Computer vision methods for 3d concrete printing process monitoring

Rodrigo Rill García

▶ **To cite this version:**

Rodrigo Rill García. Computer vision methods for 3d concrete printing process monitoring. Computer Vision and Pattern Recognition [cs.CV]. École des Ponts ParisTech, 2022. English. NNT : 2022ENPC0043 . tel-03964423

HAL Id: tel-03964423
https://pastel.hal.science/tel-03964423

Submitted on 31 Jan 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

École des Ponts ParisTech

THÈSE DE DOCTORAT
de l'École des Ponts ParisTech

Computer Vision Methods for 3D Concrete Printing Process Monitoring

École doctorale N°532, Mathématiques et Sciences et Technologies de l'Information et de la Communication (MSTIC)

Spécialité Informatique

Thèse préparée au LIGM, UMR 8049, et au Laboratoire Navier, UMR 8205, École des Ponts ParisTech.
CNRS, Université Gustave Eiffel, Marne-la-Vallée, France

Soutenance prévue le 13 décembre 2022, par

Rodrigo RILL GARCÍA

Composition du jury:

Sylvie LE HÉGARAT-MASCLE Professeur, Univ Paris-Saclay	*Rapporteur*
Dominique GINHAC Professeur, Univ Bourgogne Franche-Comté	*Rapporteur*
Thierry GERAUD Professeur, EPITA	*Examinateur/Président*
Jean-François CARON Professeur, École des Ponts ParisTech	*Examinateur*
Eva DOKLADALOVA Professeur, Univ Gustave Eiffel	*Directrice de thèse*
Petr DOKLADAL Chargé de recherche, MINES Paris	*Co-directeur de thèse*

Acknowledgments

I take this space to thank all the persons that helped to finish, one way or another, this thesis. In first place, my family: they will be always a part of any of my achievements. In second place, my advisors Eva and Petr: their support has been really important even since before my arrival to France. In third place, my colleagues from the École des Ponts: not only their ideas contributed to this thesis, but a lot of the experiments presented in this work were only possible because of their help in the printing lab. In fourth place, my Mexican colleagues Rosemberg and Elias: my stay in France was greatly eased because of their support.

As an additional round of acknowledgements, I take the opportunity to thank the internal review committee: for their valuable time, interest and suggestions. Lastly, but not less important, the jury members: for agreeing to review this thesis, participating in the jury board, and their contributions towards making this manuscript as good as possible.

*This work has been supported by the project DiXite. Initiated in 2018, DiXite (Digital Construction Site) is a project of the I-SITE FUTURE, a French initiative to answer the challenges of sustainable city. This research project was financially supported by I-SITE FUTURE (ANR-16-IDEX-0003).

Abstract

Construction is one of the largest sectors of the world economy, and an important part of our daily life: it is in charge of building the structures we use to live, work and travel. However, construction has suffered from poor productivity relative to other sectors in the past decades. To boost this productivity, construction based on 3D concrete printing has become an attractive option, infusing advanced automation and digital technologies to the sector.

Because of the late introduction of digitalization in construction, as well as the relative novelty of 3D concrete printing, research on computer vision for the monitoring of construction based on this technology is scarce. Motivated by this, and to contribute to the digitalization of the sector, in this work we propose methods for automatic monitoring of construction based on computer vision. The scope of this work is divided into two stages of the construction life cycle: 1) damage monitoring, after the construction is finished; 2) quality monitoring, performed inline during a 3D concrete printing process.

For damage monitoring, among the possible damages in constructions, cracks are an evident sign that can be identified through a visual inspection. Properties of the cracks such as their orientation, length and number allow inferring the mechanisms that generated the observed defect. Other geometrical properties of the cracks such as their width are important indicators of the severity of the damage, as well as the future durability. In the context of this PhD, a focus is given to pixel-accurate segmentation of cracks, so that it becomes possible to automatically determine their width.

Concerning the utilization of machine learning for crack segmentation, the annotations used for supervised training tend to be inaccurate because the annotation task is very prone to human error, specially at the pixel level. Particularly, the provided annotations tend to be wider than the visible cracks, and this bias in the annotations is reflected in the predictions of the trained models. In this work, we approach the problem of supervised crack segmentation in presence of inaccurate annotations. However, it is difficult to collect images of cracks in 3D printed structures: early cracks appear during the period in which the concrete is drying, and posterior cracks appear during the lifetime of the printed piece once it is put into service. Consequently, we perform our experiments on public datasets for a highly studied and closely related task: road crack segmentation.

As baseline model for the experiments in this work, we proposed U-VGG19: an encoder-decoder neural network we built inspired by U-net, using the convolutional layers of VGG19 pre-trained on ImageNet as an encoder. By training this network on the public CrackForest dataset, we obtain an F-score of 71%; this score is superior to other, more complex, state-of-the-art networks when the evaluation is performed at pixel level without any kind of tolerance. However, maximizing this score is not a reliable way to determine the quality of a model when the annotations used for evaluation are known to be inaccurate.

To overcome this problem, we used the Syncrack generator as benchmark –a tool we developed for a parametrizable generation of synthetic images emulating cracked constructions. The Syncrack generator provides a ground truth segmentation of the visible cracks; it also includes a module to introduce parametrizable

noise in the ground truths, emulating the human inaccuracy.

On Syncrack-generated data, we show that inaccurate training annotations deteriorate the accuracy of the predicted crack width. With respect to training with accurate ground truths, the predictions of a model trained with noisy annotations exhibit a decrease of F-score up to 12%. This decrease is essentially due to a drastic decrease of precision, up to 25%. On the other side, we observe an increase of recall up to 6%. These behaviors are associated to a higher crack detection rate (specially of thin cracks), at the cost of producing segmentations typically wider than the visible cracks. It is worth noticing that this behavior is observed in the presence of bi-directional noise: crack pixels are mislabeled as background and vice versa in the training images. Since the width of the cracks is an important property to determine their severity, we work towards improving the accuracy of the predicted crack width when training in the presence of inaccurate annotations.

Regarding this goal, we present two different approaches in the current manuscript: one method based on learning with inaccurate supervision, and one based on transfer learning from synthetic data. The first method, inspired by the literature on weakly supervised learning, consists of two mains steps: 1) producing new pseudo-labels per pixel with the help of a model trained on the raw inaccurate annotations, and 2) training a final model with the help of these pseudo-labels. For the generation of pseudo-labels, we test 4 different methods inspired by weakly supervised classification; from these methods, the most successful one was the one we referred to as *5-nn voting* (based on the k-NN algorithm). By training with the help of the pseudo-labels generated by this method, we observed an increase of F-score up to 11% with respect to training with inaccurate annotations. The obtained score is less than 2% below the F-score obtained by training with accurate annotations, showing that the proposed method can effectively deal with the noise introduced to the annotations.

The aforementioned results were obtained on Syncrack-generated data. To test the proposed method on real data, the previously used supervised scores (F-score, precision, recall) are not reliable since the manual annotations are expected to be inaccurate. Because of this, we proposed a set of unsupervised scores as an additional option for evaluation. These scores are based on metrics previously used for unsupervised segmentation: the first and second order region entropies, and the Kolmogorov-Smirnov score. The entropies evaluate the uniformity of the regions predicted as crack; these scores increase when the regions predicted as cracks are contaminated by background pixels, since the diversity of pixel intensities in the region increases. The Kolmogorov-Smirnov score measures the difference between the distributions of pixels predicted as crack and as background, respectively; this score decreases when the regions predicted as crack are contaminated by background pixels, since they quickly become more similar to the background region. In summary, these scores allow detecting when the predictions are wider than the visible cracks. This claim is supported by the evidence on Syncrack-generated data, where we show that the proposed scores have a direct relation with the quality of the predicted segmentations in terms of supervised precision.

With the help of these auxiliary scores, we show that the proposed method in-

spired by weakly supervised learning is applicable to real data too. Specifically, we tested it on a dataset composed by merging the CrackForest and the Aigle-RN datasets. After training with the pseudo-labels generated by 5-nn voting, the predicted crack width accuracy improves with respect to training with the available manual annotations, as measured by the proposed unsupervised metrics.

The second method to improve the accuracy of the predicted crack width is based on transfer learning. We show that models trained solely on Syncrack-generated images are able to segment cracks on real images. For these experiments, the parameters of the Syncrack generator were adjusted so that the image resolution and average crack width were similar to those of the CrackForest dataset. The predictions obtained on the CrackForest dataset by the models trained on synthetic images show that these models are transferable to real images. Furthermore, these predictions are more precise in terms of crack width accuracy with respect to the models trained on manual annotations, as measured by the proposed unsupervised scores.

Remember that, in 3D printed pieces, the appearance of cracks typically occurs when the construction material is drying, and during its lifetime once it is put into service. Prior to this, quality monitoring during the construction process is important to ensure that the final product meets a minimum quality standard before being put into service.

For quality monitoring, we work on inline anomaly detection for 3D concrete printing. Specifically, the monitoring of pieces produced by contour crafting (i.e. layer-wise extrusion of concrete). In these pieces, the simplest way to differentiate one layer from its neighbors is to locate the transition regions between adjacent layers; we refer to these regions, that determine the boundaries of individual layers, as interlayer lines. The interlayer lines provide information about the geometry of the extruded layers, such as their orientation, if they present bending, if there are overlapping layers, etc. The analysis of these lines can be automatized using computer vision; this requires segmenting the interlayer lines in an image.

For post-printing analysis, the interlayer lines in dry pieces tend to take the appearance of thin, black edges in the surface. However, for inline monitoring, segmenting these lines exhibits a higher difficulty because we have to deal with fresh material. On one hand, the extruded concrete presents specular reflectance due to its –considerable and variable-– water content and the different additives in the mixture. On the other hand, the fluidness of the fresh concrete can produce layer merging and superposition. Consequently, the interlayer lines during the printing process can take very different aspects: black, bright or even not visible at all. Additionally to the challenges associated to the analyzed material, the segmentation task has to deal with the constraints imposed by the physical setup: the environment is hazardous and it is difficult to set ideal acquisition conditions.

In this work, we analyze images from I3DCP: a dataset we collected during a printing session in the 3D concrete laboratory of the École des Ponts ParisTech. The chosen setup used a camera fixed to the extrusion nozzle, in order to have a fixed lateral view of the extrusion zone: this allows observing the interlayer lines of the analyzed piece.

For segmentation, we use U-VGG19 as baseline model. We train the network from an initial set of manually labeled images, using a semi-supervised approach. With this method, we iteratively improve the trained model while producing additional annotated images. We produce a total of 128 new annotations, and we use the initial 32 manually annotated images as test set. After training with the 128 images, U-VGG19 obtains an F-score of 91%.

The analyses based on these lines allow detecting anomalies in the printed piece. On one hand, by analyzing the lines themselves, we can determine the presence of geometrical anomalies. On the other hand, these lines allow us to segment each layer independently in the input image; consequently, we can analyze the texture of each layer separately in search of anomalies.

For geometrical characterization and anomaly detection, we propose to perform local measurements of the geometry of the layers using image processing. Specifically, we measure, at pixel level: the orientation and curvature of the interlayer lines, the width of the layers (i.e. the distance between their bounding interlayer lines), and the vertical distance from the deposition layer to the extrusion nozzle. Then, for each type of measurement, we obtain a distribution of the values measured per pixel: these distributions serve as a descriptor to characterize the analyzed image. In a good printing process, each of these distributions should be contained and centered inside a range of admissible values, which represents the expected normality of the process. In this work, these ranges are defined by the user. Consequently, the anomalies are located as the regions in the analyzed image with measured values outside the user-defined ranges. With this approach, our method can return the location, nature and severity of the detected anomalies.

Additionally to the geometrical anomalies, we analyze the observed surface for anomaly detection in terms of the visible texture. This approach is based on the premise that one of the factors that change the texture of the extruded layers is their water content; at the same time, an excess or shortage of water in the mixture is a potential indicator of issues which may impact the structural performance of the printed piece. With this in mind, we use a machine learning approach to classify the extruded layers, window-wise, either as *good* or as one of three defective cases: excessively fluid, excessively dry, or presenting superficial tearing. The feature extraction is based on gray level co-occurrence matrices and local binary patterns. The selected model is a small convolutional neural network. The training and testing are performed on manually annotated windows extracted from the I3DCP dataset; on the test set, our model obtained a macro-averaged F-score of 94%.

As a whole, the obtained results show the wide range of possibilities for automatic monitoring based on computer vision, during different stages of the life cycle of a construction. On one hand, we improved the geometry of predicted crack segmentations when training with inaccurate annotations, a topic that is generally overlooked in the crack segmentation literature. On the other hand, we extended the principle of crack segmentation to interlayer line segmentation in 3D printed concrete pieces. Our results show how an inline monitoring of the printing process can be based on the geometrical analysis of these lines, particularly in search of anomalies in the piece. With this, we extend the catalogue

of possible geometrical measurements using computer vision with respect to the current proposals in the literature. Similarly, our results show that the detection of anomalies in the process can be based on a textural analysis of the extruded layers: an approach that has not been explored before for 3D concrete printing in the literature.

The multi-factorial method (cracks, geometry, textures) we have developed, based on computer-vision, could be used in industry, after validation, for control and delivery to customer.

Keywords: Computer vision, 3D concrete printing, Weakly supervised segmentation, Texture classification, Anomaly detection.

Résumé

La construction est un des plus grands secteurs de l'économie mondiale et une partie importante de notre vie quotidienne : elle est chargée de construire les structures que nous utilisons pour vivre, travailler et voyager. Cependant, la construction a souffert d'une productivité médiocre par rapport à d'autres secteurs au cours des dernières décennies. Pour augmenter cette productivité, la construction basée sur l'impression 3D de béton est devenue une option attrayante, infusant des technologies avancées d'automatisation et numériques au secteur.

En raison de l'introduction tardive de la numérisation dans la construction, ainsi que de la relative nouveauté de l'impression 3D du béton, les recherches sur la vision par ordinateur pour la surveillance de la construction basée sur cette technologie sont rares. Motivés par cela, et pour contribuer à la numérisation du secteur, nous proposons dans ce travail des méthodes de surveillance automatique de la construction basées sur la vision par ordinateur. Le cadre de ce travail est divisé en deux étapes du cycle de vie de la construction : 1) détection des défauts, une fois que la construction est terminée; 2) contrôle qualité, effectué en ligne lors d'un processus d'impression 3D de béton.

Pour la détection des défauts, parmi les dommages possibles dans les constructions, les fissures sont un signe évident qui peut être identifié grâce à une inspection visuelle. Les propriétés des fissures telles que leur orientation, leur longueur et leur nombre permettent de déduire les mécanismes qui ont généré le défaut observé. D'autres propriétés géométriques des fissures telles que leur largeur sont des indicateurs importants de la gravité des dommages, ainsi que de la durabilité future. Dans le cadre de cette thèse, l'accent est mis sur la segmentation au pixel près des fissures, afin qu'il soit possible de déterminer automatiquement leur largeur.

Concernant à l'utilisation de l'apprentissage automatique pour la segmentation des fissures, les annotations utilisées pour l'apprentissage supervisée ont tendance à être imprécises parce que la tâche d'annotation est très sujette à l'erreur humaine, spécialement au niveau du pixel. En particulier, les annotations fournies ont tendance à être plus larges que les fissures visibles, et ce biais dans les annotations se reflète dans les prédictions des modèles entraînés. Dans ce travail, nous abordons le problème de la segmentation supervisée des fissures en présence d'annotations imprécises. Cependant, il est difficile de collecter des images de fissures dans les structures imprimées en 3D : les fissures précoces apparaissent pendant la période de séchage du béton, et les fissures postérieures apparaissent pendant la durée de vie de la pièce imprimée une fois celle-ci mise en service. Par conséquent, nous effectuons nos expériences sur des bases de données publiques pour une tâche très étudiée et étroitement liée : la segmentation des fissures de chaussée.

Comme modèle de base pour les expériences de ce travail, nous avons proposé U-VGG19 : un réseau neuronal encodeur-décodeur que nous avons construit inspiré d'U-net, en utilisant comme encodeur les couches convolutives de VGG19 pré-entraînées sur ImageNet. En entraînant ce réseau sur la base de données publique CrackForest, nous obtenons un F-score de 71%; ce score est supérieur à celui d'autres réseaux plus complexes de l'état de l'art lorsque l'évaluation est

effectuée au niveau du pixel sans aucune sorte de tolérance. Cependant, maximiser ce score n'est pas un moyen fiable de déterminer la qualité d'un modèle lorsque les annotations utilisées pour l'évaluation sont connues pour être imprécises.

Pour surmonter ce problème, nous avons utilisé le générateur Syncrack comme référence –un outil que nous avons développé pour une génération paramétrable d'images synthétiques émulant des constructions fissurées. Le générateur Syncrack fournit la vérité terrain de la segmentation des fissures visibles; il comprend également un module pour introduire du bruit paramétrable dans la vérité terrain, imitant l'inexactitude humaine.

Sur les données générées par Syncrack, nous montrons que les annotations d'entraînement imprécises détériorent la précision de la largeur des fissures prédites. Par rapport à l'entraînement avec une vérité terrain précise, les prédictions d'un modèle entraîné avec des annotations bruitées présentent une diminution de la F-mesure jusqu'à 12%. Cette diminution est essentiellement due à une diminution drastique de la précision, jusqu'à 25%. De l'autre côté, on observe une augmentation du rappel jusqu'à 6%. Ces comportements sont associés à un taux de détection de fissures plus élevé (en particulier des fissures fines), au prix de la production de segmentations généralement plus larges que les fissures visibles. Il convient de noter que ce comportement est observé en présence de bruit bidirectionnel : les pixels des fissures sont mal étiquetés comme arrière-plan et vice versa dans les images d'apprentissage. Étant donné que la largeur des fissures est une propriété importante pour déterminer leur gravité, nous travaillons à améliorer la précision de la largeur des fissures prédites lors d'un entraînement en présence d'annotations imprécises.

En ce qui concerne à cet objectif, nous présentons deux approches différentes dans ce manuscrit : une méthode basée sur l'apprentissage avec une supervision imprécise et une méthode basée sur l'apprentissage par transfert à partir de données synthétiques. La première méthode, inspirée de la littérature sur l'apprentissage faiblement supervisé, consiste à deux étapes principales : 1) produire de nouvelles pseudo-étiquettes par pixel à l'aide d'un modèle entraîné sur les annotations brutes imprécises, et 2) entraîner un modèle final à l'aide de ces pseudo-étiquettes. Pour la génération des pseudo-étiquettes, nous testons 4 méthodes différentes inspirées de la classification faiblement supervisée; parmi ces méthodes, la plus performante est celle que nous appelons *5-nn voting* (basée sur l'algorithme k-NN). En entraînant à l'aide des pseudo-étiquettes générées par cette méthode, nous avons observé une augmentation de la F-mesure jusqu'à 11% par rapport à l'entraînement avec des annotations imprécises. Le score obtenu est moins que 2% en dessous de la F-mesure obtenu en s'entraînant avec des annotations précises, ce qui montre que la méthode proposée est capable de traiter efficacement le bruit introduit dans les annotations.

Les résultats susmentionnés ont été obtenus sur des données générées par Syncrack. Pour tester la méthode proposée sur des données réelles, les scores supervisés précédemment utilisés (F-mesure, précision, rappel) ne sont pas fiables car on s'attend que les annotations manuelles soient imprécises. Pour cette raison, nous avons proposé un ensemble de scores non supervisés comme option supplémentaire d'évaluation. Ces scores sont basés sur des métriques

précédemment utilisées pour la segmentation non supervisée : les entropies de région de premier et second ordre, et le score de Kolmogorov-Smirnov. Les entropies évaluent l'uniformité des régions prédites comme fissures; ces scores augmentent lorsque les régions prédites comme des fissures sont contaminées par des pixels d'arrière-plan, puisque la diversité des intensités des pixels dans la région augmente. Le score de Kolmogorov-Smirnov mesure la différence entre les distributions de pixels prédits respectivement comme fissure et comme arrière-plan; ce score diminue lorsque les régions prédites comme des fissures sont contaminées par des pixels d'arrière-plan, car elles deviennent progressivement similaires à la région d'arrière-plan. En résumé, ces scores permettent de détecter quand les prédictions sont plus larges que les fissures visibles. Cette affirmation est soutenue par les expériences sur les données générées par Syncrack, où nous montrons que les scores proposés ont une relation directe avec la qualité des segmentations prédites en termes de précision supervisée.

A l'aide de ces scores auxiliaires, nous montrons que la méthode proposée inspirée de l'apprentissage faiblement supervisé est également applicable aux données réelles. Plus précisément, nous l'avons testé sur une base de données composée en fusionnant les bases de données CrackForest et Aigle-RN. Après l'entraînement avec les pseudo-étiquettes générées par 5-nn voting, l'exactitude de la largeur des fissures prédites s'améliore par rapport à l'entraînement avec les annotations manuelles disponibles, telle que mesuré par les métriques non supervisées proposées.

La deuxième méthode pour améliorer l'exactitude de la largeur des fissures prédites est basée sur l'apprentissage par transfert. Nous montrons que les modèles entraînés uniquement sur des images générées par Syncrack sont capables de segmenter les fissures sur des images réelles. Pour ces expériences, les paramètres du générateur Syncrack ont été ajustés de manière que la résolution de l'image et la largeur moyenne des fissures soient similaires à celles de la base de données CrackForest. Les prédictions obtenues sur les images de CrackForest par les modèles entraînés sur des images synthétiques montrent que ces modèles sont transférables sur des images réelles. De plus, ces prédictions sont plus précises en termes de l'exactitude de la largeur des fissures par rapport aux modèles entraînés sur des annotations manuelles, telle que mesurée par les scores non supervisés proposés.

N'oubliez pas que, dans les pièces imprimées en 3D, l'apparition de fissures se produit généralement pendant que le matériau de construction sèche et pendant sa durée de vie une fois qu'elle est mise en service. Avant cela, le contrôle qualité pendant le processus de construction est important pour s'assurer que le produit final répond à une norme de qualité minimale avant d'être mis en service.

Pour le contrôle qualité, nous travaillons sur la détection d'anomalies en ligne pour l'impression 3D de béton. Plus précisément, la surveillance des pièces produites par *contour crafting* (i.e. l'extrusion de béton par couches). Dans ces pièces, la façon la plus simple de différencier une couche des couches adjacentes est de localiser les régions de transition entre les couches adjacentes; nous appelons ces régions, qui déterminent les limites des couches individuelles, des lignes intercouches. Les lignes intercouches renseignent sur la géométrie des couches extrudées, telle que leur orientation, si elles présentent des flexions, s'il

y a des couches superposées, etc. L'analyse de ces lignes peut être automatisée grâce à la vision par ordinateur; cela nécessite de segmenter les lignes intercouches dans une image.

Pour l'analyse post-impression, les lignes intercouches dans les pièces sèches ont tendance à prendre l'apparence de bords fins et noirs à la surface. Cependant, pour la surveillance en ligne, la segmentation de ces lignes présente une plus grande difficulté car nous devons traiter du matériel frais. D'une part, le béton extrudé présente une réflectance spéculaire due à sa –importante et variable– teneur en eau et aux différents additifs du mélange. D'autre part, la fluidité du béton frais peut produire des fusions et superpositions de couches. Par conséquent, les lignes intercouches lors du processus d'impression peuvent prendre des aspects très différents : noir, brillant ou même pas visible du tout. En plus des défis associés au matériau analysé, la tâche de segmentation doit faire face aux contraintes imposées par la configuration physique : l'environnement est dangereux et il est difficile d'établir des conditions d'acquisition idéales.

Dans ce travail, nous analysons des images d'I3DCP : une base de données que nous avons collectée lors d'une session d'impression dans le laboratoire béton 3D de l'École des Ponts ParisTech. La configuration choisie a utilisé une caméra fixée sur la buse d'extrusion, afin d'avoir une vision latérale fixe de la zone d'extrusion : cela permet d'observer les lignes intercouches de la pièce analysée.

Pour la segmentation, nous utilisons U-VGG19 comme modèle de base. Nous entraînons le réseau à partir d'un ensemble initial d'images étiquetées manuellement, en utilisant une approche semi-supervisée. Avec cette méthode, nous améliorons de manière itérative le modèle entraîné tout en produisant des images annotées supplémentaires. Nous produisons un total de 128 nouvelles annotations et nous utilisons les 32 premières images annotées manuellement comme ensemble de test. Après d'entraîner avec les 128 images, U-VGG19 obtient une F-mesure de 91%.

Les analyses basées sur ces lignes permettent de détecter des anomalies dans la pièce imprimée. D'une part, en analysant les lignes elles-mêmes, on peut déterminer la présence d'anomalies géométriques. D'autre part, ces lignes nous permettent de segmenter chaque couche indépendamment dans l'image analysée; par conséquent, nous pouvons analyser la texture de chaque couche séparément à la recherche d'anomalies.

Pour la caractérisation et la détection d'anomalies géométriques, nous proposons d'effectuer des mesures locales de la géométrie des couches par traitement d'image. Plus précisément, nous mesurons, au niveau du pixel : l'orientation et la courbure des lignes intercouches, la largeur des couches (i.e. la distance entre leurs lignes intercouches délimitantes) et la distance verticale entre la couche de dépôt et la buse d'extrusion. Ensuite, pour chaque type de mesure, on obtient une distribution des valeurs mesurées par pixel : ces distributions servent comme un descripteur pour caractériser l'image analysée. Dans un bon processus d'impression, chacune de ces distributions doit être contenue et centrée dans un intervalle de valeurs admissibles, qui représente la normalité attendue du processus. Dans ce travail, ces intervalles sont définis par l'utilisateur. Par conséquent, les anomalies sont localisées en tant que régions de l'image analysée

avec des valeurs mesurées en dehors des intervalles définis par l'utilisateur. Avec cette approche, notre méthode peut retourner la localisation, la nature et la sévérité des anomalies détectées.

En plus des anomalies géométriques, nous analysons la surface observée pour la détection d'anomalies en termes de la texture visible. Cette approche est basée sur la prémisse qu'un des facteurs qui modifient la texture des couches extrudées est leur teneur en eau; en même temps, un excès ou un manque d'eau dans le mélange est un indicateur potentiel de problèmes pouvant avoir un impact sur les performances structurelles de la pièce imprimée. En tenant compte de ceci, nous utilisons une approche d'apprentissage automatique pour classer les couches extrudées, par fenêtres, soit comme *bonne*, soit comme un des trois cas défectueux : excessivement fluide, excessivement sèche ou présentant un déchirement superficiel. L'extraction des caractéristiques est basée sur des matrices de co-occurrence des niveaux de gris et des motifs binaires locaux. Le modèle sélectionné est un petit réseau neuronal convolutif. L'entraînement et le test sont effectués sur des fenêtres annotées manuellement, extraites de la base de données I3DCP; sur l'ensemble de test, notre modèle a obtenu une F-mesure macro-moyenne de 94%.

Dans l'ensemble, les résultats obtenus montrent la grande variété de possibilités de surveillance automatique basée sur la vision par ordinateur, au cours de différentes étapes du cycle de vie d'une construction. D'une part, nous avons amélioré la géométrie des segmentations prédites de fissures lors de l'entraînement avec des annotations imprécises, un sujet qui est généralement ignoré dans la littérature sur la segmentation des fissures. D'autre part, nous avons étendu le principe de segmentation des fissures à la segmentation des lignes intercouches dans des pièces en béton imprimées en 3D. Nos résultats montrent comment une surveillance en ligne du processus d'impression peut s'appuyer sur l'analyse géométrique de ces lignes, notamment à la recherche d'anomalies dans la pièce. Avec cela, nous élargissons le catalogue des mesures géométriques possibles basées sur la vision par ordinateur par rapport aux propositions actuelles de la littérature. De même, nos résultats montrent que la détection d'anomalies dans le processus peut être basée sur une analyse texturale des couches extrudées : une approche qui n'a pas été explorée auparavant pour l'impression 3D de béton dans la littérature.

La méthode multifactorielle (fissures, géométrie, textures) que nous avons développée, basée sur la vision par ordinateur, pourra être utilisée dans l'industrie, après validation, pour le contrôle et la livraison au client.

Mots clés : Vision par ordinateur, Impression 3D de béton, Segmentation faiblement supervisée, Classement des textures, Détection d'anomalies.

Table of contents

1 Introduction **1**
1.1 Digitalization of the Construction Industry 1
1.1.1 Computer Vision for Quality Assessment of 3D Concrete Printing 4
1.1.2 Monitoring of 3D Concrete Printing in the Literature 6
1.2 Thesis Contributions 8
1.2.1 Crack Segmentation 8
1.2.2 Inline Characterization and Anomaly Detection 11
1.3 Thesis Structure 12

2 Crack Segmentation in Construction Materials **15**
2.1 State of the Art on Crack Segmentation 17
2.1.1 Conventional methods 17
2.1.2 Machine Learning Methods 18
2.1.3 Deep Learning Methods 19
Convolutional Networks for Crack Detection 19
Generative Networks for Crack Segmentation 20
2.2 U-VGG19: Crack Segmentation in Construction Materials 22
2.3 Road Crack Segmentation 24
2.3.1 Road Crack Segmentation Datasets 25
2.3.2 Loss Function and Evaluation Scores for Crack Segmentation 25
2.3.3 Experimental Setup for Crack Segmentation 27
2.3.4 Preliminary Experiments and Results on Crack Segmentation 27
2.4 Discussion on the Obtained Results for Crack Segmentation 31

3 Pixel-Accurate Crack Segmentation in Presence of Inaccurate Labels **33**
3.1 Inaccurate Segmentation Annotations as Label Noise 35
3.2 Learning in Presence of Label Noise: A Brief Review 35
3.3 Syncrack 37
3.3.1 Background Generation 38
3.3.2 Crack Shape Generation 40
3.3.3 Crack Introduction 41
3.3.4 Label Noise Generator 42
3.4 Impact of Inaccurate Training Annotations when Evaluating on Clean Ground Truths 43

3.4.1 Experimental Setup to Evaluate the Detrimental Impact of Noisy Labels . . . 45
3.5 Weakly Supervised Learning . . . 48
3.5.1 Pseudo-Label Generation . . . 48
3.5.2 Experiments and Results Training with Pseudo-labels . . . 50
3.5.3 5-nn Pseudo-Label Generation in Real-life Images . . . 54
3.6 Transfer Learning from Synthetic Images . . . 58
3.6.1 Transfer Learning Setup . . . 59
3.6.2 Experiments and Results with the Different Syncrack Difficulty Levels . . . 59
3.7 Discussion on Crack Segmentation in Presence of Inaccurate Annotations and Future Perspectives . . . 65

4 Interlayer Line Segmentation in 3D Concrete Printing 69
4.1 Image Acquisition During 3D Concrete Printing . . . 71
4.1.1 Inline 3D Concrete Printing (I3DCP) Dataset . . . 72
4.2 U-VGG19 for Interlayer Line Segmentation . . . 74
4.3 Semi-Supervised Learning for Interlayer Line Segmentation . . . 75
4.3.1 Initial Training with a Reduced Amount of Annotated Images 77
4.3.2 Training with Automatically Segmented Images . . . 78
4.3.3 Fine-Tuning with Corrected Difficult Images . . . 78
4.4 Experimental Setup for Interlayer Line Segmentation . . . 78
4.5 Experiments and Results on Interlayer Line Segmentation . . . 79
4.6 Discussion on Interlayer Line Segmentation and Future Perspectives 84

5 Geometrical Characterization and Anomaly Detection in 3D Concrete Printing 87
5.1 Geometrical Monitoring of 3D Concrete Printing in the Literature . 88
5.2 Geometrical Measurements Based on Interlayer Lines . . . 90
5.2.1 Geometry of Interlayer Lines . . . 91
Orientation . . . 91
Curvature . . . 92
5.2.2 Local Thickness of Layers . . . 93
5.2.3 Relative Nozzle Height . . . 94
5.3 Geometrical Anomaly Detection . . . 95
5.4 Experiments and Results on Local Geometrical Characterization and Anomaly Detection . . . 96
5.4.1 Post-Extrusion Characterization and Anomaly Detection . . . 96
5.5 Discussion on Geometrical Assessment of 3D Concrete Printing and Future Perspectives . . . 104

6 Textural Characterization and Anomaly Detection in 3D Concrete Printing 107
6.1 Textural Monitoring of Small-Scale Additive Manufacturing in the Literature . . . 108
6.2 Characterization of 3D Printed Layers Based on Texture . . . 109
6.2.1 Pre-Processing . . . 110
6.2.2 Texture Descriptors . . . 111

Gray-Level Co-Occurrence Matrices 112
Local Binary Patterns 114
6.3 Data Analysis and Texture Classification 115
6.4 Experiments and Results for Texture Classification 117
6.5 Discussion on Textural Assessment of 3D Concrete Printing and Future Perspectives 119

7 Conclusions and Future Works **123**
7.1 Contributions 124
7.1.1 Crack Segmentation in Constructions 124
7.1.2 Inline Characterization and Anomaly Detection in 3D Concrete Printing 128
7.2 Future Research Lines 130

List of research communications **133**

Appendix **135**

References **137**

Chapter 1

Introduction

Construction is one of the largest sectors of the world economy. The importance of this sector is prominent in our daily life, since it is in charge of building the structures we use to live, work and travel. However, construction has suffered from remarkably poor productivity relative to other sectors in the past decades. The McKinsey Global Institute identified seven areas that could boost the productivity of the construction sector by 50 to 60 percent (Barbosa et al. [2017]): 1) reshape regulation; 2) rewire the contractual framework to reshape industry dynamics; 3) rethink design and engineering processes; 4) improve procurement and supply-chain management; 5) reskill the workforce; 6) improve on-site execution; and 7) infuse digital technology, new materials, and advanced automation.

In the present thesis, we focus in the last two areas. Specifically, the use of digital technologies for the monitoring of construction during on-site execution and posterior to the construction process.

1.1 Digitalization of the Construction Industry

Unlike many other industry sectors, the construction sector has been slow at adopting digitalization and automation (Buswell et al. [2020]). Beyond the fact that adopting these technologies could drastically increase the productivity of this sector, the current environmental issues make the transformation of the construction industry a priority.

Towards this goal, construction based on additive manufacturing (a.k.a. 3D printing) has become an attractive technology in recent years. Using 3D printed concrete for building construction has numerous advantages. In the first place, the time and monetary cost of construction can be considerably reduced compared to traditional construction methods. These advantages are accompanied by environment-friendly ones such as reduced material transportation, less material wastes and fewer dust generation (Hager et al. [2016]).

Although there are many methods for 3D concrete printing (3DCP), nowadays the most common one is Contour Crafting. This method consists of layer-wise extrusion (see Figure 1.1), allowing a rapid construction of large-scale and complex-shape objects (Khoshnevis and Hwang [2006]). Using the 3DCP tech-

Figure 1.1: Printing a wall by layer-wise concrete extrusion.

Figure 1.2: 3D printing of a house by (Peri [2020]).

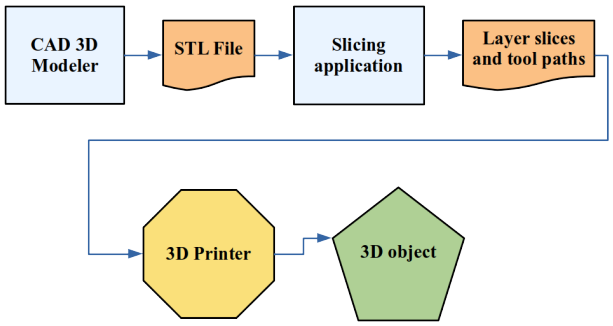


Figure 1.3: Typical workflow of a 3D printing process (Hager et al. [2016]).

nology, we can see impressive and highly publicized experimental applications (All3DP Pro [2021]) (see Figure 1.2).

In principle, extrusion-based 3D printing is based on an already established workflow. This workflow can be summarized as in Figure 1.3: 1) piece design: creating a 3D model (typically a STL file) using a CAD modeler; 2) printing path definition: using a slicing application to produce layer slices and tool paths; 3) concrete extrusion: following the tool paths with the printer to produce the 3D object. In the case of contour crafting, the printer consists of a robot with an extrusion nozzle as actuator. The concrete to extrude is contained in a mixer and

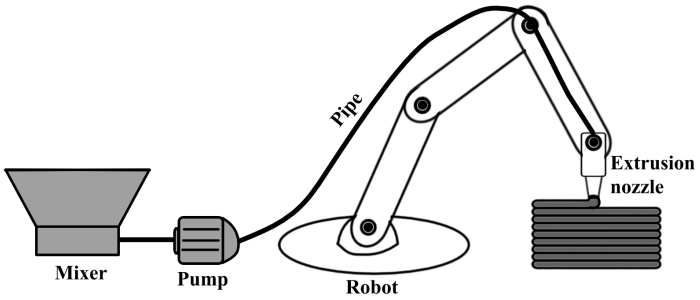


Figure 1.4: A robotic system for construction based on concrete extrusion (3D concrete printing).

transported to the nuzzle through pipes with the help of a pump. An example of a typical physical setup using a robotic arm is illustrated in Figure 1.4.

However, each of the steps listed above has many sub-tasks to be addressed to achieve a successful printing. Indeed, 3DCP is a multiparametric process: the variation of one parameter can cause defects on the process, producing deviations or defects in the printed part which can lead to its refusal (Buswell et al. [2018]), with economic and environmental consequences. Nowadays, construction based on additive manufacturing is heading towards its maturity. To reach this maturity, there is still work to be done on material developments, construction strategies and processes to make the technology certifiable, repeatable and competitive at industrial level.

In Figure 1.5, we observe examples of defective pieces refused during post-printing quality assessment. In 1.5a, we observe an excess of material on the surface and a thin crack propagated along the layers. In 1.5b, we observe a fracture propagated along the layers. In 1.5c, we observe layer superposition. In 1.5d, we observe superficial tearing at single-layer level. In 1.5e, we observe a local collapse of the printed part.

(a) (b) (c) (d) (e)

Figure 1.5: Examples of refused defectives pieces.

The appearance of defects such as cracks and fractures occurs during the period on which the concrete dries i.e. when it hardens. Because of this, it is important to monitor the printed pieces after the extrusion process has finished. Furthermore, periodic monitoring for this type of defects is necessary to ensure the security and future durability of the produced pieces once they are placed into service.

Although post-printing monitoring for quality assessment is a necessary process, it is not sufficient. There is no purpose on finishing a piece that will be refused for certain (as in Figure 1.5e). Furthermore, by early identification, it is

possible to solve many problems rising during the process (such as the ones presented in Figure 1.5c-d). Therefore, monitoring is important during the printing itself.

As discussed before, many different parameters influence the actual printing process. Furthermore, it is difficult to accurately predict the joint behavior of these parameters during the extrusion (Buswell et al. [2018]). Because of this, there has been an interest on introducing digital tools for inline quality monitoring and inspection.

In the ideal context, the robotic system used for construction should be equipped with an automated process-monitoring module to measure relevant process parameters. This would allow, of course, to identify anomalies in the piece during printing. This kind of module can be extended further to identify post-printing defects, including the ones that are inherent to dry concrete (e.g. cracks and fractures).

1.1.1 Computer Vision for Quality Assessment of 3D Concrete Printing

As illustrated in Figure 1.5, many defects can be easily identified in printed pieces by visual inspection. However, since this inspection is expected to be performed continuously for 3DCP, it becomes a tedious and complicated task. Furthermore, it is dangerous for humans to be inside the robot's workspace for inline monitoring. Therefore, auxiliary sensors such as cameras become necessary.

With the use of cameras, the visual inspection post or during printing can be automatized using computer vision. As a sub-field of Artificial Intelligence (AI), computer vision provides tools to derive meaningful information from digital inputs such as images or videos (e.g. a stream acquired from a digital camera). This information is then used to make conclusions about the images obtained from the input devices.

As an automatic –non-invasive, non-destructive– inspection method, machine vision systems have been widely used in many industry sectors. However, both because of the late introduction of digitalization in construction and the relative novelty of 3DCP, research on computer vision for 3DCP construction is scarce. The work presented in this thesis is the result of a collaboration between the LIGM and Navier laboratories to extend the frontier of knowledge for 3DCP assisted by computer vision. This research is part of DiXite (Digital Construction Site), a project of the I-SITE FUTURE, a French initiative to answer the challenges of sustainable city.

From preliminary discussions between members of both laboratories, in Figure 1.6 we propose a general roadmap for 3DCP assisted by computer vision, in the form of a flowchart. As observed in the figure, there is a wide variety of tasks related to the problem presented in this work. Once an image to process is acquired, preliminary tasks include the segmentation of constructions in the image; this segmentation can be based on material, for example. Once these areas of interest are isolated in the image, it is possible to characterize the observed construction by inspecting their geometry and surface.

In the case of the geometry, a possible characterization revolves around the

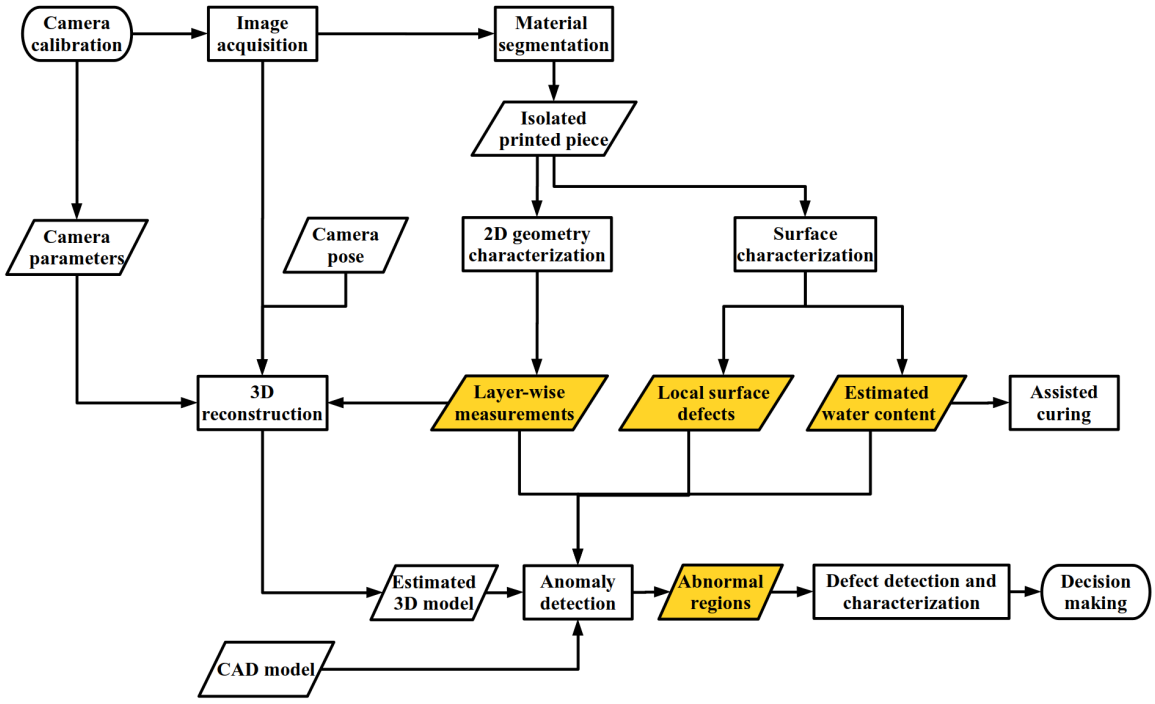


Figure 1.6: A roadmap for research on 3D concrete printing assisted by computer vision.

geometrical properties of the observed layers (e.g. width, height, orientation, etc., as discussed in subsection 1.1.2). Furthermore, by adding information such as the camera parameters and its pose, it is possible to perform a 3D reconstruction to obtain an estimated 3D model of the construction, observed simultaneously from different positions.

In terms of the surface, we can inspect the printed piece in search of superficial defects such as cracks and tearing; furthermore, by analyzing attributes such as its texture, we can gain information about the properties of the extruded layers. One example of the factors that modify the visible texture is the water content. Monitoring the change of the water content during the drying process is important to avoid the appearance of cracks and fractures; the automatic monitoring of this water content can provide assistance for tasks such as the curing of the concrete.

Furthermore, with the characterization based on geometry and surface analysis, it is possible to perform an automatic anomaly detection to locate abnormal regions in the processed image. These abnormal regions can be further analyzed to detect and characterize defects on the printed piece, allowing to create a catalogue of defects (for the posterior creation of standard norms). Finally, this information can be used to take decisions about the inspected construction e.g. rejecting the piece during post-printing monitoring or performing corrective actions to the printing process during inline monitoring.

As highlighted in yellow in the roadmap, the focus of this work is centered on the detection of abnormal regions. After the construction is finished, we detect local surface defects by performing crack segmentation; crack segmentation is an active topic in many sectors of construction besides 3DCP. During the construction process itself, we measure 2D geometrical properties of the observed layers and estimate their water content based on a textural analysis. This cha-

racterization allows detecting anomalies in the printed piece when the measured values differ from the admissible ones.

In the next section, we present a literature review of related work on 3DCP monitoring. The discussion is mainly centered around inline monitoring using computer vision.

1.1.2 Monitoring of 3D Concrete Printing in the Literature

Introducing monitoring tools for inline 3DCP printing assessment has been a relatively new field of research. The particular interest for this is to develop closed-loop control systems to correct the printing process according to the feedback of such monitoring tools.

Early attempts focused their attention on evaluating mechanical properties of the extruded material during printing (Lloret et al. [2015]; Panda et al. [2017]). Posterior works began to give attention to measurements about the geometry of the printed pieces too (Davtalab et al. [2018]). In recent years, we have seen approaches based on optical sensors such as time of flight (Wolfs et al. [2018]) and laser triangulation (Lindemann et al. [2019]). The objective of these 2 works was determining the distance between the extrusion nuzzle and the deposition surface: as this distance increases, there is a higher likelihood of unexpected layer deformation in its way from the nozzle to the surface.

We see one of the first uses of color cameras in (Kazemian et al. [2019]). With a camera attached to the printing nozzle (see Figure 1.7), the authors segment and measure the width of the observable layer using a traditional computer vision pipeline. The printing bed is white to maximize the contrast of the concrete with respect to the background, making easier to segment the printed layer. The estimated layer width corresponds to the average width of the object segmented in the image. Low extrusion rates are expected to create layers thinner than expected, while wider layers are produced by big extrusion rates.

To solve the problem of segmenting the printed material in images with more complex backgrounds, (Davtalab et al. [2020]) use a deep encoder-decoder network. In this work, however, the camera sees a lateral view of the piece post-printing. Once the region of interest (the printed piece) has been segmented,

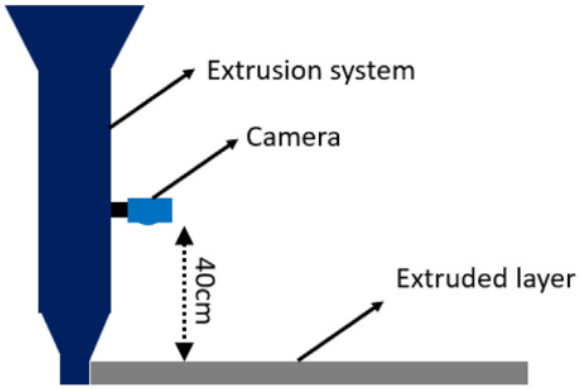


Figure 1.7: Schematic provided by (Kazemian et al. [2019]) with respect to the layout used in their experiments.

the researchers inspect it for geometrical defects. Particularly, the authors look for bended layers by inspecting the interlayer lines that separate the layers. To do this, the lines are detected as borders using the Canny algorithm and their orientation is estimated using the Hough transform.

The approaches of (Kazemian et al. [2019]) and (Davtalab et al. [2020]) are examples of local geometry assessment. A different perspective is proposed by (Nair et al. [2021]), comparing the 3D geometry of the printed piece with the 3D model used for printing. This analysis is performed post printing by scanning the piece on a rotatory base (see Figure 1.8) to obtain a point cloud.

The geometry assessment is based on mathematical morphology. Once the printed piece and the reference model are represented by their corresponding 3D point cloud, both clouds are divided into bins using an octree structure. Then, using topologic set theory, the type and percentage of overlapping between both structures is calculated. The overlapping of both structures is then used to calculate the distribution of cloud-to-cloud distances. This distribution then can be used for global (all piece) and local (layer-wise) error quantification.

