false positives, true negatives and false negatives with $TP$, $FP$, $TN$ and $FN$, respectively, the patch-level metrics are defined as follows:

1) Accuracy: The first patch-level metric accuracy is defined as follows:

$$\text{accuracy} = \frac{TP + TN}{TP + FP + FN + TN}.$$  \hspace{1cm} (2)

2) Matthews Correlation Coefficient: The accuracy metric can be misleading in the presence of imbalanced data. Therefore, we also use the Matthews Correlation Coefficient
In the best case, the well the classifiers work for the tiles and not just for patches. Therefore, there is a need to quantify how accurately can the underlying technique detect the defective tiles. For a given tile \( t \) detected cracks is available, we can calculate Recall, Precision and F1 score for every tile \( t \) as:

\[
\text{Recall} (R_t) = \begin{cases} 
\frac{n(G_t \cap D_t)}{n(G_t)}, & \text{if } n(G_t) > 0, \\
1, & \text{otherwise}, 
\end{cases}
\]

(6) For \( N \) tiles, crack presence accuracy (CPA) is calculated as:

\[
\text{CPA} = \text{average}(PM) = \frac{1}{N} \sum_{t=1}^{N} (PM)_t. 
\]

(5) In the best case, the CPA = 1.0 indicating that all the tiles having at least one crack were correctly identified. On the other hand, CPA = 0.0 indicates that no tile was correctly identified for having presence/absence of the cracks.

2) Crack Count F1 Score: The next proposed image-level metric quantifies how close is the number of correctly detected cracks to the number of actual cracks in the ground truth. In other words, it indicates how accurately the underlying technique can detect the cracks. This metric (which we call as the crack count F1 score) is most relevant when the goal is to determine how many cracks can be correctly detected in the given tile.

In order to calculate this metric, we first need to identify the cracks that are correctly detected, i.e., detected cracks that are also present in the ground truth. To do so we follow the procedure given in Procedure 1. Once the set of correctly detected cracks is available, we can calculate Recall, Precision and F1 score for every tile \( t \) as:

\[
\text{Recall} (R_t) = \begin{cases} 
\frac{n(G_t \cap D_t)}{n(G_t)}, & \text{if } n(G_t) > 0, \\
1, & \text{otherwise}, 
\end{cases}
\]

(6) For \( N \) tiles, the Crack Count F1 Score (CCF1) is calculated as the average of the F1 score given by:

\[
CCF1 = \frac{\sum_{t=1}^{N} F1_t \cdot a_t}{\sum_{t=1}^{N} a_t}, \quad \text{where } a_t = n(G_t) + 1. 
\]

(9) The maximum value \( CCF1 = 1.0 \) happens when the detected cracks are the same as the cracks present in the ground truth for all the tiles. Higher the value, more is the match between these quantities. In the worst case, i.e., when these quantities differ to their maximum we get \( CCF1 = 0.0 \). In this case none of the cracks are correctly detected in all the tiles.

V. RESULTS AND DISCUSSION

Our dataset consists of 88 ceramic tile images acquired in five different lighting configurations using the procedure described in section III-A. For each lighting configuration, the dataset is organized into 10 folds such that every fold is assigned about 70-76 and 8-12 tiles for training and test phases, respectively. Models corresponding to all the architectures

\[
MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}. 
\]

(3)
discussed in section III-C have been trained on a NVIDIA GeForce RTX 2080 Ti GPU using Adam optimizer [33] with a learning rate $= 0.0001$. The input size and batch size for training models using low-resolution patches is $50 \times 50$ and 128, respectively. For models trained using high-resolution patches, the input size is $299 \times 299$ while the batch size is reduced to 16 for managing the computational overhead. The training is done for about 1300 epochs in both cases where classifiers are trained using (a) our custom architectures, and (b) features extracted from pre-trained models. The number of epochs is reduced to approximately 55 for the classifiers that are only fine-tuned over the pre-trained weights. The total time spent for training all the classifiers for the different lighting configurations, resolutions, folds and architectures is about 30 days. The results obtained by using these classifiers are presented below.

A. Results

Examples of crack detection on two tiles for the different lighting configurations are shown in Fig. 6. The patch classification accuracy defined in Equation (2) is compared for the trained models in Fig. 7a. Similarly, Fig. 7b compares the Matthew’s correlation coefficient defined in Equation (3) for these models. The crack presence accuracy defined in Equation (5) is compared in Fig. 7c, while Fig. 7d compares the crack count F1 score defined in Equation (9) for these models.

B. Discussion

Accuracy: Looking at Fig. 7a, we observe that the training accuracy for almost all the models saturates to 1 while for validation it is about 0.8; an indication of overfitting. For test data also the accuracy is closer to 0.85 for most of the models. If only the feature-based models are considered, the model trained using Xception-features appears to be slightly better than the others. Nevertheless, the models trained using TileNet6 provide better accuracy in comparison to feature-based models. Also, having an additional convolution layer (i.e., TileNet7) helped in slightly improving the accuracy. Since the Xception-features have the best accuracy among the feature-based models, we also trained classifiers by fine-tuning all layers of the Xception architecture. These classifiers provided the best results reaching an average accuracy of 0.89 and 0.95 on the validation and test data, respectively. To study the effect of using high-resolution inputs, we also trained classifiers again by fine-tuning all layers of the Xception architecture but using high-resolution patches (Xception-finetune_all HR). Here, we observe that the use of high-resolution inputs provides further improvement in the results. The average validation accuracy jumped from 0.89 to 0.95 while the average test accuracy improved from 0.95 to 0.98 and the training accuracy is also slightly improved.