As seen in this literature review, the use of automatic geometrical assessment tools for 3DCP is a young field of research to improve this type of construction systems (see summary in Table 1.1). Furthermore, computer vision is a promising approach towards this goal both for inline and post-printing monitoring. While geometrical assessment is a popular trend in methods using computer

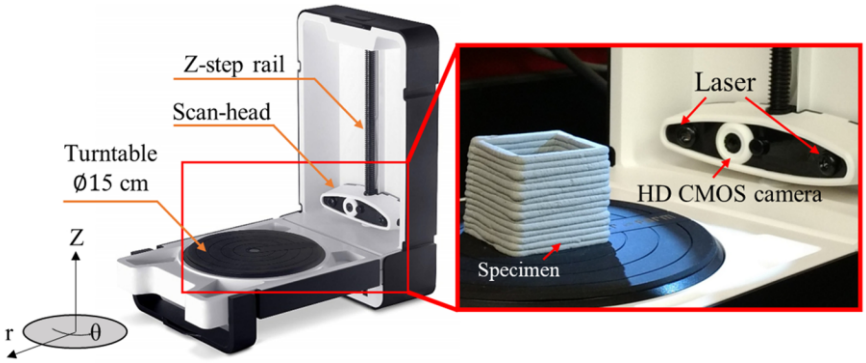


Figure 1.8: Setup for scanning as provided by (Nair et al. [2021]).

Table 1.1: Summary of methods based on optics for 3DCP monitoring.

Work	Sensor	Measurements
Wolfs et al. [2018]	Time of flight	Layer to nozzle height
Lindemann et al. [2019]	Laser triangulation	Layer to nozzle height
Kazemian et al. [2019]	RGB camera	Top layer width (top view)
Davtalab et al. [2020]	RGB camera	Layer orientation (lateral view)
Nair et al. [2021]	3D scanner	Reference 3D cloud to piece 3D cloud distance distribution (lateral view)

vision, monitoring of the surface has been ignored in 3DCP. In fact, surface analysis for inline monitoring of small-scale extrusion is an active topic of research (Oleff et al. [2021]). Among the methods based on computer vision, the ones based on texture analysis for surface quality assessment can be extended for the monitoring of extruded concrete. In this work, we propose to detect anomalies in terms of both layer geometry and surface analysis.

Towards this goal, there are different methodological restrictions. On one hand, since this type of monitoring of the printed pieces is a relatively new field of study, there is an absence of standard norms for quality evaluation. Because of this, there are no strict guides to be used at the moment of proposing monitoring methods. In fact, there is a wide variety of possible defects that can appear in the piece, both in terms of its geometry and texture (see Figure 1.5 for some examples). Furthermore, given the complexity of the printing process, it is difficult to reproduce these errors; moreover, producing defective pieces on purpose represents a time and monetary lose. Additionally, the robots used for printing were not designed to integrate the additional sensors that are currently being proposed for the monitoring of the printed pieces. Consequently, a proper integration of these devices (fixing, wiring, calibrating, etc.) is a challenge to solve in advance.

The above-mentioned restrictions can explain the absence of public data for the evaluation of defects in pieces produced by 3DCP. Consequently, there is an absence of benchmarks for evaluation and comparison between different monitoring methods. Given the context discussed in this section, next we describe the contributions of this thesis.

1.2 Thesis Contributions

As discussed previously, there are many areas of opportunity for the digitalization and automatization of the construction industry. In this work, we propose to perform automatic construction monitoring based on computer vision. The scope of this work is divided into two stages of the construction life cycle: damage monitoring after the construction is finished, and quality monitoring of a 3DCP-based construction process.

For damage monitoring, we work on pixel-accurate crack segmentation. For quality monitoring, we work on inline characterization of the printed pieces for anomaly detection based on geometrical and textural analyses. Next, we present a summary of the contributions of this thesis based on these topics (see Figure 1.9).

1.2.1 Crack Segmentation

Among the possible damages in constructions, cracks are an evident sign that can be identified through a visual inspection. Properties of the cracks such as their orientation, length and number allow inferring the mechanisms that generated the defect. Among other properties, the width is one important indicator of damage severity and future durability. To allow an accurate, automatic mea-

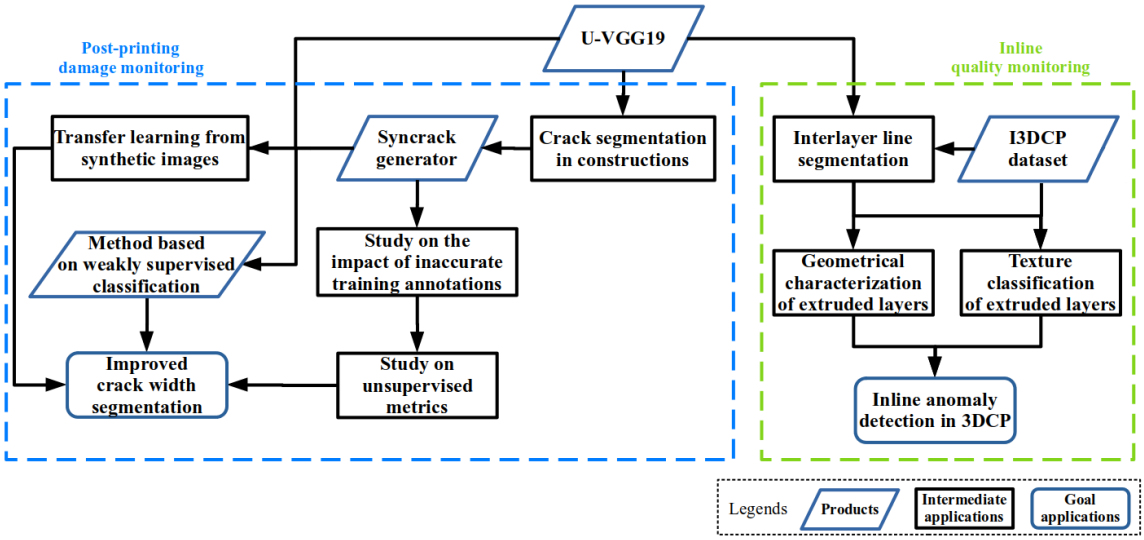


Figure 1.9: Summary of this thesis' contributions and their interactions.

surement of the geometrical properties of the located cracks, it is important to provide a pixel-accurate segmentation.

The U-VGG19 network. For the automatic segmentation of cracks, we proposed the U-VGG19 architecture (Rill-García et al. [2022e]). This network consists of an encoder-decoder architecture inspired by U-net, which uses skip connections between the encoder and the decoder at different resolutions. As an encoder, we use the convolutional layers of VGG19; the decoder is built to be symmetrical. To ease the training, we use a transfer learning approach by initializing our encoder with the weights of VGG19 trained on ImageNet.

U-VGG19 is used with a supervised learning approach i.e. it is trained with previously produced segmentation maps. However, since this segmentation task is difficult even for humans, the manual annotations are prone to multiple errors. Particularly, these annotations tend to be wider than the cracks in the analyzed images. This introduces the problem of learning in the presence of inaccurate annotations.

The Syncrack generator. To study the problem of inaccurate annotations for supervised crack segmentation, we developed the Syncrack generator. This tool allows generating parametrizable synthetic images, emulating pavement/-concrete textures with cracks in them; these images are accompanied with a corresponding ground truth segmentation of the cracks. Additionally, the Syncrack generator includes a module that allows introducing inaccuracy to the segmentation maps, emulating the inaccuracy of manual annotations. The inaccuracy of the annotations produces mislabeling at the level of pixels, which is considered as label noise. Similarly to the module in charge of producing synthetic images, this module is parametrizable to control the noise introduced to the annotations. A dataset obtained with the beta version of Syncrack was used in (Rill-García et al. [2022e]), and the generator was formally presented in (Rill-García et al. [2022f]).

A study on the impact of inaccurate training annotations and unsupervised metrics for crack segmentation. With the help of Syncrack, we studied

the detrimental impact on the predicted segmentations when training with inaccurate labels (Rill-García et al. [2022e,f]). To the best of our knowledge, we are the first to provide this kind of analysis. With this study, we showed that using inaccurate annotations for training deteriorates the accuracy of the width of the located cracks. Specifically, we observe an increase of recall and a decrease of precision, even when the noise in the annotations is bi-directional i.e. some crack pixels are mislabeled as background and vice versa.

The evaluation based on precision and recall is reliable only when accurate ground truths are available. This is the case for Syncrack-generated data, but not necessarily for manual annotations. As an additional option for evaluation, we proposed the use of unsupervised scores (Rill-García et al. [2022f]), based on metrics previously used for unsupervised segmentation. These metrics are the first and second order region entropies, and the Kolmogorov-Smirnov score. On Syncrack-generated data, we showed that these scores have a direct relation with the quality of the predicted segmentations in terms of supervised precision. This allows detecting when the predictions are wider than the located cracks i.e. the produced segmentations contain background pixels surrounding the visible cracks.

An improved crack segmentation based on weakly supervised learning. From a machine learning perspective, the inaccuracy of the annotations is a particular case of noisy labels at pixel level. Training when mislabeling exists is called inaccurate supervision, a sub-case of weakly supervised learning. To improve the geometry of the predicted crack segmentations when training in presence of inaccurate annotations, we proposed a method based on two steps (Rill-García et al. [2022e]): 1) getting a set of new pseudo-labels from the original raw data and 2) training a final model using these pseudo-labels. For the generation of these pseudo-labels, we tested 4 methods inspired from weakly supervised classification.

In Syncrack-generated data, we showed that our approach produces a significant improvement with respect to training with inaccurate annotations. In terms of F-score, this improvement goes up to 11%; the model trained with pseudo-labels is less than 2% below the model trained with accurate annotations. We applied this method on real images too, showing that it is applicable to real data; the produced segmentations exhibit a geometry closer to the visible cracks, particularly in terms of crack width.

An improved crack segmentation based on transfer learning. Previously, we used Syncrack-generated data as a benchmark to evaluate the detrimental impact of inaccurate labels during training, as well as to measure the improvement brought by our method based on weakly supervised learning. In addition to that, in this work we proposed a second approach to improve crack segmentation based on transfer learning: training a model with Syncrack-generated images and transferring the learned model to real images (Rill-García et al. [2022f]).

The models trained solely with Syncrack-generated images were able to segment cracks on real-life images. Furthermore, these segmentations are more accurate in terms of crack width.

1.2.2 Inline Characterization and Anomaly Detection

The presence of defects during a 3D printing process leads to anomalies in the constructed piece. In this work, we provide an inline characterization of 3D concrete printing for anomaly detection. This characterization is based on the analysis of the geometry of the extruded layers, as well as their texture (Rill-García et al. [2022g]).

The I3DCP dataset. This dataset, used for the experiments presented in this work, was collected during a printing session by fixing a camera and a lamp to the extrusion nozzle of the robot. This fixed point of view with respect to the extrusion zone allows performing a local inline monitoring of the printed piece. The camera position and orientation are fixed so that the acquired images show a lateral view of the piece; this allows observing the interlayer lines i.e. the lines separating independent layers. Our geometrical characterization is based on the analysis of these lines.

Interlayer line segmentation. To analyze the interlayer lines, first it is necessary to segment them. To provide this segmentation, we used U-VGG19 as baseline model. However, this network requires a supervised training. To train our final model, we used a semi-supervised learning approach, beginning from an initial set of 32 manually annotated images. This allowed us to iteratively improve the learned model and to increase the number of annotated images, producing a total of 128 additional annotated images. The final model tuned on these 128 images obtained an F-score of 91% on the manually annotated images (which were never seen during training by the final model). The segmentation maps produced by this model can be used to characterize the printed piece in search of anomalies.

A monitoring method for 3D concrete printing based on geometrical characterization and anomaly detection. We use image processing to determine the local geometry of the printed piece, based on an analysis of its observable interlayer lines. This analysis consists of first measuring their orientation and curvature, the thickness of the layers contained between each pair of lines, and the distance between the deposition layer and the extrusion nozzle.

These measurements are provided at pixel level. Consequently, per measurement type, we obtain a distribution of the values measured per pixel. These distributions serve as a summary that characterizes the analyzed image. The geometrical anomalies are then defined and located as the regions with measured values outside the ranges of admissible values, as defined by the operator. This allows returning the location, nature and severity of the detected anomalies.

Additionally to analyzing the geometry of the layers, the interlayer line segmentation allows segmenting each printed layer separately. In this way, we can analyze the surface of each layer independently; particularly, we analyze their texture.

A monitoring method for 3D concrete printing based on textural characterization and anomaly detection. The principle behind this method is that the water content of the extruded material is an indicator of potential issues, with impact in the structural performance of the printed piece. Since the excess or shortage of water changes the visible texture of the printed layers, an anomaly

in the observed texture is an indicator of possible defects in the process. To the best of our knowledge, we are the first to propose a monitoring method for 3DCP based on texture.

This method uses a machine learning approach to classify a printed layer, window-wise, using textural descriptors. These descriptors are based on gray level co-occurrence matrices and local binary patterns. After feature extraction using such descriptors, the texture windows are classified either as *good* or as one of three defective cases: excessively fluid, excessively dry or containing superficial tearing. The anomalies are then defined as the windows in an image that are assigned with a class other than good by the classifier.

For our experiments, we produced a set of 111 labeled windows as a subset of I3DCP. In the test split of this set, our trained model (a small convolutional network) obtained a macro-averaged F-score of 94%.

In the next section, we close this chapter by summarizing the structure of the remaining content in this manuscript.

1.3 Thesis Structure

As described in section 1.2, our contributions are grouped into two topics: pixel-accurate crack segmentation for damage monitoring (chapter 2 and chapter 3) and inline characterization for anomaly detection in the printed pieces (chapter 4, chapter 5 and chapter 6).

In chapter 2, we begin to study the problem of crack segmentation. First, we introduce the topic of crack segmentation in construction. Then, we present a literature review on methods used for crack segmentation in constructions. Afterwards, we discuss our baseline model proposed for crack segmentation: the novel U-VGG19 architecture using a transfer learning approach. Next, we present preliminary results comparing U-VGG19 with state-of-the-art methods, ignoring the inaccuracy of the manual annotations (as commonly done in the literature). We close this chapter with a discussion on the obtained results.

In chapter 3, we formalize the problem of learning in the presence of inaccurate labels and discuss its importance for crack segmentation. After this discussion, we introduce Syncrack; consequently, we use Syncrack-generated data as a benchmark to study the detrimental impact of training with inaccurate labels. Next, to improve segmentation in the presence of inaccurate annotations, we discuss our method inspired by weakly supervised classification. Afterwards, to approach the problem of inaccurate annotations during labeling itself, we study the transfer of models trained on Syncrack-generated data to real-life images. Finally, we close the chapter with a global discussion about the obtained results and future perspectives.

In chapter 4, we study the problem of interlayer line segmentation. First, we provide an introduction on the relevance and difficulty of segmenting these lines for the inline monitoring of 3DCP. Then, we discuss the setup used to acquire the images used in our experiments, as well as the properties of the images in the resulting dataset: I3DCP. Next, we provide a summary of U-VGG19, the model selected as baseline for our experiments. Afterwards, we discuss our method for

semi-supervised learning beginning from a small number of annotated images. Then, we show our experiments and results for the segmentation of interlayer lines. Finally, we close the chapter with a discussion on the obtained results and future perspectives.

In chapter 5, we study the proposed method for layer characterization and anomaly detection based on the geometrical analysis of segmented interlayer lines. First, we introduce the concept behind our method. Next, we provide a brief summary of the state of the art on geometrical monitoring of 3DCP. Then, we describe the methods used to measure the properties proposed in this work: the interlayer lines' orientation and curvature, the layers' thickness, and the relative height of the printing nozzle with respect to the last printed layer. Subsequently, we discuss how these measures can be summarized to characterize the observed layers and to detect anomalies. Afterwards, we illustrate our technique applied to two study cases. After that, we show preliminary experiments on additional applications of the proposed method. We close the chapter with a final discussion on the obtained results and some future perspectives.

In chapter 6, we study the proposed method for layer characterization and anomaly detection based on texture classification. First, we discuss the principle behind the proposed method. Next, we provide a brief literature review on texture analysis focused on (small-scale) additive manufacturing. Then, we discuss the feature extraction approach that we used for classification (we describe the selected classes and texture descriptors). Consequently, we analyze the produced features and discuss about the model used to classify the textures. Afterwards, we show the results of our experiments using the proposed method. We close the chapter with a discussion about the obtained results and some future perspectives.

Finally, in chapter 7, we conclude the manuscript. First, we provide a summary of the scope of this work, as well as its contributions. We close the chapter proposing some future research lines.

Chapter 2

Crack Segmentation in Construction Materials

Periodic supervision of constructions is important to ensure their security and estimate their future durability. For structural monitoring, crack inspection plays an important role since the cracks are an evident sign of damage in concrete structures (Yang et al. [2018]) and roads (Coquelle et al. [2012]). The cracks can be identified by visual inspection, therefore crack detection can be approached from the perspective of computer vision.

According to the Oxford Advanced Learner's Dictionary, a crack is a line on the surface of something where it has broken but not split into separate parts. From the point of view of computer vision, crack detection consists of locating these lines from background in a given image. When this detection is performed at the level of pixels, we talk specifically about segmentation i.e. deciding, per pixel, a class: either crack or background. An accurate segmentation of the cracks allows calculating many properties such as their position, orientation, density, length, width, etc. The severity and potential causes of the detected cracks can be inferred from this type of properties (Gao and Mosalam [2018]). In the context of this work, our final goal is to perform an accurate crack segmentation in construction materials.

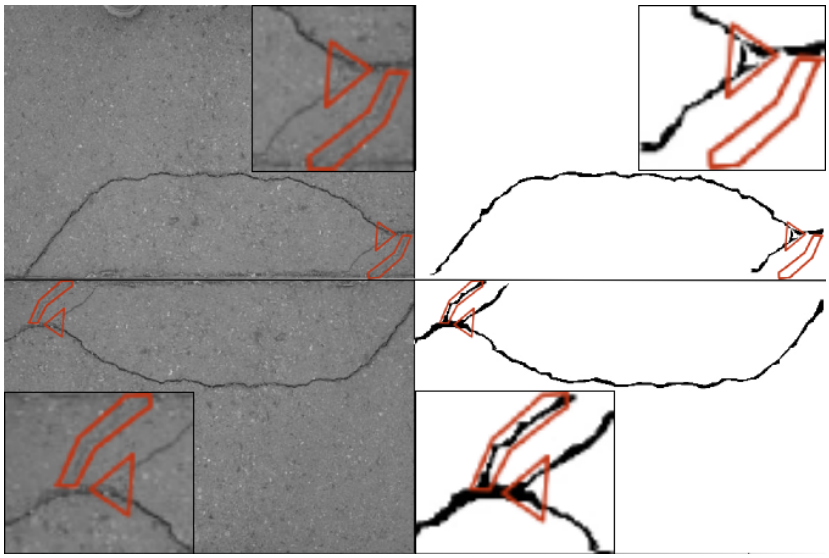
The most successful methods for crack segmentation in constructions are based on supervised learning. However, an accurate segmentation is still a challenging problem. The background textures are diverse, noisy and usually non-stationary; also, the cracks exhibit complex and diverse geometric shapes: this increases the probability of missing low-contrast cracks or even confusing background noise as a crack. Furthermore, the images analyzed to detect cracks are typically acquired from a distance that maximizes the area captured in a single image. In consequence, the width of the cracks in the images is usually on the limit of the resolution of the acquisition devices. Because of this, the boundary between crack and background in an image is often fuzzy. Moreover, the accumulated area of the cracks typically represents a very small percentage of the total area of the analyzed images; from a machine learning perspective, this means that the class of interest is underrepresented.

Given all the aforementioned challenges, obtaining an accurate segmentation for training is a difficult task even for humans; consequently, manual annotations

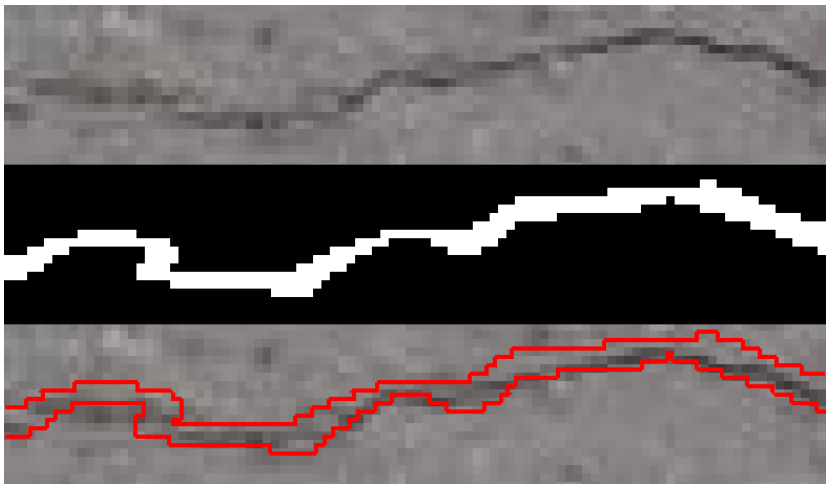
are prone to error. These errors can be divided into two levels (see Figure 2.1):

- On the level of objects. The false negatives on this level refer to mislabeling whole cracks or segments as background; the false positives refer to mislabeling noise in the background as an independent crack. We can relate this level to the crack detection task

- On the level of pixels. The false negatives on this level refer to mislabeling the pixels in the outer regions of a crack as background (making the annotation thinner than the crack); the false positives refer to mislabeling the pixels around a crack as part of the crack (making the annotation wider than the crack). A combination of both false negatives and positives on this level can also result in an inaccurately placed annotation. We can relate this level to the geometry of the detected cracks



(a)



(b)

Figure 2.1: Examples of inaccurate annotations in the public CrackForest dataset (Shi et al. [2016]). a) Errors on the level of objects. b) A zoom to errors on the level of pixels; the first row shows the raw image, the second one shows a manual annotation, and the third one shows the image with a red border surrounding the annotation.

Using these inaccurate annotations for supervised learning adds another layer of difficulty to the problem. This is particularly relevant when we expect that the produced segmentations respect geometrical properties of the cracks such as their width.

Regarding the context of this thesis, we did not have access to enough samples of 3D printed pieces with cracks; to validate our proposals, we studied the problem on a closely related domain with publicly available datasets: road crack segmentation.

The rest of the chapter is structured as follows: first, we present a literature review on methods used for crack segmentation in constructions. Then, we discuss our baseline model proposed for crack segmentation: the novel U-VGG19 architecture using a transfer learning approach. Next, we present preliminary results comparing U-VGG19 with state-of-the-art methods, ignoring the inaccuracy of the manual annotations (as commonly done in the literature). We close this chapter with a discussion about these results.

2.1 State of the Art on Crack Segmentation

Surface analysis for automatic crack detection in constructions has been a topic of interest for many years. Early attempts on crack segmentation, which could be categorized as conventional methods, relied on a particular property: the cracks tend to be darker than the background because of the shadow produced within (see Figure 2.2).

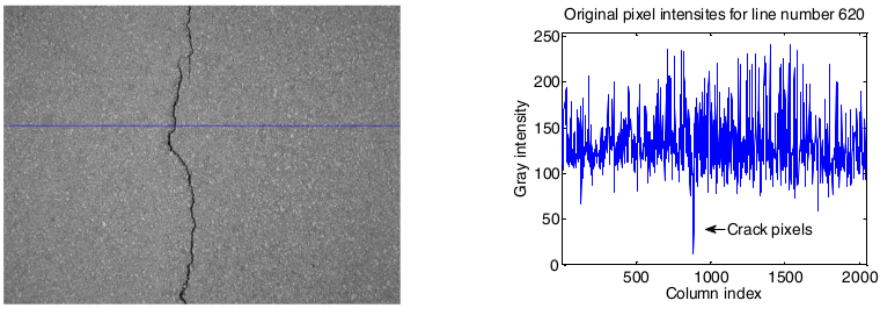


Figure 2.2: Pixel-intensity profile along a row including a crack as illustrated by (Oliveira and Correia [2014]).

2.1.1 Conventional methods

Among these methods, we can find diverse approaches:

Thresholding. The simplest conventional methods were based only on threshold selection to detect dark pixels. This selection has been performed with diverse techniques such as *ad hoc* histogram analysis (Acosta et al. [1992]; Kirschke and Velinsky [1992]), the Otsu's method (Akagic et al. [2018]), entropy-based fuzzy logic (Cheng et al. [1999]), dynamic thresholding based on stacked thresholds (Oliveira and Correia [2009]), etc.

Edge detection. Other approaches were based on edge detection with techniques such as the fast Haar transform, the fast Fourier transform, Sobel and

Canny (Abdel-Qader et al. [2003]). Sobel and Canny are examples of techniques based on gradient calculation using oriented filters. The use of steerable filters can be considered as an extension of this concept (Aldea and Hégarat-Mascle [2015]).

Although the thresholding and/or edge detection are usually performed after some pre-processing to reduce the background noise, these techniques are prone to produce false positives, specially in complex textures. Indeed, thresholding and edge detection ignore the spatial properties of the cracks; with this in mind, they are usually followed by other strategies such as:

Mathematical morphology. The use of mathematical morphological tools allows cleaning the segmentation by connecting discontinuous crack segments, identifying elements with privileged directions (high likelihood of being cracks), and discarding small, disconnected elements with a low likelihood of belonging to cracks, as discussed by (Tanaka and Uematsu [1998]). Further post-processing is possible, such as in the work presented by (Tang and Gu [2013]); the authors proposed to use a B-spline snake model, in order to refine the crack segmentation after a cleaning based on morphological operations.

Probability. Another approach using morphological, incomplete path openings, was proposed in (Dokládal [2017]). There, the threshold selection was obtained using a binomial distribution, inspired from the perception law stipulating that a shorter crack must be more contrasted to be visible in noise. In (Vandoni et al. [2016]), the cracks are represented as groups of line segments. A line segment is interpreted as a marked point, and an algorithm based on a marked point process is used to determine if a line segment belongs to a crack; the algorithm also can produce new line segments connecting existing ones. This method is complemented with a post-processing step, based on minimum cost paths to improve connectivity.

Graph theory. Similarly, we find approaches that analyze the pixels obtained after thresholding/edge detection from a graph perspective. Among these, we find the use of Minimal Path Selection (Amhaz et al. [2016]) to connect these pixels in independent image patches seen as graphs. The CrackTree method (Zou et al. [2012]) builds a crack probability map using tensor voting to enhance the connection of the crack fragments with good proximity and curve continuity. It represents a set of crack seeds from the map as graph, derives minimum spanning trees from that graph, and performs recursive tree-edge pruning to keep only desirable cracks.

Although these approaches allow reducing false positives, addressing some spatial properties of the cracks to discard noise, they were superseded by machine learning approaches.

2.1.2 Machine Learning Methods

Supervised learning approaches provided higher scores on crack segmentation than conventional methods, and they showed to be able to cope with more complex images, even containing intrusive objects. Among these methods, we find examples such as CrackIt (Oliveira and Correia [2014]), which performs both pixel and region-based classification based on block mean and standard devia-

tion. In this work, both supervised and non-supervised machine learning methods are applied. In CrackForest (Shi et al. [2016]), a method based on random structured forests is used to predict a crack patch of structured tokens as a preliminary crack detection result. Then, the structured tokens are used to construct a crack descriptor fed to a SVM to discriminate cracks from the noise. In (Wu et al. [2014]), a bottom-hat transform is used to enhance crack-like regions and a MorphLink-C method is used to connect regions; connected regions are classified as crack or non-crack by extracting 6 geometrical features and feeding them to a multilayer perceptron.

Other works perform feature extraction through filtering. In (Cord and Chambon [2012]), for example, the classification is based on multiple descriptors relying on structural analysis, the Fourier transform and steerable filters. In (Zalama et al. [2014]), the features are extracted using Gabor filter banks. In both cases, the learning is based on the AdaBoost algorithm: each weak classifier corresponds to a threshold selection for each individual feature in the extracted feature vector.

The works presented in this section follow a similar strategy: identifying patch-wise the apparent existence of cracks, extracting features from a crack patch, and using those features to identify crack pixels/regions. The supervised learning approach allowed the reduction of false positives by discriminating noise and intrusive elements from actual cracks, but it still lacked pixel-wise precision. This lack of precision was constrained mainly by the hyperparameter selection for the features used for classification.

2.1.3 Deep Learning Methods

As in many computer vision fields, methods based on neural networks surpassed traditional learning algorithms for crack detection and, subsequently, segmentation. A typical convolutional neural network is composed of two main sections: 1) a feature extractor, composed by convolutional layers, and 2) a classifier, composed by fully connected layers. In simple terms, the convolutional layers are filters that are learned during training, allowing automatic feature extraction.

Convolutional Networks for Crack Detection

Beginning by crack detection, in 2017, (Eisenbach et al. [2017]) cropped high-resolution pavement images (1920×1080) into small patches to train ASINVOS-net, a CNN aiming to classify each patch (64×64) as containing or not distress. In 2018, (Kim and Cho [2018]) took a similar approach for crack detection on concrete structures by fine-tuning AlexNet. To achieve more precise detections, the authors use overlapping sliding windows and get crack-probability maps based on the average predicted probability of overlapping windows. For simultaneous distress segmentation, (Mandal et al. [2018]) used the YOLO v2 framework; this avoided the necessity of cropping images or using sliding windows but provided detections in terms of rough bounding boxes.

In fact, the fully connected layers of a typical convolutional neural network require fixed input and output sizes; this limited the possibility of producing

single-pixel predictions without cropping an image. The posterior introduction of fully convolutional networks (FCN) solved this issue, and it brought the possibility of efficiently providing pixel-accurate detections of multiple cracks in a single image. This type of networks, which are considered as generative models and do not use fully connected layers, is the current state of the art for crack segmentation.

Generative Networks for Crack Segmentation

Fully convolutional neural networks allow generating segmentation maps from images i.e. a class label is assigned to each pixel individually. For crack segmentation, many models nowadays are inspired by the original concept of U-net. U-net was first introduced for biomedical image segmentation (Ronneberger et al. [2015]). The base architecture of U-net is the one of an auto-encoder: a contracting path (encoder) to capture context information at different resolutions, followed by a symmetric expansive path (decoder) to enable precise localization. The context captured by the encoder at each resolution is transmitted to the decoder through skip connections as seen in Figure 2.3. This provides the decoder with variate information about the encoding process at different resolutions, allowing it to produce more detailed output images.

Standard U-nets. The work of (Escalona et al. [2019]) compared the performance of U-net architectures for road crack segmentation with different encoder depths. Their work showed that U-net architectures could surpass a modified version of the pre-trained VGG16, in which the fully-connected layers were re-

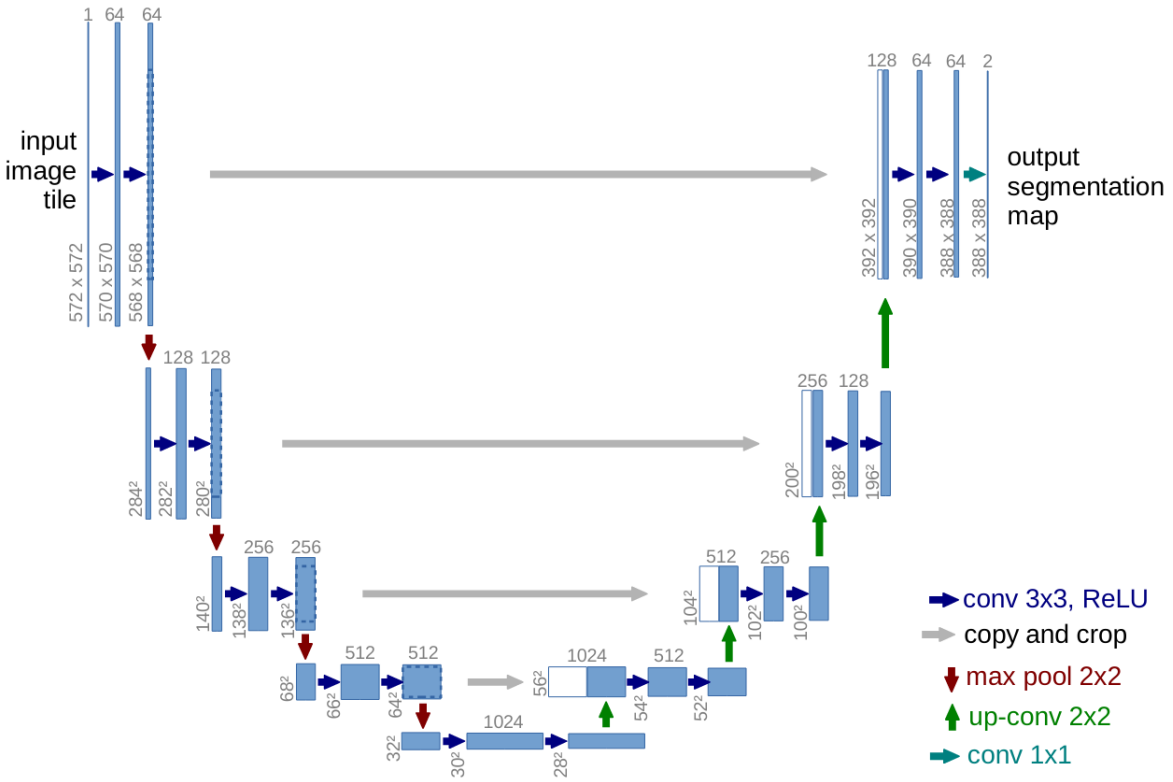


Figure 2.3: U-net architecture as illustrated by (Ronneberger et al. [2015]). Blue boxes correspond to multi-channel feature maps. White boxes represent copied feature maps.

placed by deconvolutional layers. For concrete crack segmentation, (Liu et al. [2019c]) showed the superiority of U-net with respect to previous FCNs in which the decoder was composed by deconvolutional layers only (without connections with the encoder).

Multi-resolution analysis and attention mechanisms. To enhance the ability to extract contextual information at different resolution scales, a U-net variant inspired by PSPNet was proposed by (Sun et al. [2020]). In this architecture, the blocks of convolutional layers from the standard U-net are substituted with multi-scale blocks. Additionally, the bottleneck between the encoder and the decoder is substituted with residual blocks, surpassing the scores of the original U-net. A similar concept was proposed by (König et al. [2019]), with the addition of attention gates to propagate only relevant activations from the encoder to the decoder. Similarly, this approach surpassed the scores obtained by the original U-net.

Following the idea of attention mechanisms, (Fang et al. [2021]) added a self-attention layer on top of the last bottleneck layer of a U-net. This architecture is one of the three components of their method: image preprocessing, deep neural network and data augmentation (dilating the annotations used for training). They found out that augmenting data with wider (dilated) crack annotations helped to increase the recall of the predictions; however, models trained with this data augmentation approach are susceptible to rapidly decrease their precision as the annotations become wider.

Another approach that used different resolution scales was proposed by (Yang et al. [2020]). In this work, the feature maps from the last layer at each resolution in the decoder are deconvolved to be compared with the target annotation. These deconvolved feature maps are further merged simply by a concatenation followed by a convolution with a 1×1 kernel. The loss function to optimize consists of a weighted sum of the independent losses of each deconvolved map and the final fusion. A similar approach was used before by (Zou et al. [2019]), but using SegNet as a basis. In this work, SegNet was extended in order to decode using information from the encoder at different resolutions, similarly to the principle of U-net's skip connections.

To further exploit multi-scale information extraction, similarly to (Sun et al. [2020]), the architecture presented by (Yang et al. [2020]) was extended with a multi-dilation module at the bottleneck by (Fan et al. [2020b]). This module allows obtaining multiple-context-size features by using dilation convolutions with different dilation rates.

Generative adversarial learning. Another recent tendency to produce better crack segmentation is based on generative adversarial networks (GANs). The key concept behind GANs is training two models simultaneously: 1) a generative model, used to capture the distribution of interest and to create synthetic samples as if they were part of that distribution; 2) a discriminative model, used to identify if an input sample belongs to a distribution different from the one of interest. The final goal is having a generator good enough to produce outputs indistinguishable from the real data, even for an adversary that has been specifically trained to discriminate synthetically generated samples. For example, (Gao et al. [2019]) proposed to use a GAN approach to create crack segmentation

maps resembling manual annotations. In their work, they compared different U-net variants as generative models. U-net was used again with adversarial learning by (Zhang et al. [2020b]), substituting the last block of convolutional layers in U-net's decoder with a discriminator. However, instead of producing segmentation maps, this work classifies small patches as containing or not a crack.

As seen from the literature review in this section, generative models based on DL have dominated crack segmentation in construction surfaces. However, the complexity of these models has been increasing drastically. Unlike other fields in computer vision, the amount of annotated data for crack segmentation has not increased enough to cope with the increasing complexity of the networks proposed to solve the task. Furthermore, because of the difficulty to provide accurate annotations, the labels used for training are usually inaccurate at pixel level. As a result, the latent risk of overfitting to small, inaccurate datasets increases over time. The problem of low cross-dataset generalization has been discussed in works such as the one presented by (Shi et al. [2022]). Rather than developing a complex network, the authors propose to introduce a novel image transformation that can be plugged at the input of any CNN, allowing the extraction of more meaningful features for curvilinear object segmentation.

However, this type of approach is not able to deal with inaccurate annotations in supervised learning. Indeed, in parallel to increasing the amount of accurate data available for training, it is necessary to develop crack segmentation methods able to effectively learn in the presence of inaccurate annotations (rather than creating more complex methods that maximize scores with respect to inaccurate ground truths). In the next section, we discuss our baseline model used to study this problem: U-VGG19.

2.2 U-VGG19: Crack Segmentation in Construction Materials

As discussed in the literature review from the previous section, methods based on U-net have shown very promising results for crack segmentation. Particularly, the skip connections between a contracting path and an expansive path allow capturing contextual information useful to connect pixels within a thin but long structure. This intuition is supported by the findings of (Ai et al. [2018]), who demonstrated that neighborhoods contain critical information for crack detection. They also demonstrated that different neighborhood sizes have significant impact on the segmentation results.

However, these supervised methods require segmentation ground truths to be trained. Not only accurate segmentation ground truths are difficult and expensive to obtain, but methods based on DL require high volumes of data to avoid overfitting. When producing more labeled data is not a feasible option, a common approach is fine-tuning models trained previously for other tasks in the same domain.

A standard fine-tuning strategy for image classification consists of keeping the convolutional layers trained for feature extraction and substituting the last layers used for classification with an architecture fitting the new problem. This

new architecture is then trained with images from the new target domain. Since the convolutional layers from the pre-trained model are expected to be able already to extract features, it is usual to "freeze" their weights during the new training (i.e. not updating their values). Although, depending on the new target domain, this is not always useful.

In this way, the knowledge acquired from training on a certain domain can be transferred to a new one to avoid learning from scratch: a transfer learning. In the current work, we propose to use transfer learning to deal with few annotated data on public datasets.

Similarly to other works focused on crack segmentation, we base our solution on the principle of U-net: a symmetrical encoder-decoder network with skip connections to feed the decoder with the feature maps generated by the encoder at different resolutions. The encoder is used to extract representative features from the input image at different scales (it is a contracting path). As such, any feature extractor used originally for image classification can be used as an encoder.

In our context, the objective is to find thin, long elements in a given texture (such as concrete and pavement). We propose to use, as an encoder, a pre-trained network employed for texture recognition. The selected network was VGG, the architecture developed by the Visual Geometry Group at the University of Oxford (Simonyan and Zisserman [2014]). Although this network is meant for image classification, (Gatys et al. [2015]) showed the promising ability of the feature maps obtained from VGG's convolutional layers (trained on ImageNet) to recognize textures.

Furthermore, VGG has been an attractive feature extractor to detect and segment cracks. For example, (Escalona et al. [2019]) used the pre-trained VGG16 (the 16-layers version of VGG) by substituting its dense layers with deconvolutional ones. For black box images, (Bang et al. [2019]) built a generic decoder able to be concatenated with different encoders. Using a pre-trained VGG16 encoder allowed the model to surpass even a model based on ResNet152 (without using pre-trained ResNet weights). To improve U-net for crack segmentation, (Zhang et al. [2020c]) tested different encoders created by cropping the VGG19 architecture at different depths.

Moreover, loss functions based on VGG-extracted feature maps have been used for texture generation. Particularly, as an alternative to reduce the impact of limited amounts of available annotated data, (Mazzini et al. [2020]) used a semantic texture generation approach for data augmentation. This improved their model's scores in real images.

Inspired by this, we used the convolutional layers of VGG19, pre-trained on ImageNet, as an encoder (see Figure 2.4.A). Then, inspired by U-net, we created a symmetrical decoder (see Figure 2.4.B) to replace the dense layers used by VGG19 for classification. As discussed previously, the skip connections between the encoder and the decoder (see Figure 2.4.C) are a key element to preserve contextual information for the final crack segmentation.

Our resulting network, referred to as U-VGG19, provides a way to obtain accurate segmentation maps of cracks. All the convolutional layers use kernel size 3, stride 1, and a ReLu activation. The only exception is the final layer (Figure 2.4.D), using kernel size 1 with a sigmoid activation to obtain a single

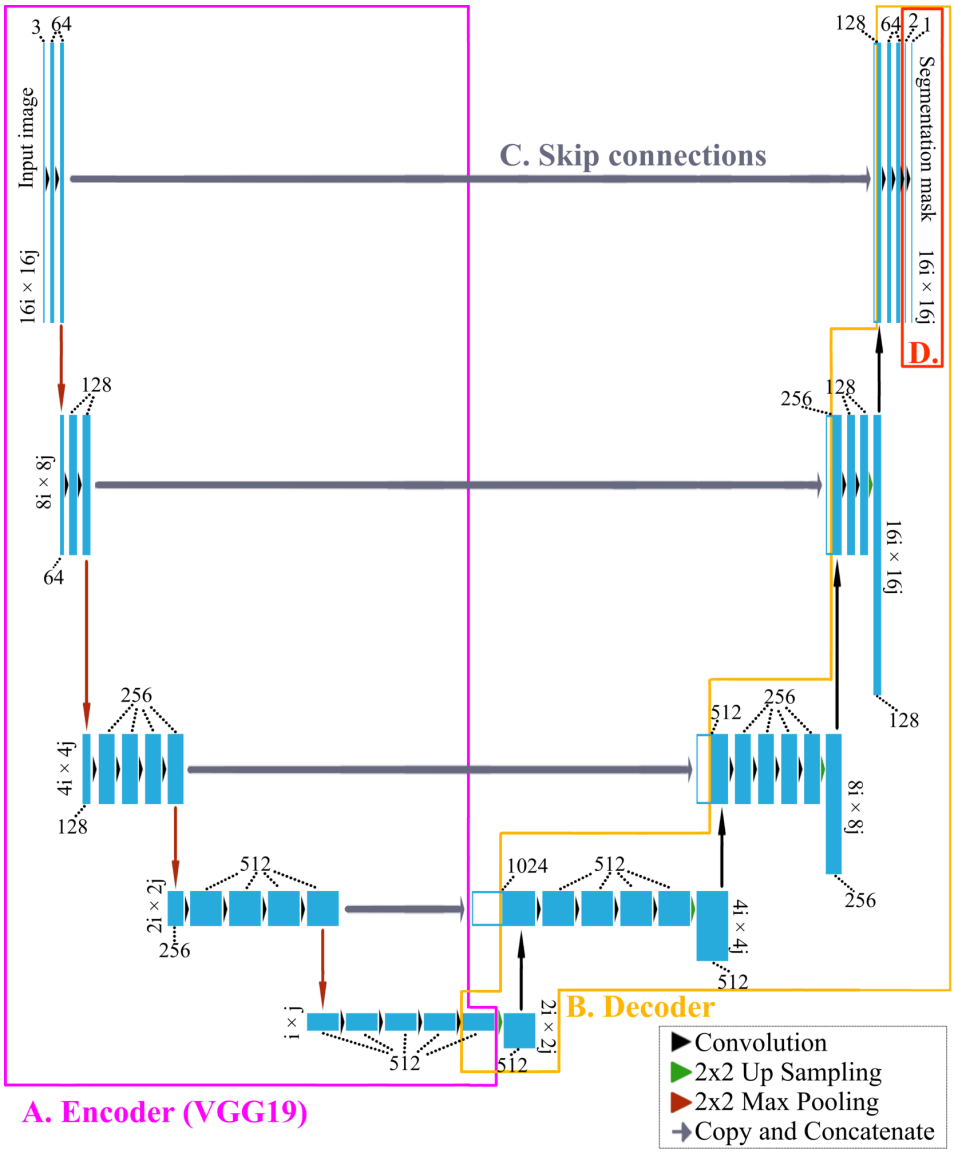


Figure 2.4: Our U-VGG19 architecture.

channel image with values in the range (0, 1): per-pixel probability of belonging to a crack. The downsampling uses 2×2 max pooling; the upsampling uses 2×2 nearest-neighbor interpolation. Since we downscale the image 4 times, the resolution of the input image should be in multiples of 16 ($16i \times 16j$, with $i, j \in \mathbb{Z}^+$, as shown in Figure 2.4). U-VGG19 has 39,236,101 trainable parameters, in contrast with the 143,667,240 parameters from the full VGG19 network.

Next, we test U-VGG19 in publicly available road crack segmentation datasets.

2.3 Road Crack Segmentation

As previously discussed in this chapter, crack segmentation in constructions is not a trivial task. When it comes to segmenting cracks in roads, the problem becomes more challenging because of uncontrolled image acquisition conditions: irregular and variable shadows and illumination conditions; intrusive background textures due to external surface conditions (humidity, dirt, sand,

painting); intrusive objects (leaves, lane signs, manhole covers); etc.

In the following sections, we compare the results of U-VGG19 with the state of the art on publicly available images with manual annotations for road crack segmentation. Based on these results, we show and discuss the shortcomings of the current approaches based on supervised learning using these inaccurate labels for training and testing.

2.3.1 Road Crack Segmentation Datasets

We perform our experiments on two different datasets:

CrackForest Dataset (CFD): This public dataset presented in (Shi et al. [2016]) contains 118 images collected from urban roads in Beijing, China. The images were taken by an iPhone5 with focus of 4mm, aperture of f/2.4 and exposure time of 1/134s. The collected images contain perturbations such as shadows, oil spots, and water stains. The image size is 480×320. Two independent annotations are provided per image: crack borders and crack segmentation; for our experiments, we used only the segmentation annotations. As suggested by (Sun et al. [2020]), we removed some images with clear annotation errors; we preserved a total of 108 images.

Aigle-RN: A subset from a bigger database presented in (Amhaz et al. [2016]). Unlike the other subsets (collected by laser), Aigle-RN is captured using cameras with stroboscopic lights. It contains 38 annotated images with variable sizes collected at traffic speed for periodically monitoring the French pavement surface condition. They have been pre-processed to mitigate the influence of non-uniform lighting conditions.

2.3.2 Loss Function and Evaluation Scores for Crack Segmentation

The cracks in an image typically represent a very small percentage of all the pixels. Given this high class imbalance, as pointed out by (Zhang et al. [2020b]), a naive loss function will lead to the "all black" problem: the network will simply converge to treating the entire input image as background.

Classical approaches to deal with this include class weighting (Yang et al. [2020]). However, by overweighting the under-represented class, the model will have a bias towards false positives. Instead, we adopted the approach proposed by (Sun et al. [2020]): a hybrid loss function using the Dice Score Coefficient (DSC). This score, used for segmentation evaluation, represents the ratio of the area of intersection of two objects to the total area. Given a ground truth set A and a prediction set B, both of which contain the pixels identified as the class of interest in a segmentation map:

$$\mathrm{DSC} = \frac{2|\mathrm{A} \cap \mathrm{B}|}{|\mathrm{A}| + |\mathrm{B}|} \tag{2.1}$$

This definition is mathematically equivalent to the F-score, which is the evaluation score typically used in the literature for crack segmentation. Expressed

in terms of boolean predictions at pixel level, using the definition of true positive (TP), false positive (FP), and false negative (FN):

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2.2}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2.3}$$

$$\text{F-score} = \frac{2}{\text{Precision}^{-1} + \text{Recall}^{-1}} \tag{2.4}$$

$$\text{DSC} = \frac{2\text{TP}}{2\text{TP} + \text{FP} + \text{FN}} = \frac{2}{\frac{2\text{TP}+\text{FP}+\text{FN}}{\text{TP}}} = \frac{2}{\frac{\text{TP}+\text{FP}}{\text{TP}} + \frac{\text{TP}+\text{FN}}{\text{TP}}} = \text{F-score} \tag{2.5}$$

Therefore, the DSC is a harmonic mean of the precision and recall of the predicted segmentation maps. As such, it ranges from 0 (bad segmentation) to 1 (perfect segmentation). If we flatten the ground truth (GT) and the predicted (Pred) segmentation maps (both in the range [0, 1]), we can express the DSC in terms of vector operations:

$$\text{DSC} = \frac{2|\text{GT} \cdot \text{Pred}|}{|\text{GT}| + |\text{Pred}|} \tag{2.6}$$

Since this expression is differentiable, it can be used to create a loss function. However, it is undefined for cases in which $|\text{GT}| = |\text{Pred}| = 0$ (i.e. there is no region of interest in the ground truth and the prediction is consistent with the ground truth). To deal with this, we add a constant as a smoothing factor. Finally, since we want the optimizer to minimize the loss, our loss function based on DSC is defined as:

$$\text{DICE} = 1 - \frac{2|\text{GT} \cdot \text{Pred}| + 1}{|\text{GT}| + |\text{Pred}| + 1} \tag{2.7}$$

This loss function, based on a measurement of segmentation quality, addresses naturally the problem of class imbalance. However, due to its complexity, this function has convergence problems, sometimes falling into local optima. To alleviate this problem, as proposed by (Sun et al. [2020]), we extend our loss function using the binary cross-entropy loss (BCE) typically used for other segmentation tasks. Given an image with N pixels:

$$\text{BCE} = -\frac{1}{\text{N}} \sum_{i=1}^{\text{N}} (\text{GT}_i \cdot log(\text{Pred}_i) + (1 - \text{GT}_i) \cdot log(1 - \text{Pred}_i)) \tag{2.8}$$

$$\text{loss} = \text{BCE} + \alpha \cdot \text{DICE} \tag{2.9}$$

In BCE, GT_i and Pred_i represent the value of the i-th pixel in GT and Pred, respectively. In the final loss, α is a user defined constant; in our experiments, we set α empirically to 3 to give more importance to DICE. In this way, BCE helps to achieve convergence while DICE severely punishes an “all black” output.

2.3.3 Experimental Setup for Crack Segmentation

The U-VGG19 architecture was implemented using Tensorflow 2 (Abadi et al. [2015]). The initial weights from the encoder are the weights of the convolutional layers of VGG19 pre-trained on ImageNet. From this starting point, the whole U-VGG19 is trained together. The loss function is the one based on DICE and BCE presented in Equation 2.9. The optimizer is Adam with default parameters and an initial learning rate of 10^{-4}.

For each dataset, we randomly split the available images into 80% training and 20% testing. Then, a model per dataset is trained using 256×256 cropped patches fed in 4-patch batches; the patch approach was used to deal with variable-size training images. To refine the results at late epochs, we reduce the learning rate on test loss plateau (by 2, with 5 epochs tolerance). The network is trained during 150 epochs at most, stopping early to avoid overfitting if the test loss does not improve during 20 consecutive epochs. The images for testing are fed without cropping using batch size 1.

We evaluate our results in terms of precision, recall and Dice score coefficient, as commonly done in the literature. We calculate the scores per image and we record the averages in the test split at the epoch with the minimum test loss. Due to the random initialization of the network weights, we repeat the training 10 times and we report the average ± standard deviation of the obtained scores. Additionally, we provide an F-score as the harmonic mean of the average precision and the average recall.

2.3.4 Preliminary Experiments and Results on Crack Segmentation

In Table 2.1, we show the scores of the segmentation maps generated by U-VGG19 in the test splits of CFD and Aigle-Rn. Additionally, to study the cross-dataset generalization ability of U-VGG19, we share the results of three additional cases: 1) merging CFD and Aigle-RN into a single dataset, 2) training on CFD and testing on Aigle-RN, and 3) training on Aigle-RN and testing on CFD. For the first case, the dataset (CFD+Aigle-RN) is split into 80% training and 20% testing. For the latter two cases, the training is done with the training split of one dataset and testing is performed on the test split of the other dataset.

The best average DSC is obtained in Aigle-RN (72.6%). This is followed by CFD+Aigle-RN (71.7%), where we can see the best balance between the average precision and the average recall (70.9% and 74.9%, respectively). This

Table 2.1: Scores of U-VGG19 predictions on the different testing images.

Training/Testing	DSC(%)	Pr(%)	Re(%)	F(%)
CFD / CFD	70.9±0.5	68.3±2.9	76.2±3.7	72.0
Aigle-RN / Aigle-RN	72.6±5.1	70.4±6.4	80.8±4.6	75.2
CFD+Aigle-RN / CFD+Aigle-RN	71.7±0.4	70.9±1.7	74.9±2.1	72.8
CFD / Aigle-RN	33.9±5.3	21.8±4.4	91.4±3.0	35.2
Aigle-RN / CFD	18.3±4.5	62.6±4.7	11.2±3.1	19.0

balance decreases when training and evaluating only on CFD (DSC=70.9%) and decreases even more in Aigle-RN, with a difference around 10% between precision and recall (70.4% and 80.8%, respectively). In fact, this dataset exhibits the highest standard deviations among the three, while CFD+Aigle-RN exhibits the lowest; this behavior is associated to the number of training samples per dataset, being 30 for Aigle-RN and 116 for CFD+Aigle-RN.

There is an interesting behavior in the inter-dataset experiments. Both cases exhibit a low DSC; however, in terms of precision and recall, we see opposite behaviors. When training on CFD and evaluating on Aigle-RN, the precision is low (21.8%) but the recall is high (91.4%); when training on Aigle-RN and evaluating on CFD, the precision is high (62.6%) but the recall is low (11.2%).

When evaluating on Aigle-RN, the low precision of the model trained on CFD is, on one hand, explained by the presence of noisy predictions outside the cracks. On the other hand, the predicted cracks are considerably wider than the manual annotations. When evaluating on CFD, the low recall of the model trained on Aigle-RN is, on one hand, explained by missing some cracks in the prediction. On the other hand, the predicted cracks are considerably thinner than the manual annotations. These phenomena are illustrated in Figure 2.5, along with examples of the other three experiments.

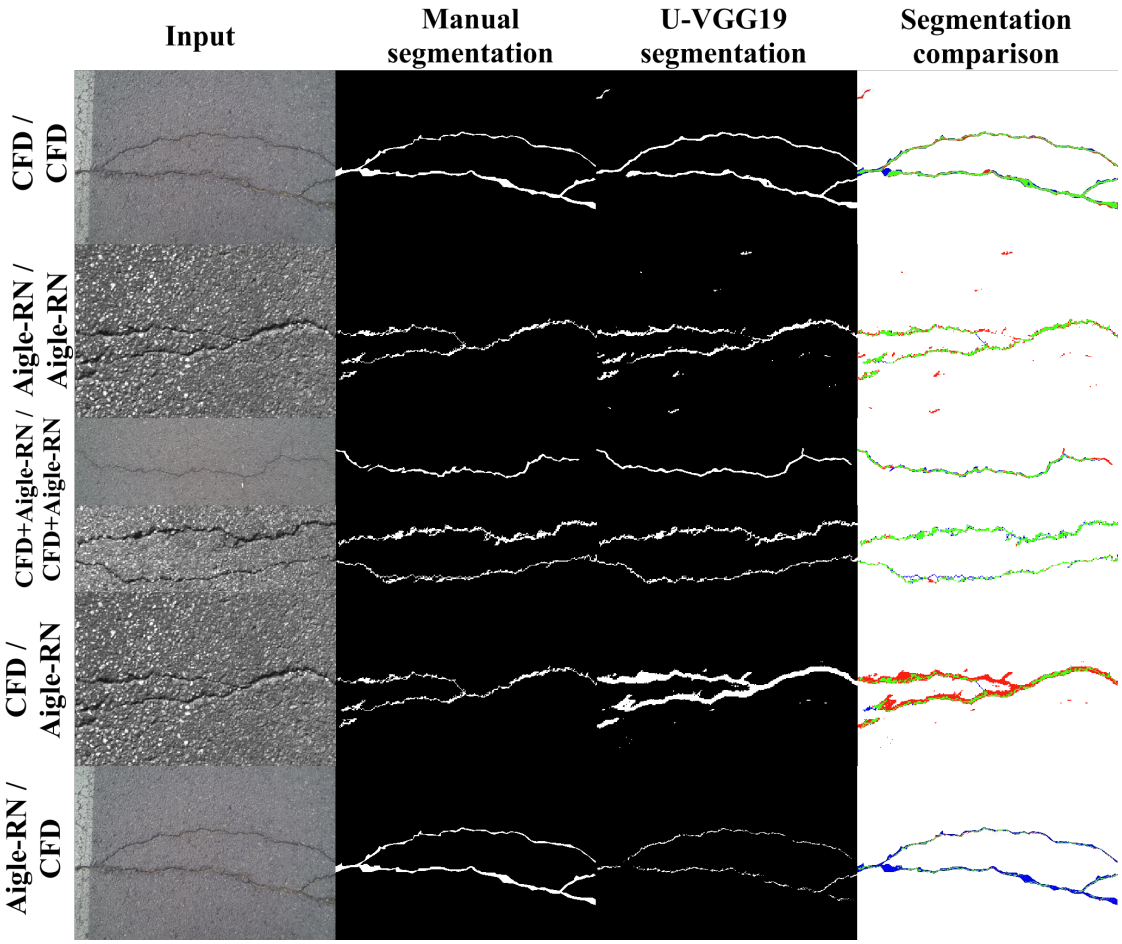


Figure 2.5: Examples of predictions from the experiments presented in Table 2.1. For the segmentation comparison, the color code is: (Green) True positives; (Blue) False negatives; (Red) False positives.

Since CFD and Aigle-RN represent two very different distributions, there is no wonder on the low inter-dataset scores. Not only the raw images were acquired under different conditions, but the quality of the annotations is different. Indeed, the annotations from Aigle-RN look closer to the real crack width with respect to CFD's annotations (which could explain the reduced number of annotated images). However, U-VGG19 exhibited good cross-dataset generalization by effectively learning from these two datasets simultaneously (see CFD+Aigle-RN in Table 2.1).

In Table 2.2 we compare the results of U-VGG19 with other methods for pixel-accurate segmentation. These state-of-the-art methods, similarly to ours, are based on U-net. We chose CrackForest as the reference dataset and DSC[1] as the score since they are the most popular in literature.

Table 2.2: Comparison of Dice score coefficients on CFD.

Method	DSC(%)
U-net (Gao et al. [2019])	60.5
GANs (Gao et al. [2019])	64.1
Multi-scale Convolutional Blocks (Sun et al. [2020])[a]	72.1
Feature Pyramid Hierarchical Boosting (Yang et al. [2020])[b]	70.5
Distribution equalization learning (Fang et al. [2021])[c]	42.2
– U-VGG19 (ours) –	70.9±0.5

[a] Training and evaluation are done with a CFD+Aigle-RN dataset.
[b] GT and Pred are thinned to 1-pixel edges for evaluation. This score is obtained by using, per image prediction, the decision threshold that maximizes the score
[c] Using a weighted F-score to prioritize the precision: $F_\beta = \frac{(1+\beta^2)\cdot Pr\cdot Re}{\beta^2\cdot Pr+Re}$, $\beta^2 = 0.3$

Our method is just below a U-net with multi-scale convolutional blocks (Sun et al. [2020]). However, those results were obtained by using a CFD+Aigle-RN dataset. By comparing them to our results using the same dataset fusion approach, the difference is only 0.4% – see Table 2.1 (CFD+Aigle-RN, DSC) and Table 2.2 (Multi-scale Convolutional Blocks). Nonetheless, their architecture is much more complex than ours; moreover, our training converged around the same number of epochs reported by them (~90, see Figure 2.6), implying an advantage in terms of hardware requirements and time. On CFD alone, the transfer learning strategy of U-VGG19 outperforms other, more complex, approaches too (GANs, multi-scale hierarchical boosting, distribution equalization learning).

When we analyzed the predictions of U-VGG19 qualitatively, we observed two main types of error: 1) missing low-contrast thin cracks and 2) questionable cracks that could be or not annotated as cracks depending on the observer.

We expected to improve the recall by solving the problem of missing thin cracks. To do so, we used a data augmentation approach. It consisted of randomly transforming an image (and its corresponding annotation) immediately before feeding it to the neural network for training. To do this, a random value for each of the 6 following operations is chosen: adding noise, changing illumination, flipping, zooming, rotating and shearing. Every image undergoes the 6

[1]The score commonly used in the literature is F-score. However, as shown in Equation 2.5, both scores are equivalent.

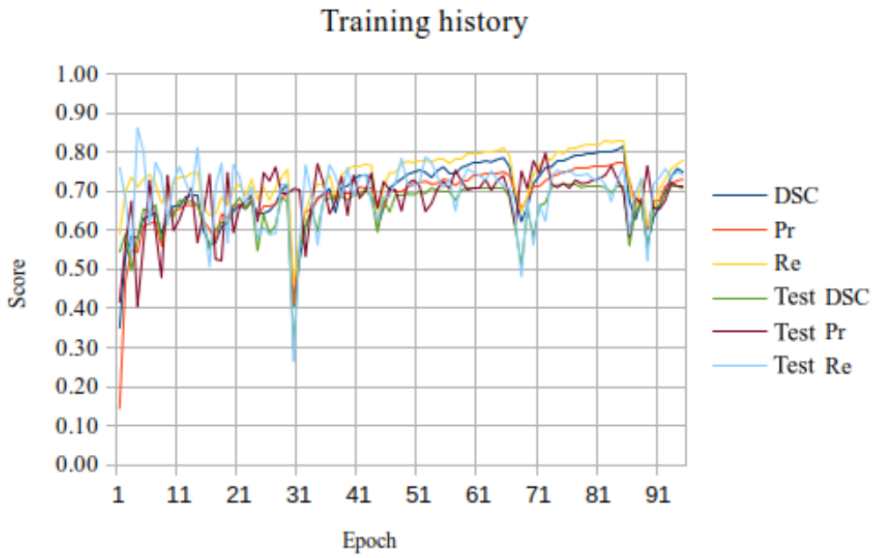


Figure 2.6: Example of the evolution of the evaluation scores in the training and the test splits during training on the CFD+Aigle-RN dataset.

operations in the given order. Therefore, there is a low likelihood that the model sees the very same image twice during training.

We performed this new experiment on the CFD+Aigle-RN dataset. In Table 2.3, we can see that the data augmentation approach actually reduced our DSC. A t-test ($p<0.05$) confirmed that, contrary to the expected outcome, there is a significant decrease of the average recall (and no significant difference of precision).

Table 2.3: Results of U-VGG19 on the CFD+Aigle-RN dataset.

Training	DSC(%)	Pr(%)	Re(%)
Without data augmentation	71.7±0.4	70.9±1.7	74.9±2.1
With data augmentation	70.5±0.8	70.7±2.7	72.3±3.6

When we compared the predictions with and without data augmentation, we noticed a particular phenomenon in the images from CFD: the predictions of the model trained with data augmentation tended to be thinner. This phenomenon, illustrated in Figure 2.7, is actually similar to the one observed in the predictions done on CFD by the model trained solely on Aigle-RN (see fifth row in Figure 2.5).

Despite obtaining a lower recall, by inspecting the false negatives (in blue) of U-VGG19 trained with data augmentation, it is clear that the predicted crack is closer to the real width and shape. As discussed before, the quality of the Aigle-RN's annotations is better than the ones of CFD (the annotations look closer to the real crack width). By using the CFD images, we deal with the reduced number of available images in Aigle-RN: since the model is able to learn from a wider variety of backgrounds, it misses less cracks than training with Aigle-RN alone. By increasing the number of Aigle-RN images through data augmentation, we force the network to produce more accurate segmentations in a greater number of images during training.

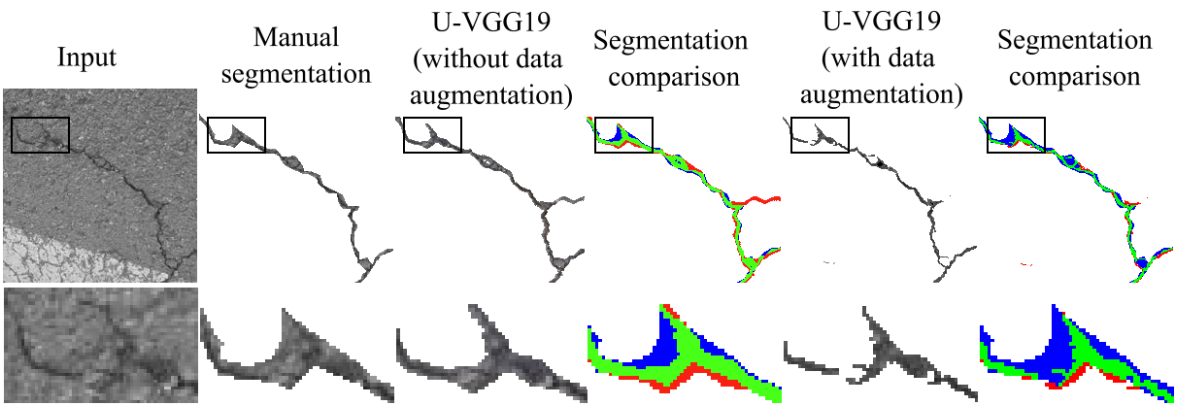


Figure 2.7: Comparison of manual segmentation and the segmentations of U-VGG19 trained with and without data augmentation. The color code is the same as in Figure 2.5.

2.4 Discussion on the Obtained Results for Crack Segmentation

In this chapter, we approached the problem of segmenting cracks: defects that appear along the lifetime of a construction and that can be identified through visual inspection. By providing a pixel-accurate segmentation, the geometrical and spatial properties of the segmented cracks can be used to infer their severity and potential causes; particularly, the width is an indicator of the cracks' age and severity.

In order to provide this segmentation, we proposed the U-VGG19 architecture. To compare our network with the state of the art, we evaluated U-VGG19 on two public datasets for road crack segmentation: CFD and Aigle-RN. In these datasets, U-VGG19 achieved a DSC of 70.9% and 72.6%, respectively. When merging both datasets as one (CFD+Aigle-RN), the DSC was 71.7%. By comparing these scores with the state of the art, we confirmed that U-VGG19 is a competitive model: our scores are greater or equal than those of other, more complex, models in the literature. This implies that the transfer learning approach of U-VGG19 (using an encoder pre-trained on ImageNet) is an effective way to reduce training costs, particularly in terms of trainable parameters.

By training on CFD+Aigle-RN, we proved the cross-dataset learning ability of U-VGG9 (i.e. learning from simultaneous datasets). However, we noticed an interesting phenomenon related to the manual segmentation quality on both datasets. In fact, the annotations from Aigle-RN exhibit –visually– a better quality in terms of crack width. This difference in the annotations is evidenced in inter-dataset experiments: the predicted cracks on Aigle-RN are wider when training with CFD images; in contrast, the predicted cracks on CFD are thinner when training with Aigle-RN images. When using data augmentation to train on the CFD+Aigle-RN dataset, we see that the influence of thinner manual annotations in Aigle-RN increases: the predictions on CFD images tend to be thinner than when training without data augmentation. These thinner predictions, in fact, look more similar to the visible crack.

These observations lead us to the problem of learning in presence of inaccurate annotations. As previously discussed, accurate segmentations are neces-

sary to correctly measure geometrical features from the segmented cracks. As observed here, on one hand, it is difficult to produce accurate segmentations when the manual segmentations used for training are inaccurate. On the other hand, it is difficult to produce accurate manual segmentations because the annotation task is highly time consuming and very prone to human error at different levels. In the next chapter, we discuss our approaches to produce pixel-accurate segmentation while training with inaccurate annotations.

Chapter 3

Pixel-Accurate Crack Segmentation in Presence of Inaccurate Labels

For the automatic analysis of detected defects such as cracks, it is necessary to provide pixel-accurate segmentations that preserve their actual location and shape. While some tolerance may be admissible when segmenting objects such as animals and pedestrians, small errors in cracks create huge differences in the measured geometry.

The state of the art on crack segmentation provides methods with a good capability of locating cracks in images. Among these, the most successful ones are based on supervised learning, using manual annotations to train DL models. However, as discussed in the previous chapter, providing an accurate segmentation is a difficult task even for humans: the boundary between crack and background is often fuzzy; the background is composed of diverse confusing textures; the cracks have complex geometric shapes; the image resolution can be on the limit of thin cracks' width; etc.

The aforementioned constraints make manual annotations prone to error. Among these errors, we focus on those committed on the level of pixels: the annotations are either too wide, too thin or inaccurately placed (see Figure 3.1).

Because of this inaccuracy in annotations, fundamental works on road crack segmentation such as CrackTree (Zou et al. [2012]) proposed margins of tolerance for evaluation: "a detected crack pixel is still considered to be a true positive if it is located no more than 2 pixels away from human annotated crack curves". Many posterior works have opted to use margins of tolerance for evaluation too. While this tolerant approach is sufficient for counting and locating cracks, it is not for measuring important properties such as their width – an indicator of the cracks' age and severity (Bhat et al. [2020]).

In fact, there is a huge score gap between works that use margins of tolerance and works which do not. State-of-the-art methods using these margins of tolerance exhibit overwhelming F-scores as high as 95% (Escalona et al. [2019]; Fan et al. [2020a]). On the other side, when no tolerance is used for evaluation, the F-scores decrease down to ~70%, as revised in the previous chapter (Gao et al. [2019]; Sun et al. [2020]; Yang et al. [2020]).

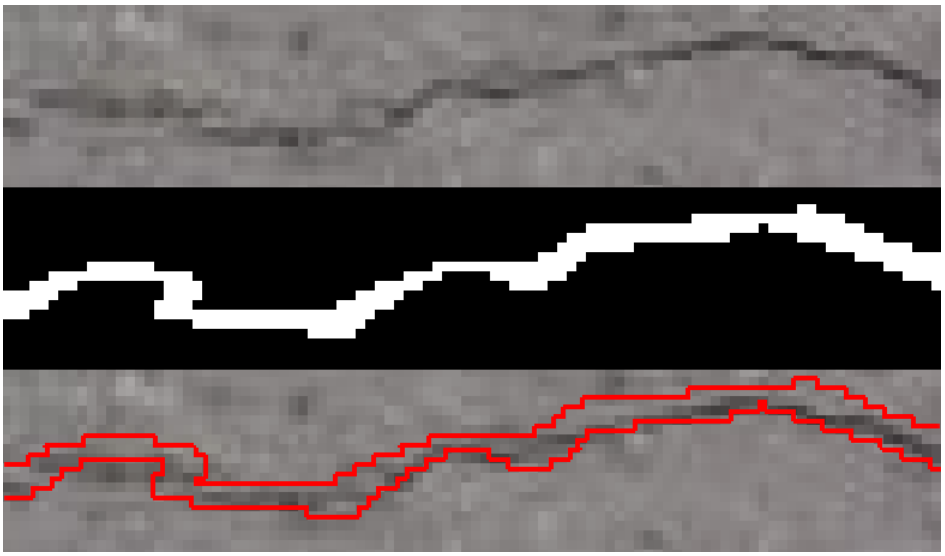


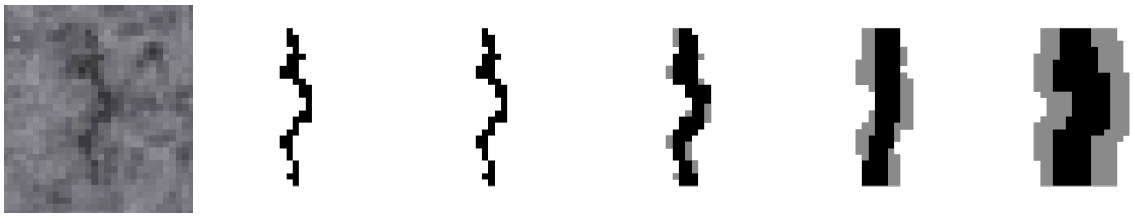
Figure 3.1: A zoom to errors on the level of pixels in the CrackForest dataset. The first row shows the raw image, the second one shows a manual annotation, and the third one shows the image with a red border surrounding the annotation.

This difference is no surprise because margins of tolerance are lax to errors on the pixel level, providing too optimistic scores. In (Ai et al. [2018]), we see a study of the impact of different margins of tolerance on the scores: without tolerance, their method achieved a precision of 47.1% and a F-score of 56.7%. With a 1-pixel tolerance, these scores increased drastically to 78.9% and 80.0%, respectively. Using the more common tolerance in the literature (2 pixels), their scores rose to 90.7% and 87.0%, respectively. From this study, we can conclude that increasing the margin of tolerance increases the precision drastically. In Table 3.1, we confirm this hypothesis using one of the U-VGG19 models trained on CFD: evaluating with increasing margins of tolerance, we increase drastically the precision without changing the predictions. Thus, tolerance allows the predicted cracks to be wider than reality, artificially preserving high precision scores.

Furthermore, the width is not the only geometrical property that can be severely changed if margins of tolerance are allowed (see Figure 3.2). Models that achieve high scores without tolerance will provide a higher degree of confidence in the geometric properties obtained from detected cracks e.g. shape, length and width. However, increasing the scores without tolerance is difficult due to the inaccuracy of manual annotations: optimizing with respect to inaccurate ground truths will provide biased scores.

Table 3.1: Crack segmentation scores on CFD using different margins of tolerance.

Metric	Margin of tolerance			
	0px	1px	2px	5px
Precision	71.75	90.22	95.82	97.70
Recall	71.49	75.80	76.87	77.21
F-measure	70.78	81.87	84.87	85.84



(a) Image (b) Crack (c) 0px (d) 1px (e) 2px (f) 5px

Figure 3.2: Evaluation with margins of tolerance. Color code: (Black/White) Crack/No crack; (Grey) Ground truth for evaluation with margins of tolerance. (a) Input image. (b) Crack segmentation ground truth. (c) Perfect prediction with no tolerance. (d-f) Predictions inside their corresponding margin of tolerance; all black pixels are true positives, but the geometry of the real crack is not respected.

3.1 Inaccurate Segmentation Annotations as Label Noise

Since image segmentation returns a single class per pixel, we can consider each pixel as a single, independent sample. Indeed, by feeding an image to a FCNN, we produce a vector of values per pixel before making the final class decision. This, in concept, is the same as a conventional CNN: extracting a fixed-size set of features to be fed to a fully connected network (or classifier).

Let us consider an image, I, a distribution of pixels, each of them represented by a vector x_i obtained by a feature extractor. Then, each vector has an associated label y_i depending on whether x_i belongs to a crack or to the background. Because of the inaccuracy in the manual annotations, some pixels in I are mislabeled in this binary classification problem.

The rest of this chapter is organized as follows. First, we provide a literature review about learning in the presence of label noise caused by mislabeling. Next, we introduce our Syncrack generator. Subsequently, we use Syncrack to evaluate the detrimental impact of training with inaccurate labels in the predicted segmentations. Afterwards, we discuss and evaluate our proposed methods to improve crack segmentation, particularly in terms of crack width. The chapter is closed with a discussion on the obtained results and future perspectives.

3.2 Learning in Presence of Label Noise: A Brief Review

Training when given labels are not always ground truth (i.e. mislabeling exists) is called inaccurate supervision. Inaccurate supervision is a sub-case of weakly supervised learning (Zhou [2017]). Weakly supervised learning is an intensively studied topic. The approaches to deal with label noise automatically can be categorized into three main groups (Frenay and Verleysen [2014]):

1. Noise-robust algorithm design: some algorithms are less influenced than others by label noise. The core idea is to create classifiers with low sensitivity to the presence of label noise during training.

2. Noise filtering: the principle is filtering the training data to identify the noisy labels before learning. The focus is given on the data filters.

3. Noise-tolerant algorithm design: it consists of developing algorithms able to model label noise during learning. Alternatively, existing classifiers can be modified to consider label noise in an embedded fashion.

From these types of approaches, we will focus on the first two.

Regarding noise-robust algorithms, classifiers such as SVM and k-NN have shown some robustness to noisy labels, particularly as the number of training samples increases (Cannings et al. [2018]). However, traditional machine learning algorithms using handcrafted features have been overpassed by DL for crack segmentation.

Regarding pre-learning filtering, increasing the number of independent annotators can improve label quality and model quality. It has been shown that, when labels are noisy, repeated labeling can be preferable to single labeling even if labels are not particularly cheap (Sheng et al. [2008]). Classifier ensembles have shown to be robust to inaccurate data too. These ensembles produce a joint decision based on the classification of individual classifiers or "voters". The labels provided by individual labelers (either humans or classifiers) can be used to filter data prior to training a final model. Particularly, there are two types of filters based on voting: consensus and majority filters (Brodley and Friedl [1999]). The biggest weakness of these approaches is the necessity of multiple labelers; this may be unfeasible for human labeling, and it is computationally expensive when training multiple classifiers.

On the other hand, single-classifier-based filters have been used successfully too (Brodley and Friedl [1999]). The predictions of a classifier can be used to iteratively identify mislabels, clean the data, and re-train the classifier with the cleaned data (Young et al. [2013]). A particular case of this approach is called self-training: the classifier is trained using the raw predictions of the classifier trained during the previous iteration. Each iteration produces new, potentially cleaner, labels per sample. This strategy, in fact, has been used to segment images using image-level labels (Liang et al. [2020]). However, this kind of approaches has the inherent risk of enhancing the base classifier's biases.

Other approaches have been proposed to study the influence of single data pairs (input vector and assigned label) during training to identify outliers (i.e. potential mislabels) (Cook [1977]; Cook and Weisberg [1982]). Nonetheless, the influence functions require expensive second derivative calculations and assume model differentiability and convexity. To integrate influence functions to CNN training, (Koh and Liang [2017]) proposed to approximate them using second-order optimization techniques. They show that, even on non-convex/differentiable models, their approximation can provide valuable information. These influence functions have been used for classification with inaccurate labels using DL (Hao et al. [2020]). However, the detection of influential observations is complex and computationally expensive.

Alternatively, efforts have been made towards building loss functions able to deal with outliers. For example, (Zhang et al. [2020d]) proposes a framework with a noise-robust loss that allows updating the parameters of the network

and correcting the inaccurate labels simultaneously. However, this kind of approaches, as well as the influence-based ones, is typically used for image classification. In image segmentation, tens of thousands of predictions are performed per image instead of a single one. Because of this, the cost of the aforementioned approaches increases drastically.

Some alternatives to this kind of problem have been proposed based on the difficulty of classifying each individual pixel. One example is the method presented by (Zhang et al. [2020a]) for blood vessel segmentation from weak automatic annotations. First, the loss focuses on easy-to-classify pixels. However, this has the potential risk of ignoring systematic biases in noisy labels (losing crucial pixels). Thus, an online active learning component is used to refine updated labels: a small number of valuable pixels with potentially incorrect labels are annotated and then manually refined in each iteration during training.

In blood vessel segmentation, choosing these valuable pixels is challenging because of the highly imbalanced foreground and background. Similarly, crack segmentation has an inherent severe class imbalance. Training in presence of class imbalance is difficult by itself. Naturally, doing this in presence of label noise is still a challenging problem (Koziarski et al. [2019]). Although some efforts have been made to learn in the presence of noise with some class imbalance (Cannings et al. [2018]), severe class imbalance represents an open challenge for all the previous methods discussed in this section. In (Sáez et al. [2015]), for example, we see an approach based on re-sampling using noise filters. Although the proposed method improves the area under the ROC curve with respect to training with raw data, it was used on datasets with imbalance ratios below 9 i.e. less than 9 majority samples per minority sample.

In fact, no strategy is straightforwardly usable on heavily class-imbalanced data. Cracks typically represent even less than 1% of the pixels from road images i.e. an imbalance ratio >99; such imbalance ratio represents a considerable challenge to effectively train a model with inaccurate human labels.

In this work, we focus our efforts on data filtering strategies. Once the mislabeled samples are identified, two typical approaches are taken to clean the training data: either removing the mislabeled samples or relabeling them (Zhou [2017]). Each of these approaches has their own advantages and disadvantages; however, these have been studied in contexts with lower class imbalance.

To study these advantages and disadvantages in a highly class-imbalanced context, as well as to objectively measure the impact of inaccurate labels on pixel-accurate crack segmentation, we introduce a novel tool for synthetic dataset generation: Syncrack.

3.3 Syncrack

One inherent problem when evaluating methods learning from inaccurate annotations is that ground truths are hard to obtain. Indeed, there are only few datasets where incorrect labels have been identified. A common approach is to introduce label noise on purpose on datasets with reliable labels (Frenay and Verleysen [2014]). On the other hand, the use of synthetic data to train machine

learning models is an increasing trend in artificial intelligence.

Inspired by this, we created Syncrack: an open-source tool aimed to generate parametrizable synthetic images of cracked pavement/concrete-like textures. Syncrack provides accurate ground truth segmentations of the cracks and allows creating parametrizable noisy versions of these annotations. In this way, we provide a benchmark to study the robustness of crack segmentation methods to inaccurate annotations. This benchmark also allows evaluating noise filtering and label correction methods for segmentation tasks with severe class imbalance. Additionally, the use of transfer learning can allow adapting the models and methods developed for Syncrack-generated distributions to real-life ones e.g. the images generated by Syncrack with default values are perceptually similar to pictures of real-life roads.

The Syncrack generator is composed of 4 main modules. The goals of each model are enumerated below:

1. Creating a background image
2. Creating crack shapes
3. Adding cracks to a background
4. Creating noisy annotations from pixel-accurate ground truths

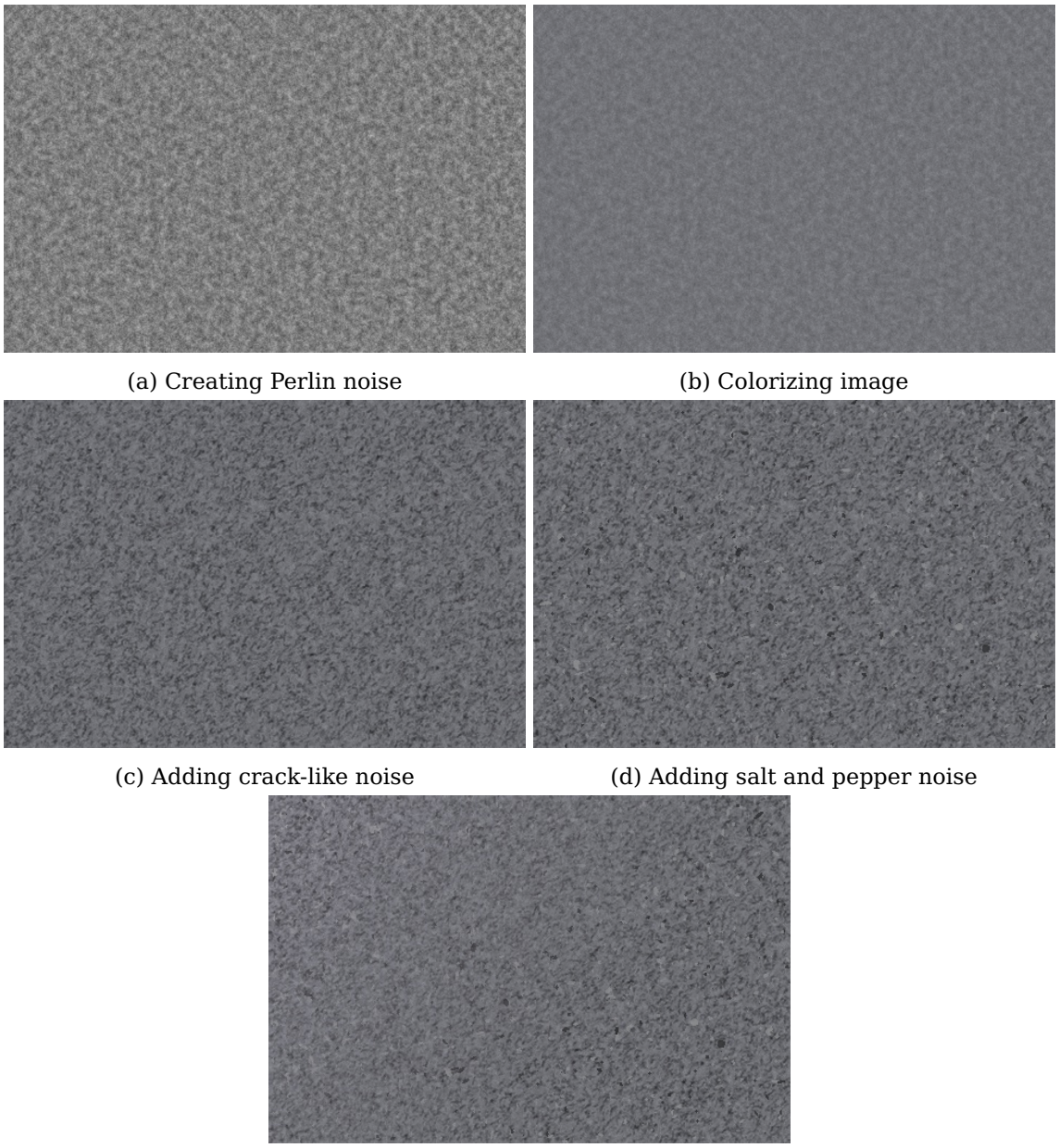
To ensure diversity among images, each module uses random values for its internal parameters. Some of these values are calculated based on user-provided parameters, allowing user customization of the output. To ensure reproducibility, the random value selection is done with a fixed seed. Each one of the modules is discussed next.

3.3.1 Background Generation

Our method consists of 5 main steps performed progressively, as illustrated in Figure 3.3):

1) Creating a 2D Perlin noise. Our primary texture is based on Perlin noise calculated in 2D. By using the noise module (Duncan [2018]) as a basis, we create a grayscale image normalized in the range [0, 1]. To modify the smoothness of this primary texture, we adjust the 'scale' of the Perlin noise. The value of the scale is obtained following a normal law defined by the average smoothness and the smoothness standard deviation provided as user-accessible parameters.

2) Converting to color image. Even though the materials generally look gray, public crack datasets usually provide RGB images. To emulate these real materials, we colorize the grayscale image obtained from step 1. First, we select two random grayish colors in RGB; to get the grayish appearance, the dominating channel is the blue one. One color has greater intensities to represent a bright color; the other one has lower intensities to represent a dark color. Let $\mathrm{P}(x, y)$ be the grayscale image obtained in step 1, and B and D be the size-3 vectors representing the bright and dark color, respectively. The colorized image is $\mathrm{C}(x, y) = \mathrm{B} \cdot \mathrm{P}(x, y) + \mathrm{D} \cdot [1 - \mathrm{P}(x, y)]$.



(a) Creating Perlin noise

(b) Colorizing image

(c) Adding crack-like noise

(d) Adding salt and pepper noise

(e) Combining textures

Figure 3.3: Illustration of the Syncrack's background generation steps.

3) Adding crack-like noise. One common problem in road crack segmentation is the presence of artifacts on the background that look similar to cracks. To make the images generated by Syncrack challenging and more similar to real-life pictures, Syncrack adds crack-like artifacts to the background. First, random positions for the noise are selected; the amount of artifacts is inversely correlated to the background average smoothness parameter in step 1. These positions are used as centers of random size and orientation ellipses. One arc per ellipse is introduced to the colorized image from step 2 similarly to how cracks are introduced (see subsection 3.3.3). Therefore, these arcs will look similar to cracks but will have very different geometrical properties. Thus, thresholding is not enough to distinguish actual cracks from these artifacts.

4) Adding additional noise. Real images are prone to invasive elements.

We add salt and pepper noise to simulate dirt on the surface and acquisition artifacts. The grains are introduced as filled circles with random variable radii. The number of grains is inversely correlated to the background average smoothness parameter too. The grains share the same intensity in the three color channels. Let the image with noisy artifacts obtained in step 3 be N. For salt noise, the value is $max(\mathrm{N})$; for pepper, it is $min(\mathrm{N})$.

5) Combining different textures. Stationary textures are not frequent in real-life constructions. This is specially true in roads, that suffer constant surface wear. We create non-stationary background by iteratively joining pairs of textures. Each stationary texture is generated with steps 1-4. Given a base texture T_1 and a new texture T_2 to be combined, we fuse them together using a 2D grayscale linear gradient map $\mathrm{G}(x, y) \in [0, 1]$. The resulting texture is $\mathrm{T}(x, y) = \mathrm{T}_1 \cdot [1 - \mathrm{G}(x, y)] + \mathrm{T}_2 \cdot \mathrm{G}(x, y)$. These allows creating smooth transitions between the fused textures.

The image with non-stationary textures is the final product of the background generation module. Once we have a background image, a crack is added.

3.3.2 Crack Shape Generation

Our crack shapes are based on a modification of the 1D Perlin noise. The method is illustrated in Figure 3.4. First, we calculate a set of vertices using the Perlin noise generator from the noise module. The domain of these vertices is determined by a randomly chosen crack bounding width W.

However, these vertices have two problems: 1) their height scale (y-axis) is blind to the image size and the crack length, and 2) they use a straight axis as

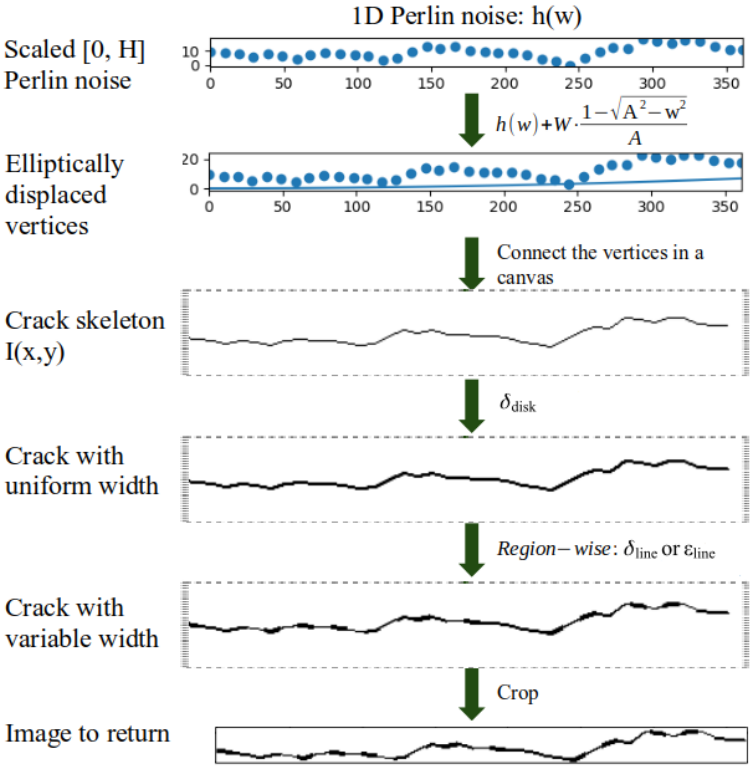


Figure 3.4: Summary of the Syncrack's crack shape generation steps.

origin, so the cracks would always tend to straight lines. To solve this, we select a random crack bounding height H and normalize the y-coordinates of the vertices to the range $[0, H]$. To obtain curve cracks, we displace the y-coordinates of the vertices using an elliptic function with a random major axis $A > W$.

These vertices are connected using 1-pixel-wide straight lines in a canvas to create a crack skeleton. To provide the crack with width, we dilate the skeleton by a disk-shaped structuring element.

Dilation. Let us define $I(p)$ as the image to dilate; this function returns the intensity value for each pixel location $p \in \mathbb{Z}^2$. Given a binary structuring element (SE) S, the dilation of an image consists of assigning to each pixel the maximum value found over the neighborhood of the SE centered on the pixel of interest. With p the pixel of interest, this is formalized as (Soille [2004])

$$[\delta_S(I)](p) = \max_{s \in S} I(p+s)$$

The diameter of the SE used to provide the crack with width is chosen randomly from a normal distribution centered at 2 with standard deviation 0.5 (default user-accessible parameters). Since real cracks do not have constant widths, a sliding-window approach is used to introduce dilations and erosions in random crack regions.

Erosion. Contrary to dilation, erosion consists of assigning to each pixel the minimum value found over the neighborhood of the SE centered on the pixel of interest. This is formalized as

$$[\varepsilon_S(I)](p) = \min_{s \in S} I(p+s)$$

To avoid drastic width changes, the chosen SE is a 2-pixel long vertical line. Additionally, a dilation is never followed by an erosion or vice-versa.

The final product of the crack shape generation module is an image containing the crack, cropped to the minimum bounding box. This image is used to introduce a crack with such shape into a background image.

3.3.3 Crack Introduction

Once a crack shape is obtained, we generate a weight map (same size as the background image) by creating an empty white image (weight=1.0). The crack shape image is randomly rotated and introduced in a random position of this map, with a value of 0 for pixels corresponding to the crack. At this point, the weight map is a binary image equivalent to the crack segmentation ground truth (see center of Figure 3.5).

An average contrast value, A, is randomly chosen from a normal distribution with center and standard deviation provided as user parameters. Then, within the weight map, we set each crack-labeled pixel to a new independent value. Per pixel, this value is obtained from a new Gaussian distribution centered around A with standard deviation $A/10$. We call this a contrast map $M \in [0, 1]$ (see top-right of Figure 3.5).

Since different crack regions tend to have different lighting conditions in real life, we change the contrast of certain zones in M within the crack using a sliding

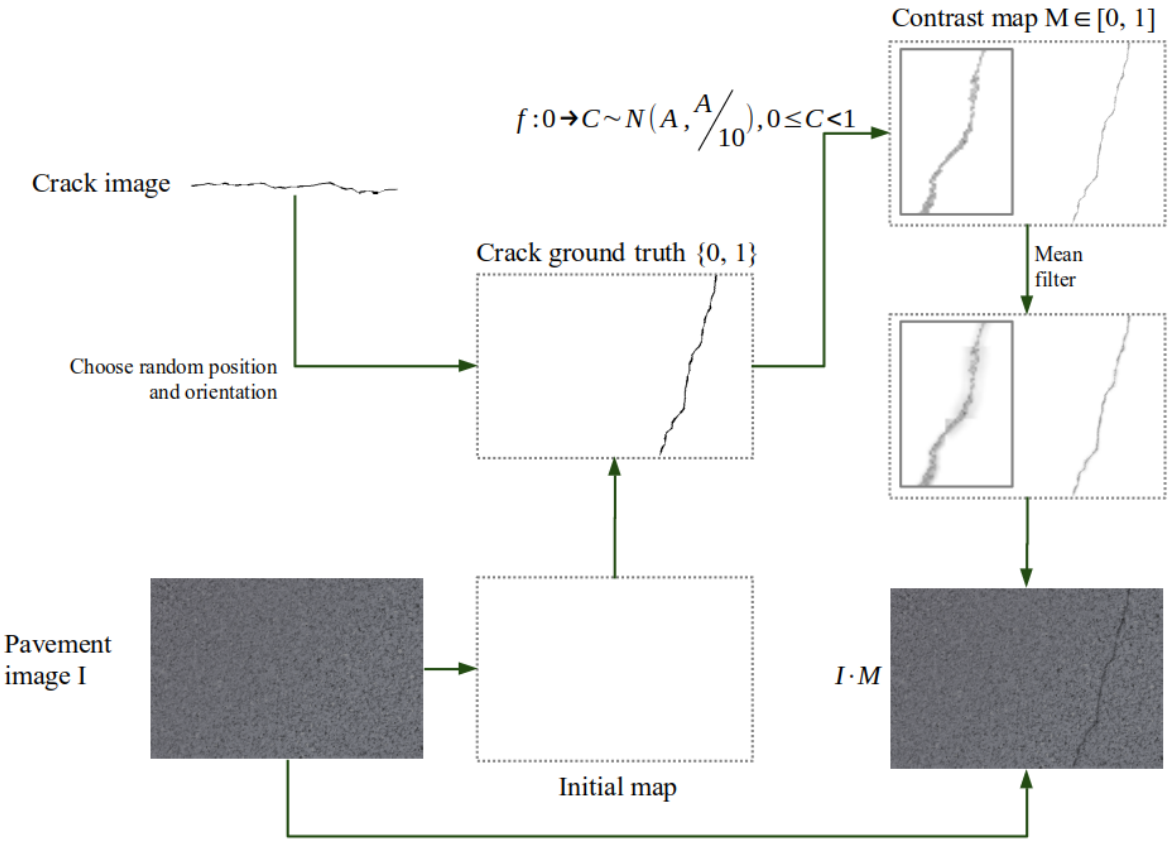


Figure 3.5: Summary of the Syncrack's crack introduction steps.

window. Finally, by using disk mean filters, we create a transition region between the actual crack and near-to-crack background. The filtering is performed window-wise, choosing a random disk radius between 2 and 5 per window. The possible radii are equally probable, and each window undergoes a second filtering using a disk with twice the chosen radius. This is the final contrast map M used to introduce the crack into the background image (see middle-right of Figure 3.5).