MCC: We observe that MCC (in Fig. 7b) for most of the feature-based models is close to 0.5 (for both validation and test data), while it is slightly higher for the models trained on our custom architectures. The MCC for Xception architecture is comparable with that of our custom architectures. However, the MCC is highest for Xception_finetune_all with values 0.80 and 0.79 for for validation and test data, respectively. For the same, training with high-resolution patches shows further improvement with the MCC increasing from 0.80 to 0.91 for validation and from 0.79 to 0.90 for test data.

Crack Presence Accuracy: Since the crack presence is quantified at image-level (as opposed to patch-level on which training is performed), the CPA is calculated for training data and test data that consists of images (and not image-patches) of tiles selected for training and test, respectively. Fig. 7c shows that crack presence accuracy = 1.0 for for almost all the trained models in both training and testing phases. Similar is the case when using models trained on high-resolution patches. This metric can be more meaningful if (a) additional data containing tiles not having cracks is also considered for testing or (b) the detection of cracks is to be performed in only specific regions of the tiles. However, currently, we only have with us a dataset of tiles having cracks for which we perform crack detection over the complete tile-surface.

Crack Count F1 score: The crack count F1 scores shown in Fig. 7d indicate that the feature-based models are substantially less accurate in correctly detecting the cracks in
comparison to those trained on TileNet6 (CCF1 ≈ 0.51). Adding an extra convolution layer (i.e., TileNet7) also helps in increasing this score (CCF1 ≈ 0.56). However, fine-tuning the entire Xception architecture leads to even better results (CCF1 ≈ 0.65). Also, the use of high-resolution inputs further increases the score (CCF1 ≈ 0.71), indicating a more accurate crack detection.

Effect of illumination sources placed at various heights: The overall results in Fig. 7 indicate that both patch-level and image-level metrics improve as we move from Only Level 1 to Only Level 4. In other words, higher the placement of the illumination source, better is the performance and less is the number of false positives (as seen in Fig. 6). This trend is visible in all the results but more prominent for the models trained using high-resolution patches. Secondly, the performance for Only Level 4 is more-or-less similar to that of All Lights. This indicates that an illumination source placed at a greater height is as good as having a denser illumination setup (with sources at placed at various heights), for the purpose of crack detection.

VI. CONCLUSION AND FUTURE WORK

Our proposed height-varying illumination setup, which is designed for field work with constraints in its maximum dimensions, has been effectively used to acquire images of defective tiles. Crack detection has been performed on these images by means of patch-classification. Our unique study on height-varying illumination conditions for crack detection.
suggests that lights placed at greater heights are more effective as compared to those placed near the tile’s surface for crack detection. In fact, their performance is as good as that of using together the lights at all the levels. Our study also indicates that fine-tuning of all the pre-trained weights of the Xception architecture provide the best results in comparison to all the other trained models across all the illumination condition. Moreover, use of high-resolution patches (i.e., without downsampling the acquired images) for training further improves the results. Thus, the intuition of performance improvement with the use of high-resolution patches is also validated across all the lighting conditions in our study. This should help in deciding the resolution versus performance trade-off when designing a real-time crack detection system for field use. The effectiveness of different illumination conditions on crack detection has been demonstrated using evaluation performed on classifiers trained with the state-of-the-art as well as our customized architectures.

The present work describes the experiments and results obtained from a relatively small number of tiles, which are difficult to procure. Also, the training images were annotated by highly specialized personnel in their very limited available time. Nevertheless, the effectiveness of portable setup has been clearly demonstrated by our experiments. In fact, its use can be generalized for automatic visual inspection of any object having a relatively planar surface.

Following encouraging preliminary results, in the future we will be focusing on using different sensor modalities towards extending the acquisition of the tiles by means of longwave infrared (LWIR) thermal sensors. Last but not least, we want to explore further the effect of both image and light resolution in the performance of the classification models.

REFERENCES

[1] A. Landstrom, M. J. Thurley, and H. Jonsson, “Sub-millimeter crack detection in casted steel using color photometric stereo,” in 2013 International Conference on Digital Image Computing: Techniques and Applications (DICTA), Nov 2013, pp. 1–7.

[2] F. Chen and M. R. Jahan-shahi, “Nb-cnn: Deep learning-based crack detection using convolutional neural network and naîve bydes data fusion,” IEEE Trans. on Industrial Electronics, vol. 65, no. 5, pp. 4392–4400, May 2018.

[3] L. Yang, B. Li, G. Yang, Y. Chang, Z. Liu, B. Jiang, and J. Xiao, “Deep neural network based visual inspection with 3d metric measurement of concrete defects using wall-climbing robot,” in 2019 IEEE IROS, Nov 2019, pp. 2849–2854.

[4] M. Mohajeri and P. J. Manning, “Aria: an operating system of pavement distress diagnosis by image processing,” Transportation Research Record, vol. 1311, pp. 120–130, 1991.

[5] R. S. Walker and R. L. Harris, “Noncontact pavement crack detection system,” Transportation Research Record, vol. 1311, pp. 149–157, 1991.

[6] P. W. Fieguth and S. K. Sinha, “Automated analysis and detection of cracks in underground scanned pipes,” in Proceedings 1999 ICIP (Cat. 99CH36348), vol. 4, Oct 1999, pp. 395–399 vol.4.

[7] A. Ammouche, J. Riss, D. Breysse, and J. Marchand, “Image analysis for the automated study of microcracks in concrete,” Cement and Concrete Composites, vol. 23, no. 2, pp. 267 – 278, 2001, special Theme Issue on Image Analysis.

[8] S. Iyer and S. K. Sinha, “A robust approach for automatic detection and segmentation of cracks in underground pipeline images,” Image and Vision Computing, vol. 23, no. 10, pp. 921 – 933, 2005.