The introduction of the crack is done by multiplying the background image and the contrast map M. This multiplication is performed element-wise per color channel in the background image.

The final product of the crack introduction module is an image with a crack and its corresponding ground truth segmentation. This ground truth annotation can be modified by the next module to emulate human inaccurate annotations.

3.3.4 Label Noise Generator

To simulate inaccurate labeling, we introduce noise in the annotations through morphological operations. This process is illustrated in Figure 3.6. First, we divide the annotation image into tiles. Then, we alter randomly chosen tiles by performing an erosion or a dilation by a disk with a random diameter. With default user parameters, the tile size is $0.05 * image_height \times 0.05 * image_width$. The diameter of the disk follows a normal law centered around 3 for dilation and 2 for erosion, as default values. In both cases, the standard deviation is 0.5.

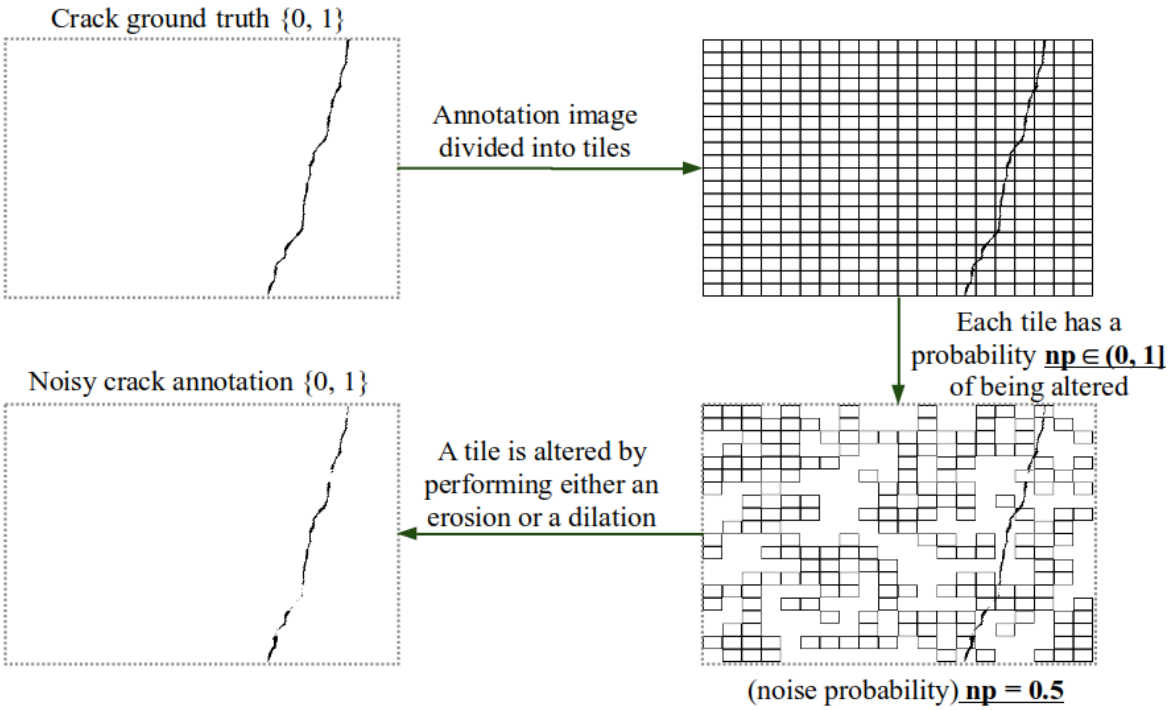


Figure 3.6: Summary of the Syncrack's label noise generator.

The user parameter 'noise_percentage' (np from now on) is the independent probability, $np \in (0,1]$, for a tile to be altered; if 1, all the tiles are altered. Since the tiles are obtained by dividing the whole image, and the probability is unconditional, an altered tile may contain no cracks.

With $np < 1$, we expect inaccuracy similar to a manual one: some crack segments wider (a dilation was applied), some segments thinner (a small erosion), missing crack segments (big erosion), and some accurate segments.

In the next section, we perform experiments with a dataset generated by Syncrack. We use both clean and noisy versions of the annotations, as provided by the Syncrack generator.

3.4 Impact of Inaccurate Training Annotations when Evaluating on Clean Ground Truths

In this section, we present the results of training U-VGG19 (see section 2.2) on a dataset generated with Syncrack. We generate a dataset of 200 images with default Syncrack parameters. Similarly to the public CrackForest Dataset (CFD), the resolution is 480×320 and the crack width is around 1-3 pixels. For each of these images, we produce 5 different annotations with different label noise levels. The noise levels range from 0 (no noise) to 4 (the maximum amount of noise); these levels are created solely by varying the np parameter, as described in the previous section. The noise levels are illustrated in Figure 3.7.

A descriptive analysis of the generated noise is presented in Table 3.2. The crack-to-background and background-to-crack mislabeling percentages are calculated with respect to the total number of crack pixels. For example, with 100

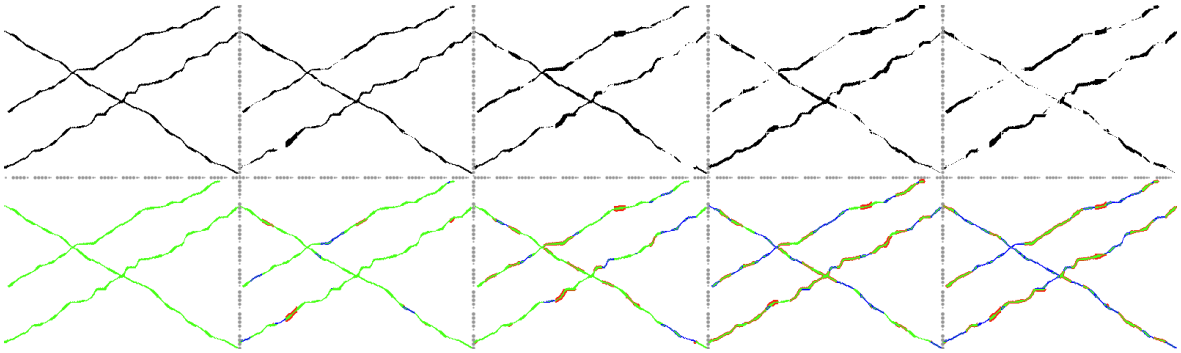


Figure 3.7: Examples of different label noise levels. From left to right, levels 0 to 4 (see Table 3.2). The second row shows: Green) Crack pixels; Blue) Crack pixels mislabeled as background; Red) Background pixels mislabeled as crack.

Table 3.2: Label noise levels used for experiments.

Label noise level	0	1	2	3	4
np	0.00	0.25	0.50	0.75	1.00
Crack → Background mislabeling (%)	0.00	9.68	19.35	29.03	38.71
Background → Crack mislabeling (%)	0.00	12.90	24.19	32.26	43.55
Mislabeling (%)	0.00	22.58	43.55	61.29	82.26
DSC (%)	100.0	88.93	78.41	68.39	58.94
Pr (%)	100.0	88.44	77.18	68.53	58.29
Re (%)	100.0	89.92	80.61	69.41	60.66
H_{crack}	3.984	4.064	4.117	4.138	4.159
H^2_{crack}	7.391	7.498	7.589	7.613	7.675
K-S	0.6846	0.6299	0.5768	0.5373	0.4911

crack pixels in the dataset: if 10 crack pixels were labeled as background, Crack → Background mislabeling (%) = 10; if 5 background pixels were labeled as crack, Background → Crack mislabeling (%) = 5. Mislabeling (%) is the sum of both percentages.

In order to provide a better intuition of the meaning of the aforementioned percentages, we also show the average precision, recall and DSC per image with respect to the clean annotations. These scores, which are supervised, are meaningful when an accurate ground truth is available. However, in the presence of inaccurate annotations, these scores are unreliable. That is why, in Table 3.2, we propose 3 additional scores to evaluate the quality of the noisy annotations. These unsupervised scores, which do not require a ground truth for comparison, are proposed for further evaluation on real images, in which the supervised scores are biased because of the inaccuracy of the manual annotations.

The first of these unsupervised scores is the crack region entropy H_{crack}, based on the region entropy (Pal and Bhandari [1993]). Given the set R_{crack}

of pixels within a crack-predicted region, define a set V_{crack} of all the possible pixel intensities within R_{crack}. Define p_m as the probability of finding a pixel with intensity m in R_{crack}. Then

$$H_{crack}(R_{crack}) = -\sum_{m \in V_{crack}} p_m log(p_m) \tag{3.1}$$

This is a way to measure the intra-crack uniformity. With a good segmentation, the intensity distribution in R_{crack} is dominated by dark pixels. This will minimize H_{crack}. Introducing background pixels into R_{crack}, on the other side, will increase H_{crack}.

We also use a second-order crack region entropy H^2_{crack}. The second-order entropy relies on co-occurrence matrices (Haralick et al. [1973]) rather than pixel intensities. This allows inspecting the intra-crack region for textures. Given p the probability from the co-occurrence matrix calculated in R_{crack}:

$$H^2_{crack}(R_{crack}) = -\sum_{i \in V_{crack}} \sum_{j \in V_{crack}} p_{ij} log(p_{ij}) \tag{3.2}$$

Similarly to the first order entropy, including background pixels in R_{crack} will increase the entropy because new, more diverse textures will be taken into account. A good segmentation will produce a reduced amount of textures within the crack-predicted region, reducing H^2_{crack}.

From a probabilistic point of view, we assume that background and cracks are two different intensity distributions. By using the Kolmogorov-Smirnov test (Smirnov [1939]), we measure the distance between the accumulated distributions of pixels predicted as background ($F_1(x)$) and as cracks ($F_2(x)$), respectively:

$$\text{K-S} = sup_x |F_1(x) - F_2(x)| \tag{3.3}$$

The Kolmogorov-Smirnov score (K-S) increases with respect to how much two distributions differ. Since the cracks are thin, introducing background pixels in the segmentation will quickly make the distribution of pixels segmented as cracks similar to the one of pixels segmented as background; this will decrease the K-S score. Therefore, a good segmentation should maximize this score.

These scores were selected because they have been previously used for the evaluation of unsupervised segmentation tasks (Zhang et al. [2008]). As shown in Table 3.2, a decrease in annotation quality implies an increase of the entropies and a decrease of the K-S. This behavior is exactly the one expected from the previous discussion. Next, we study the quality of the predictions of U-VGG19 trained on the different label noise levels of the Syncrack-generated dataset.

3.4.1 Experimental Setup to Evaluate the Detrimental Impact of Noisy Labels

Using Syncrack as a benchmark, we study the detrimental impact of training with inaccurate labels on the predicted segmentations. Since Syncrack provides accurate ground truths for evaluation, we can measure this impact in a supervised manner; additionally, this allows us to validate the proposed unsupervised scores.

For these experiments, we split the Syncrack-generated images into 50% training and 50% testing. In this way, we have a number of training images similar to the training split used in subsection 2.3.4 when training on CFD. The training is performed using 256×256 cropped patches with batch size 4, while testing is performed by using entire images without cropping using batch size 1. To refine the results at late epochs, we reduce the learning rate on test loss plateau (by 2, with 5 epochs tolerance). The training is performed using 150 epochs at most; to avoid overfitting, we add an early stop if the test loss does not decrease during 20 consecutive epochs. The loss function is again the one based on DICE and BCE presented in Equation 2.9. The optimizer is Adam with an initial learning rate of 10^{-4} and default parameters in Tensorflow 2.

The final scores that we report are calculated using the network weights at the epoch with the minimum test loss. We perform a supervised evaluation calculating precision, recall and DSC with respect to the clean (accurate) annotations. Additionally, we perform an unsupervised evaluation in terms of H_{crack}, H^2_{crack} and K-S. Each score is calculated per image, and we record the averages of the test split. As before, we repeat the training 10 times and we report the average ± standard deviation in Table 3.3.

Table 3.3: Results of training with different label noise levels on Syncrack. The supervised scores are calculated with respect to accurate annotations.

Score	Label noise level				
	0	1	2	3	4
DSC(%)	78.6±0.7	77.0±0.6	73.1±1.4	68.6±1.1	66.2±1.4
Pr(%)	79.9±0.9	75.0±2.1	66.5±3.3	58.5±2.6	55.3±2.3
Re(%)	78.4±1.9	80.5±2.4	82.7±2.4	84.9±3.2	84.5±2.4
H_{crack}	3.96±0.01	3.99±0.02	4.05±0.02	4.10±0.02	4.13±0.01
H^2_{crack}	7.36±0.03	7.46±0.05	7.62±0.06	7.76±0.05	7.81±0.03
K-S	0.698±0.006	0.676±0.011	0.619±0.022	0.566±0.018	0.536±0.014

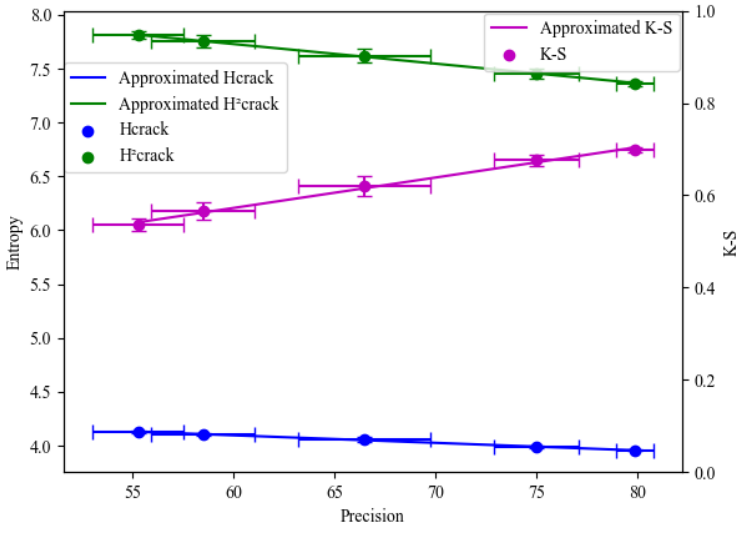


Figure 3.8: Linear relations between the precision and the unsupervised scores of the predictions of U-VGG19 trained with different label noise levels.

As expected, the DSC decreases as we increase the label noise level. However, among the supervised scores, the most interesting behaviors are observed with respect to precision and recall. When increasing the noise, we see a tendency of the recall to actually improve. On the other hand, the precision decreases. We see an overall decrease of the DSC because the decrease in precision is greater than the increase in recall.

The reasoning behind these results is that, as we increase the label noise level, the model becomes more conservative to discard potential crack segments. This allows decreasing the amount of missed cracks, but at the cost of increasing the amount of false positives. Although some of these false positives are structures apart from the actual cracks, most of the false positives are located surrounding the actual cracks, as illustrated in Figure 3.9. In other words, as we increase the noise during training, the predictions tend to be wider. In the figure, we can see that the predicted cracks can be twice as wide as the ground truth.

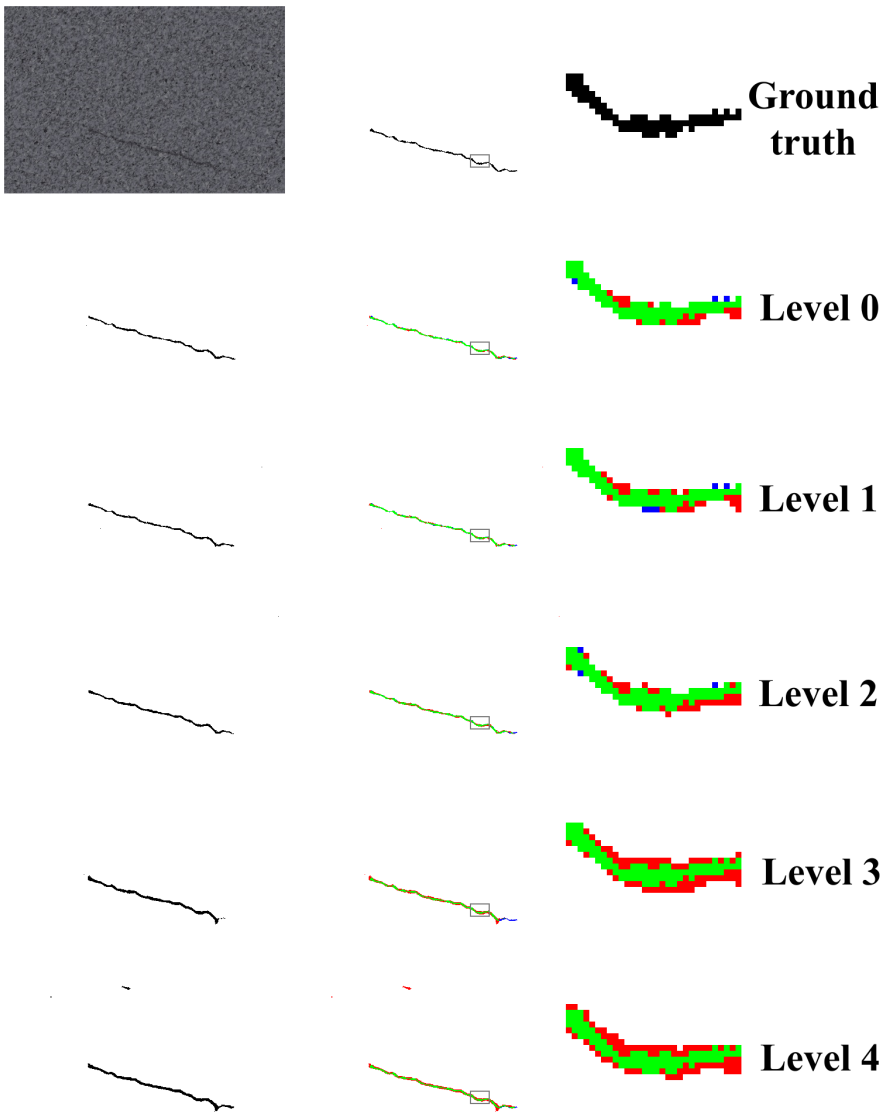


Figure 3.9: A comparison of the crack width predicted by models trained with different label noise levels. The color code is: (Green) True positives; (Blue) False negatives; (Red) False positives.

Regarding location of cracks, the network was able to deal with missing crack segments in the annotations during training. However, when it comes to the cracks' geometrical properties, the predictions of the network trained with inaccurate labels seem to overfit to the excessive width of the inaccurate annotations. This kind of behavior, which is precisely the one we expect to solve, is identified by the unsupervised scores.

In Table 3.3, we see a direct relation between the supervised precision and the unsupervised scores: when the precision decreases, the K-S decreases and the entropies increase. In fact, the relations between the precision and the unsupervised scores can be approximated as linear functions as depicted in Figure 3.8

Remember, the specific focus in this chapter is improving the crack segmentation in terms of width, achieving predictions closer to the real crack shape. In the following sections, we use both supervised and unsupervised scores to evaluate the methods proposed to improve learning in the presence of inaccurate annotations.

3.5 Weakly Supervised Learning

As discussed in the previous section, increasing the label noise level makes the predicted cracks wider even if the label noise is bi-directional i.e. from background to crack and vice versa. In this section, we explore noise filtering methods used traditionally for weakly supervised classification (Frenay and Verleysen [2014]). Then, we study the quality of predictions trained with data that was corrected using such filtering methods.

The method proposed to improve crack segmentation can be summarized in 2 main steps: 1) getting a set of new pseudo-labels from the original data and 2) training a model with these pseudo-labels instead of the original labels, which are known to be inaccurate. The goal is to improve the width of the predicted cracks with respect to training with raw inaccurate labels. We propose 4 different methods for the creation of pseudo-labels based on the k-NN algorithm, ensemble majority voting, ensemble consensus voting and self-training.

3.5.1 Pseudo-Label Generation

It has been shown that the k-NN algorithm is robust to inaccurate labels, as long as the number of samples is big enough (Cannings et al. [2018]). In our work, first we use U-VGG19 trained on the original annotations as a feature extractor: the feature maps from the second to last convolutional layer (see Figure 2.4.D) represent each pixel as a 2D vector (before the ReLu activation). We use the k-NN algorithm with k=5 in this 2D space to assign new pseudo-labels per pixel. The algorithm is applied to each image individually, because of its scalability with respect to the number of pixels. An image for pavement crack segmentation with a 480 × 320 resolution contains >150k pixels; this fulfills the "big enough" requirement (Cannings et al. [2018]). Increasing the number of images to be used simultaneously would increase drastically the processing time per image. We refer to this approach as *5-nn voting*.

Ensemble voting strategies have shown to improve the performance of models trained with noisy labels too (Brodley and Friedl [1999]; Sheng et al. [2008]). Similarly to (Zhou et al. [2021]), we use a bagging strategy with a k-fold approach. Specifically, we train 10 independent instances of U-VGG19 with a different subset each (composed of 9 folds each). Once the 10 voters are trained, they are used to predict all the available images. Unlike Zhou et al., we get pseudo-labels per image using all the trained models simultaneously before training the final model. We use two different ensemble strategies to produce the pseudo-labels: *majority voting* and *consensus voting*.

Finally, we use the predictions of a single model to produce pseudo-labels through self-training. This consists of training models recursively: each new model takes the output of the previous one as training input, and the new model predicts new pseudo-labels (Liang et al. [2020]). In this work, we use as pseudo-labels the predictions of U-VGG19 trained with the original labels. We refer to this approach as *self-training*.

Although we use U-VGG19 as a baseline model, the aforementioned methods can be used with any segmentation network. The computational complexity of these methods depends directly on the baseline model used. For U-VGG19, we assume its complexity constant per pixel, and linear with the size of the image, denoted by $\mathcal{O}(n)$ with n being the number of pixels.

The four methods can be repeated more than once, but currently there is no guarantee of improvement by increasing the number of iterations. Next, we discuss the computational complexity of performing 1 iteration per method.

In the case of self-training, the method requires one full training of the baseline model and one prediction of the full dataset; no further operations are needed. For the voting methods, M instances of the baseline model (considered as weak learners) are trained using each one $\frac{\mathrm{M}-1}{\mathrm{M}}$ of the dataset. With the predictions of the weak learners, the final pseudo-labels are obtained through voting. Each voting strategy can be solved in $\mathcal{O}(\mathrm{M})$ per pixel (Boyer and Moore [1991]). Therefore, the complexity of voting pseudo-label generation is $\mathcal{O}(\mathrm{M}n)$ per image. In the case of 5-nn voting, the final pseudo-labels are obtained using a k-NN algorithm on a d-dimensional space as projected by the baseline model. The projection is done in $\mathcal{O}(n)$. Then, when using an efficient tree structure for the k-NN algorithm (Bentley [1975]; Omohundro [1989]), available in (Pedregosa et al. [2011]), the complexity of the tree construction is $\mathcal{O}(dn \cdot log(n))$. The k-NN prediction per pixel is performed in $\mathcal{O}(k \cdot log(n))$. Finally, the pseudo-labels for all the pixels are obtained in $\mathcal{O}(kn \cdot log(n))$ per image.

The pseudo-labels generated by the methods described before (see summary in Table 3.4) are used to filter and correct mislabels in the original labels. This correction can consist either of removing or relabeling the identified suspicious samples (Zhou [2017]). Evidence suggests that removing the identified mislabeled samples reduces the error in clean data with respect to relabeling them (Young et al. [2013]). Nevertheless, compared with removal methods, relabeling ones have their accuracy fall off much more slowly when increasing the label noise. As discussed by Young et al., this phenomenon may be explained by the limited number of remaining training samples after removing suspicious ones. In our context, with few labeled images and a very low rate of positive-class in-

Table 3.4: Comprehensive comparison of the proposed pseudo-label generation methods. The number of pixels per image is denoted by n.

Method	Basis	Advantages	Disadvantages	Computational complexity
Self-training	Single-model prediction	Easy to implement, no hyperparameter selection	Amplification of the baseline model's bias	Cost of the baseline model $\mathcal{O}(n)$
Majority voting	Ensemble of M classifiers using bagging	A balanced probability of discarding good data and retaining bad data (Brodley and Friedl [1999])	Empirical selection of M	$\mathcal{O}(\mathrm{M}n)$
Consensus voting	Ensemble of M classifiers using bagging	Conservative in discarding good data (Brodley and Friedl [1999])	Empirical selection of M	$\mathcal{O}(\mathrm{M}n)$
5-nn voting	k-NN algorithm on projected d-dimensional space	Robust to decision boundary overfitting	Empirical selection of k	- Pixel projection $\mathcal{O}(n)$ - Tree creation $\mathcal{O}(dn \cdot log(n))$ - Pseudo-label generation $\mathcal{O}(kn \cdot log(n))$

stances, the impact of removing data points could be worse. Thus, we study both approaches.

3.5.2 Experiments and Results Training with Pseudo-labels

In our method, the pseudo-labels are produced by methods expected to be robust to label noise. Therefore, we assume that these pseudo-labels are cleaner than the original noisy (inaccurate) annotations. To relabel, we simply replace the original labels by the new pseudo-labels. To remove, we use the pseudo-labels to weight pixels at loss calculation during training: we assign a weight of 1 to the pixels where both the raw and the pseudo-label agree, and 0 to all others.

Before evaluating the results using pseudo-labels, we propose a naive approach as baseline. As discussed at the end of subsection 2.3.4, training with data augmentation can potentially improve the predicted crack width. Since data augmentation is a common practice in DL, we use the scores of training with this approach as a baseline. To evaluate if this approach can benefit from the proposed pseudo-label generation methods, we base these ones on U-VGG19

trained with data augmentation.

We used the same data augmentation approach as when U-VGG19 was tested with real-life images in subsection 2.3.4: during training, each image undergoes 6 operations before being fed to the network (adding noise, changing illumination, flipping, zooming, rotating and shearing). We compare the scores of training on Syncrack with and without data augmentation, under different label noise levels, in Table 3.5.

Table 3.5: Results training with and without data augmentation on Syncrack. The supervised scores are calculated with respect to accurate annotations.

Data Aug.?	Score	Label noise level				
		0	1	2	3	4
No	DSC(%)	78.6±0.7	77.0±0.6	73.1±1.4	68.6±1.1	66.2±1.4
	Pr(%)	79.9±0.9	75.0±2.1	66.5±3.3	58.5±2.6	55.3±2.3
	Re(%)	78.4±1.9	80.5±2.4	82.7±2.4	84.9±3.2	84.5±2.4
	H_{crack}	3.96±0.01	3.99±0.02	4.05±0.02	4.10±0.02	4.13±0.01
	H^2_{crack}	7.36±0.03	7.46±0.05	7.62±0.06	7.76±0.05	7.81±0.03
	K-S	0.698±0.006	0.676±0.011	0.619±0.022	0.566±0.018	0.536±0.014
Yes	DSC(%)	79.0±0.5	78.5±0.8	77.6±0.9	75.2±1.5	72.5±1.4
	Pr(%)	79.8±2.1	76.6±2.5	72.8±3.0	66.8±3.7	62.5±2.9
	Re(%)	79.0±2.7	81.6±3.1	84.1±2.3	87.4±2.6	87.8±2.0
	H_{crack}	3.94±0.01	3.96±0.02	4.01±0.02	4.05±0.02	4.08±0.02
	H^2_{crack}	7.34±0.05	7.43±0.06	7.53±0.06	7.65±0.07	7.72±0.06
	K-S	0.719±0.009	0.699±.013	0.667±0.016	0.625±0.024	0.595±0.021

As hypothesized, training with data augmentation can actually improve the predicted crack width. This improvement is denoted by the precision, H_{crack}, H^2_{crack} and K-S. However, according to the results used to produce Table 3.5, this improvement begins to be significant at label noise level 2 (t-test with $p<0.5$ for the supervised precision).

As such, we confirm that data augmentation can be very useful to deal with inaccurate annotations when this inaccuracy is big enough. Using Table 3.2 as reference, we talk about inaccurate annotations with a DSC below 80% with respect to the ground truth. This principle can be used to train crack-locator networks from very rough annotations. However, for pixel-accurate crack segmentation, this is not enough. Since in real scenarios it is hard to objectively measure the quality of the annotations, a blind naive data augmentation approach is not sufficient to actually improve the width of the predicted cracks.

In Table 3.6, we compare the results obtained by training with the help of the pseudo-labels produced by the proposed methods; both relabeling and removing were tested. For an easier interpretation of the results, we plot the scores per pseudo-label generation method in Figure 3.10; per plot, we show the baseline scores of training with the original raw annotations (with and without noise, see row with data augmentation in Table 3.5).

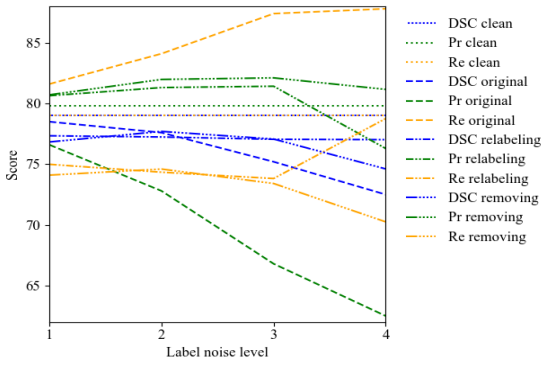
5-nn voting. Training with the pseudo-labels produced by k-NN voting produced the best results among the 4 methods. It is considered to be the best because, as we increase the label noise level, the 6 evaluation metrics hold values

Table 3.6: U-VGG19 prediction scores on Syncrack using the different proposed pseudo-label generation methods at different label noise levels. Per cell, we show the results of relabeling/removing. The best supervised scores per label noise level are **highlighted**. The overall best method is denoted in **bold**. The "Original" row shows the baseline training with data augmentation using raw noisy annotations in Table 3.5.

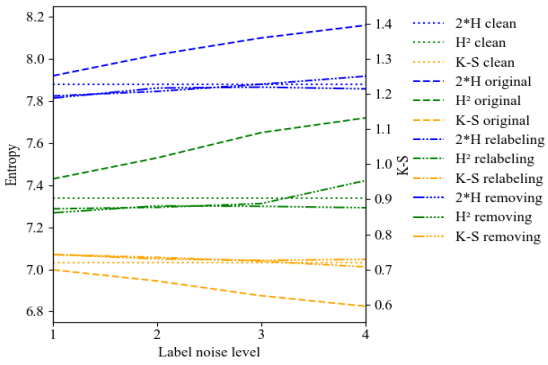
Labels	Score	Label noise level			
		1	2	3	4
Original	DSC(%)	78.5	77.6	75.2	72.5
	Pr(%)	76.6	72.8	66.8	62.5
	Re(%)	81.6	84.1	87.4	87.8
	H_{crack}	3.96	4.01	4.05	4.08
	H^2_{crack}	7.43	7.53	7.65	7.72
	K-S	0.699	0.667	0.625	0.595
5-nn voting	DSC(%)	77.34/76.83	77.24/77.70	77.05/77.07	**77.02**/74.61
	Pr(%)	80.64/80.71	81.31/81.98	81.42/**82.11**	76.29/**81.16**
	Re(%)	74.99/74.10	74.34/74.60	73.82/73.41	78.76/70.25
	H_{crack}	3.912/3.907	3.923/3.931	3.940/3.933	3.959/3.929
	H^2_{crack}	7.288/7.269	7.295/7.302	7.313/7.300	7.422/7.293
	K-S	0.7422/0.7430	0.7346/0.7297	0.7234/0.7261	0.7074/0.7287
Consensus voting	DSC(%)	72.60/73.52	75.11/74.43	74.94/72.98	73.71/74.39
	Pr(%)	**85.70**/84.06	81.67/**84.98**	77.39/78.87	75.86/75.92
	Re(%)	63.89/66.16	70.46/67.20	74.15/69.38	73.33/74.56
	H_{crack}	3.871/3.891	3.913/3.899	3.920/3.912	3.952/3.956
	H^2_{crack}	7.102/7.158	7.252/7.178	7.342/7.274	7.393/7.400
	K-S	0.7719/0.7589	0.7401/0.7531	0.7315/0.7405	0.7081/0.7069
Majority voting	DSC(%)	**77.94**/75.99	77.85/76.62	75.99/76.98	74.48/74.77
	Pr(%)	77.41/79.95	79.33/80.91	68.22/73.10	66.61/67.71
	Re(%)	79.36/73.24	77.24/73.66	86.79/82.52	**85.67**/84.76
	H_{crack}	3.929/3.913	3.939/3.927	4.008/3.968	4.029/4.034
	H^2_{crack}	7.368/7.275	7.354/7.304	7.589/7.478	7.624/7.618
	K-S	0.7253/0.7430	0.7219/0.7316	0.6586/0.6920	0.6454/0.6404
Self-training	DSC(%)	77.48/77.05	77.70/**78.47**	76.09/**77.22**	75.56/76.47
	Pr(%)	75.78/80.54	71.74/75.50	67.76/71.42	69.11/74.20
	Re(%)	**80.14**/74.75	**85.75**/82.62	**87.90**/85.00	84.35/80.19
	H_{crack}	3.935/3.918	4.000/3.977	4.021/3.999	4.017/3.991
	H^2_{crack}	7.393/7.302	7.542/7.467	7.617/7.547	7.581/7.491
	K-S	0.7210/0.7344	0.6711/0.6905	0.6483/0.6680	0.6558/0.6799

very similar to training with clean labels. Furthermore, according to precision, H_{crack}, H^2_{crack} and K-S, the width of the predicted cracks is considerably better than training with raw noisy labels. Although the scores of removing and relabeling are similar to each other, removing emphasized precision while relabeling emphasized recall; this is particularly visible at label noise level 4. The DSC is more constant for relabeling than for removing.

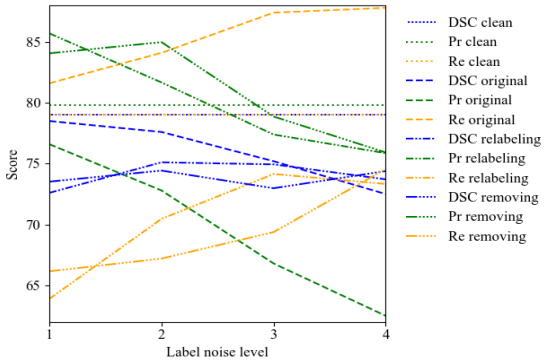
Consensus voting. These pseudo-labels produced the less satisfactory results. While the precision and the unsupervised scores are better than training



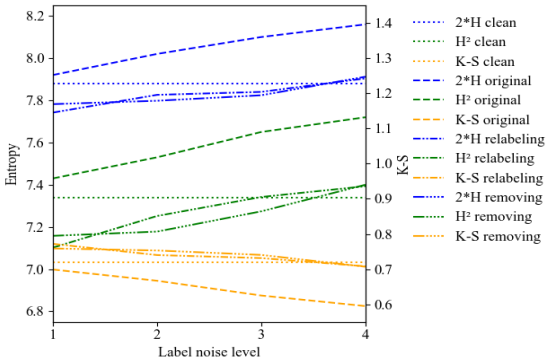
(a) 5-nn voting: supervised



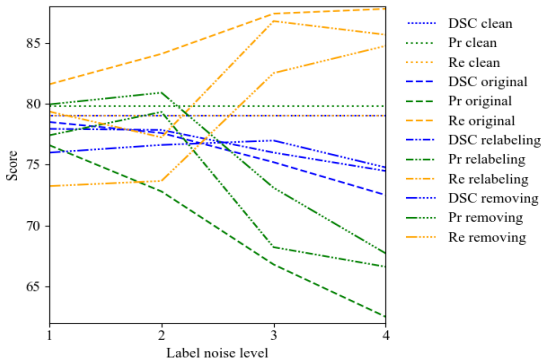
(b) 5-nn voting: unsupervised



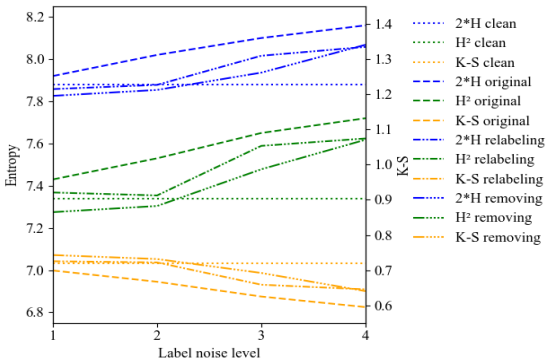
(c) Consensus voting: supervised



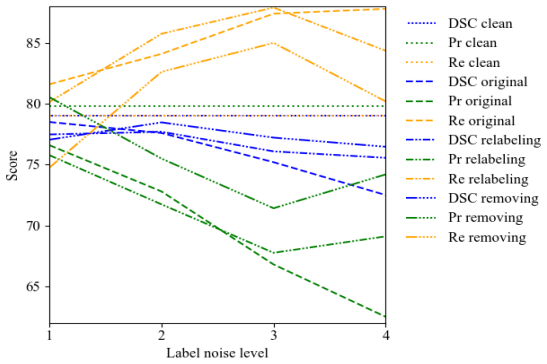
(d) Consensus voting: unsupervised



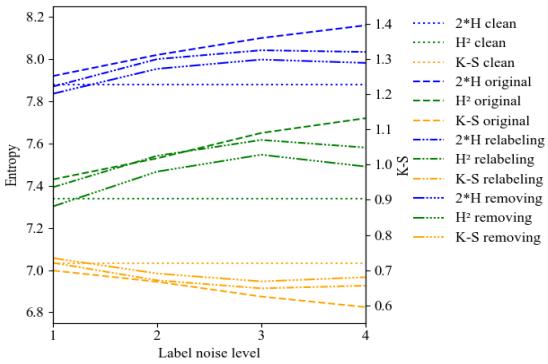
(e) Majority voting: supervised



(f) Majority voting: unsupervised



(g) Self-training: supervised



(h) Self-training: unsupervised

Figure 3.10: U-VGG19 prediction scores on Syncrack using the different proposed pseudo-label generation methods at different label noise levels.

with noisy labels, the decrease in recall is considerably high with respect to training with clean labels. In fact, the DSC is overall lower when training with the consensus-generated pseudo-labels than when training with noisy labels. However, this difference decreases as we increase the label noise level: the precision decreases but the recall increases more. Therefore, when the noise in the labels is severe, this method can indeed improve the predicted crack width while preserving a DSC and a recall similar to training with accurate labels. Similarly to 5-nn voting, removing emphasized precision while relabeling emphasized recall; however, the difference between the scores along the label noise levels is more evident.

Majority voting. Contrary to consensus, majority showed robustness to noise mainly at low noise levels. From level 3, we see a drastic drop of precision and improvent of recall. Unlike consensus, compared to training with noisy labels, training with majority-voting pseudo-labels exhibits a similar but often greater DSC. Furthermore, in terms of precision and the unsupervised metrics, we can observe that the predicted crack width is better using these pseudo-labels instead of the raw inaccurate annotations. With these observations, it can be stated that majority voting is a better strategy than consensus voting to generate the pseudo-labels. As with 5-nn and consensus voting, removing emphasized precision while relabeling emphasized recall. However, the difference in precision and recall between removing and recall is more constant than for the two other pseudo-label generation methods.

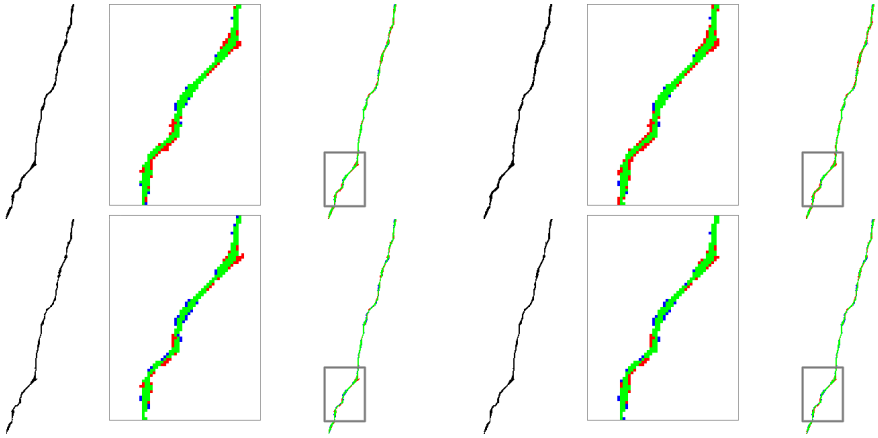
Self-training. When training with the pseudo-labels generated by self-training, the 6 evaluation scores follow a behavior very similar to when training with noisy labels. However, the precision, the DSC and the unsupervised scores become better as the noise increases. While the recall tends to be lower than training with raw inaccurate annotations, this recall is still greater than training with clean labels. In fact, training with self-training-generated labels shows to be a way to reduce the influence of noise during training. Similarly to majority voting, removing emphasized precision while relabeling emphasized recall, and the difference in both scores is relatively constant along the label noise levels.

With the analysis of the 4 proposed pseudo-label generation methods on Syncrack, we selected 5-nn voting as the best one. In Figure 3.11, we show a comparison of the predicted cracks training with 5-nn voting using relabeling under different label noise levels. We compare the predictions trained with raw noisy labels and pseudo-labels; this comparison shows the improvement with respect to training with raw noisy annotations. Next, we study the proposed method using 5-nn voting in real-life images.

3.5.3 5-nn Pseudo-Label Generation in Real-life Images

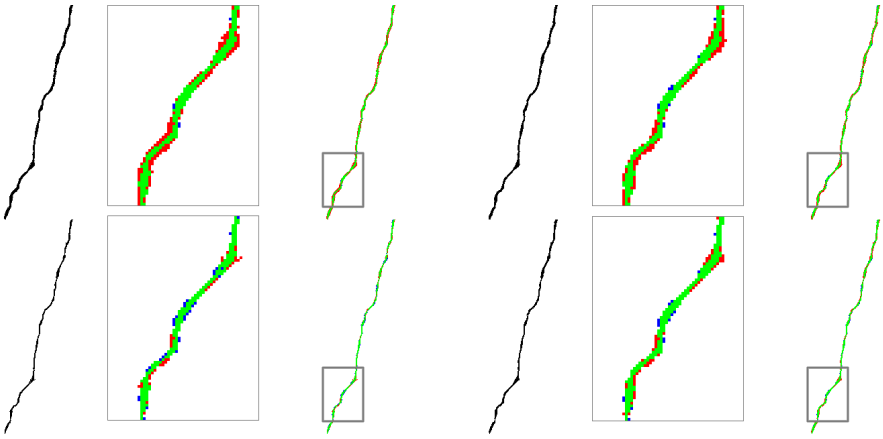
The approach that provided the best scores on Syncrack, 5-nn voting, is tested on real data in this section. To continue with the experiments proposed in subsection 2.3.4, we will perform our experiments on the CFD+Aigle-RN dataset; this dataset is created by merging the public CrackForest and Aigle-RN datasets. For simplicity, we used only a relabeling strategy for training. This time, our only baselines are training with manual annotations and training with data augmen-

(a) Clean labels



(b) Label noise level 1

(c) Label noise level 2



(d) Label noise level 3

(e) Label noise level 4

Figure 3.11: a) From left to right: input image, crack ground truth, prediction training with clean labels, ground truth vs prediction comparison. b-e) Comparison of predictions training with raw noisy labels (top row) and pseudo-labels obtained with 5-nn voting (bottom row) at different label noise levels. From left to right: prediction and ground truth vs prediction comparison. For the comparison, the color code is: (Green) True positives; (Blue) False negatives; (Red) False positives.

tation (as reported in Table 2.3). As for Syncrack, we obtain the pseudo-labels for CFD+Aigle-RN using the model trained with data augmentation; the model trained with pseudo-labels uses data augmentation during training too. Table 3.7 shows the scores of the baselines and the proposed approach based on weak supervision, using the original manual annotations as ground truth for evaluation. As in previous cases, we report the average ± standard deviation obtained by training 10 times.

As discussed in subsection 2.3.4, the DSC of training with data augmentation is lower than when training without it. Furthermore, the DSC obtained by training with pseudo-labels is also lower than the raw manual labels baseline.

Table 3.7: Comparison of the prediction scores on CFD+Aigle-RN training U-VGG19 with manual annotations and pseudo-labels.

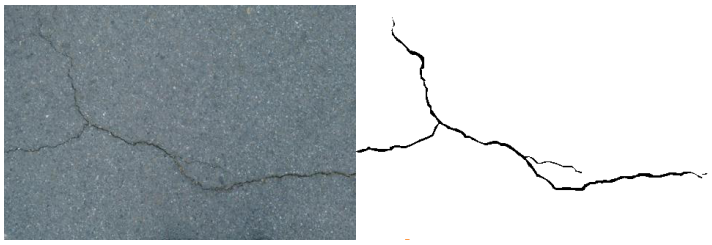
Score	Training with manual annotations	Training with manual annotations using data augmentation	Training with pseudo-labels generated by 5-nn voting
DSC(%)	71.7±0.4	70.5±0.8	67.6±2.1
Pr(%)	70.9±1.7	70.7±2.7	73.3±5.0
Re(%)	74.9±2.1	72.3±3.6	65.7±7.2
H_{crack}	4.34±0.01	4.30±0.03	4.25±0.06
H^2_{crack}	8.07±0.04	8.01±0.07	7.87±0.17
K-S	0.633±0.015	0.677±0.023	0.718±0.046

Nonetheless, under the context of noisy labels, this is not a reliable measure of the quality of the model. As expected from the results obtained on Syncrack, the decrease of DSC is explained by a reduced recall in both cases. Furthermore, the precision actually improves with respect to training with manual annotations. Manual annotations have a bias towards labeling cracks wider than reality. Learning this bias will improve the scores evaluating with those inaccurate annotations. A better crack width prediction, on the other side, will decrease the recall score in this context.

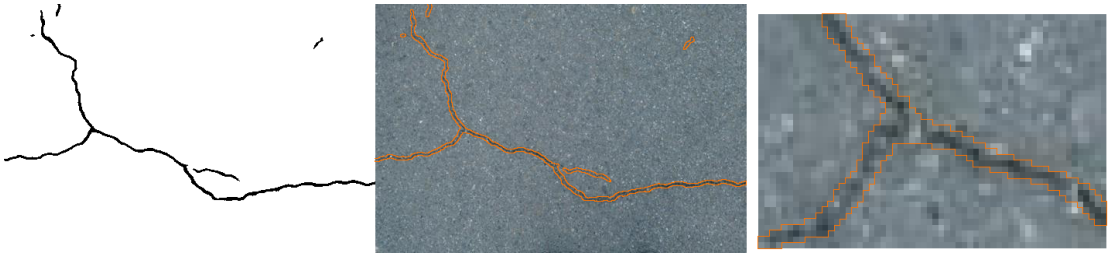
Unlike with Syncrack, this time we do not have accurate ground truth annotations to measure if our correction approaches are actually improving the prediction in terms of supervised scores. Therefore, the analysis is centered on the proposed unsupervised scores. Regarding H_{crack} and H^2_{crack}, training with data augmentation and with 5-nn-voting pseudo-labels reduces the entropies with respect to training with raw manual annotations. Furthermore, training with pseudo-labels reduces the entropies with respect to the naive data augmentation approach. This is congruent with the behavior of K-S: training with data augmentation and with 5-nn-voting pseudo-labels increases the K-S with respect to training with raw manual annotations. As expected, training with pseudo-labels increases the K-S with respect to the naive data augmentation approach.

The decrease of the entropies and the increase of the K-S denote an improvement of the predicted crack widths: this behavior means that the regions segmented as cracks are less contaminated by background pixels. It is worth to notice that training with 5-nn-voting pseudo-labels exhibits the best unsupervised scores; this is explained by the relative decrease of recall. On one hand, the model trained with pseudo-labels preserves the amount of correctly labeled elements in the regions manually annotated as background; since no false positives appear in those regions, the precision keeps similar. On the other hand, the model improves the width of the predicted cracks by classifying the pixels surrounding the cracks as background; since the manual annotations label these regions as crack, the recall decreases but the unsupervised scores improve.

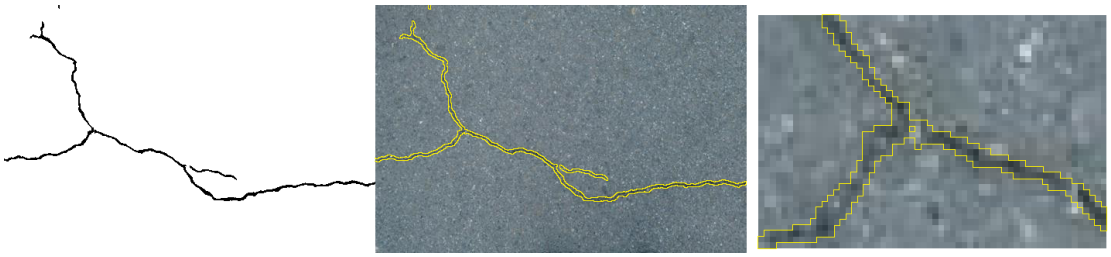
A comparison of the predictions of models used to obtain the scores reported in Table 3.7 is shown in Figure 3.12. There, we can observe that the prediction training with manual annotations without any additional measure produces a crack segmentation around 3 times wider than the observed crack. Congruently



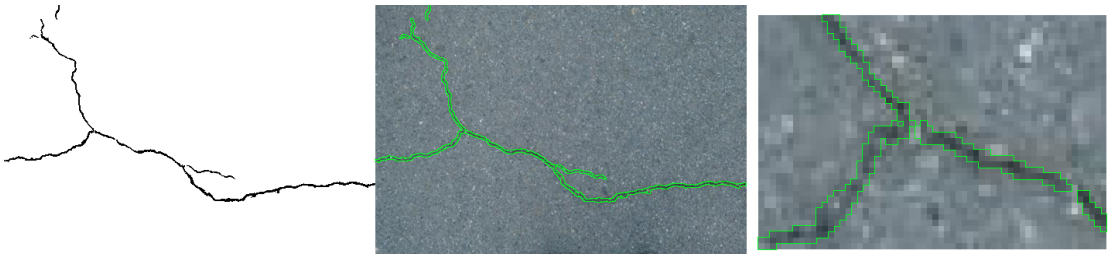
(a) Input image and manual annotation



(b) Training: manual annotations



(c) Training: manual annotations with data augmentation



(d) Training: 5-nn voting pseudo-labels

Figure 3.12: a) An image from CFD along with its manual annotation. b-d) The predictions of the models reported in Table 3.7; the middle column shows the border of the prediction over the input image; the right column shows a zoomed detailed view to appreciate better the predicted crack width with respect to the input image.

with the analysis of the unsupervised scores, using data augmentation makes the segmentation thinner; this prediction looks more similar to the visible crack. Finally, training with 5-nn-voting pseudo-labels produces a segmentation with a more detailed shape. This predicted shape looks even more similar to the visible crack. Again, this behavior is congruent with the unsupervised scores reported in Table 3.7.

Our results show that the proposed methodology reduces the impact of inaccurate labels at pixel-level. Inaccurate annotations are the result of the difficult and time-consuming nature of labeling this kind of images. Rather than dealing with the consequences of labeling errors, in the next section we propose an approach based on the images generated by Syncrack. Specifically, we propose a transfer learning approach to learn from synthetic images with accurate crack

segmentation ground truths.

3.6 Transfer Learning from Synthetic Images

The inaccuracy in manual annotations for crack segmentation is caused by the difficulty to obtain correct, pixel-accurate labels. Because of this, it is not feasible to produce big amounts of annotated images with accurate annotations. The Syncrack generator was originally developed with the purpose of producing benchmark datasets to evaluate methods intended to learn in the presence of inaccurate annotations with high class imbalance. However, there is an additional use for Syncrack-generated datasets: transferring models trained on these synthetic datasets to real-life images.

To test this possibility, in this section we provide experiments with 3 different versions of datasets produced by the Syncrack generator. To produce these datasets, we modified 2 user-accessible parameters: the background average smoothness (referred to as *bas*) and the crack average contrast (referred to as *cac*). The background smoothness intervenes both in the Perlin noise generation and the amount of added noise (see subsection 3.3.1). The crack contrast modifies the relative contrast between the background and the added cracks (see subsection 3.3.3). The selected values were chosen to create Syncrack datasets with 3 difficulty levels as described in Table 3.8.

The easy difficulty has a smooth background and well-contrasted cracks. The medium difficulty has rough textures with low-contrast cracks. It is worth noticing that this difficulty uses default values; this version is, in fact, the same one used for the experiments in section 3.4 and 3.5.2. The hard difficulty has the same contrast as the medium one, but the background is rougher. The 3 datasets, composed by 200 images each, share the same cracks (see Figure 3.13 for exam-

Table 3.8: Parameters used to create the 3 difficulty versions.

Difficulty level	Easy	Medium	Hard
Background smoothness (*bas*)	6.0	3.0	1.5
Crack contrast (*cac*)	0.5	0.7	0.7

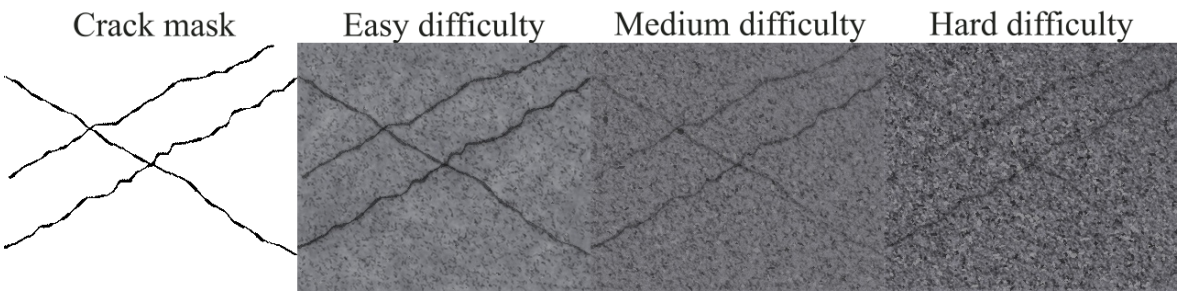


Figure 3.13: Syncrack images generated with different user parameters. The crack mask is the same, but the background generation and crack introduction parameter values differ to create different levels of difficulty for crack segmentation.

ple).

In the next experiments, we evaluate the predictions of models trained with synthetic data when predicting real-life images. We use CFD as the baseline for real images.

3.6.1 Transfer Learning Setup

As for previous experiments in this work, we implemented U-VGG19 using Tensorflow 2. For training, input images are cropped to 256×256 patches and fed as 4-patch batches. We used the Adam optimizer with an initial learning rate of 10^{-4} and default parameters. The initial weights from the encoder are the weights from VGG19 pre-trained on ImageNet and the whole U-VGG19 is trained together. Each synthetic dataset is randomly split into 50% training and 50% validation.

Our preliminary models trained on synthetic images showed promising results but failed to detect cracks in a small number of pictures. Specifically, images depicting bluish-looking pavement in contrast with the normal grayish colors. In order to easily transfer models learned from Syncrack, we decided to neglect the effect of color saturation due to the acquisition process, instead of trying to emulate this saturation with the Syncrack generator. To do this, all images are converted to grayscale and converted back to 3 channels by concatenation.

As discussed in subsection 3.5.2, data augmentation can help to improve the predicted crack width. In this transfer learning context, data augmentation is also useful to avoid overfitting to the distributions used by the Syncrack generator. We use data augmentation for all the experiments in this section. As in previous experiments in this work, our data augmentation consisted of randomly transforming an image (and its corresponding annotation) immediately before feeding it to the neural network for training. To do this, a random value for each of the 6 following operations is chosen: adding noise, changing illumination, flipping, zooming, rotating and shearing. Every image undergoes the 6 operations in the given order.

To refine the results at late epochs, we reduce the learning rate on validation loss plateau (by 2, with 5 epochs tolerance). To avoid overfitting, we add an early stop if the validation loss does not decrease during 20 consecutive epochs. The loss function is the one presented in Equation 2.9, based on BCE and DICE. For evaluation, images are fed without cropping using batch size 1.

3.6.2 Experiments and Results with the Different Syncrack Difficulty Levels

In this section, we evaluate the trained models again in terms of supervised (precision, recall, DSC) and unsupervised (H_{crack}, H^2_{crack}, K-S) scores. For synthetic images, we record the scores calculated on the validation split using the network weights with the minimum validation loss. We use these weights to evaluate the model on the test split of the real images. These scores are calculated per image, and we calculate the average over all the images in the split used to evaluate. As

in previous experiments, we report the average ± standard deviation obtained by training 10 times.

We divide our results into two categories: 1) training on Syncrack, evaluating on Syncrack; 2) training on Syncrack, evaluating on CFD. In these experiments, we train U-VGG19 using the clean accurate ground truths provided by the Syncrack generator.

Training on synthetic images, evaluating on synthetic images. Before moving to real images, we evaluate the impact of different background distributions when predicting the same cracks. To do this, we train U-VGG19 with each Syncrack difficulty level independently. The results of these experiments are shown in Table 3.9.

Table 3.9: Prediction scores obtained in the validation splits of the three Syncrack difficulty levels.

Score	Difficulty level		
	Easy	Medium	Hard
DSC(%)	95.9±0.6	78.7±0.7	66.7±0.7
Pre(%)	95.3±1.2	79.0±1.7	67.1±2.7
Re(%)	96.6±0.8	79.3±2.6	68.2±3.0
H_{crack}	3.59±0.02	3.94±0.01	4.07±0.01
H^2_{crack}	6.82±0.03	7.35±0.04	7.50±0.07
K-S	0.978±0.003	0.719±0.010	0.603±0.010

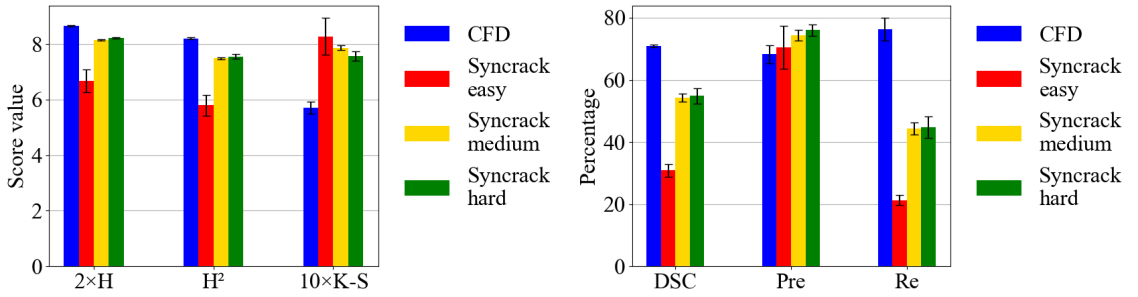
As expected, the DSC decreases as we increase the difficulty level (from 96% down to 67%). The precision and the recall decrease too; however, the rate of change is slightly different for both. When moving from the easy to the medium difficulty, the background becomes rougher, and the crack contrast decreases. In this context, the precision was reduced around 16% while the recall decreased around 17%. On the other side, when moving from medium to hard difficulty, the change of difficulty is produced only by increasing the background roughness (the crack contrast is kept equal). Now, the decrease in precision is around 12% while the decrease of recall is around 11%. In general, there is a tendency of the precision to decrease more than the recall when increasing the difficulty level; therefore, we observe that increasing the noise in the background texture had a greater impact at rising false positives.

Regarding the last point, it is relevant to remember that the background's roughness is increased, in part, by adding confusing crack-like artifacts to the image (see subsection 3.3.1). Therefore, as we increase the Syncrack difficulty level, the background becomes rougher and more populated with these hard-to-discriminate artifacts. The better a model performs as we increase the difficulty level, the more robust it is to false positives. This is directly related to the objective of improving the predicted crack width. Next, we evaluate the models trained on Syncrack predicting real-life images from CFD.

Training on synthetic images, evaluating on real images. In these experiments, our baseline is U-VGG19 trained with CFD (as originally reported in Table 2.1). We compare this baseline with the predictions of the models trained on each Syncrack difficulty level; we evaluate each model by predicting the CFD

Table 3.10: Scores obtained on the CFD test set by models trained on different datasets.

Score	Trained on CFD	Trained on Syncrack easy	Trained on Syncrack medium	Trained on Syncrack hard
DSC(%)	70.9±0.5	30.9±2.0	54.3±1.3	54.8±2.5
Pre(%)	68.3±2.9	70.5±6.8	74.3±1.7	76.1±1.8
Re(%)	76.2±3.7	21.3±1.6	44.4±1.9	44.8±3.5
H_{crack}	4.33±0.01	3.34±0.21	4.07±0.01	4.10±0.01
H^2_{crack}	8.20±0.04	5.79±0.37	7.48±0.05	7.54±0.08
K-S	0.570±0.022	0.827±0.066	0.786±0.009	0.756±0.016



(a) Unsupervised scores

(b) Supervised scores

Figure 3.14: Scores obtained on the CFD test set by using models trained on different datasets. For the unsupervised the scores: the lower the entropies (H, H^2), the better; the greater the Kolmogorov-Smirnov score (K-S), the better.

test split. These results are presented in Table 3.10. Separate plots for the supervised and unsupervised scores of this table are presented in Figure 3.14.

With respect to the supervised scores, we can see that the models trained with Syncrack datasets exhibit a higher precision than the one trained with real images. On the other side, the recall is lower. The model trained on the easy Syncrack achieves the lowest supervised scores among the three datasets. As we increase the Syncrack difficulty, all the supervised scores increase; the scores that increase the most are the DSC and the recall.

Regarding the easy Syncrack, the dataset has only easy-to-detect cracks. Consequently, a model trained with this dataset will struggle to detect cracks with low contrast or rough backgrounds. This explains why there is a high confidence in the pixels predicted as cracks but the model misses hard-to-detect cracks. As we increase the Syncrack difficulty level, the learned models detect harder cracks (increasing recall) and incur in less false positives (increasing precision).

Among the models trained with synthetic images, the model trained with the hard Syncrack has the better trade-off between precision (76%) and recall (45%). The DSC of this model is 55%, in contrast with the 71% obtained by training with CFD; this DSC difference is caused only by a decreased recall. However, in the context of inaccurate annotations, these scores are not reliable to measure the quality of the trained models.

As discussed in this chapter, the manual annotations tend to be wider than the actual cracks. Therefore, when evaluating with manual annotations, a more accurate segmentation preserves the precision but leads indeed to a lower recall.

Let us assume a scenario where, in average, the manual annotations are twice as wide as the cracks: the recall of a perfectly accurate segmentation would be merely around 50%. Although the proposed scenario may sound extreme, in practice it is not, as illustrated in Figure 3.15.

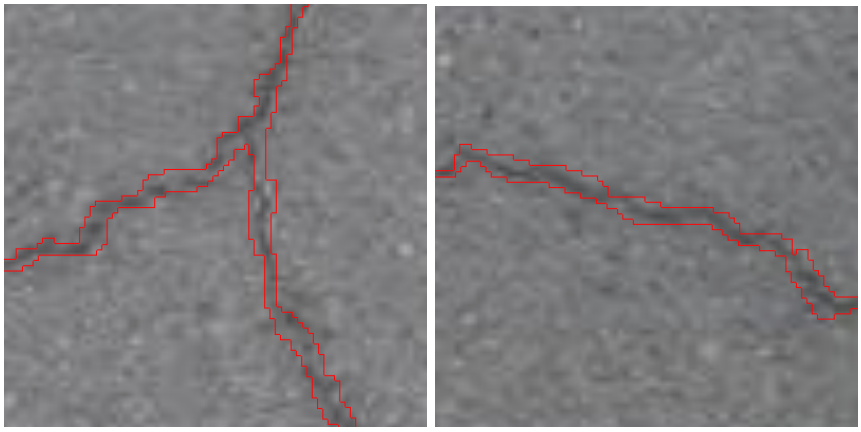


Figure 3.15: Borders of manual annotations in CFD. The annotations can be more than twice as wide as the visible crack in many regions.

Since we don't have accurate ground truths for evaluation, we rely on unsupervised scores to evaluate if the models trained on synthetic images are actually improving the prediction. From the plot in Figure 3.14a, we can observe that the models trained with Syncrack have better scores than the one trained on CFD. All the models trained with synthetic data exhibit lower entropies; this denotes that the regions segmented as cracks have a high uniformity in terms of pixel intensity and textures. This is likely to occur if such regions are dominated by crack pixels. As a complement to this, we see that all the models trained with synthetic data exhibit a greater Kolmogorov-Smirnov score; this denotes that there is a difference between the distributions of the regions segmented as cracks and the ones segmented as background. This is only possible if the regions segmented as cracks have a very low (or no) amount of background pixels.

In summary, the unsupervised scores support that the decrease in recall, with respect to the model trained on real data, is caused mainly by correctly classifying pixels in the excessively wide annotations as background. This is particularly true for the medium and hard difficulty, which have a considerably greater recall with respect to the easy Syncrack. A qualitative analysis of the predictions confirmed this: as illustrated in Figure 3.16, the models trained with the medium and hard Syncrack perform a good job at locating cracks (not missing crack segments); moreover, the predicted crack geometry (shape and width) looks more similar to the visible crack that both the manual annotation and the prediction training with CFD (see Figure 3.17).

The model trained on easy Syncrack produces refined segmentations but prone to contain small discontinuities; it has a higher crack-segment-miss rate than the other tested models. The models trained on medium and hard Syncrack tend to fill these gaps, but their predictions look coarser. Particularly, the predictions of the model trained on hard Syncrack are coarser than the predictions of the model trained on medium Syncrack. However, the model trained on hard

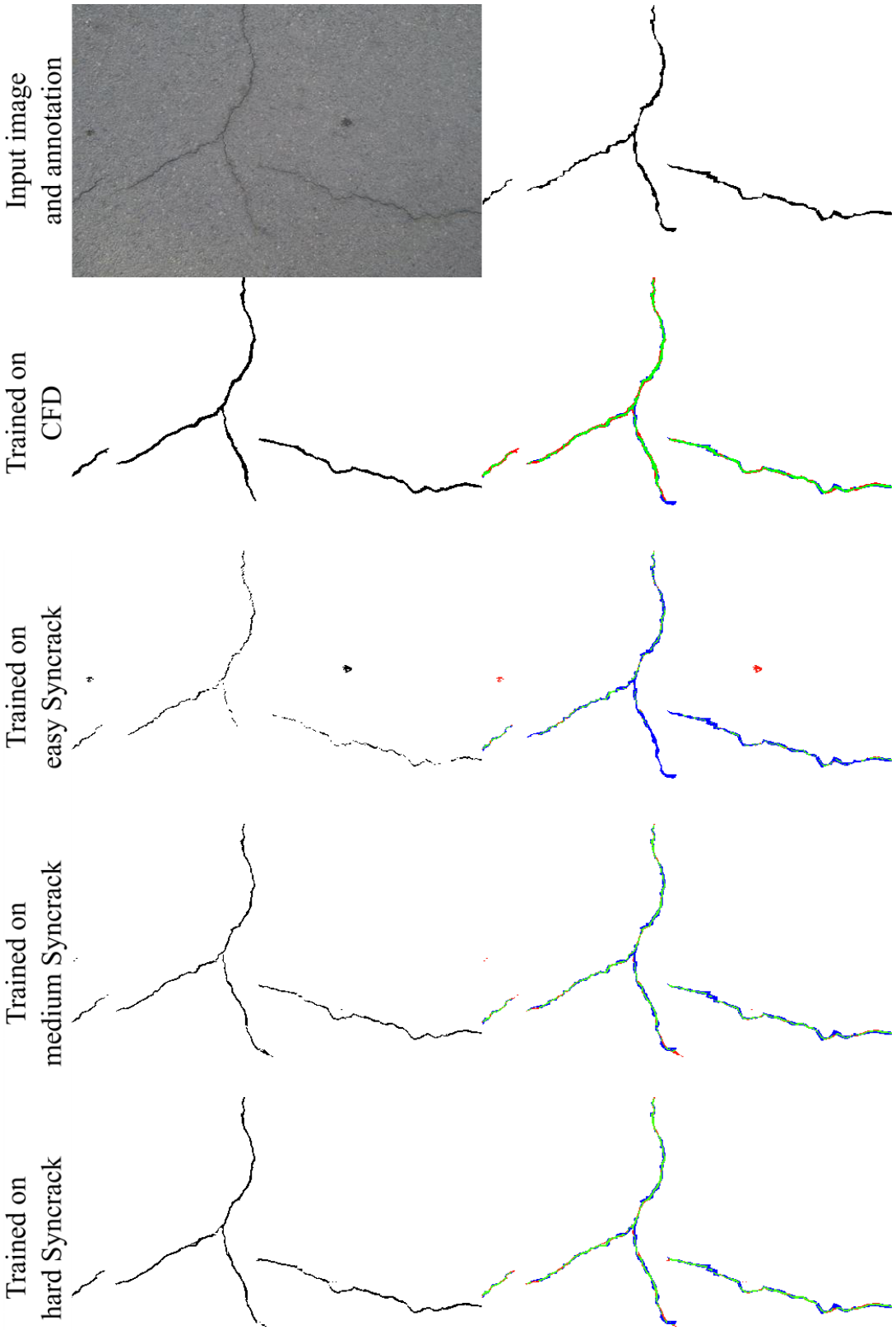


Figure 3.16: Example of predictions on a CFD image. The first row shows the input image and the provided annotation. The next rows show the predictions of models trained on the different proposed datasets and its comparison with the manual annotation; the color code is: (Green) True positives; (Blue) False negatives; (Red) False positives.

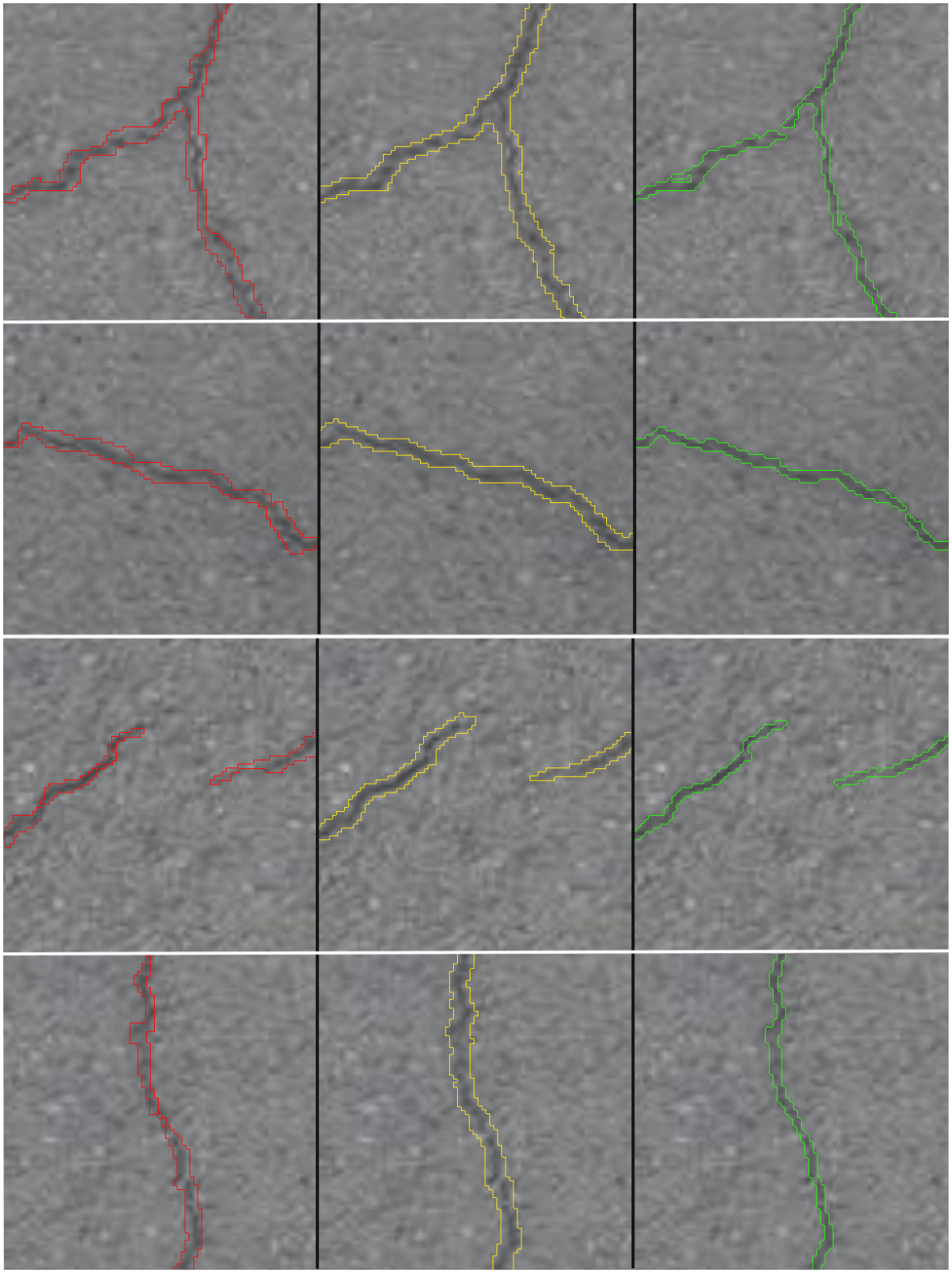


Figure 3.17: A close-up detailed view to different regions of the picture depicted in Figure 3.16. The red line is the border of the manual annotation, the yellow line is the border of the prediction obtained by training on CFD, and the green line is the border of the prediction obtained by training on Syncrack hard.

Syncrack is more conservative at the moment of discarding crack segments (i.e. it is conservative to provide false negatives).

In this section, we studied the potential of using Syncrack generated images for real-life supervised crack segmentation without requiring manually labeled real-life images. The models trained solely on Syncrack-generated datasets showed to be transferable to real images; furthermore, the predictions of these models are more precise in terms of crack width. This means that synthetic image generation is indeed a promissory approach to deal with the problem of inaccurate annotations: there is no need for humans labeling images, prone to error, if the images and their corresponding ground truths can be generated synthetically. The parametrizable nature of Syncrack allows adapting the generated datasets to be similar to different distributions corresponding to different real contexts.

With these results, we finish our contributions regarding the problem of accurate segmentation of cracks in presence of inaccurate annotations. Next, we close the current chapter with a joint discussion of all the presented results, as well as some future perspectives.

3.7 Discussion on Crack Segmentation in Presence of Inaccurate Annotations and Future Perspectives

In this chapter, we approached the problem of segmenting cracks in presence of inaccurate annotations. On one hand, it is difficult to produce accurate segmentations when the manual segmentations used for training are inaccurate. On the other hand, it is difficult to produce accurate manual segmentations because the annotation task is highly time consuming and very prone to human error at different levels.

Among the possible errors, the scope of this work was centered on the errors on the level of pixels: labels excessively wide, excessively thin or inaccurately placed. Particularly, manual annotations tend to be wider because of the fuzzy boundaries between cracks and background. This bias is reflected on the models trained with these annotations, providing segmentations typically wider than the visible cracks. In this context, we aim to produce pixel-accurate segmentations that respect the geometry of the cracks.

To study the problem and possible solutions to inaccurate annotations for supervised crack segmentation, we introduced the Syncrack generator as a benchmark. With the help of the accurate ground truth segmentations provided by Syncrack, as well as its module to introduce noise into clean annotations, we found that increasing the inaccuracy of the annotations used for training is prone to produce segmentations wider than the observable cracks. Nonetheless, increasing this inaccuracy actually can be helpful to reduce the amount of pixels predicted as false negatives. In fact, purposely increasing the width of the manual annotation (e.g. through dilations) can improve the ability of a model to locate cracks; this comes with the cost of drastically reducing the geometrical quality

of the produced segmentations.

This principle can be used in the future for nested methods: a first stage dedicated to locating cracks with low emphasis in the geometry and a second stage dedicated to refine the segmentations in order to provide predictions with accurate shapes corresponding to the visible cracks. In this chapter, we centered our efforts in the later.

The accurate ground truth annotations from Syncrack allowed a supervised scoring of models trained using annotations with different label noise levels. From these analyses, we confirmed that increasing the label noise during training reduces the precision and increases the recall (supporting the principle mentioned in the last paragraph). This supports the claim that this kind of scores is not reliable in real-life contexts where ground truths are not accurate.

To expand our evaluations to real-life images, we proposed three scores used previously for unsupervised segmentation: H_{crack}, H^2_{crack} and K-S. In our experiments training with Syncrack, we showed that these scores exhibit an approximately linear relation with the precision; this relation is useful to evaluate if the crack-predicted regions contain or not actually background pixels. When background pixels contaminate the cracks, the entropies increase and the K-S decreases, rapidly, because the accumulated area of crack pixels is typically very low. Although these metrics allow to measure if the produced segmentations are wider than the visible crack, it is worth to notice that they are not very sensitive to the opposite case: mislabeling crack pixels as background. Because of the heavy class imbalance associated to crack segmentation, even mislabeling all the crack pixels as background will not produce relevant changes in the intensity distribution of pixels segmented as background. Further work is needed towards unsupervised metrics that allow measuring the overall quality of models' predictions, beyond identifying when the predictions are wider than the cracks.

With the help of supervised and unsupervised scores, we evaluated the performance of a method inspired by weakly supervised learning. The method consists of 1) generating pseudo-labels through a method used for learning in the presence of inaccurate labels, and 2) using these pseudo-labels to train a final model. Although the 4 proposed pseudo-label generation methods (summarized in Table 3.4) improved the precision and the unsupervised scores, the one with best results was 5-nn voting. This method relies on using a backbone network (U-VGG19 in our case) as a per-pixel feature extractor and applying a 5-nn algorithm to produce a pseudo-label per pixel. By using these pseudo-labels, the trained models at different label noise levels kept a quasi-constant DSC between 1-2% below training with clean labels. The same holds true for the precision.

We tested the 5-nn voting approach with real-life images to evaluate its suitability on non-synthetic distributions; we performed this experiment on the CFD+ Aigle-RN dataset. Through the decrease of the entropies and the increase of the K-S, along with a qualitative analysis of the segmentations, we confirmed that these segmentations improved in terms of crack width with respect to training with manual annotations.

5-nn voting allows avoiding overfitted decision boundaries. However, the number of neighbors is a manual hyperparameter to tune. Increasing this value will increase the complexity without a guarantee of improving the expected re-

sults. In the current work, the feature vector per pixel was 2D. Increasing this dimensionality could improve the pseudo-label generation by providing more separable clusters between classes; this could be combined with models or loss functions that produce feature spaces requiring less complex boundaries between classes (e.g. linear boundaries).

Finally, we approached the problem of inaccurate annotations during the annotation stage itself. Since Syncrack produces datasets with ground truth annotations, we studied the potential of using models trained on Syncrack-generated datasets to segment real-life images. The models trained solely on synthetic data showed to be transferable to images from the CrackForest dataset. Moreover, as measured by the decrease of the entropies and the increase of the K-S, the predictions are more precise in terms of crack width than training with real images.

These results suggest that synthetic image generation is a good approach to deal with inaccurate annotations from the root. The user-accessible parameters from Syncrack already allow generating datasets similar to different real-life distributions. Moreover, this generator can be extended. Indeed, the modularity of Syncrack allows extending or replacing each module by more sophisticated algorithms or techniques such as generative networks. For example, the Syncrack generator could be extended by substituting the background generator with a Generative Adversarial Network; this would allow minimizing the difference between the Syncrack generated distribution and a distribution of real-life images. At the same time, the modularity of Syncrack allows working with real, non-cracked, textures: the background image for crack introduction can be a real picture rather than a synthetic image.

The focus of this chapter was on the monitoring of cracks: defects that appear on the finished construction over time. For the specific case of 3D concrete printing, there is an additional type of object that is perceptually similar to the cracks: the interlayer lines dividing two extruded layers. In the next chapter, we use U-VGG19 to segment these lines, with a focus on inline monitoring of the additive manufacturing process.

Chapter 4

Interlayer Line Segmentation in 3D Concrete Printing

3D concrete printing based on contour crafting consists of extruding concrete in a layer-wise manner. Within a lateral view of a piece produced by 3DCP, we observe the extruded layers one over the other. The simplest way to differentiate one layer from its neighbors is to find the transition regions between such layers. These regions, referred to as interlayer lines from now on, determine the boundaries of individual layers (see Figure 4.1).

Figure 4.1: Example of interlayer lines on a dry piece produced by 3DCP. The interlayer lines are highlighted in green.

For post-printing analysis, the interlayer lines in dry pieces tend to take the appearance of thin, black lines in the surface (similarly to cracks). Although this holds true for good quality printing, the behavior changes drastically as the printing quality decreases. In this scenario, detecting the interlayer lines becomes more difficult. Although we are still in the case of locating thin, long lines, finding black lines is no longer sufficient (see Figure 4.2).

In fact, cases like these, in which the interlayer lines show abnormal behaviors, are the most informative. As discussed in the next chapter, the geometry of the interlayer lines provides information about the interaction between layers, allowing to identify defects. Using the Figure 4.2 as reference, we can identify behaviors such as: overlapping layers (first and second picture), leaning layers (second picture), variable intra and inter-layer thickness (all the pictures), etc.

The analysis of the interlayer lines can be automatized using computer vision.

Figure 4.2: Examples of bad quality pieces in which the interlayer lines are difficult to recognize.

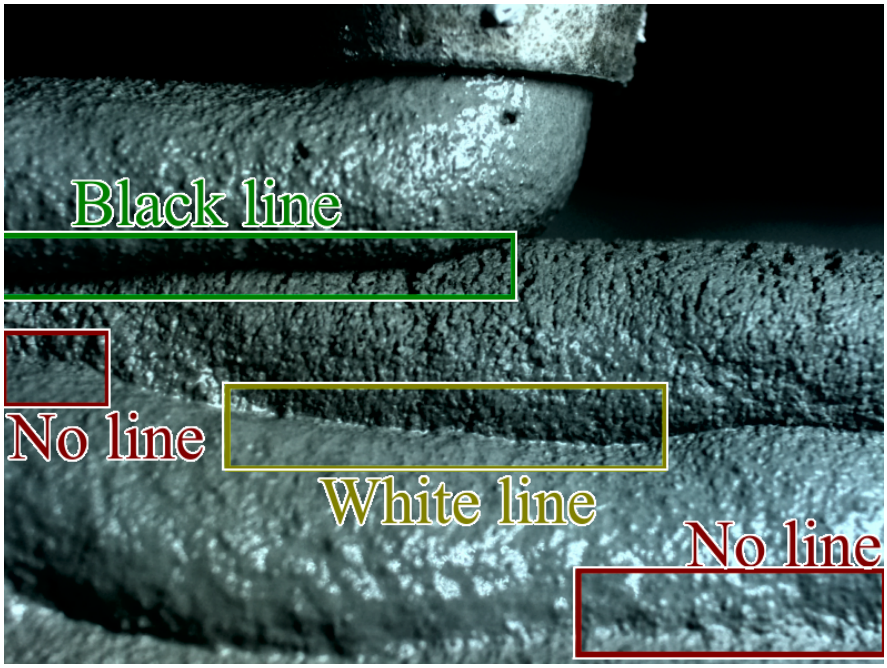


Figure 4.3: The interlayer lines in fresh concrete can appear black, white, or invisible, because of layer superposition and specular reflectance caused by the material's humidity.

To do this, the first necessary step is to segment the interlayer lines in an image. This chapter is dedicated to study the problem of interlayer line segmentation for inline monitoring of the 3DCP process.

Inline monitoring exhibits a higher difficulty with respect to segmenting the interlayer lines after printing (when the concrete is dry), since we must deal with fresh material. On one hand, the extruded concrete has an increased yet variable specular reflectance due to its humidity and the additives in the mixture. On the other hand, the fluidness of fresh concrete can produce layer merging and superposition. Consequently, the interlayer lines can take different aspects: black, bright or even not visible at all (see Figure 4.3).

Additionally to the challenges added by fresh concrete, inline monitoring has to deal with the constraints imposed by the physical setup used for image acquisition. Given the nature of the process, the environment is hazardous and it is difficult to set ideal acquisition conditions, specially in terms of proper illumina-

tion and camera position.

The rest of this chapter is organized as follows. First, we discuss the setup used to acquire the images used in our experiments, as well as the properties of the images in the resulting dataset: I3DCP. Next, we provide a summary of the baseline model used for our experiments: U-VGG19, first introduced in chapter 2. Afterwards, we discuss a method for semi-supervised learning beginning from a small number of annotated images. Then, we show our experiments and results for the segmentation of interlayer lines. Finally, we close the chapter with a discussion on the obtained results and future perspectives.

4.1 Image Acquisition During 3D Concrete Printing

Several setups for image acquisition are possible regarding inline monitoring. In the current work, the chosen setup was meant to provide local segmentation, inspecting only the area underneath the printing nozzle. Particularly, as illustrated in Figure 4.4, we chose a camera fixed to the nozzle in order to have a fixed point of view with respect to the extrusion zone. In order to simplify the analysis, the camera orientation should remain perpendicular with respect to the printed wall. The problem of orientation also exists when printing with non-circular nozzles, which are commonplace in 3DCP; although out of the scope of this thesis, there are technical solutions to this issue (Liu et al. [2020]).

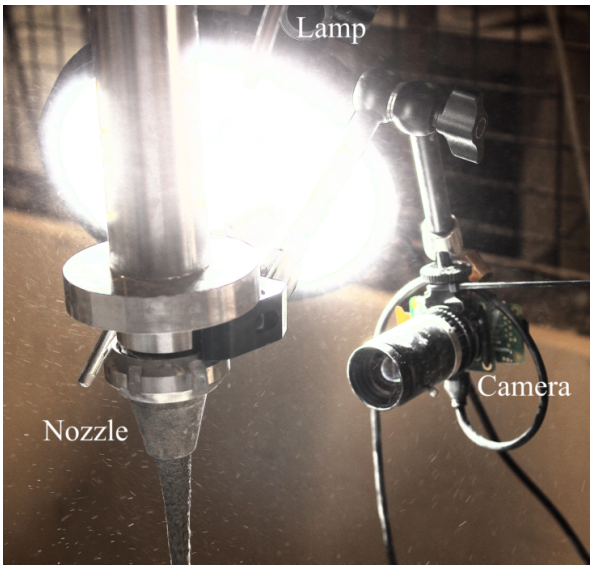


Figure 4.4: Setup used for inline image acquisition.

The image acquisition was performed using a Raspberry Pi Camera *High Quality* Module featuring a SONY IMX477 CMOS sensor with a macro lens. The distance between the lens and the nozzle was ~13cm. Due to eventual deformations or lack of control of width, small apertures in the camera are preferred in our applications: for a smaller aperture, the depth of field increases, allowing obtaining sharp images even if the distance between the observed concrete and

the lens differs from the expected value. Smaller apertures let in less light; because of this, a small aperture is usually used with slow shutters to increase the exposure time. However, mitigation of motion blur requires fast shutter speed. Here, we are mostly in the case of linear motion blur. In order to limit the blur, we get a constraint on exposure time. We mounted a led lamp on the nozzle to allow a short, and constant, exposure time. This setup is illustrated in Figure 4.4.

4.1.1 Inline 3D Concrete Printing (I3DCP) Dataset

For the results presented in this paper, we created a dataset from images recorded during a printing session run in the 3D printing laboratory of École des Ponts ParisTech. The cell is composed of a 6-axis industrial robot (ABB IRB 6620), with the xHEAD extruder developed by XTreeE. The system uses the bi-component technology, which allows modifying the rheological properties at the nozzle exit. The material is the NAG3 concrete developed by Lafarge and tested in (Esnault et al. [2018]). The nominal speed is 80mm/s, with a nozzle diameter of 20mm, while the pipe diameter is 30mm. The typical flow rate is 1L/min but may slightly vary during the experiment. The robot controller controls the position and speed of the robot, the flow rate of the dosing pump and the flow rate of accelerator. The printing parameters used during the session were dynamically varied to purposely create imperfections (see Figure 4.5).

The goal of this session was to produce a piece able to support itself (i.e. avoiding a collapse due to its own weight) but with layer deformations. These defor-

Figure 4.5: The experimental sample printed with purposely created defects.

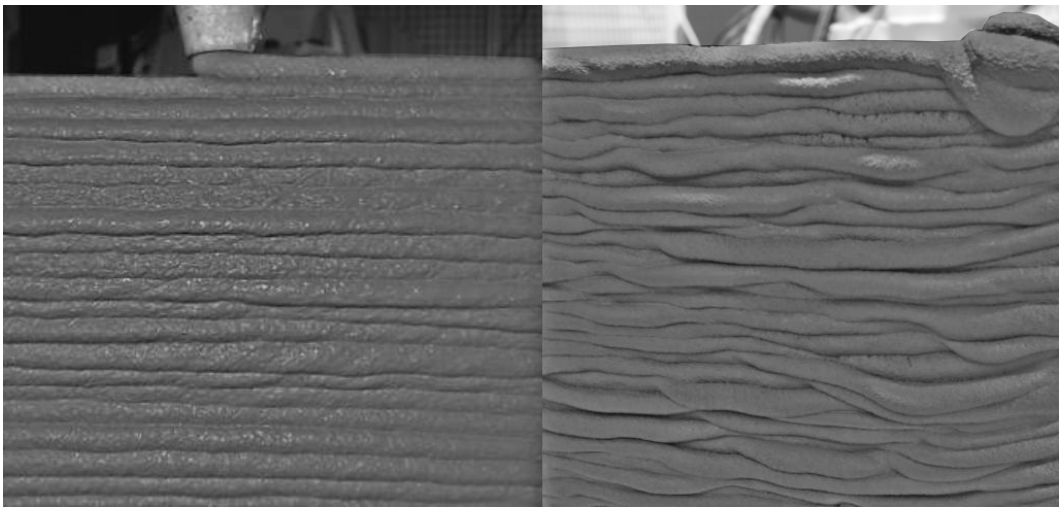


Figure 4.6: A comparison between good (left) and bad (right) layer geometry in fresh pieces.

mations would lead to the rejection of this piece. A comparison between good layer geometry and the defective piece is presented in Figure 4.6.

The distance of the lens was adjusted to contain approximately 4 layers per frame, with an expected layer width of 6mm. The image resolution is 1280 × 960 pixels; therefore, our images contain ≈40 px/mm. Regarding the camera settings, the focus, exposure time and lens aperture were manually fixed prior to printing the piece. Despite the nozzle speed during printing and the small field of view, the rolling shutter distortion was negligible.

To have uniformity in our final dataset, from the acquired images we chose a subset of pictures meeting 2 conditions: 1) no background is visible at the sides of the printed piece (similarly to the images depicted in Figure 4.6); 2) the center of the nozzle is coplanar with the center of the observed wall (as illustrated in Figure 4.7). To ensure that the totality of the currently extruded layer is visible, a small portion of the extrusion nozzle must be visible. The final dataset, referred to as I3DCP, is composed by 628 TIFF images ordered chronologically. Example images from I3DCP are illustrated in Figure 4.8.

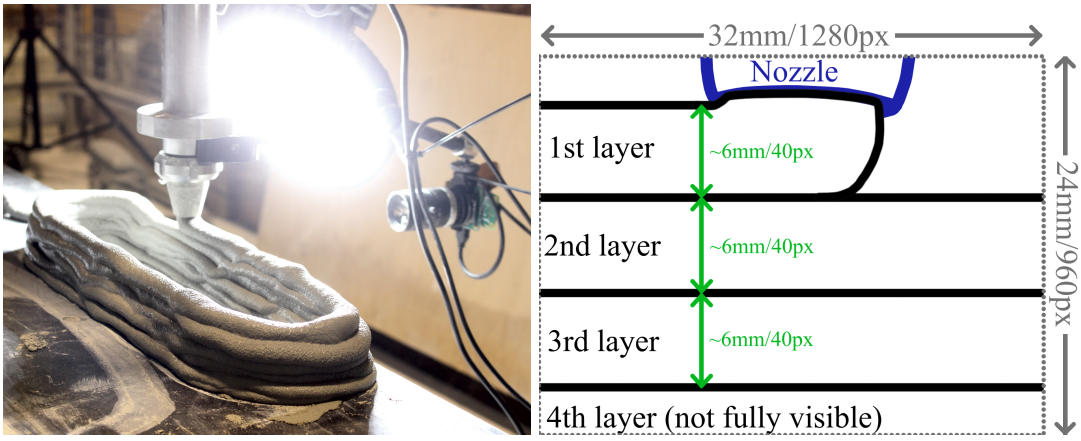


Figure 4.7: The printing process with inline image acquisition, and scheme of expected output.

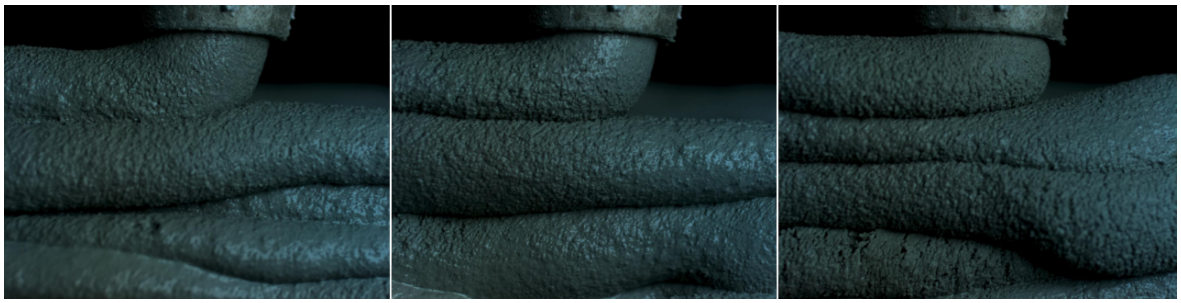


Figure 4.8: Examples from the I3DCP dataset: the first, middle and last image, respectively from left to right. The dataset is ordered chronologically.

In the next section, we provide a summary of the baseline model used to segment these raw images: U-VGG19.

4.2 U-VGG19 for Interlayer Line Segmentation

Once the images are acquired, the goal is to segment the interlayer lines in them. 3DCP interlayer line detection has been barely explored in the literature, particularly for post-printing monitoring (Davtalab et al. [2020]). Beyond the difference between dry and fresh material, there are relevant differences in the geometrical properties of the extruded layers depending on the used printing technology. For example, the printing process used in our case produces pieces with very different geometrical properties than the ones analyzed by Davtalab et al., as seen in Figure 4.9. The main difference is the size and shape of the extruded layers, having rectangular layers with vertical width of 25mm in the case of Davtalab et al. In our printing process, the layers are extruded with a cylindrical shape and a vertical width of 6mm.

Figure 4.9: Comparison of wide rectangular extrusion (left) as illustrated by (Davtalab et al. [2020]) and thin cylindrical extrusion (right) as used in our experiments.

These differences in width and shape tend to produce shadows and rougher visible textures in the images from I3DCP. Furthermore, given the constraints imposed by inline image acquisition, the problem becomes particularly hard: specular reflectance, superposition and merging of layers, loss of focus, unexpected motion blur, camera vibration, polluted air (presence of particles), etc. All these conditions hinder accurate interlayer line detection with methods based on traditional image processing as the one provided by (Davtalab et al. [2020]) (based on texture smoothing and canny edge detection).

To overcome these challenges, we proposed a method based on Deep Learning using as a basis the U-VGG19 architecture introduced in section 2.2. It is a fully convolutional neural network inspired by U-net which uses VGG19 as backbone model. The architecture of U-VGG19 can be summarized as an encoder-decoder network with skip connections between the encoder and the decoder at different resolutions. By using this network, we get as output a binary mask with the location of the interlayer lines in the input image (as illustrated in Figure 4.10).

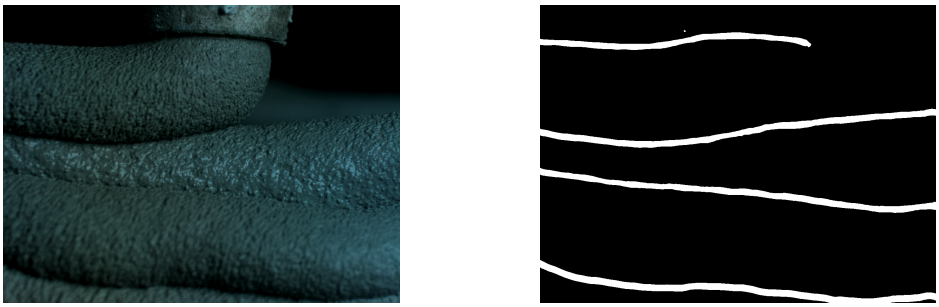


Figure 4.10: Example of interlayer line segmentation using U-VGG19.

Since U-VGG19 is a model meant for supervised segmentation, segmentation maps for training must be provided. However, manual segmentation is a highly time-consuming task. In the next section, we discuss the method used to train our final model from a reduced number of annotated images.

4.3 Semi-Supervised Learning for Interlayer Line Segmentation

To train our final model, first we annotated a small subset of images i.e. we obtained a small amount of labeled data, which is insufficient to train a good learner, while abundant unlabeled data was available. This is a case of incomplete supervision, which is again a type of weak supervision (Zhou [2017]). To learn in this type of context, active learning is extensively used: it assumes the existence of an "oracle" (such as a human expert) that can be queried to get ground truth labels for selected unlabeled instances (Settles [2009]). For the segmentation of interlayer lines, rather than producing queries at the level of pixels, queries at the level of patches are preferred –similarly to the approach proposed by (Li et al. [2020]).

During active learning, the unlabeled samples are typically ranked to select only a reduced subset of samples per query. The motivation of this is reducing the time and effort invested into labeling. Rather than choosing a criterion to rank unlabeled images, we decided to ease the annotation task by using the principle of self-training i.e. the model is retrained using its own predictions. Under this approach, the role of the oracle is to evaluate (and correct, if necessary) the predictions of the model on unlabeled images (instead of annotating the whole image). Because of this difference, we refer to our method as a semi-supervised approach rather than an active learning approach.

This method, summarized in Figure 4.11, was used as follows: 1) U-VGG19 was trained with a small set of manual annotations, 2) it was re-trained from scratch with automatically generated segmentation maps, and 3) it was fine-tuned with manually corrected automatic segmentation maps.

The first step is meant to learn a basic model from the domain. This model serves as a simple automatic annotator, allowing to increase the amount of annotated images by annotating images that are not seen during training. Since the model can be considered as a novice annotator, a human supervisor is in charge of selecting the images in which the model produced good segmentations.

In the second step, we train the network from scratch using only automatically generated segmentations. In this way, we can preserve the manually annotated images as a test set. This new model is trained with more images than the one trained in the first step, and therefore is less prone to overfitting.

In the third step, the human supervision takes a more active role. After predicting the images not seen during training, instead of selecting the images with good segmentations, the human supervisor selects images with bad segmentations. These images are considered to be difficult examples, and we wish to help the model to produce better segmentations in them. To do so, the automatically produced segmentations are first corrected by the human supervisor. Then, the

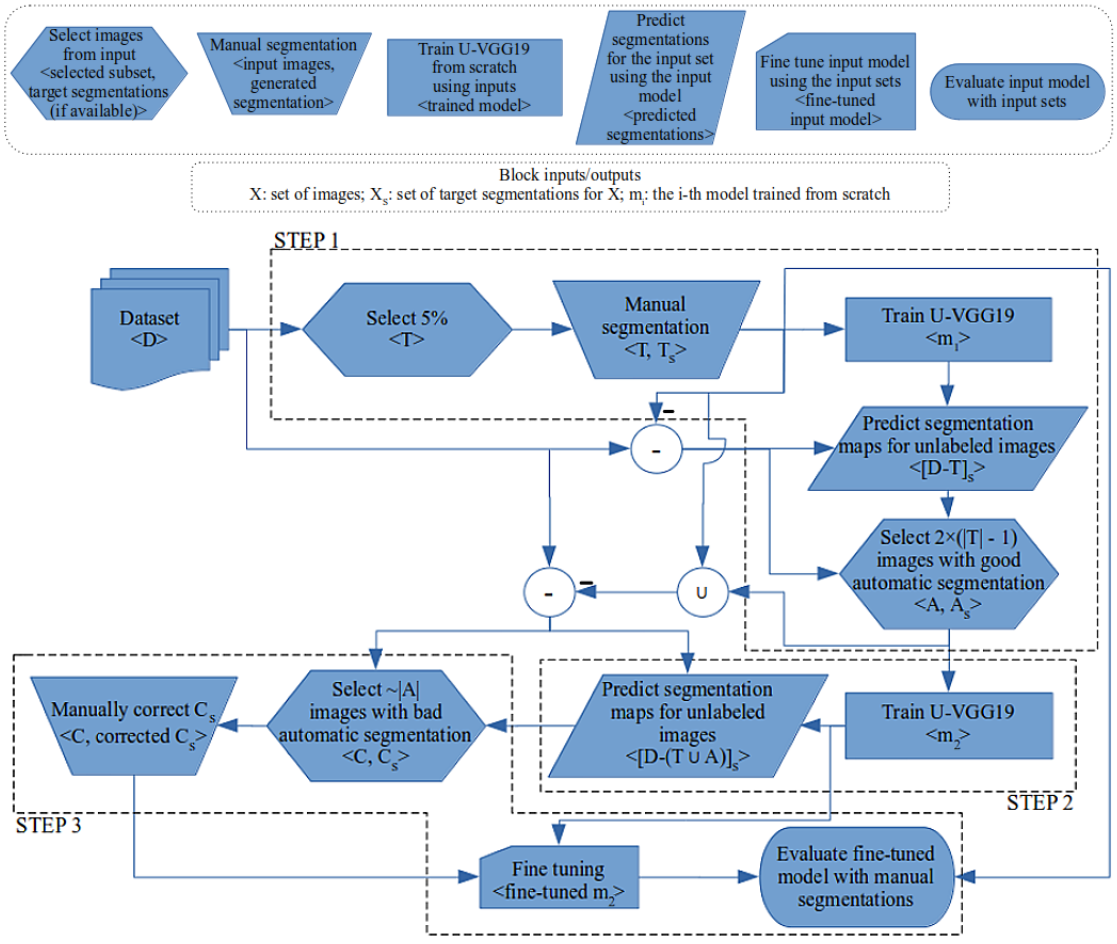


Figure 4.11: Flowchart of the method used to train the interlayer line segmentation model.

trained model is fine-tuned by adding these difficult images and their corrected segmentations to the set of images used for training.

The three aforementioned steps are formalized next.

4.3.1 Initial Training with a Reduced Amount of Annotated Images

To create the initial training set, we chose a sample of images from I3DCP. To promote diversity, the images are equidistant in the ordered dataset. Formally, let us consider the full dataset of images as $D = \{D^{(n)} | n \in [1, 2, ..., N]\}$; here, $D^{(n)}$ is the n-th image in the dataset (D) and N is the size of the dataset (628, in the case of I3DCP). Given $t(D^{(n)})$ the timestamp in which $D^{(n)}$ was captured, $t(D^{(n)}) < t(D^{(n+1)})$. To obtain 5% from the total number of images (i.e. 32), we use a step of 20. Therefore, the final subset used for training is defined as $T = \{D^{(n)} | n = 20m+1\}$, $m \in \mathbb{Z}$.

Since the dataset is sorted chronologically, this ensures to utilize images from diverse time stamps. A binary segmentation map was manually generated for each of these images, having a value of 1 for interlayer lines and 0 everywhere else. The width of the interlayer lines was set to 20 pixels (i.e. ~0.5mm). A training image and its manual annotation are illustrated in Figure 4.12. Let I_S be the set of target segmentations for a given set of images I.

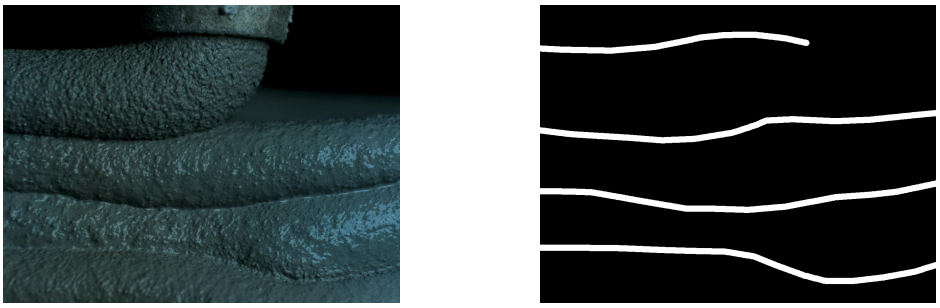


Figure 4.12: Example of manual annotation in the I3DCP dataset.

Given the set T of the selected images, and their corresponding annotations T_S, we train a first instance of U-VGG19; we refer to the fitted model as m_1. To train m_1, we use 80% of the images from T for training and the other 20% for validation. Once this fully-supervised step is done, m_1 is used to predict segmentation maps for all the remaining images without annotations in D (i.e. D−T). These automatic segmentations are used to extract a new sample of images from D.

The new sample is extracted with human supervision. Given $T^{(i)}$ the i-th image in the ordered T (i.e. $t(T^{(i)}) < t(T^{(i+1)})$), consider $P = \{(T^{(i)}, T^{(i+1)}) | i < |T|\}$ the set of all the pairs of consecutive images in T. For each pair $P^{(i)} \in P$, we select two images $D^{(n)} \in D$ s.t. $[t(T^{(i)}) < t(D^{(n)}) < t(T^{(i+1)})] \cap [D^{(n)} \in G]$. Here, G is the set of images for which m_1 predicted good segmentations, as determined by a human annotator as oracle. Let $A \subset G$ be the set of all the selected images; since $|T| = 32$, the cardinality of this new set A is 62 (i.e. $2 \times (|T| - 1)$).

4.3.2 Training with Automatically Segmented Images

The set A is composed by images that were not seen during the training of m_1 i.e. $A \cap T = \varnothing$. From now on, the images T and their corresponding manual segmentations T_S are considered to be our test set. A new instance of U-VGG19 is then trained from scratch, using the images from A and their corresponding predicted segmentations A_S (as predicted by m_1). We refer to the fitted model, trained only with automatically generated segmentation maps, as m_2. Considering that $|A| \approx 2 \times |T|$, and that the target segmentations A_S were verified by humans, m_2 is expected to produce better segmentations than m_1.

Then, m_2 is used to predict segmentations for the remaining unlabeled images in D i.e. $D - (T \cup A)$. Using the predictions of m_2 as reference, a new set of images is chosen containing only images with bad segmentations (i.e. images that are difficult to segment).

4.3.3 Fine-Tuning with Corrected Difficult Images

The images with bad predicted segmentations are cases in which the model should improve. To do so, we fine-tune m_2 with this type of images. Specifically, a set $C \subset [D - (T \cup A)]$ of difficult images is chosen so that $|C| \approx |A|$. The size of C is chosen to have a balance between –unseen– hard images and already seen images for training. For each image in C, the corresponding target segmentation predicted by m_2 is corrected by a human annotator. Then, the images in $A \cup C$ are used to fine-tune m_2 using A_S and the corrected C_S. The fine-tuned m_2 is finally evaluated using T and T_S for testing.

4.4 Experimental Setup for Interlayer Line Segmentation

For the method proposed in this chapter, we used U-VGG19 as the baseline model. We implemented this network in Tensorflow 2. As done for crack segmentation, the initial weights from the encoder are the ones from VGG19 pre-trained on ImageNet and the whole U-VGG19 is trained together. Similarly to cracks, the interlayer lines in an image typically represent a very small percentage of all the pixels. Because of this, we use the loss function presented in Equation 2.9, which is based on the Dice score coefficient and binary cross-entropy. The Adam optimizer was chosen for training, using the default parameters of Tensorflow. During training, input images are cropped to 256×256 patches to reduce space in memory while generally preserving at least one interlayer line per patch (the expected layer width is ~240 pixels). This setup also allows training with datasets containing multiple-size images without resizing.

When training a model, either from scratch or fine-tuning (see blocks including m_1 and m_2 in Figure 4.11), the input images are randomly split into 80% training and 20% validation. The training is performed using a batch size of 4, while validation is performed by using entire images without cropping using batch size of 1. To refine the results at late epochs, we reduce the learning rate

on validation loss plateau (by 2, with 5 epochs tolerance). Each training process is performed using 150 epochs at most; to avoid overfitting, we add an early stop if the validation loss does not decrease during 20 consecutive epochs.

The final scores that we report are calculated using the network weights at the epoch with the minimum validation loss. We calculate precision (Pr), recall (Re) and Dice score coefficient (DSC) per image and we report the average on the images from T as test set (the images with manually generated segmentation maps).

4.5 Experiments and Results on Interlayer Line Segmentation

In this section, we present the results obtained by training the models m_1 and m_2 during each respective step in the proposed method. The first step is to use the set of images T and their corresponding manual annotations T_S to train U-VGG19; after training, the resulting model is referred to as m_1. For this first training, we used an initial learning rate of 10^{-4}. Since in this particular case the images in the training and the validation splits are the same as in the test split (T), we show only the train and validation scores. Given that $|T| = 32$, the training and the validation sets were composed of 25 and 7 images, respectively. The scores on these sets during training are plotted in Figure 4.13.

As seen in the figure, U-VGG19 achieves a DSC above 85% on the training and on the validation splits since the first 20 epochs. The training is stopped early at epoch 92, meaning that the validation loss did not improve from epoch 72; we preserve the weights from this epoch for further steps in the method. As expected from the use of a loss function based on DICE, as the number of epochs increase, the precision and the recall tend to get closer values to each other. At

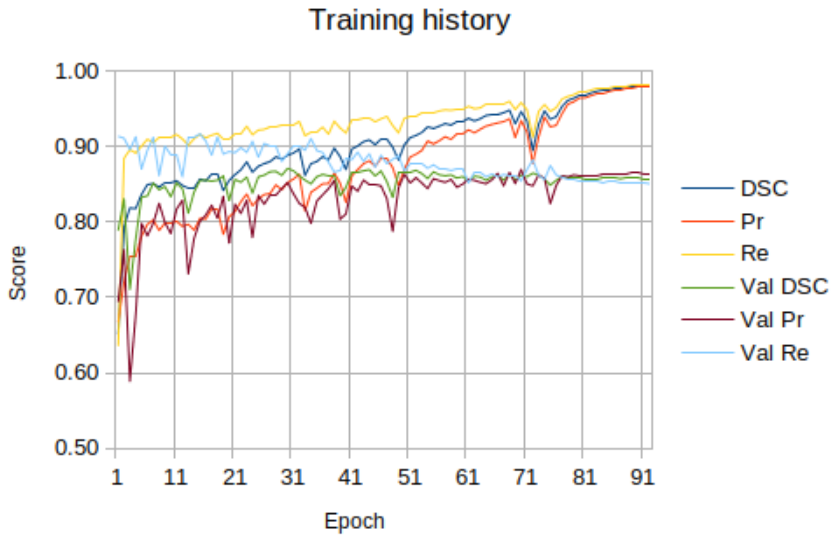


Figure 4.13: Evolution of the evaluation scores in the training and the validation sets, during the training of m_1 using manual annotations (T_S).

the epoch with the minimum validation loss, the train precision and recall are 87.7% and 91.3%, respectively; for the validation split, precision and recall are 84.8% and 88.2%, respectively. The DSC is 89.5% in the training split and 86.5% in the validation split.

The next step in our method consists of increasing the generalization power of the model by using a more extensive training set. We do this by training a new instance of U-VGG19 from scratch. The fitted model, trained using the set A of selected images with their corresponding good target segmentations predicted by m_1, is referred to as m_2; remember that the selection of images is performed by a human (semi-supervised approach). Some examples of images in A along with their predicted segmentation maps are depicted in Figure 4.14. The cardinality of A is almost the double of the cardinality of T (63 vs 32). Since we are training from scratch again, the images from T are never seen during training by m_2; we use the images from T as a test set.

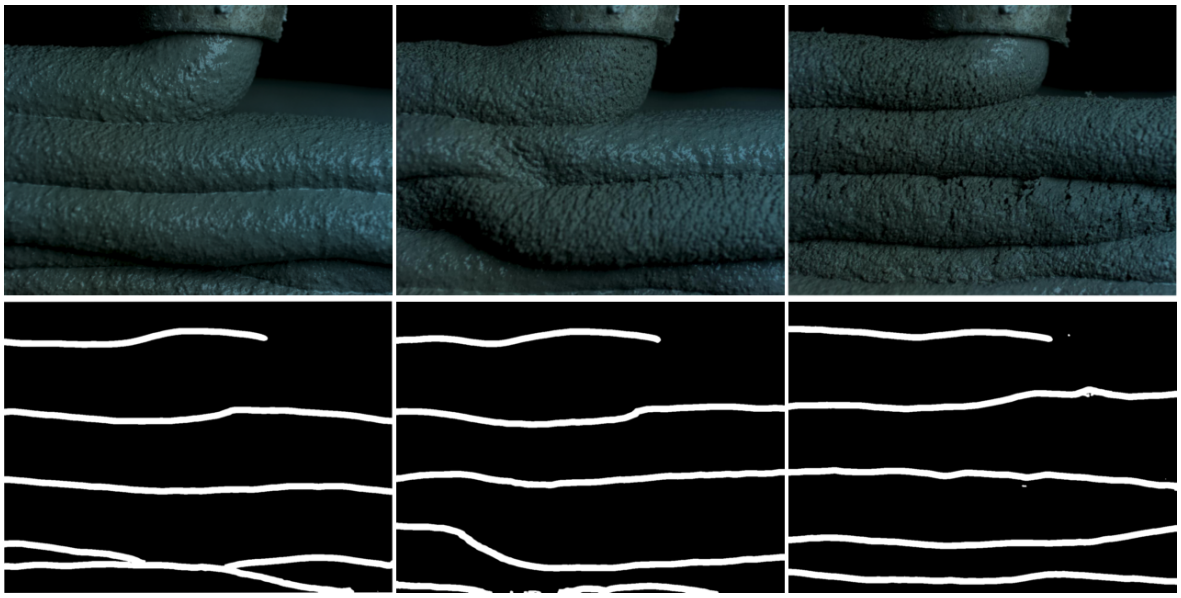


Figure 4.14: Examples of good predictions from m_1 trained on a small set of manually annotated images. The analyzed images and their corresponding predicted segmentations are part of A and A_S, respectively.

Because one of the purposes of this step is to increase the generalization power of the model, we introduced data augmentation. Our data augmentation consisted of randomly transforming an image (and its corresponding annotation) immediately before feeding it to the neural network for training. To do this, a random value for each of the 6 following operations is chosen: adding noise, changing illumination, flipping, zooming, rotating and shearing. Every image undergoes the 6 operations in the given order. Therefore, there is a low likelihood that the model sees the very same image twice during training.

Similarly to the training of m_1, we split the images from A into 80% training and 20% validation to train m_2. Figure 4.15 illustrates the scores in the training and validation splits during training. Again, the training has an early stop after ~90 epochs; however, this time the scores from both the training and the validation splits are very close to each other along the epochs. At the epoch with the

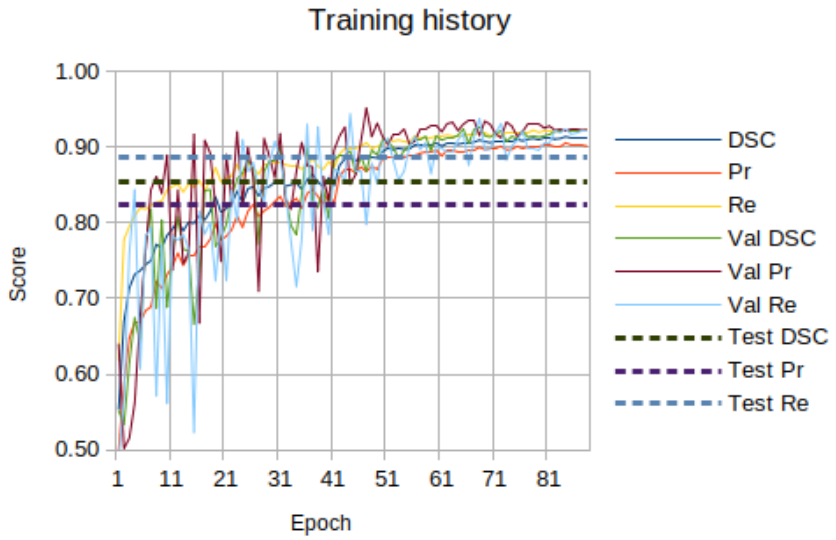


Figure 4.15: Evolution of the evaluation scores in the training and the validation sets, during the training of m_2 using target segmentations predicted by m_1 (A_S). The results on the test set (T), using the weights with minimum validation loss, are shown as constants.

minimum validation loss, the training precision and recall are 90.0% and 91.9%, respectively; for the validation split, precision and recall are 91.5.8% and 93.8%, respectively. The DSC is 90.9% in the training split and 92.6% in the validation split. The results on the test set (T) are illustrated as constants with dashed lines in Figure 4.15. The precision, recall and DSC are 82.3%, 88.6% and 85.3%, respectively.

This new model provides a better generalization power as demonstrated by the scores on unseen images, which are more abundant and diverse with respect to the subset of T used to validate m_1 at the first step of our method.

The next step is to identify the cases that are difficult to segment for m_2 (i.e. images with bad predicted segmentation). We want the model to learn from these examples. To do this, we fine-tune m_2 using the images from $A \cup C$ for training, where C is the set of chosen images with bad segmentation produced by the model trained solely on A. To make this learning effective, we manually correct the labels produced by m_2 (partial human annotations); this is illustrated in Figure 4.16.

The cardinality of $A \cup C$ is 128; as before, we split these images into 80% training and 20% validation. We fine-tune m_2 beginning from the weights learned in the previous step of our method (i.e. the weights with minimum validation loss training with A). For this fine-tuning, we reduce the initial learning rate one order (i.e. from 10^{-4} to 10^{-5}). A detailed comparison between a manual segmentation and the segmentation produced by this fine-tuned U-VGG19 is shown in Figure 4.17.

As it can be appreciated in the figure, U-VGG19 locates all the interlayer lines. However, there are a few errors at pixel level caused by slight offsets of the prediction with respect to the manual annotation. These offsets can be caused by

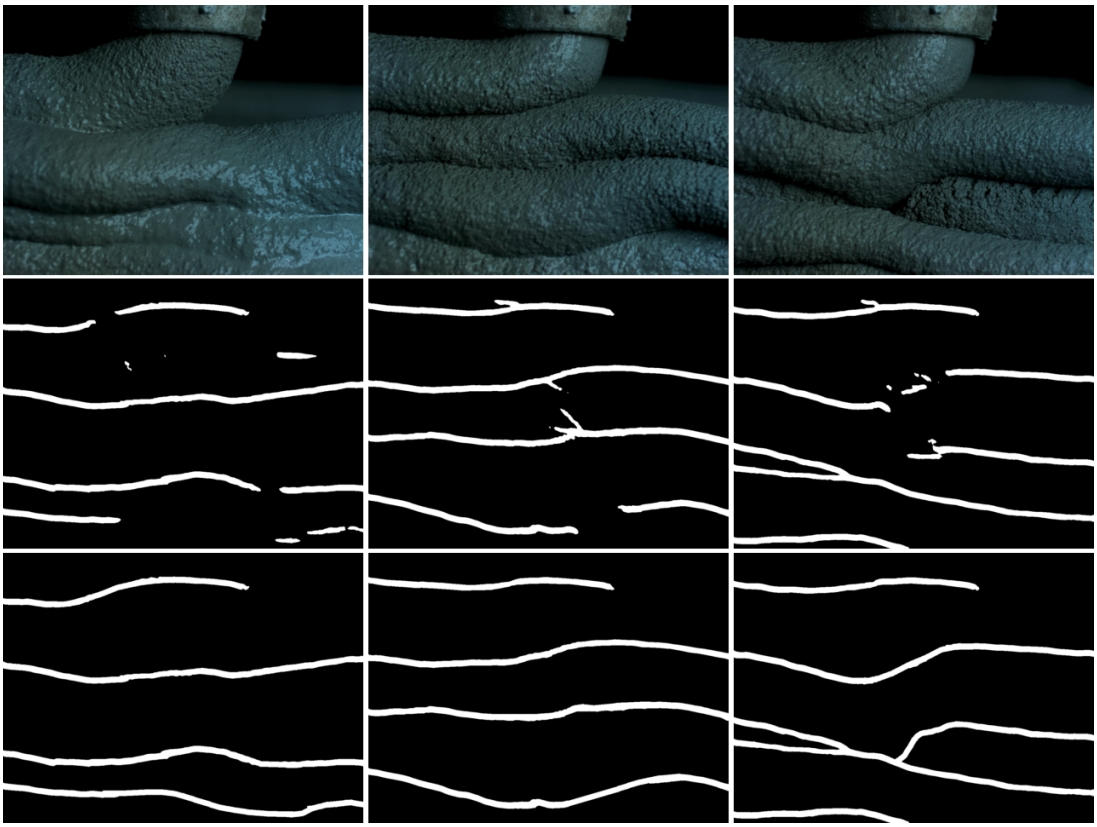


Figure 4.16: Examples of bad predictions (second row) from m_2 trained with segmentation maps predicted by m_1 (A_S), and their corresponding manual correction (third row).

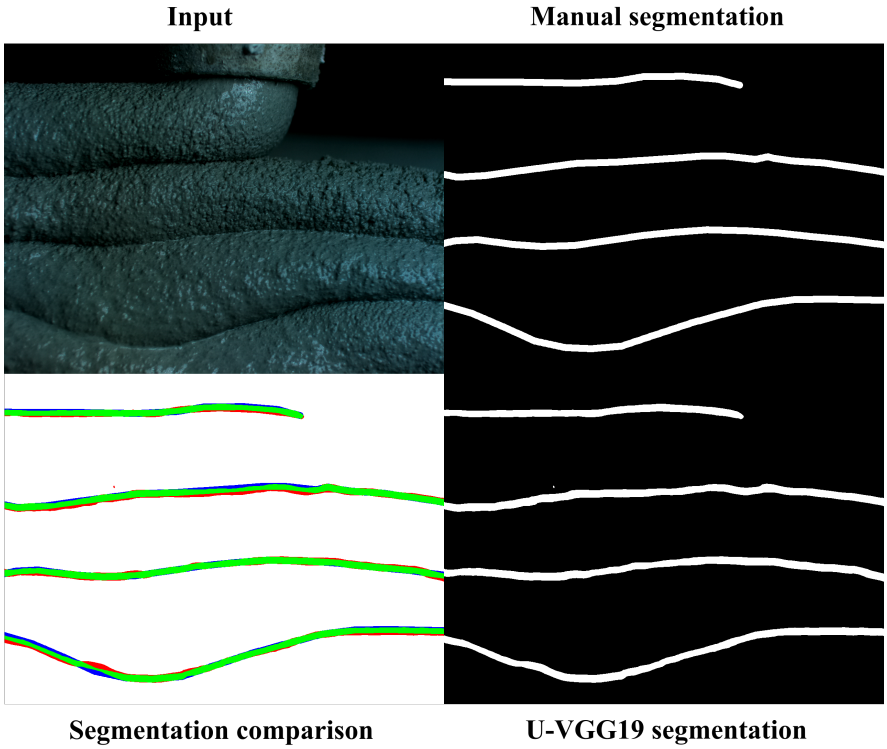


Figure 4.17: Comparison of a manual segmentation and the prediction of the final fine-tuned U-VGG19 (m_2). The color code in the bottom left image is: (Green) True positives; (Blue) False negatives; (Red) False positives.

the inaccuracy of the annotators to locate the center the center of the interlayer line. Because of these inaccuracies, for the final evaluation, we propose a margin of tolerance as previously done in works such as (Amhaz et al. [2016]; Shi et al. [2016]). In our work, the tolerance is chosen to be 2 pixels. This means that a pixel classified as part of an interlayer line is considered as true positive if it is no more than 2 pixels away from a pixel manually annotated as part of an interlayer line. This represents 10% of the diameter used for annotation and approximately 0.05mm in real life i.e. $< 1\%$ of the expected layer width (6mm).

To study the precision of the obtained scores, we repeat the training and fine-tuning of m_2 10 times. Each time, we reuse the same annotations used in the previous experiments presented in this section; we report the average $\pm$ standard deviation. We compare the results of the fine-tuned U-VGG19 with a method based on the one presented by (Davtalab et al. [2020]), since it is the most closely related work in the literature. For this alternative method, first we smooth the image texture using a bilateral filtering with diameter 30 and color sigma 100. Then, we use the Canny edge detector with threshold 15. Unlike Davtalab et al., we extend the processing to connect line segments and to filter noise. To connect segments, we perform a morphological closing with a disk of diameter 11; to filter noise, we perform an opening with a disk of diameter 9. Finally, to match the width of the manual annotations in T, we thin the resulting lines and perform a dilation with a disk of diameter 20. The results (using 2-pixels tolerance) of this method and U-VGG19 are presented in Table 4.1.

Table 4.1: Results of interlayer line segmentation in I3DCP.

Method	F-score	Pr	Re
Texture smoothing and edge detection (Davtalab et al. [2020])	38.6%	39.1%	38.7%
U-VGG19 (ours)	91.0±0.2%	92.0±0.6%	90.0±0.7%

As expected, the scores of U-VGG19 are better by a huge margin. Although both methods are intended for interlayer line segmentation, the nature of the acquired images is very different. In first place, a method based on edge detection with algorithms such as Canny is unable to deal with interlayer lines defined by a change in texture, which are not visible edges; as an example, see the third row from Figure 4.18, specifically the third interlayer line from top to bottom. In second place, using fixed hyperparameters for smoothing and edge detection is not an effective strategy in a setup like ours, in which many different textures can be observed along the time (as in the three images from Figure 4.18), and even within a single image (as in the first row of the figure).

These constraints, as demonstrated by the obtained results, justify the use of Deep Learning for interlayer segmentation during inline monitoring of 3D concrete printing. In the next section, we provide a final discussion on this topic and future perspectives

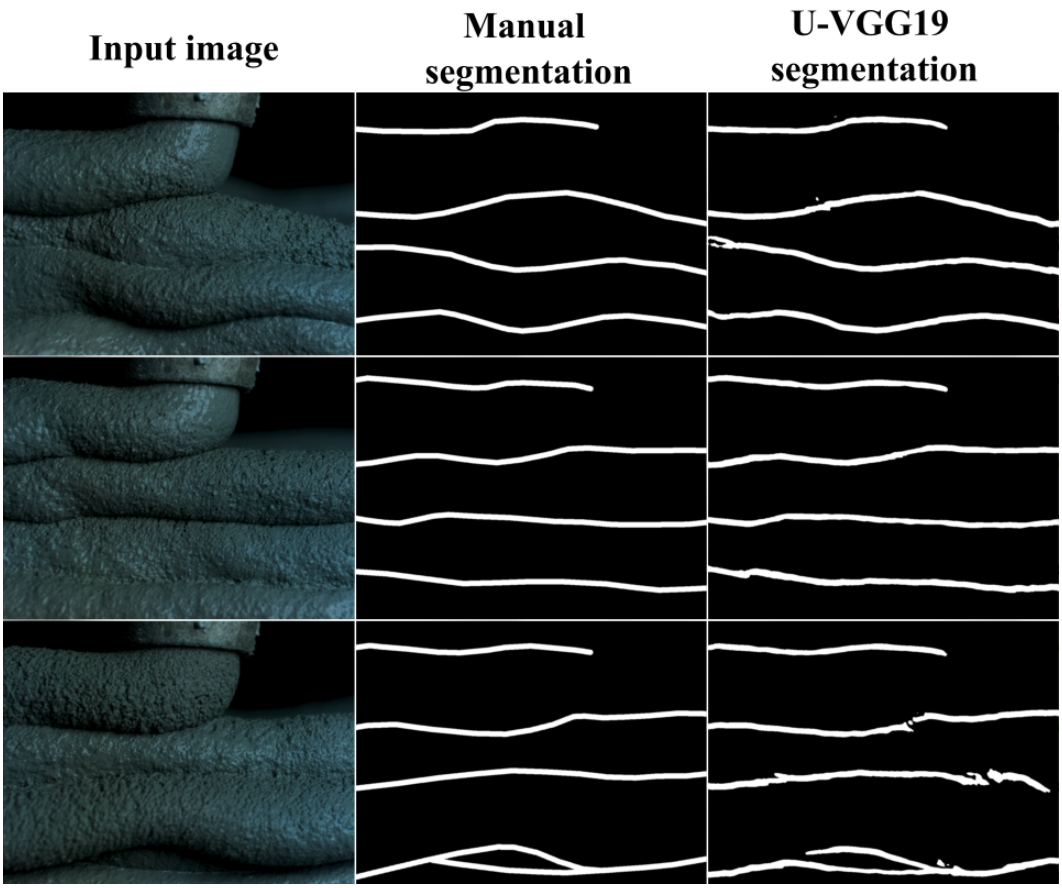


Figure 4.18: Manual and U-VGG19 segmentation of difficult images.

4.6 Discussion on Interlayer Line Segmentation and Future Perspectives

In this chapter, we approached the problem of interlayer line segmentation for inline monitoring of 3D concrete printing. First, we introduced the I3DCP dataset, which was acquired during a printing session with a camera fixed to the extrusion nozzle.

In this particular context, with fresh material, the interlayer lines can take different aspects: black, bright or even not visible at all. To segment the images from I3DCP, we proposed to use U-VGG19 as baseline model. Since this network is intended for supervised learning, we proposed a method (summarized in Figure 4.11) for semi-supervised learning inspired by active learning, starting from a reduced set of manually segmented images. The resulting U-VGG19, trained with the proposed method, achieves a Dice score coefficient of 91%. This score shows the suitability of the proposed method and U-VGG19 itself for inline interlayer line segmentation.

In the first step of the proposed method, a manual selection of images with good predictions is performed; the predictions are obtained from a model trained on the small initial set of manually annotated images. Using the principles of self-training and active learning, this selection can be iterated to progressively increase the number of annotated images. The time investment of selecting al-

ready annotated images is considerably lower than the time cost of manually segmenting the images.

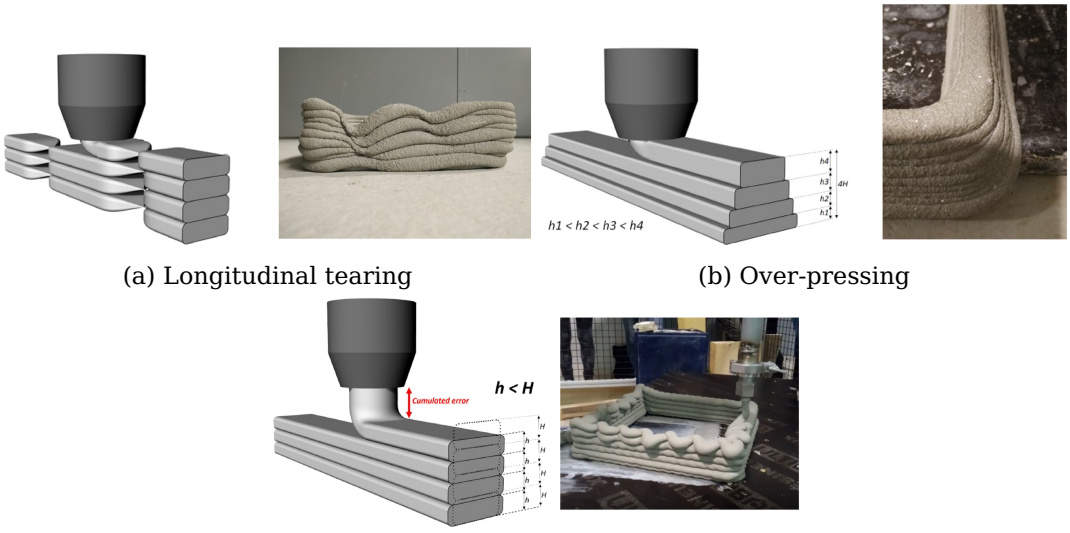
In the same direction, although the correction of annotations takes longer than just selecting images, it should generally take less time than annotating images from scratch. With this in mind, the initial set of annotations used for training can be obtained with the help of less robust but simpler methods (e.g. the one proposed by (Davtalab et al. [2020]). This is the same concept behind the third step of our method, in which a human supervisor is asked to identify and correct wrong segmentation maps produced by the trained model; this step can also benefit from techniques for learning in presence of inaccurate annotations. By fine-tuning the final model with these corrected annotations, the model explicitly learns to segment interlayer lines in difficult images, while preserving the knowledge acquired from previous training iterations.

With robust segmentation models, the detected interlayer lines can provide useful information about the printing process. In the next chapter, we discuss how the segmentations of the interlayer lines can be used to perform a geometrical characterization of the printed layers in order to detect anomalies.

Chapter 5

Geometrical Characterization and Anomaly Detection in 3D Concrete Printing

3D printed structures are subject to various defects, which have different physical origins. Some defects are due to an insufficient stiffness or strength of the fresh mortar and may result in global failure, or to an unacceptable accumulated error, as observed in (Archez et al. [2021]). Some defects may not jeopardize the printing procedure, but they would still lead to the rejection of the 3D printed piece afterwards. For illustration purposes, in Figure 5.1 we show some typical geometrical deformations caused by process defects (Carneau et al. [2022]).



(a) Longitudinal tearing

(b) Over-pressing

(c) Flow-out

Figure 5.1: Local defects in 3D printed samples, images from (Carneau et al. [2022])

As illustrated in the figure, the presence of local defects can be identified by a visual inspection. In this work, we aim to detect geometrical defects by inspection based on computer vision. The basis of our method relies on the analysis of the interlayer lines. Since these lines serve as boundaries of the observed layers, they provide relevant information about the geometry of the layers and their possible deformations.

For example, the layer thickness can be measured as the local distance between the two interlayer lines surrounding each layer. In Figure 5.2a-c, we observe a few top printed layers for local analysis, and a distribution of the layer thickness measured at pixel level from the extracted interlayer lines. Considering this distribution as the result of a defect-less printing, a defect can be defined as a deviation from the expected distribution (as illustrated in Figure 5.2d-i).

In this chapter, we propose a method for geometrical characterization and anomaly detection of 3DCP, based on image processing. The proposed characterization consists of local measurements of the geometry of the printed layers, measured with respect to their interlayer lines. The posterior anomaly detection is based on the principle illustrated in Figure 5.2, using user-defined ranges to define and locate the anomalies in the analyzed images.

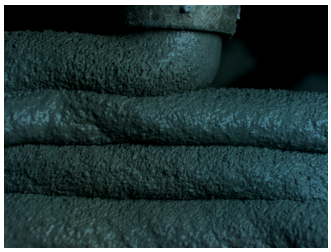
The rest of this chapter is organized as follows. First, we provide a brief summary of the state of the art on geometrical monitoring of 3DCP. Then, we describe the methods used to measure the properties proposed in this work: the interlayer lines' orientation and curvature, the layers' thickness, and the relative height of the printing nozzle with respect to the last printed layer. Subsequently, we discuss how these measures can be summarized to characterize the observed layers and to detect anomalies. Afterwards, we illustrate our technique applied on diverse study cases. We close the chapter with a final discussion on the topic of geometrical monitoring of 3DCP and some future perspectives.

5.1 Geometrical Monitoring of 3D Concrete Printing in the Literature

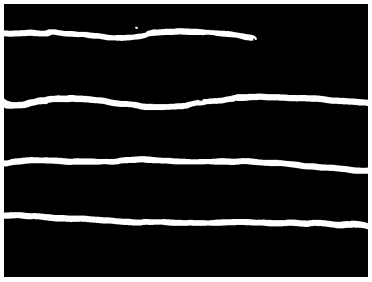
In chapter 1, we discussed how automatic geometrical monitoring of 3DCP began to gain attention in recent years. Research in this direction has shown that approaches based on optical sensors are promising strategies as non-invasive methods for inline monitoring. However, research on these methods is still limited in the literature.

Early works have used sensors such as 1D Time-of-flight (Wolfs et al. [2018]) or laser triangulation (Lindemann et al. [2019]) sensors attached to the extrusion nozzle to measure its distance to the surface of deposition. To measure the layer width from a top view, (Kazemian et al. [2019]) attached an RGB camera to the nozzle instead. In that work, the acquired images are blurred in grayscale with a Gaussian filter; the extruded material is then segmented from a white background using Otsu's threshold selection. The reported width is the average of the segmented blob at different vertical coordinates. Another similar work is presented in (Shojaei Barjuei et al. [2022]), where the borders of the printed layer are detected as edges. The reported width is the distance between the center points of the detected edges.

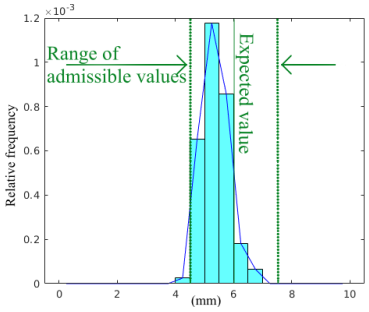
We can also identify approaches oriented towards post-printing geometrical assessment. In the case of (Davtalab et al. [2020]), a fully-convolutional network is used to segment the analyzed piece from background. Since the input images correspond to a lateral view, the interlayer lines are visible. Posterior to segmenting the piece, the region of interest is smoothed using bilateral filtering



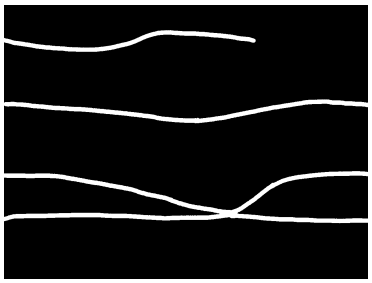
(a) Lateral view of several top printed layers



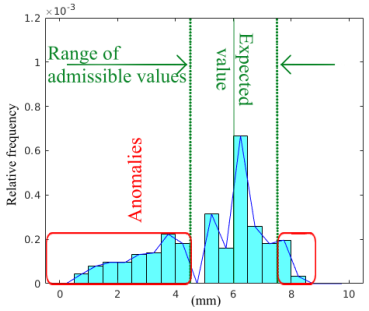
(b) Interlayer lines without defects (extracted from a)



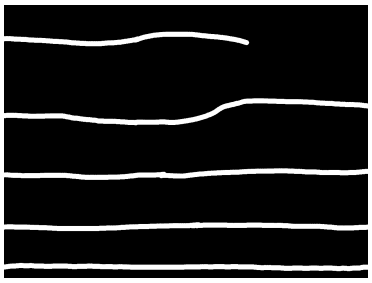
(c) Distribution of layer thickness from b



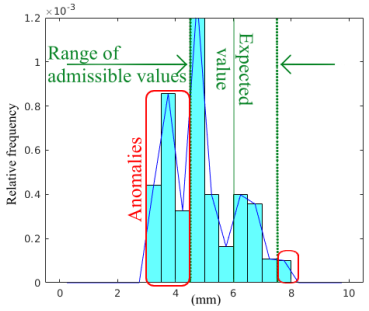
(d) Interlayer lines suggesting longitudinal tearing



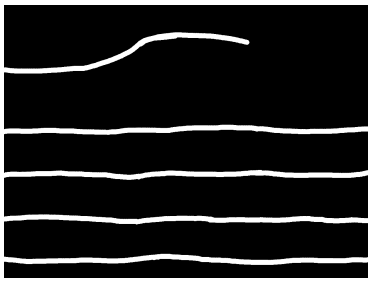
(e) Distribution of layer thickness from d



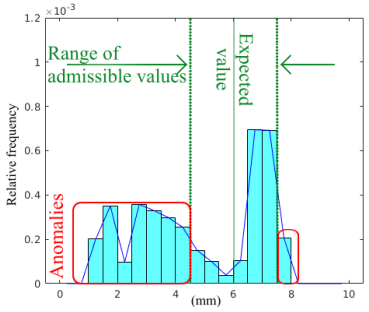
(f) Interlayer lines suggesting over-pressing



(g) Distribution of layer thickness from f



(h) Interlayer lines suggesting flow-out



(i) Distribution of layer thickness from h

Figure 5.2: Distributions of the layer thickness, measured pixel-wise. b-c) In a defect-free process, the distribution of any measured parameter is contained in an expected interval. d-i) When printing defects occur, a deviation from this interval is observed.

Table 5.1: Summary of methods based on optics for geometrical monitoring of 3DCP.

Work	Sensor	Measurements	Approximate cost[a]
Wolfs et al. [2018]	1D Time-of-Flight	Layer to nozzle height (top view)	5 USD
Lindemann et al. [2019]	Laser triangulation	Layer to nozzle height (top view)	6500 USD
Kazemian et al. [2019]	RGB camera	Layer width (top view)	40 USD
Shojaei Barjuei et al. [2022]	Monochromatic camera	Layer width (top view)	349 USD
Davtalab et al. [2020]	RGB camera	Layer orientation (lateral view)	40 USD
Nair et al. [2021]	3D scanner	Printed piece cloud to 3D model cloud distance (lateral view)	472 USD
Ours	RGB camera	Layer to nozzle height; layer thickness; interlayer line orientation and curvature (lateral view)	57 USD

[a] Estimated with base on the sensor models and/or specifications reported in the respective cited works.

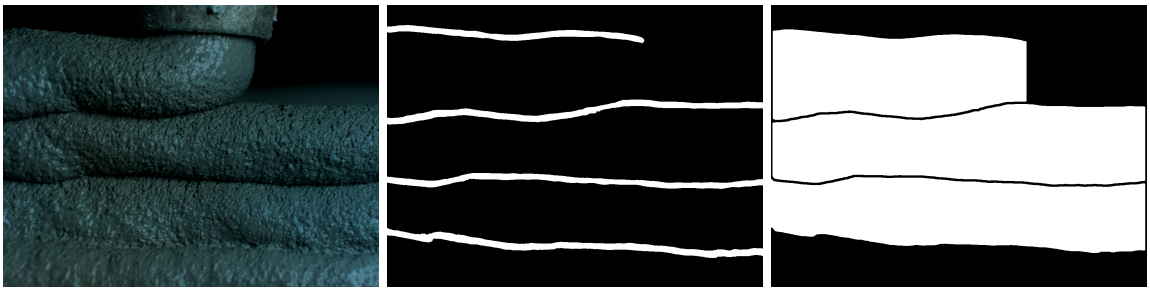
and the interlayer lines are detected as borders using the Canny algorithm. The authors look for deformations of the layers by analyzing the orientation of these lines using the Hough transform window-wise. To measure deformations in 3D, the work of (Nair et al. [2021]) produces a point cloud by analyzing the printed piece with a 3D scanner on a rotatory base. With the help of mathematical morphology, the authors compare this cloud with the 3D model used for printing.

From this literature review, we can see that monitoring based on computer vision is still scarce. While RGB cameras are typically cheaper than sensors such as 3D and laser triangulation scanners, the range of possible measurements that can be performed with a single camera is wide. In Table 5.1, we compare our proposed method with the methods based on optics discussed in this section. Our geometrical measurements, based on the analysis of the interlayer lines, are discussed in the next section.

5.2 Geometrical Measurements Based on Interlayer Lines

For the experiments and results presented in this chapter, we used the I3DCP dataset (described in subsection 4.1.1). We collected this dataset inline by fixing

a camera to the printing robot during a printing session. As stated in Table 5.1, the images are acquired from a lateral point of view. In this work, we analyze the interlayer lines as segmented by U-VGG19 (see section 4.2). Additionally, let us denote by ROI $\subset \mathbb{Z}^2$ the set of layers delimited vertically between each two lines (see Figure 5.3). The geometrical measurements proposed to characterize the observed layers are obtained using the following methods.



(a) Image to analyze (I) (b) Interlayer lines (S) (c) Regions of interest (ROI)

Figure 5.3: a) One image from I3DCP (our collected dataset). b) Interlayer lines segmented by U-VGG19. c) Regions of interest; regions within a layer that are not contained between two interlayer lines are ignored because the layer is not seen entirely in the vertical axis.

5.2.1 Geometry of Interlayer Lines

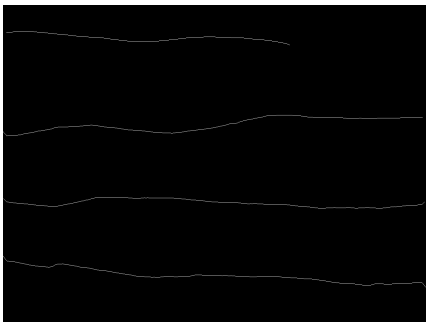
We measure two properties of the interlayer lines: orientation and curvature, at pixel level. Given an input image I, let the binary image $S \in \mathbb{Z}^2$ be the segmentation map of the interlayer lines in I. Before measuring, we pre-process S by performing a thinning (Guo and Hall [1989]); this approximates the interstitial lines to 1-pixel-wide lines. This processed image, denoted by S_T, is unaware of the width provided by the segmentation method to obtain S.

Orientation

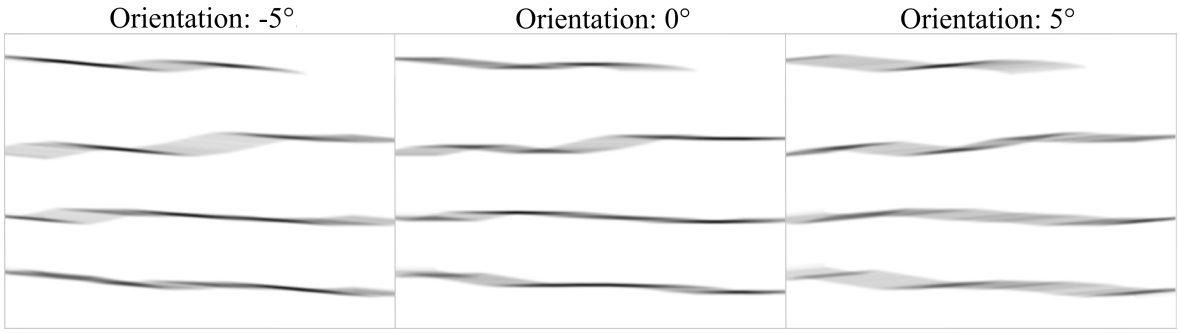
Similarly to (Davtalab et al. [2020]), we propose to measure the orientation of the interlayer lines. Let $R(d,l)$ be a line oriented by d with length l, and D the set of possible orientations $d \in [-90, 90)$. The line length l is chosen to be 1/5 of the image width. This hyperparameter presents a trade off between the available line angles and the locality of the measurement. Since the selection of this value is not critical for our method, it was selected to be similar to the expected layer thickness (256 pixels). Let $p \in \mathbb{Z}^2$ be a point in an image. Then, $F(p,d) = [S_T * R(d,l)](p)$ is the convolution of S_T with a line oriented in d (Figure 5.4c). Finally, our local orientation map ϕ (Figure 5.4d) is defined as:

$$\phi(p) = \begin{cases} \arg\max_{d \in D} F(p,d), & p \in S_T \\ \text{undefined}, & \text{otherwise} \end{cases} \tag{5.1}$$

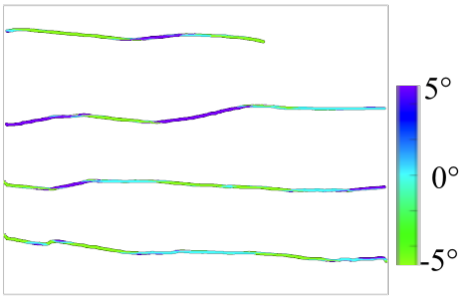
(a) Input image I



(b) Thinned segmentation map S_T



(c) Filter responses $F(p,d)$



(d) Local orientations $\phi(p)$

Figure 5.4: Measurement of the local orientation of the interlayer lines. a-b) Input image and its corresponding thinned segmentation map. c) Convolution of S_T with differently oriented lines (darker values represent higher activations). d) Local orientations of the interlayer lines for $d \in \{-5, 0, 5\}$; undefined values are shown in white.

Curvature

Let $\hat{L}_n$ be a piece-wise approximation of the n-th interlayer line in S_T, using m cubic splines (De Boor and De Boor [1978]):

$$\hat{L}_n(t) = \begin{cases} s_n^1(t - t_0), & t_0 \le t < t_1 \\ \vdots & \\ s_n^m(t - t_{m-1}), & t_{m-1} \le t < t_m \end{cases} \tag{5.2}$$

The knots are selected to split the interlayer line of interest into m fixed-length segments (Figure 5.5); the length is chosen to be 1/10 of the image width. While this hyperparameter implies a trade-off between the accuracy and the smoothness of the spline approximation, it is not critical for the method. The

Figure 5.5: Spline approximation of an interstitial line. The black line shows the original neural network prediction. The red line shows the spline approximation. The knots used to estimate the splines are shown as blue points.

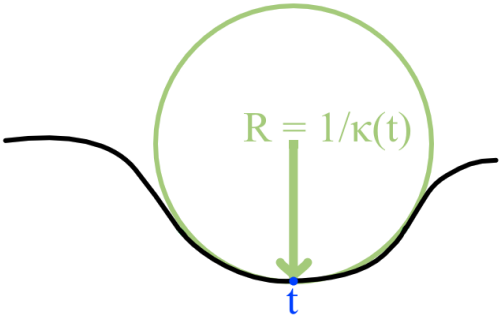


Figure 5.6: Example of osculating circle (green) at t (blue). The curvature $\kappa(t)$ of the line (black) is the inverse of the radius.

independent variable t corresponds to a parametric representation of the i-th spline $s_n^i(t) = (x(t), y(t))$.

Since this smooth approximation is twice differentiable, the local curvature κ of $\hat{L}_n$ at point t can be computed as:

$$\kappa(t) = \frac{x'y'' - y'x''}{(x'^2 + y'^2)^{\frac{3}{2}}} \tag{5.3}$$

The primes refer to the derivatives with respect to t. The magnitude of $\kappa(t)$ is the inverse of the radius of the osculating circle touching the point (x, y) defined by $\hat{L}_n(t)$ (see Figure 5.6). The sign indicates the concavity.

The curvatures of the interstitial lines, as well as their orientations, produce variable distances between adjacent interstitial lines. In the next section, we measure the local layer thickness in terms of this distance.

5.2.2 Local Thickness of Layers

The center of a printed layer can be defined as the line composed of points equidistant to both of its interstitial lines. The local thickness of a layer can be measured precisely as the diameter of a circle centered on the centerline and tangent to the closest point in both interstitial lines (see Figure 5.7).

Based on this principle, we use the fast Euclidean distance transform (EDT) (Maurer et al. [2003]) to measure the local thickness. First, recall that S_T is the image containing the 1-pixel-wide representation of the interlayer lines. The function $E : \mathbb{Z}^2 \to \mathbb{R}^+$ is the result of applying the EDT to S_T (see Figure 5.8a). The local maxima of E indicate the centerlines of the printed layers (see Figure 5.8b-c). Since we expect the layers to be nearly horizontal in the input picture, we estimate the locations of these local maxima along the vertical direction. Let $k_1 = (0, -1, 1)^T$ and $k_2 = (1, -1, 0)^T$ be vertical kernels. The convolution of the distance map E with these kernels allows detecting the maxima of E:

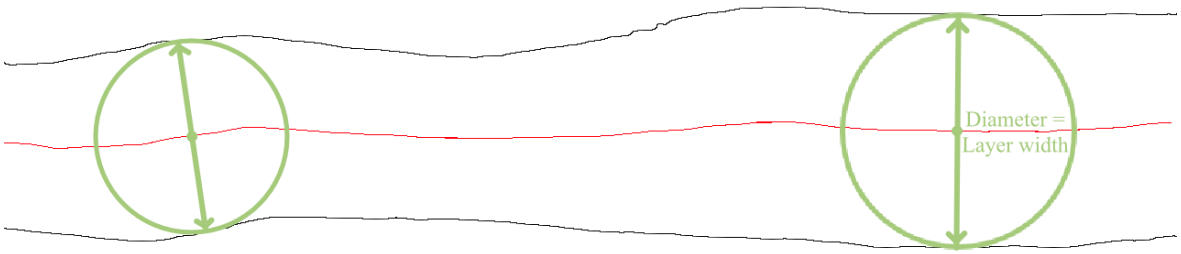
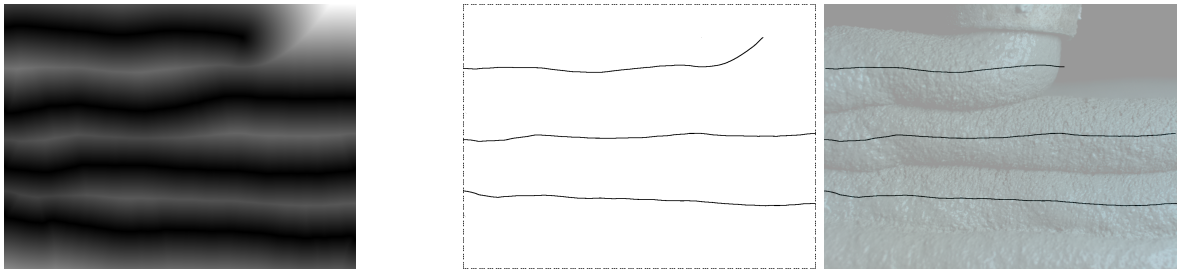


Figure 5.7: Principle to measure the local thickness of a layer. The red line is the layer's center. The local thickness is defined as the diameter of the circle centered on the line and tangent to the closest point in both interstitial lines.



(a) Distance transform (E) (b) Maxima (P_M) (c) Layer centers

Figure 5.8: Method proposed to measure the local thickness of the layers. a) Distance transform of S_T. b) Maxima from the distance transform in the vertical direction. c) Layer centerlines.

$$P_M = \{p \mid [\text{sign}(E * k_1) * k_2](p) = 2\} \tag{5.4}$$

Remember that ROI is the set of layers delimited vertically between each two lines (see Figure 5.3c). For the points located in the centerlines and inside ROI, the layer thickness W is

$$W(p) = \begin{cases} 2E(p), & p \in (P_M \cap \text{ROI}) \\ \text{undefined}, & \text{otherwise} \end{cases} \tag{5.5}$$

The factor 2 allows obtaining the thickness (the double of the shortest distance from the centerline to either line in S_T).

Additionally to the geometry of each layer, the analysis based on interlayer lines can provide information about the whole piece. For example, if the last printed layer is considerably below the printing nozzle, it may be a sign of some anomaly in the printing process (e.g. flow-out).

5.2.3 Relative Nozzle Height

Similarly to (Lindemann et al. [2019]; Wolfs et al. [2018]), we measure the relative height of the printing nozzle with respect to the surface of deposition. From the lateral point of view used to get I3DCP, this height corresponds to the vertical distance from the nozzle to the line receiving the currently printed layer.

In the setup of I3DCP, the camera is fixed to the robot; therefore, the center of the nozzle is at a constant position $c = (x_c, y_c)$ in all the images. Let L_2, the second

interlayer line from top to bottom in the image to analyze, be the set of points $p = (x_p, y_p)$ separating the currently printed layer from the layer of deposition (see Figure 5.9). Then, the relative nozzle height is:

$$\mathrm{H}(p) = \begin{cases} y_p - y_c, & p \in \mathrm{L}_2 \\ \text{undefined}, & \text{otherwise} \end{cases} \tag{5.6}$$

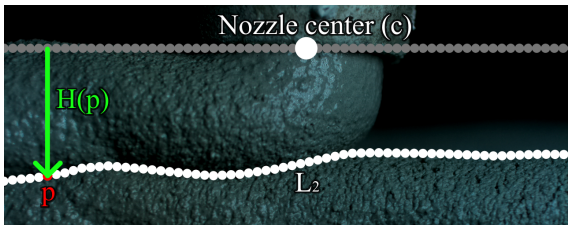


Figure 5.9: Relative nozzle height measurement.

With the vertical axis pointing downwards in an image, this height is positive as long as p is below the nozzle.

Similarly to the previously proposed measurements, a single value is measured per each point p. The distributions generated for each measure can be used as a summary of the analyzed image and to determine the presence or absence of defects.

5.3 Geometrical Anomaly Detection

Since 3DCP is intended to replicate a 3D digital model, the expected properties of the printed piece at each time stamp can be estimated beforehand. These properties are considered to be the normality of the process. Nonetheless, in real applications, there are manufacturing tolerances i.e. small deviations from the expected values are acceptable.

Regarding the geometrical properties measured in this work, under good printing conditions, they should exhibit distributions centered on the expected value and mainly concentrated in the range defined by these tolerances. The presence of anomalous values with high densities outside of these ranges is indicative of a defective process.

In this work, we propose a simple, general-purpose method to detect anomalies in the printed piece. A precise definition of the process anomalies is dependent on the specific requirements of the industrial process itself, and therefore out of the scope of this thesis.

In the proposed method, first we summarize the information retrieved from equations 5.1, 5.3, 5.5 and 5.6 (orientation, curvature, thickness and nozzle height, respectively), producing a histogram per each type of measure. The normalized version of each of these histograms allows us to obtain a probability density distribution to characterize the analyzed image.

The anomalous values in each of these distributions are defined as the values outside of a user-defined range of acceptable values (see Figure 5.2 as example). The location of the anomalies in the image then corresponds to the pixels containing measured values outside the ranges of admissible values.

5.4 Experiments and Results on Local Geometrical Characterization and Anomaly Detection

In this section, we show two study cases to be characterized using all the proposed measurements: interlayer line orientation and curvature, layer thickness, and relative nozzle height. Besides the measurement of orientation, expressed in degrees, all the other geometrical measurements are based on pixels as metric unit. We convert these measurements to millimeters using a ratio of 40 pixels per millimeter (see section 4.1 for further details).

The locally measured values are plotted using color maps; additionally, we show the distributions resulting from these measurements (see Figure 5.10 as example). The ranges of admissible values, delimited with green dotted lines in the distributions, are user-defined; deviations outside those ranges are considered as anomalies, and they are plotted in blue if below the minimum threshold and in red if above the maximum one.

For orientation, we expect near-to-horizontal layers; the user-defined range is (-10, 10) degrees. For curvature, we expect near-to-straight layers; the range is (-0.05, 0.05) mm^{-1}. The expected layer thickness is 6mm; the range is (4.5, 7.5) mm. The nozzle should be maintained above the surface of deposition, at a height similar to the expected layer thickness; the range is (5, 7) mm.

The first study case portrays an overall acceptable printing. As observed in Figure 5.10, there are almost no geometrical anomalies, except for a few high curvature segments. Consequently, the distributions obtained from these plots are contained inside the ranges of admissible values.

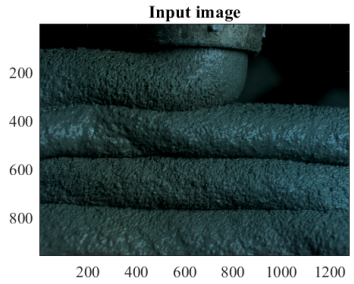
The second study case depicts a scenario with severe anomalies. As observed in Figure 5.11, the distributions of the geometrical measurements are very different from the ones in Figure 5.10: in all the distributions, there are high density regions outside the range of admissible values. The most extreme case is present in Figure 5.11e, where the distribution is concentrated outside the admissible range. This behavior is likely to cause a coiling effect on the material deposition; in fact, as depicted in Figure 5.11c, we see high curvatures that can be directly related to this phenomenon.

As shown by this study case, as well as the first one with overall acceptable printing, the proposed methodology is able to provide an inline characterization of the process based on visual inspection of the last printed layers. With this characterization, the proposed methodology detects and locates anomalies in the last extruded layers.

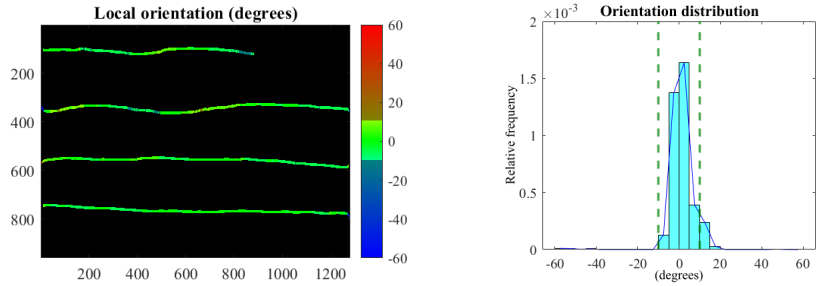
Next, we show and discuss preliminary results on additional applications of the proposed methodology.

5.4.1 Post-Extrusion Characterization and Anomaly Detection

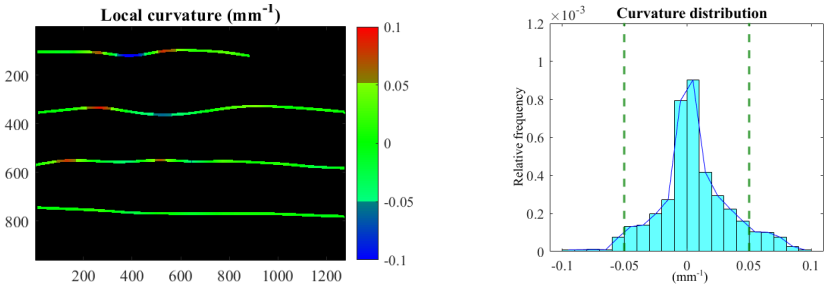
The results presented so far show how our technique can be used for inline monitoring with a small field of view. However, its usage can be easily extended to



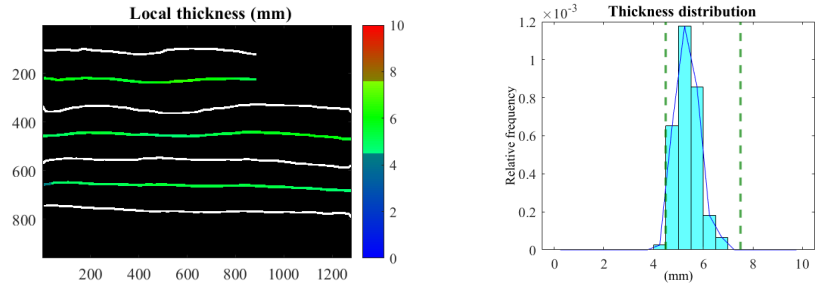
(a) Image to analyze as captured by the camera



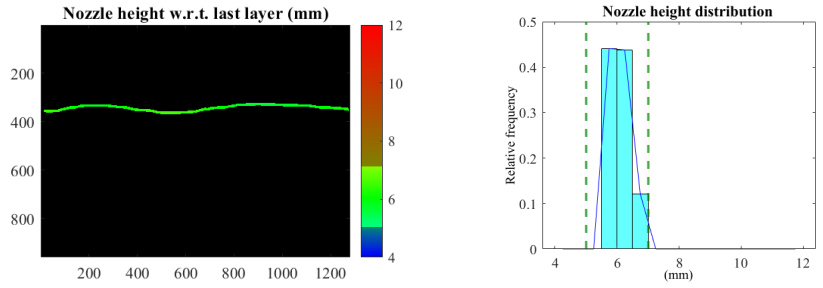
(b) Local orientations of the interlayer lines



(c) Local curvatures of the interlayer lines

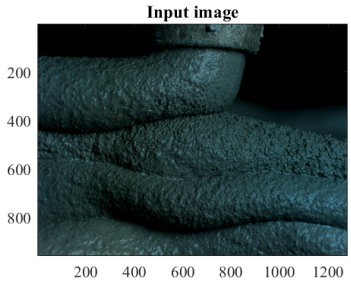


(d) Local thickness of the layers. The thickness is associated to the layer centers (in color); the white lines represent the interlayer lines

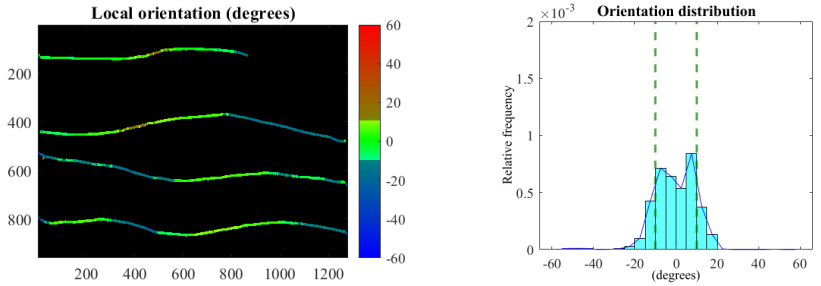


(e) Local heights of the nozzle with respect to the last printed layer

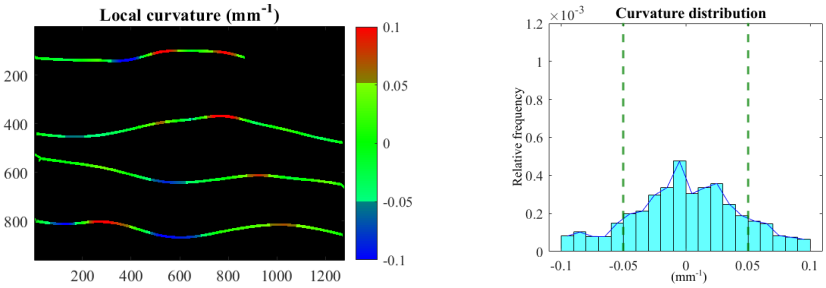
Figure 5.10: Plots and distributions from the first study case. The black pixels in the plots correspond to undefined values. The doted lines in the histograms represent the range of admissible values; values outside these ranges are detected as anomalies.



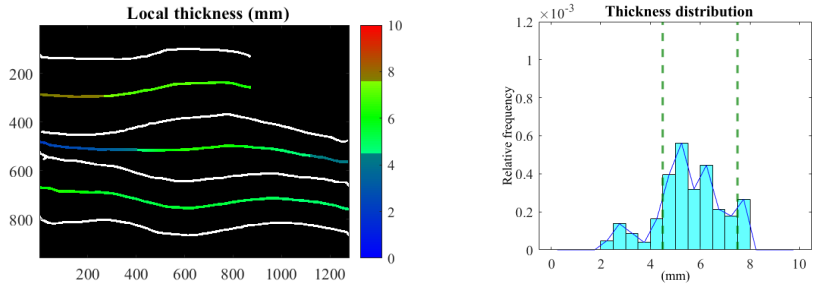
(a) Image to analyze as captured by the camera



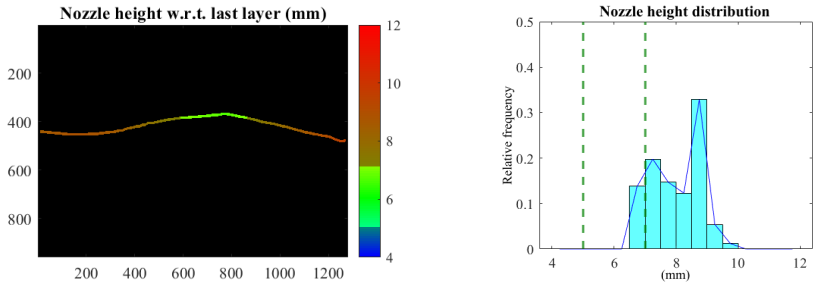
(b) Local orientations of the interlayer lines



(c) Local curvatures of the interlayer lines



(d) Local thickness of the layers. The thickness is associated to the layer centers (in color); the white lines represent the interlayer lines



(e) Local heights of the nozzle with respect to the last printed layer

Figure 5.11: Plots and distributions from the second study case. The black pixels in the plots correspond to undefined values. The doted lines in the histograms represent the range of admissible values; values outside these ranges are detected as anomalies.

larger fields of view and for post-printing stages –when the printed part is drying or has fully hardened. In this section, we explore preliminary results on the aforementioned cases. Since the analyzed images are out of the distribution of acquired images for the I3DCP dataset, we generate the interlayer line segmentation manually instead of using U-VGG19.

First, we explore 2 cases of rejected pieces after hardening, with a close field of view. In this occasion, we focus on measuring the orientation and curvature of the interlayer lines as well as the thickness of the observed layers. In the first studied case (Figure 5.12), we see a high variability of the thickness of the layers, including a case of total occlusion (denoted by a progressively reduction of thickness until 0). These changes of thickness are associated with many regions of abnormal orientation and curvature, so we see that the distributions of the three proposed measurements are highly distributed outside the range of admissible values.

The second studied case (Figure 5.13) exhibits also many defective regions. In this case, nonetheless, the printed piece has an apparently better quality. This is confirmed by the distributions of orientation and thickness: although they contain many values outside the range of admissible values, the width of these distributions is considerably lower than in Figure 5.12. However, the analyzed image contains many regions with high curvature, which is evidenced by the high density of values outside the range of admissible values in the distribution.

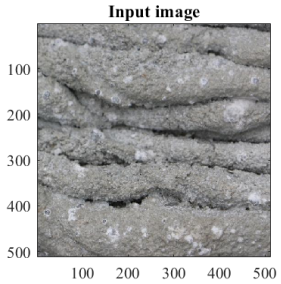
In these study cases, the fields of view were adjusted to analyze 6 and 7 layers, respectively. However, the method can be extended to an arbitrary number of layers as long as the resolution allows segmenting the interlayer lines independently. To show this, we compare two new study cases with a more global point of view.

The first study case, analyzed in Figure 5.14, shows the inline monitoring of a printing process with overall acceptable quality. Besides some layer segments with a thickness outside the admissible range, the generated distributions denote a good printing process since the measured values are centered and contained inside the ranges of admissible values.

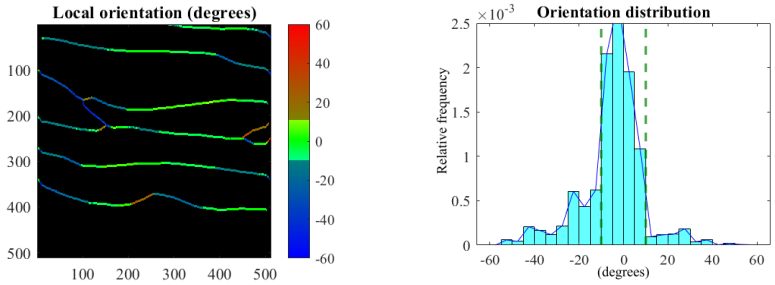
The second study case, analyzed in Figure 5.15, depicts the monitoring of a printed piece during drying. Unlike the previous case, this piece exhibits unacceptable quality. This can be concluded from the generated distributions, which are highly distributed outside the admissible ranges unlike Figure 5.14.

A remark about the curvature must be done, since the perception of the curve depends on the scale. In these experiments, we preserved the range of admissible curvatures used for inline local monitoring. That is why, even though many regions of the interlayer lines are curved, they are not detected as anomalies: the curves appreciated at this scale have curvature magnitudes considerably lower than the ones appreciated when analyzing a close view of only 4 layers. Even so, the distribution of curvatures is clearly wider than in Figure 5.14, containing out-of-range values, since the analyzed piece is in fact defective.

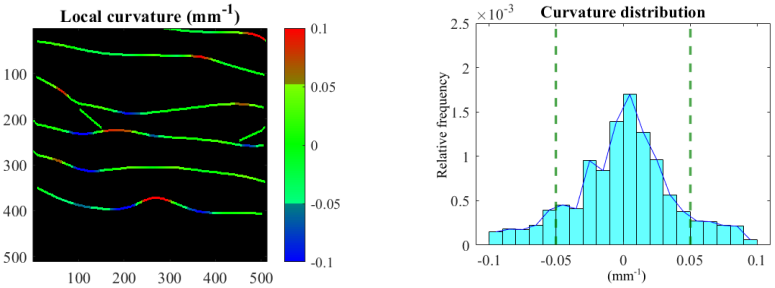
These preliminary results show the direct application of our technique in images outside the context of the I3DCP dataset. Next, we close this chapter with a final discussion on the topic of geometrical assessment in 3DCP and future perspectives of the presented work.



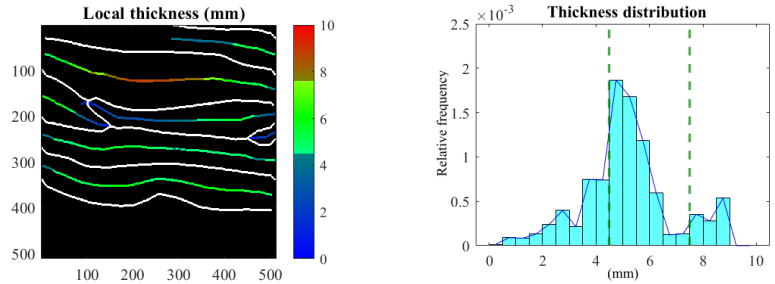
(a) Image to analyze as captured by the camera



(b) Local orientations of the interlayer lines

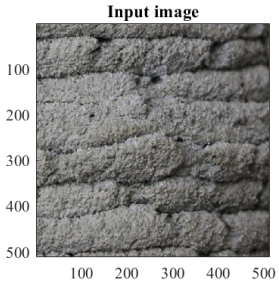


(c) Local curvatures of the interlayer lines

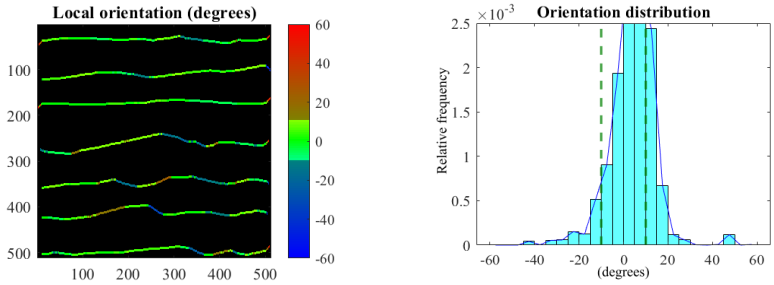


(d) Local thickness of the layers. The thickness is associated to the layer centers (in color); the white lines represent the interlayer lines

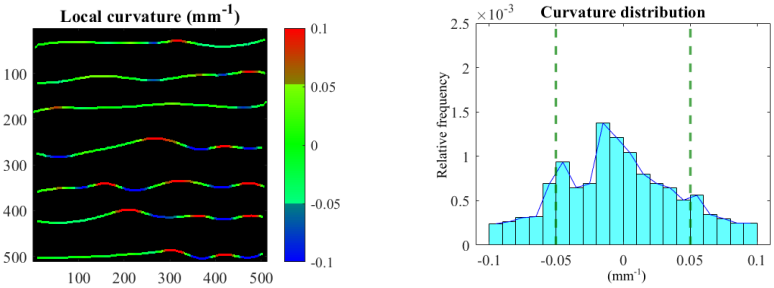
Figure 5.12: Plots and distributions from the first study case of hardened pieces.



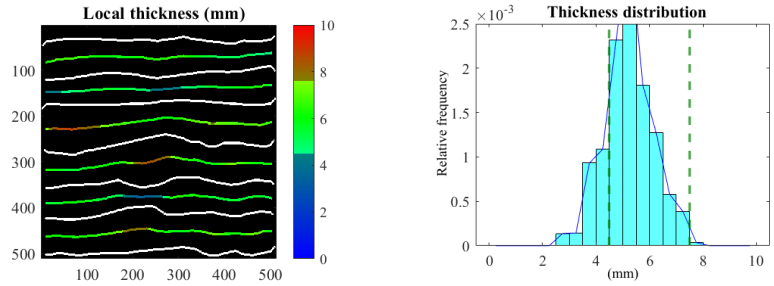
(a) Image to analyze as captured by the camera



(b) Local orientations of the interlayer lines

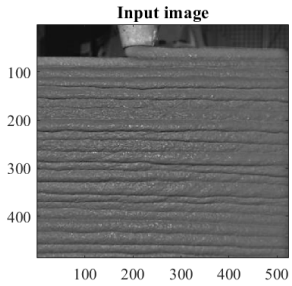


(c) Local curvatures of the interlayer lines

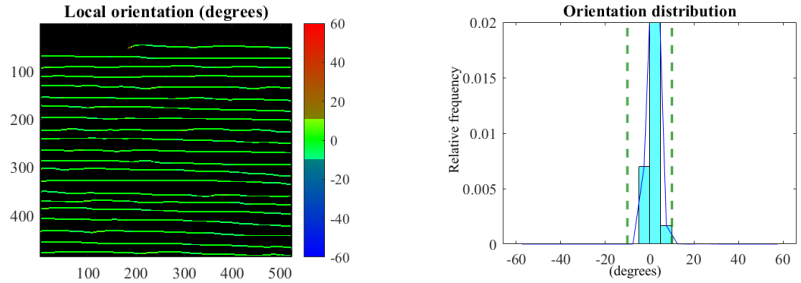


(d) Local thickness of the layers. The thickness is associated to the layer centers (in color); the white lines represent the interlayer lines

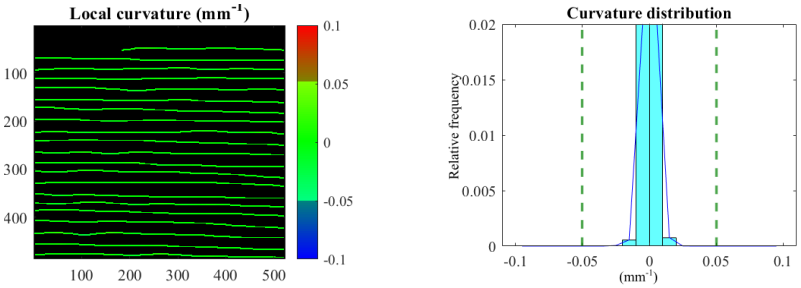
Figure 5.13: Plots and distributions from the second study case of hardened pieces.



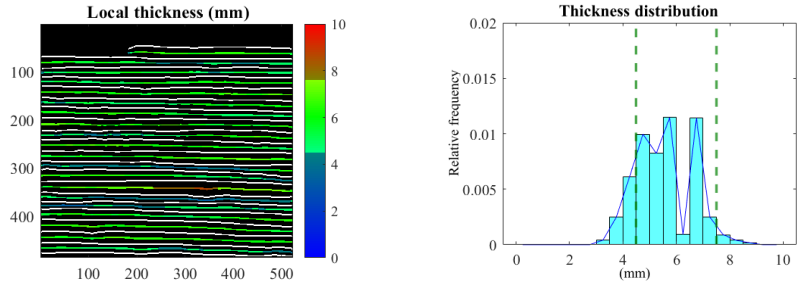
(a) Image to analyze as captured by the camera



(b) Local orientations of the interlayer lines

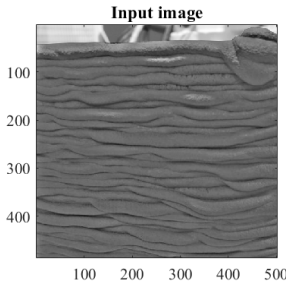


(c) Local curvatures of the interlayer lines

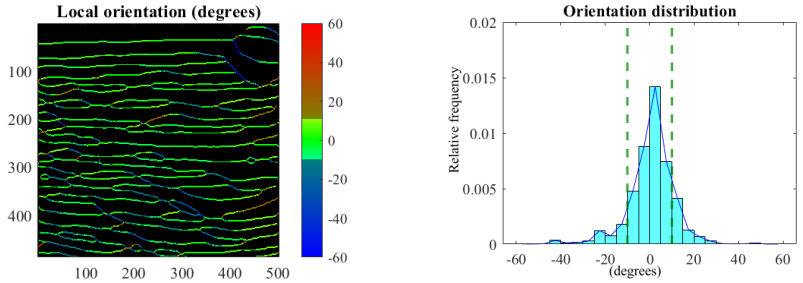


(d) Local thickness of the layers. The thickness is associated to the layer centers (in color); the white lines represent the interlayer lines

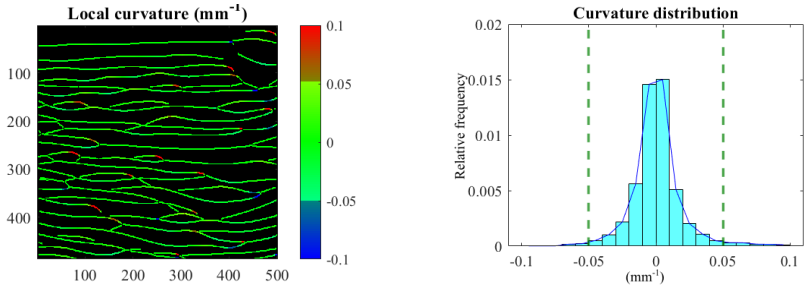
Figure 5.14: Plots and distributions of inline monitoring with a wide field of view. Example of acceptable printing.



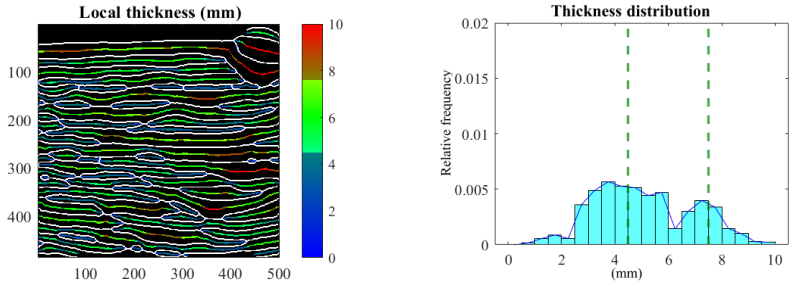
(a) Image to analyze as captured by the camera



(b) Local orientations of the interlayer lines



(c) Local curvatures of the interlayer lines



(d) Local thickness of the layers. The thickness is associated to the layer centers (in color); the white lines represent the interlayer lines

Figure 5.15: Plots and distributions of drying monitoring with a wide field of view. Example of unacceptable printing

5.5 Discussion on Geometrical Assessment of 3D Concrete Printing and Future Perspectives

In this chapter, we proposed a method for inline 3DCP geometrical monitoring. This method is based on the analysis of a binary map of the interlayer lines in the acquired image. The proposed geometrical characterization consists of the measurement of 4 properties: the local orientation and curvature of the interlayer lines, the local thickness of the layers delimited by these lines, and the height of the observed extrusion nozzle with respect to the last printed layer.

By choosing a range of normal values for each of these measures, any value below the minimum or above the maximum thresholds is considered as anomaly. The quality assessment of the printing process can be defined, in terms of geometry, by the presence or absence of these anomalies in the printed piece. Since the measurement of each of the analyzed properties is performed locally, the values retrieved by these analyses allow locating anomalies in the analyzed images. Consequently, the proposed method provides both the exact location and the type of the detected anomalies.

We showed the applicability of our method in the context of the I3DCP dataset, proving that our technique can provide an inline characterization of a printing process based on visual inspection of a few last printed layers. Furthermore, we showed that our methodology can be easily applied to rather different contexts of geometrical assessment in 3DCP: analysis of drying pieces, analysis of hardened pieces, analysis at different scales (depending on the field of view of the camera).

Future research could extend this analysis for real-time monitoring and closed-loop control. Regarding real-time monitoring, the history of the values measured from the beginning to the end of the printing could be used as a quality report of the whole printing process. This approach requires a synchronization, either to acquire frames without common regions (and performing a single analysis per region) or to evaluate the evolution of the measured values per region over time (using the time as an additional dimension for the analysis). The first approach is simple, but naive since it ignores the deformations that occur over time e.g. a progressive crushing of lower layers due to the weight of the newly added ones. The second approach provides a more complete report and can be extended to analyze the piece during its hardening period. However, it adds a non-trivial challenge: providing a line-to-line correspondence between frames under the assumption that the mesh defined by the interlayer lines is prone to change over time. To fully exploit this approach, a global perspective of the printed piece would be more useful than the local approach proposed in this chapter. The presented results show that the proposed method can be easily extended to the global perspective, although this approach is computationally more expensive.

Nonetheless, the local perspective is enough towards a closed-loop control: the nature and the severity of the detected anomalies could be used as feedback to an automated decision system, issuing corrective actions on the printing process. The decisions could stem from a rule-based expert system, vector-to-action dictionaries, supervised machine learning models, etc.

In this chapter, we showed the wide range of possibilities for 3DCP monitoring based solely on geometrical properties. However, the monitoring based on computer vision can provide characterization –and anomaly detection– in terms of other types of properties. In the next chapter, we discuss an approach currently ignored in the literature for 3DCP: monitoring based on texture.

Chapter 6

Textural Characterization and Anomaly Detection in 3D Concrete Printing

As discussed in the previous chapter, the research on methods for automatic monitoring of 3DCP using computer vision has centered its attention on geometrical assessment. Nonetheless, the data retrieved from a single camera can provide information about additional properties.

The estimation of water content in the mixture lends itself to computer vision, since it is one of the factors affecting the texture of the printed material. In fact, when the texture exhibits abnormal properties, it becomes a good visual indicator of anomalies in the printing process. For inline monitoring, a textural analysis allows identifying when the extruded material has a water content below or above the adequate values (see Figure 6.1). An excess or shortage of water is a potential indicator of issues, which may impact the structural performance. For example, the excess of humidity produces undesired deformations under the pressure imposed by newly deposed layers. On the opposite side, cold joints come from a lack of water at the interface. When the flow rate is low with respect to the robot speed, the superficial stress in the extruded layer produces defects such as sharkskin and micro-cracks. When the fluidness of the extruded layer is not enough to match the robot speed, drastic failures such as longitudinal tearing can be produced.

In this context, the problem can be approached as a classification one. In fact, texture monitoring based on classification has been already explored for defect detection in smaller-scale additive manufacturing e.g. fused filament deposition.

(a) Excessively dry (b) Good (c) Excessively wet

Figure 6.1: Texture of concrete with different water contents.

However, this approach has not yet been explored for 3DCP.

In this chapter, we propose a method for the characterization of textural anomalies in 3DCP, based on window-wise machine learning classification. The proposed characterization consists of extracting a set of local, textural features from each window in the observed layers. The obtained feature vector is then fed to a classifier, previously trained in a supervised manner to return a single label per window: either *good* (corresponding to the desired material properties) or one out of 3 defective cases: excessively fluid, excessively dry or exhibiting superficial tearing. The anomalies are then detected in windows with a class other than *good*.

In the rest of this chapter, first we provide a brief literature review on texture analysis focused on (small-scale) additive manufacturing. Then, we discuss the feature extraction we used for classification (we describe the selected classes and texture descriptors). Next, we analyze the produced data and discuss about the model used to classify the textures. Afterwards, we show the results of our experiments using the proposed method. We close the chapter with a final discussion on the topic of textural monitoring of 3DCP and some future perspectives.

6.1 Textural Monitoring of Small-Scale Additive Manufacturing in the Literature

Inline monitoring for quality control is an active topic of research for many additive manufacturing technologies. For fused filament extrusion, for example, we find many works based on computer vision (Oleff et al. [2021]). Among these, one line of research is the inspection of surface quality based on texture analysis. One example is the work presented by (Liu et al. [2019b]) for inline closed-loop quality control. The authors fix digital microscopes to the 3D printer, near to the extruder, to collect high-quality images of the printed part's surface. The researchers isolate the region just below the nozzle as the region of interest, and extract features based on statistical properties from gray-level co-occurrence matrices. Using the k-NN algorithm, the authors classify the regions of interest into 1 of 4 classes related to under/over-filling (including a class for good printing).

Similarly, (Jin et al. [2019]) proposed a method based on DL for defect detection based on under/good/over-extrusion. The authors fix a camera near the nozzle and adjust the focal length to get the best image quality at the printed region beneath the nozzle. The extracted images are fed to a modified ResNet-50 to classify each region of interest into 1 of the 3 classes. The use of this CNN allows omitting the feature extraction prior to training the classifier.

The aforementioned works performed their monitoring on the level of cropped windows, and were used to analyze flat, wide printed surfaces. In (Petsiuk and Pearce [2020]), the analyzed pieces have more complex shapes: instead of surfaces with fixed borders, the authors analyzed pieces with variable contours along the vertical printing axis. These contours are considered as the outer shell of the printed piece, and the texture irregularities are inspected only in the region bounded by this shell at the top surface. The contours are determined

prior to the textural analysis, and the region contained within the contour is segmented pixel wise. Each pixel is represented through a texton obtained by the Leung-Malik filter bank. The segmentation is performed through Gaussian mixture model clustering followed by agglomerative hierarchical clustering. The final number of clusters is 6, from which one is considered normal and all the others are considered defects (a semi-supervised approach).

These works are examples of the popularity of texture analysis for inline monitoring of fused filament deposition. Despite of this fact, the existing literature with respect to concrete extrusion ignores these properties, focusing only on geometrical properties as discussed in the previous chapter. For 3DCP, a textural analysis can be performed similarly to the approaches used in fused filament deposition i.e. by inspecting the patterns generated by the multiple extruded layers. This approach can be considered as an analysis at global level.

However, the extruded concrete layers are considerably wider than the fused filaments. This implies that the texture of a single layer can be easily inspected in 3DCP without requiring specialized acquisition devices such as microscopes. Inspecting the texture within a single layer rather than the patterns generated by a set of extruded layers can be considered as an analysis at local level. In this work, we use this approach with a camera fixed to the extrusion nozzle.

6.2 Characterization of 3D Printed Layers Based on Texture

We propose a method based on image processing and machine learning to characterize the textures of the printed layers at local level. Similarly to (Jin et al. [2019]; Liu et al. [2019a]), the analysis is performed at the level of windows, assigning a single class per window; specifically, one among 4 possible classes: fluid, good, dry or tearing. These classes are directly related to the water content of the extruded material, in descending order:

Fluid. This class corresponds to excessively fluid material caused by excessive water content. When the material has the desired properties, the aggregates in the mixture take the appearance of visible grains on the surface of the extruded material; however, when the overall fluidness of the material increases, these grains become invisible on the surface. Therefore, the “fluid” class is characterized by a smooth surface. Since this fluidness is associated with high water content, this class is also characterized by a high specular reflectance.

Good. This class corresponds to the desired material properties. In this case, the water content is adequate, producing the appearance of more visible grains on the surface and a lower reflectance. The distribution of the visible grains in the “good” class exhibits a higher density and homogeneity relative to the fluid class.

Dry. This class corresponds to low fluidness caused by a shortage of water. This class is characterized by a rougher texture, caused by an increased amount of visible grains with bigger size. Since the water content is reduced, the “dry” class exhibits very low to no reflectance. This class is also determined by the absence of cracks.

Tearing. This class corresponds to the limit case in which the lack of water produces tearing in the extruded layer. The "tearing" class is characterized by the appearance of cracks. It is also identified by the lack of reflectance and often by a rougher texture than the dry class.

We identify the 4 classes in the I3DCP dataset (first presented in subsection 4.1.1), as illustrated in Figure 6.2. Before analyzing each window for classification, the image is pre-processed at global and window level.

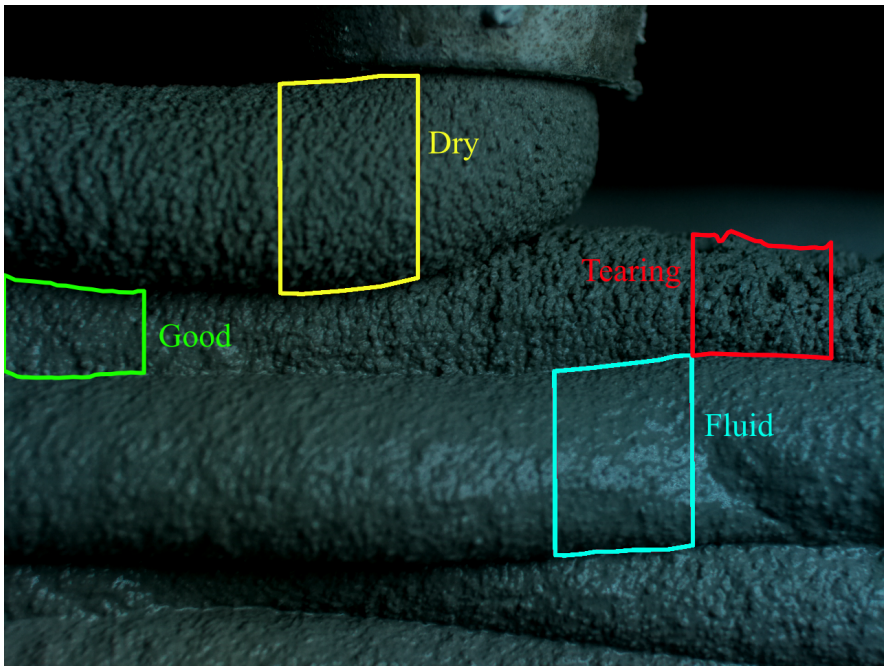


Figure 6.2: Image from the I3DP dataset containing the 4 proposed classes.

6.2.1 Pre-Processing

The analysis of the printed layers is performed at window level. Before extracting the windows, we level the whole image in grayscale to preserve textural information while reducing the effects of lighting and shadows. To do this, a local-mean-intensity image is obtained using a Gaussian filter with $\sigma = 40$. The filtered image is subtracted from the input image, producing a new image with an intensity distribution centered around 0. To avoid negative values, we subtract from this image its minimum intensity. An example of input image and the result of the described pre-processing is shown in Figure 6.3.

The windows to analyze are extracted from the resulting image. The height is adapted per window to contain the totality of the layer in the vertical direction. The width is fixed to 200 pixels (~5mm), and the windows are horizontally adjacent. The width presents a trade-off between the available area to analyze per window and the locality of the analysis; the value used in this work was chosen to have near-square windows. In Figure 6.4, we illustrate the analyzed regions for a sample image. Before extracting a window from a layer, the image is masked to ignore textures corresponding to other layers (those pixels are set to 0); this approach is similar to the one used by (Petsiuk and Pearce [2020]), in which the region of interest is delimited by a physical boundary (in this case, the interlayer

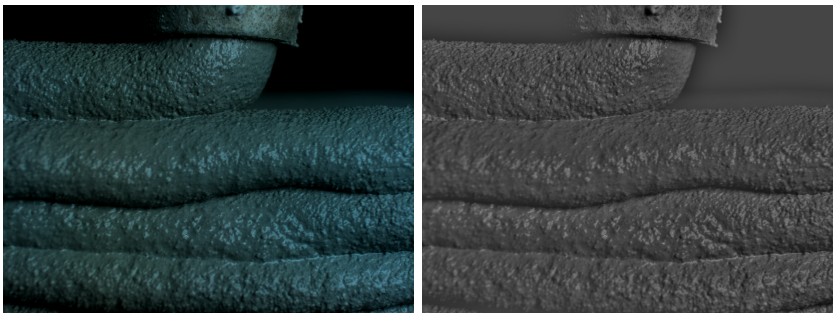


Figure 6.3: Raw input (left) and leveled image (right) for texture analysis.

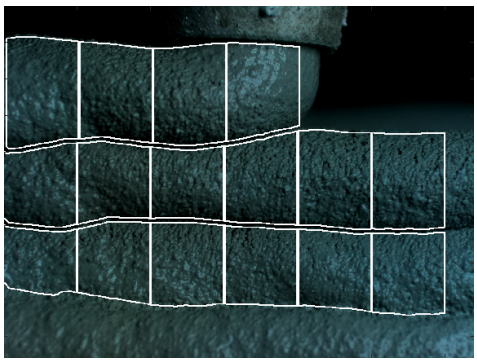


Figure 6.4: Example of windows used to analyze the textures of the observed layers. The width is fixed and the height is adapted to contain the entirety of the layer (vertically).

lines). Some examples of extracted windows ordered per class are shown in Figure 6.5. The texture descriptors are obtained from the maximum box contained within concrete pixels in the analyzed window (in order to ignore the zero-valued pixels outside the layer). Let each of these cropped boxes be called a T_{box}.

6.2.2 Texture Descriptors

As observed in Figure 6.5, each T_{box} can exhibit a different height when the thickness of the printed layers varies. Furthermore, this height varies according to the orientation and curvature of the layer of interest (remember, each T_{box} is bounded by the zero-valued pixels corresponding to adjacent layers). Because of this, we need a feature extraction method that allows extracting a fixed-size vector independently from the size of the input window. Additionally, since one of the factors that modifies the visible texture is the size of the aggregates in the mixture, resizing is not an adequate option.

Furthermore, similarly to the case of the interlayer lines, labeled windows are required for training. In this case, a single label per window is required; however, unlike the interlayer lines, deciding the class of the observed texture requires some expertise on the process. Moreover, texture recognition is not a trivial task for humans. These two aspects make it difficult to acquire reliable labels, limiting the amount of labeled samples that are available for training.

Because of the limitations discussed on feature extraction and window labeling, we decided to use an approach based on traditional texture descriptors. Specifically, inspired by previous works for texture analysis of small-scale addi-

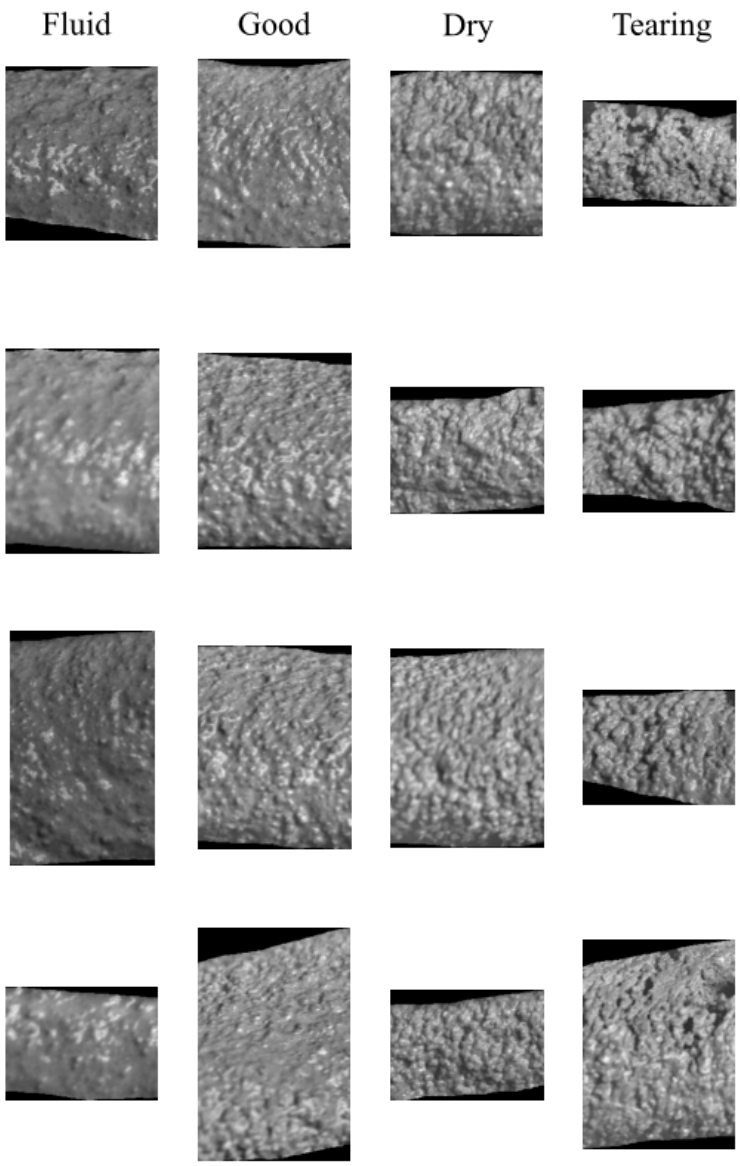


Figure 6.5: Examples of extracted windows for the four proposed classes (with improved contrast for visualization).

tive manufacturing (Liu et al. [2019b]; Shen et al. [2018]), our texture classification is based on two visual descriptors commonly used for texture: gray-level co-occurrence matrices (GLCMs) and local binary patterns (LBP). Before extracting features, each T_{box} is quantized to q binary values. To avoid that outlier intensity values have an undesired influence, each T_{box} is clipped to values in the range $mean(T_{box}) \pm 3.1 \times std(T_{box})$ before quantization. Let the quantized version of the clipped image, in the range [0, q-1], be called a T^{q}_{box}.

Gray-Level Co-Occurrence Matrices

Let us define a GLCM (Haralick et al. [1973]) as:

$$\mathrm{GLCM}^{\mathrm{I}}_{\Delta x,\Delta y}(i,j) = \sum_{x=1}^{w}\sum_{y=1}^{h}\begin{cases}1, & \text{if I(x, y)=i and I(x + } \Delta \text{ x, y + } \Delta \text{ y)=j}\\ 0, & \text{otherwise}\end{cases} \tag{6.1}$$

Here, h and w are the height and the width of the analyzed image I, respectively; Δx and Δy are a horizontal and a vertical offset, respectively. $\mathrm{I}(x, y)$ returns the intensity value of the pixel at the position x, y in I.

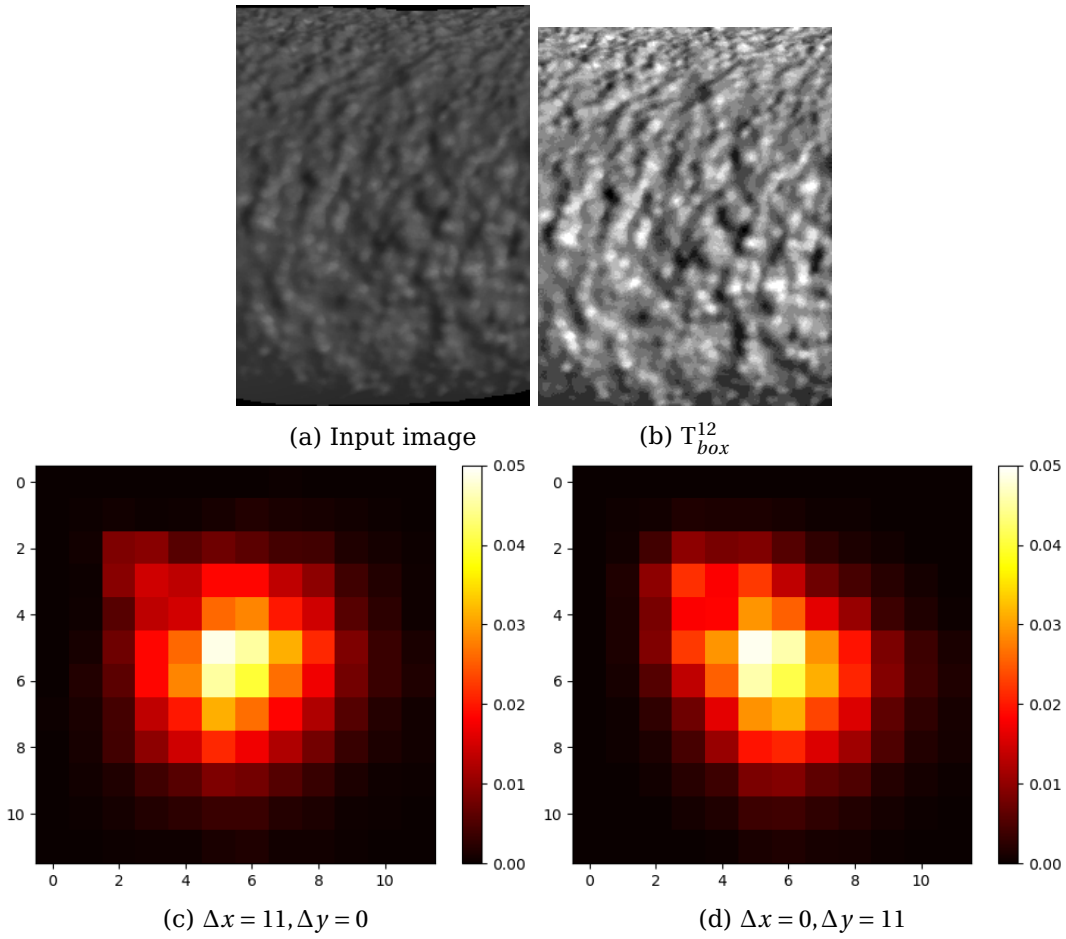
A GLCM is a distribution of co-occurring pixel values at a given scale (the chosen offset). With q intensity levels in a T^{q}_{box}, per each offset we obtain a square $q \times q$ matrix (see Figure 6.6). To reduce the dimensionality of the features obtained through GLCMs, we can reduce a GLCM to a single scalar in terms of statistical properties. With $\mathrm{P}_{i,j}$ the value of the normalized GLCM at the position i, j (i.e. the probability of the intensity pair (i, j)), we calculate 5 properties:

Contrast: the expected squared difference of intensities in the GLCM. It is defined as:

$$\sum_{i,j=0}^{levels-1} \mathrm{P}_{i,j} \cdot (i-j)^2 \tag{6.2}$$

Dissimilarity: the expected absolute difference of intensities in the GLCM. It is defined as:

$$\sum_{i,j=0}^{levels-1} \mathrm{P}_{i,j} \cdot |i-j| \tag{6.3}$$



(a) Input image (b) T^{12}_{box}

(c) $\Delta x = 11, \Delta y = 0$ (d) $\Delta x = 0, \Delta y = 11$

Figure 6.6: Example of GLCM extraction. a) The image to analyze (200 × 273 pixels). b) The maximum box of concrete pixels, with 12-levels quantization. c) GLCM extracted with a 11-pixels horizontal offset. d) GLCM extracted with a 11-pixels vertical offset.

Homogeneity: a measure of the closeness of the distribution of elements in the GLCM to its diagonal. It is defined as:

$$\sum_{i,j=0}^{levels-1} \frac{P_{i,j}}{1+(i-j)^2} \cdot |i-j| \tag{6.4}$$

Energy: the square root of the sum of squared elements in the GLCM. It is defined as:

$$\sqrt{\sum_{i,j=0}^{levels-1} P_{i,j}^2} \tag{6.5}$$

Correlation: a measure of how correlated are the pixels to their neighbors with the given offset. It is defined as:

$$\sum_{i,j=0}^{levels-1} P_{i,j} \cdot \frac{(i-\mu_i)(j-\mu_j)}{\sqrt{\sigma_i^2 \cdot \sigma_j^2}} \tag{6.6}$$

Due to the printing process, the texture properties are anisotropic, as observed in Figure 6.6(c-d). Because of this, we extract GLCMs for both the horizontal and the vertical direction independently. To capture information at many scales, we use multiple offset magnitudes (detailed in section 6.3).

Local Binary Patterns

A LBP transformation consists of encoding each pixel to a n-bits binary number depending on its n neighbors with radius R (Ojala et al. [1994]). Following the neighbors along the hypothetical circle of radius R, each bit is assigned a 0 if the center pixel's intensity value is greater than the current neighbor's value. Otherwise, the bit is assigned a 1. The transformed image contains, per pixel, the decimal equivalent of the binary number calculated according to their neighbors. The process for a single pixel is illustrated in Figure 6.7

We use 8 neighbors for all the LBP transforms; therefore, the transformed images contain values in the range [0, 255]. Per each T_{box}^q, similarly to GLCMs, we obtain LBP transforms with different radii to obtain information at different scales (detailed in section 6.3). Some examples of LBP transforms are shown in Figure 6.8.

After getting the LBP transform of a T_{box}^q, we calculate the histogram of intensities of the transformed image to extract a fixed-size vector per T_{box}^q. To reduce the dimensionality of this vector, the histograms can be calculated using $b < 255$

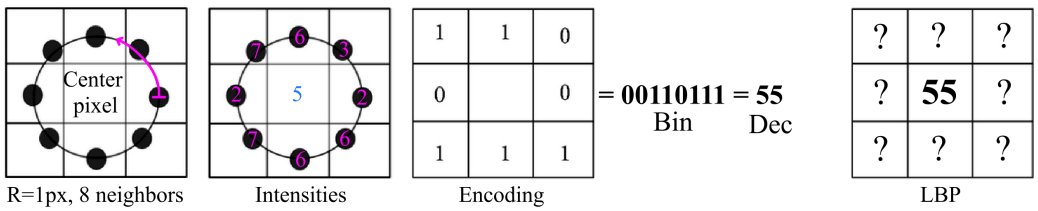
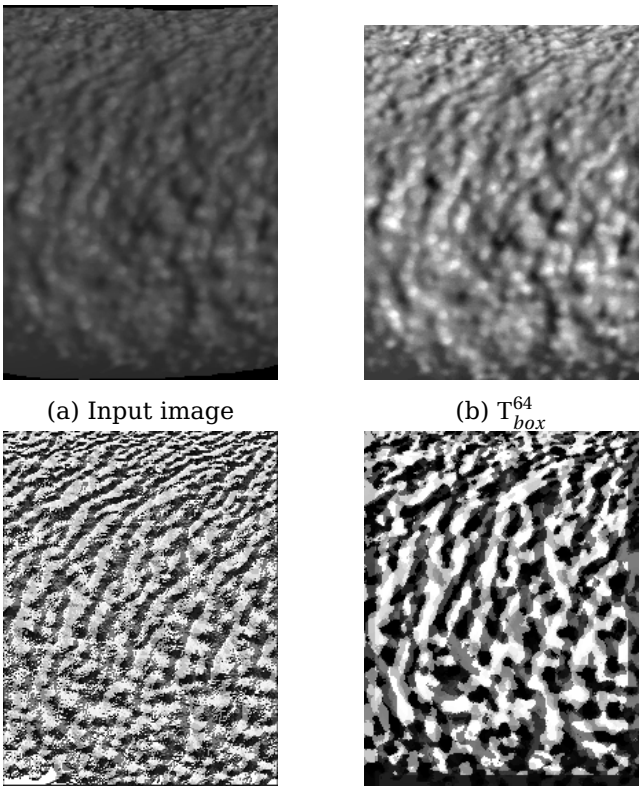


Figure 6.7: Example of LBP for a single pixel.



(a) Input image (b) T^{64}_{box}

(c) R = 1, neighbors = 8 (d) R = 11, neighbors = 8

Figure 6.8: Example of LBP transforms. a) The image to analyze; resolution: 200 × 273 pixels. b) The input image after 64-levels quantization. c) LBP transform obtained with a 1-pixel radius. d) LBP transform obtained with a 11-pixel radius.

bins. The histograms are normalized so that their magnitudes are not influenced by the size of T^{q}_{box}.

In the next section, we discuss the parameters used for the final feature extraction and texture classification.

6.3 Data Analysis and Texture Classification

For GLCM extraction, a texture window T_{box} is quantized to get a T^{q}_{box} with $q = 12$. The offset distances are selected to go from 1 to 50, using a step size 1, in both directions (horizontal and vertical). These parameters are chosen from preliminary experiments using distributed asynchronous hyperparameter optimization with Hyperopt (Bergstra et al. [2013]). Using this setup, the total number of GLCM features per T^{q}_{box} is 500: 5 properties × 2 angles × 50 distances. The features are obtained respecting that order, such that the feature resulting from <property x, angle y, distance z> is followed by the feature resulting from <property x, angle y, distance z+1> in the final feature vector.

Regarding LBP, $q = 64$ is chosen for T_{box} quantization. The radii are selected to go from 1 to 31, using step size 10. The histogram extraction is performed using 8 bins. Again, these parameters were chosen from preliminary experiments using Hyperopt. With this setup, we extract 32 LBP features per T^{q}_{box}: 4 radii × 8 bins. The features are organized respecting that order.

The final feature vector per texture window, with size 532, is the concatenation of the GLCM and LBP features. It is worth to notice that this vector is dominated by the GLCM features, with the LBP features representing only ~6%. The proportion between both descriptors is mainly due to the hyperparameter search performed with Hyperopt: the step of the offsets for the GLCMs is 1, while it is 10 for the LBP. This suggests that the GLCMs are more informative at different offset scales.

From the raw images in the I3DCP dataset, we extracted and labeled a total of 111 texture windows. Each of these windows, labeled by the agreement of two annotators, is assigned with a single numerical label: 0/fluid, 1/good, 2/dry, 3/tearing. The distribution of classes is: 24 fluid, 27 good, 24 dry and 36 tearing. The base dataset is composed by the 532 features calculated per each of these images. To avoid that some features dominate the classifier because of their magnitude, we standardize each one of them for training.

The model used to learn from this domain is a small convolutional neural network (CNN) with one convolutional input layer and one fully-connected output layer (see Figure 6.9). The decision of using a convolutional layer instead of training a multilayer perceptron comes from the implementation used for feature extraction: adjacent elements in the feature vector represent the same property but with different offsets (see Figure 6.10); consequently, adjacent features have a high likelihood of being correlated. The convolutional layer allows capturing meaningful information from the neighborhood of a feature, summarizing the information retrieved from similar offsets. The kernel size is 6, with a ReLu activation to obtain non-linear outputs. To reduce the likelihood of overfitting, we use dropout regularization with 30% probability. The output layer has 4 neurons and uses a softmax activation so that $\sum_{c \in classes} \mathrm{P}(c|\text{features}) = 1$.

In the next section, we present the results obtained with this model using the proposed dataset.

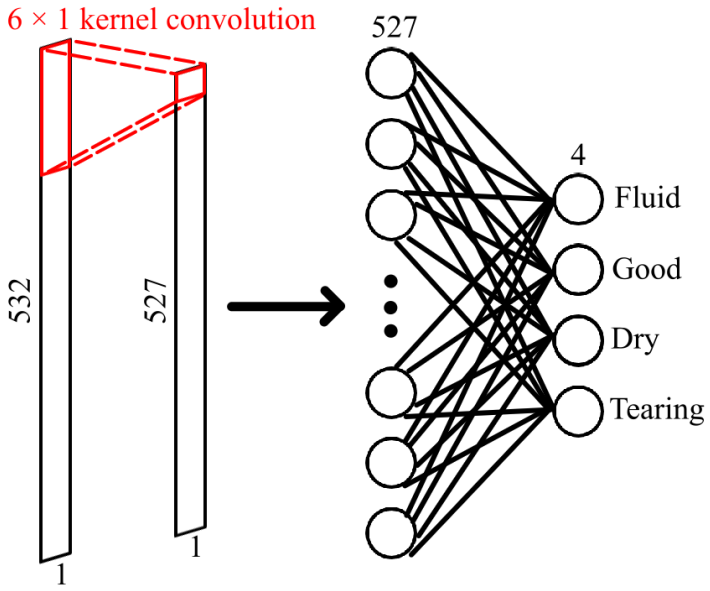


Figure 6.9: CNN for texture classification.

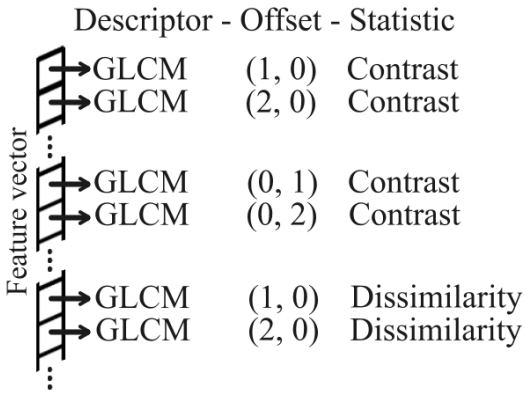


Figure 6.10: Feature vector for texture classification. Adjacent elements represent the same property but with different offsets.

6.4 Experiments and Results for Texture Classification

In this multi-class context, we evaluate using a 5-fold stratified cross-validation i.e. each fold has the same proportion of observations with a given class. Per iteration, we perform a stratified split to divide all the samples from the training folds into a training and a validation split; the size of the resulting validation split is half the size of the test fold. Finally, we perform data augmentation on the images from the resulting training split, obtaining 16 additional images per texture window. The transformations are rotations (with angles -4, -3, -2, 2, 3, 4 degrees), zooms (with scale factors 0.97, 0.98, 0.99, 1.01, 1.02, 1.03) and illumination rescales and shifts defined by $\alpha * \text{intensity} + \beta$ (with pairs α/β: 0.9/0, 1.1/0, 1.0/-50, 1.0/50).

Per iteration, the approximate number of samples per split is 1326 training, 11 validation and 22 test. Validation and testing are performed on real, raw images only. The training split is composed by raw images and their corresponding augmented versions. The approximate class distribution per split is 22% fluid, 24% good, 22% dry and 32% tearing.

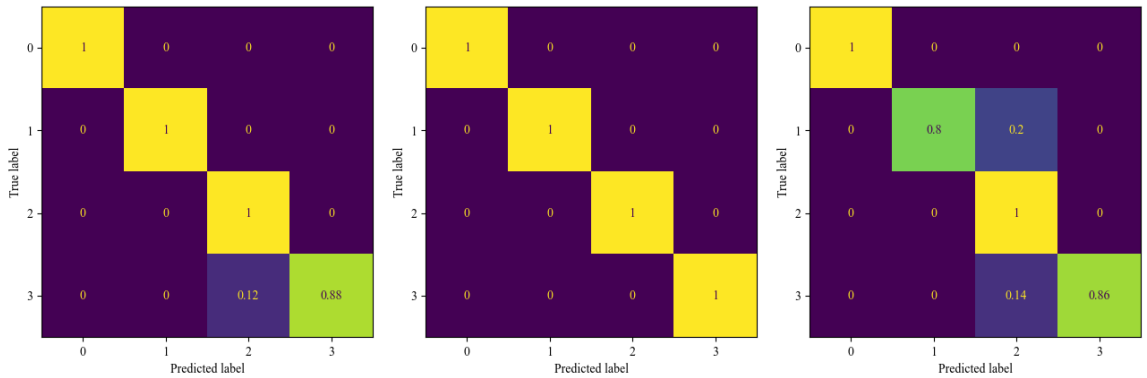
The CNN is trained using the categorical cross-entropy loss. The optimizer was Adam with default parameters in Tensorflow 2.1.0. The network is trained for a maximum of 2000 epochs with batch size of 32. An early stop is applied if the loss in the validation split does not improve during 100 consecutive epochs.

As evaluation metric for this multi-class –single-label– task, we selected the macro-averaged F-score (± standard deviation). This score is the average of the F-scores per class, giving the same importance to all the classes. Additionally, we calculate the macro F-score per class as the average over the 5 folds (± standard deviation). These results, including the single F-scores per class and fold, are presented in Table 6.1.

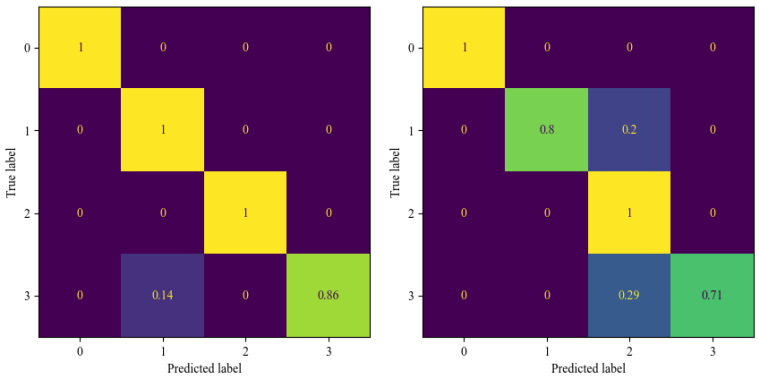
Among the four classes, fluid exhibited the best scores (100% in all folds). On the other side, the most difficult class was dry, with a 90.2% average over the 5 folds. The good and tearing classes have slightly better averages: 93.7 and 92.2%, respectively. From an analysis of the resulting confusion matrices per fold (see Figure 6.11), we discovered that the main source of these errors was

Table 6.1: F-scores(%) of texture classification using 5-fold stratified cross-validation.

Class	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Class macro-average
Fluid	100	100	100	100	100	100±0.0
Good	100	100	88.9	90.9	88.9	93.7±5.8
Dry	90.9	100	83.3	100	76.9	90.2±10.2
Tearing	93.3	100	92.3	92.3	83.3	92.2±5.9
Fold macro-average	96.1±4.7	100±0.0	91.1±7.0	95.8±4.9	87.3±9.8	



(a) Fold 1 (b) Fold 2 (c) Fold 3

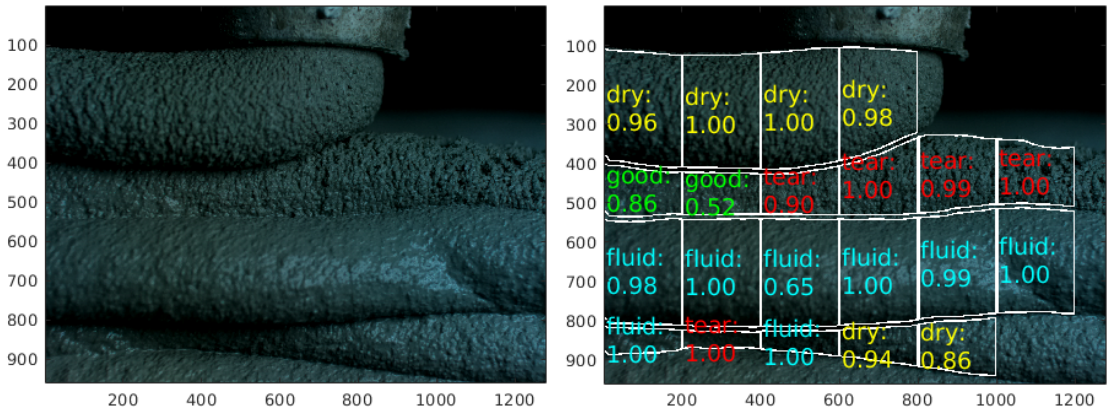


(d) Fold 4 (e) Fold 5

Figure 6.11: Confusion matrices of the 5 testing folds for texture classification. The rows are normalized. The number code is: 0/fluid, 1/good, 2/dry, 3/tearing.

classifying good and tearing images as dry. Nonetheless, this kind of confusion is understandable since the dry class is, physically, a transitory state between good printing and tearing. Furthermore, these errors are infrequent, as denoted by the average of the "Fold macro-average" row: 94.1% (±4.9).

In Figure 6.12, we show the output of our method on a sample image. There, we observe that the analyzed image exhibits the four proposed classes: dry on the currently extruded layer, good and tearing in the layer below, and fluid in the third layer from top to bottom. Additionally to the label, the image shows the confidence of the classifier as the probability of the class given the extracted features. In the second layer, we can observe that the second window from left has low confidence in the good class; this texture window is precisely a transition



(a) Input image (b) Region-wise classification

Figure 6.12: Texture classification of an image from the I3DCP dataset (the same as in Figure 6.2). The white lines in (b) show the boundaries of the analyzed region. Per region, we provide the predicted label as well as the probability of the class according to the classifier.

between a good texture and the beginning of tearing. Using this case as example, we observe that the confidence of the classifier can add an additional piece of information for the characterization of the analyzed layer segment.

One potential weakness of the proposed method is the analysis of windows with considerably low surface. An example is in the bottom layer, where there are small windows with not enough information to produce an effective classification. This explains the drastic class differences in adjacent windows (fluid and tearing). Nonetheless, these regions have a geometrical defect since they are excessively thin. Using the monitoring method proposed in the previous chapter, these regions can be filtered according to the local layer width. A more comprehensive discussion of the advantages and limitations of the proposed method for texture analysis is presented in the next section.

6.5 Discussion on Textural Assessment of 3D Concrete Printing and Future Perspectives

In the current chapter, we proposed a method for layer characterization and anomaly detection based on local, window-wise, texture classification. This supervised machine learning approach is defined as a single-label, multi-class problem. The proposed classes are: fluid, good, dry and tearing. Each of these classes is related to the water content of the mixture. During printing, an excess or shortage of water is a potential indicator of issues, which may impact the structural performance. As such, any class other than good is considered as an anomaly.

In the proposed method, first we pre-process the images to reduce the impact of undesired shadows and overall uneven illumination during image acquisition. It is worth noticing that the pre-processing can be simplified with more sophisticated lighting systems, providing uniform illumination to all the regions of interest.

The feature extraction is based on two visual descriptors used for texture classification: GLCMs and LBP. The proposed classifier consisted of an input convolutional layer and an output, fully-connected layer. We tested the proposed method in the I3DCP dataset, using 5-fold stratified cross-validation and a macro-averaged F-score. In our experiments, we obtained an average score of 94% over the 5 testing folds.

Regarding the annotation of the images from I3DCP, it is important to highlight that there was often a disagreement between independent annotators. Originally, 447 texture windows were extracted to be labeled. In an initial attempt to label these images, with 4 independent annotators, there was an agreement only on 112 images. The agreement between annotators was around 0.38 as measured by the Fleiss' kappa –with 1 corresponding to a perfect agreement (Fleiss [1971]).

As a result from this experience, a formal ruleset was decided between the original annotators (the ruleset is available at the I3DCP dataset's repository, see Appendix). The final dataset was labeled using these rules, beginning from the images that had, originally, a consensus between the four annotators: some images and their labels were preserved, and new ones were added with the consensus of two annotators, so that the resulting dataset contained a balanced amount of images per class. Indeed, texture classification is a hard task for humans, but the results obtained by our model show that this task can be automatized with our method for 3DCP monitoring.

Future research could be focused on extending the generalization ability of the produced models. The descriptors currently used for feature extraction are defined primarily by the distance between the pixel of interest and its neighbors (offset). The offset selection is a hyperparameter to tune each time the camera setup is changed, and the classifier should be retrained to match the new parameter selection. In an industrial setup, it is sufficient to train the classifier once and use it to inspect future instances of the same piece. However, it would be desirable to extend the method with automatic hyperparameter selection, so that it can easily be transferred to different camera setups.

In the current work, we used distributed asynchronous hyperparameter optimization to select the limits and the step size of the offsets. This approach optimizes over the search space by taking into account the scores of the model trained with different hyperparameters. Alternatively, future work could study models meant to select the more important offsets directly during training. In line with the current selection of a CNN for the final classification, the concept of channel attention module (Woo et al. [2018]) is directly applicable.

In fact, the use of convolutional layers allows extracting automatic features from the input image. However, there are technical challenges for this specific problem. The main problem is the input size, since the layers can easily differ in width and overall shape during printing. Since the size of the visible grains is a relevant feature in the texture, and it is directly related to the physical properties of the analyzed material, resizing should be avoided. This, at the same time, imposes a limitation on a common strategy used to increase the generalization power of a trained model: data augmentation. The texture properties of the extruded layers are anisotropic and dependent on the size of the aggregates in

the mixture: common augmentation approaches such as rotation and zoom in/out would produce images that are not representative of the phenomena of interest.

As an alternative to exploit the feature extraction ability of convolutional layers, future research could be focused on the use of fully convolutional networks, such as in chapter 3. Since pixel level annotations are not feasible nor really useful for this problem, the current window-wise labeling could be used for semantic segmentation. For a more precise segmentation, independent from the size of the annotated windows, weakly supervised learning strategies such as the ones discussed in section 3.5 could be explored.

Similarly to the discussion provided about geometrical monitoring proposed in chapter 5, future research could extend the analysis of texture for real-time monitoring and closed-loop control. This chapter closes the topic of inline monitoring of 3D concrete printing. The next chapter presents the conclusions of the work presented along the current thesis manuscript.

Chapter 7

Conclusions and Future Works

Nowadays, there are many areas of opportunity for the digitalization of the construction industry –a sector that has been slow at introducing digital tools with respect to other industry sectors. One of these areas of opportunity is additive manufacturing based on concrete extrusion, better known as 3D concrete printing. This technology is heading towards its maturity; however, there is still work to be done to make this technology reproducible and certifiable. To contribute to this maturity, in the present thesis we approach the problem of automatic monitoring: inspecting the produced pieces during and after their construction in search of defects and overall anomalies.

Towards non-invasive inspection, in the literature we find approaches based on optical devices such as laser triangulation sensors, 3D scanners and time-of-flight cameras. This type of sensors allows measuring geometrical properties of the analyzed pieces, during and after the printing process. Recently, we find works based on computer vision using monochromatic or RGB cameras. While research in this direction is still limited, the use of cameras has a wider range of possibilities than the other type of sensors mentioned in this paragraph. Among these possibilities, the most important is that the analyses are not limited to geometrical properties, allowing to extract meaningful information of other attributes such as the texture of the extruded layers.

To extend the frontier of knowledge in this domain, our work proposed methods for automatic monitoring based on computer vision. These methods are used in two stages of the construction life cycle: during the construction process itself and posterior to the construction process. For quality monitoring during the construction process itself, we approached the problem of inline 3D concrete printing characterization and anomaly detection. For damage monitoring during the lifetime of a construction once it is finished, we approached the problem of crack segmentation. From these two goals, we derive a variety of sub-problems to be addressed through computer vision.

To conclude this manuscript, we summarize our contributions derived from these problems and future work perspectives to extend the presented work.

7.1 Contributions

7.1.1 Crack Segmentation in Constructions

To ensure the security of constructions over their lifetime, as well as to estimate their future durability, it is important to inspect them periodically for damage. Among the possible damages, cracks are an evident sign that can be identified through visual inspection. Information of the cracks such as their length, width, orientation and number can be used to determine their origin and severity. For an automatic and accurate measurement of this type of properties, an accurate segmentation of the cracks is needed. In chapter 2, we studied the problem of crack segmentation.

The U-VGG19 network. In order to obtain a pixel-accurate segmentation of the cracks in an image, we proposed the U-VGG19 architecture. U-VGG19 is an encoder-decoder network inspired by U-net, using the convolutional layers of VGG19 as encoder. To ease the training, we adopted a transfer learning approach by initializing our encoder with the weights of VGG19 pre-trained on ImageNet.

We tested our architecture on public datasets for crack segmentation. Specifically, road crack segmentation, which is a popular and challenging problem in the literature. In the CrackForest dataset, we achieved an F-score of 71%. This score is similar and generally better than other – more complex – approaches in the literature. In a bigger dataset, created by merging the CrackForest and the Aigle-RN datasets, the score slightly improved to 72%; this score is, again, similar to the ones obtained by the state of the art.

To push this score, we used a data augmentation approach. As a result, the recall of our model decreased, decreasing the F-score too. By visually inspecting the predictions of U-VGG19, we determined that the main reason behind this decrease of recall was that the predictions of the network were thinner than the manual annotations used for evaluation. Nevertheless, those thinner predictions looked more similar to the visible cracks. This fact is relevant, because the cracks' width is one important indicator of damage severity and future durability.

Indeed, this observation leads to the problem of learning in presence of inaccurate annotations. It is difficult to produce accurate segmentations when training with inaccurate segmentations. However, the annotation task is highly time consuming and very prone to human error. This error is specially seen on the pixel level, with the annotations usually being excessively wide. However, this problem is generally overlooked in the crack segmentation literature.

The Syncrack generator. We introduced this tool in chapter 3 to study the problem of inaccurate annotations for supervised crack segmentation, as well as to evaluate possible solutions. Syncrack produces parametrizable synthetic images of pavement/concrete-like textures with cracks in them, providing accurate ground truth segmentation maps. The user-accessible parameters include: the image size; the smoothness of the background and its variability between images; the average and standard deviation of the contrast of the noise added to the background; the average and standard deviation of the cracks' width; the average and standard deviation of the contrast of the cracks.

Additionally, the Syncrack generator includes a module to introduce noise to

the segmentation maps, emulating human error. This noise is parametrizable too; the user-accessible parameters include: the approximate percentage of regions in the segmentation map to add noise to; the type of noise to add (dilation and/or erosion of the annotation); the probability of performing a dilation or an erosion when both are performed; the average size of the disk used for dilation and the one used for erosion.

A study on the impact of inaccurate training annotations and unsupervised metrics for crack segmentation. With the help of Syncrack, we studied the effect of training with inaccurate labels. To the best of our knowledge, we are the first to provide this type of study for crack segmentation. From this study, we learned that the predictions of models trained with inaccurate annotations showed a tendency to increase the recall and to reduce the precision. This behavior is observed even though the noise is bi-directional i.e. some crack pixels are mislabeled as background and vice versa.

By training on Syncrack with accurate annotations, we got a precision of 80% and a recall of 78%. After adding noise to the annotations, the precision decreased up to the 55% and the recall increased up to the 85%. The increase in recall is associated to locating cracks harder to identify. In fact, it is possible to increase the importance of thin cracks during training by increasing the width of their annotations. On the other side, the decrease of precision is due primarily to the predicted crack segmentations being wider than the real cracks; this is precisely what we expect to avoid.

From the previous experiments on the CrackForest dataset, we had the hypothesis that training with data augmentation could reduce the impact of inaccurate annotations during training. The results on Syncrack confirmed this hypothesis, seeing an increase of precision up to the 63% and increasing the recall up to the 88%. However, the precision was still 17% below the one obtained by the model trained with accurate annotations.

In the case of Syncrack-generated data, we have accurate ground truths to evaluate the predicted segmentations in terms of supervised scores. However, in presence of real-life inaccurate annotations, maximizing supervised scores as commonly done in the literature is not a reliable way to measure the quality of the crack segmentation. This is particularly true when we are interested in the geometry of the cracks rather than only their location.

As an additional option for evaluation, we proposed the use of unsupervised metrics, based on metrics previously used for unsupervised segmentation. These scores are the first and second order region entropies (where the regions of interest are the pixels predicted as cracks) and the Kolmogorov-Smirnov distance (measuring the difference between the distributions of pixels predicted as crack and background, respectively).

The first order entropy measures the intra-crack intensity uniformity. With a good segmentation, this uniformity increases, minimizing the entropy. Introducing surrounding background pixels to the region segmented as crack, on the other side, increases the entropy. The second order entropy is a similar case, but it is based on co-occurring intensities rather than individual values.

Regarding the Kolmogorov-Smirnov score, introducing surrounding background pixels in the segmentation minimizes the score, because both distribu-

tions become similar to each other. On the other hand, with a good segmentation, the distribution of pixels classified as cracks becomes clearly distinguishable from the one of background pixels; this maximizes the score.

The expected behaviors of the entropies and the Kolmogorov-Smirnov score were verified during our experiments with Syncrack. On Syncrack-generated data, we showed that these scores have a direct relation with the quality of the segmentations in terms of supervised precision, which is the score we want to improve because of its relation to the predicted crack width. These scores are later used as additional metrics to evaluate the methods that we proposed to improve crack segmentation in terms of crack width.

An improved crack segmentation based on weakly supervised learning. Since image segmentation returns a single class per pixel, we can consider each pixel as a single, independent sample. Due to the inaccuracy of the manual annotations, some pixels are mislabeled in this binary classification problem. Training when some mislabeling exists is called inaccurate supervision: a subcase of weakly supervised learning. To learn in the presence of inaccurate labels, we proposed a method based on two steps: 1) getting a set of new pseudo-labels with the help of a model trained on the original, raw data and 2) training a final model using these pseudo-labels. For pseudo-label generation, we tested 4 methods inspired from weakly supervised classification: *5-nn voting*, *consensus voting*, *majority voting* and *self-training*. Among these, the most successful one was *5-nn voting*. This method consists of: 1) training U-VGG19 with noisy labels and using the feature maps from the second-to-last layer as 2D features per pixel; 2) obtaining a new pseudo-label per pixel using the 5-NN algorithm at image level. Although we used U-VGG19 as baseline model for our experiments, any model that produces a feature vector per pixel can be used.

We tested our method on Syncrack-generated data (see Table 7.1). Remember, the model trained with accurate annotations obtained a precision of 80% and a recall of 78%. By training with noisy labels, the model obtained a precision of 55% and a recall of 85%. When trained with pseudo-labels generated by 5-nn voting, the model obtained a precision of 76% and a recall of 79%. The precision increased by 21% with respect to training with noisy labels. Although the recall decreased by 6%, it is still better than training with accurate annotations. In terms of F-score, our method based on pseudo-label generation achieved 77%. This score is no more than 2% below the 79% obtained by training with accurate annotations; this holds true for different label noise levels.

We also tested our method on a dataset composed by merging the CrackFor-

Table 7.1: Results obtained on Syncrack-generated data, using accurate annotations for evaluation.

Training \ Score	Precision	Recall	F-score
Training with accurate labels	80%	78%	79%
Training with pseudo-labels	76%	79%	77%
Training with noisy labels[a]	55%	85%	66%

[a] Label noise level 4 (see section 3.4).

Table 7.2: Results obtained on the test split of the CFD+Aigle-RN dataset, using manual annotations for supervised evaluation.

Score / Training	Precision	Recall	H_{crack}	H^2_{crack}	K-S
Training with manual annotations	71%	75%	4.34	8.07	0.633
Training with pseudo-labels	73%	66%	4.25	7.87	0.718

est and Aigle-RN datasets (real-life public datasets). By improving the quality of the predicted cracks, the expected behavior was preserving the precision while reducing the recall. The decrease of recall should be caused by having segmentations thinner than the annotations and, consequently, closer to the real crack width. This behavior was, indeed, captured by the unsupervised metrics (see Table 7.2).

By training with manual annotations, we obtained a precision of 71% and a recall of 75%. We generated pseudo-labels from the manual annotations using 5-nn voting; by training with these pseudo-labels, the precision increased to 73% and the recall decreased to 66%. Training with manual annotations, the first order and second order entropies were 4.34 and 8.07, respectively. When training with pseudo-labels, these scores improved to 4.25 and 7.87, respectively. The decrease of the entropies means that the regions segmented as cracks have a greater uniformity when training with pseudo-labels. Training with manual annotations, the obtained Kolmogorov-Smirnov score was 0.633. Training with pseudo-labels, this score improved to 0.718. The increase of the Kolmogorov-Smirnov score means that the regions segmented as cracks are less contaminated by background pixels when training with pseudo-labels.

The unsupervised scores, along with a qualitative analysis, showed that the apparent decrease of recall was due primarily to crack segmentations with reduced width. These thinner segmentations, obtained by training with pseudo-labels, exhibited a geometry closer to the visible cracks in the images, particularly in terms of crack width.

An improved crack segmentation based on transfer learning from synthetic data. We presented a second proposal to improve the predicted crack geometry, based on transfer learning: training a model with Syncrack-generated images and transferring the learned model to real images. To do this, we generated 3 Syncrack datasets with different crack segmentation difficulty levels. These datasets were generated to emulate the CrackForest dataset (particularly in terms of image size and crack width). By evaluating on CrackForest the different models trained with Syncrack, we observed that the models trained on Syncrack were transferable to real images.

As the difficulty level of Syncrack increased, the trained models became more conservative at discarding crack segments as false negatives. The predictions tended to become coarser as we increased the difficulty level. Nevertheless, the segmentations obtained by training with synthetic data were closer to the real crack geometries, with respect to training with real data. Here, we take only the model trained on the hard difficulty as reference (see Table 7.3). By training with manual annotations, the precision and recall were 68 and 76%, respectively.

Table 7.3: Results obtained on the test split of the CFD dataset, using manual annotations for supervised evaluation.

Score / Training images	Precision	Recall	H_{crack}	H^2_{crack}	K-S
CFD images (manual annotations)	68%	76%	4.33	8.20	0.570
Syncrak-generated images (accurate ground truths)	76%	45%	4.10	7.54	0.756

Training with hard Syncrack images, the precision increased to 76% and the recall decreased to 45%. Similarly to before, the unsupervised scores support that the apparent decrease of recall is mainly caused by producing predictions thinner than the excessively wide manual annotations. On one hand, the entropies improved: from 4.33 and 8.20, to 4.10 and 7.54, respectively. On the other hand, the Kolmogorov-Smirnov score improved too: from 0.570 to 0.756.

Complemented by a qualitative analysis, these scores showed that the model trained solely on synthetic data was transferable to real data and, furthermore, it provided more accurate segmentations in terms of crack width. With this improvement, we closed the topic of crack segmentation for post-construction lifetime monitoring.

7.1.2 Inline Characterization and Anomaly Detection in 3D Concrete Printing

The presence of local defects during construction based on 3D concrete printing can be identified by visual inspection. Specifically, the interlayer lines separating adjacent printed layers provide relevant information about the geometry of the layers and their possible deformations. In chapter 4, we introduce the experimental images acquired for our experiments and study the problem of interlayer line segmentation for inline monitoring.

The I3DCP dataset. For the experiments presented in this manuscript, we collected a dataset during a printing session by fixing a camera and a lamp to the extrusion nozzle. This setup allows obtaining images with a fixed point of view with respect to the extrusion zone; consequently, a local inline monitoring is possible. The camera position and orientation are fixed to obtain a lateral view of the printed piece, such that the interlayer lines are visible. We collected a total of 628 images, composing the Inline 3D Concrete Printing (I3DCP) dataset.

Interlayer line segmentation. To analyze the interlayer lines, the first step is to segment them. Similarly to cracks, these lines are thin, long elements that occupy a very small percentage in a noisy background. In the context of inline monitoring, with fresh material, the interlayer lines can take different aspects: black, bright or even not visible at all.

To provide the interlayer line segmentation, we used U-VGG19 as baseline model. However, this network is intended for supervised learning i.e. segmentation masks must be provided during training. To train our final model, we used a semi-supervised learning approach, beginning from an initial set of 32 manually

annotated images (used at the end as a test set). At the end of this method, we produced a total of 128 annotated images with the help of U-VGG19.

The final model was trained and fine-tuned using those 128 images. On the test set, this U-VGG19 trained for interlayer line segmentation obtained an F-score of 91%. The public release of I3DCP includes the segmentation maps used for training and testing. By analyzing the segmented interlayer lines, it is possible to characterize the observed region in search of anomalies.

A monitoring method for 3D concrete printing based on geometrical characterization and anomaly detection. Using the binary segmentation maps that locate the interlayer lines, we use image processing to determine the local geometry. In chapter 5, we proposed a method for layer characterization and anomaly detection based on geometrical analyses of the segmented interlayer lines.

Given an interlayer segmentation map (as the ones provided by U-VGG19), first we thin the regions predicted as interlayer lines to 1-pixel-width lines. Then, we perform a local measurement of the orientations and the curvatures of the resulting lines. We also estimate the local width of the observed layers as the distance between their two interlayer lines. Additionally, me measure the vertical distance between the extrusion nozzle and the line that serves as deposition surface. All these measurements are provided at pixel level.

This allows generating, per measurement type, a distribution of the values measured per pixel. These distributions, at the same time, serve as a summary of the analyzed image. By assigning a range of acceptable values per measurement, as defined by the operator, the distributions that contain high density values outside of the admissible ranges are a consequence of a defective process. This allows deciding if the analyzed image exhibits defects.

Consequently, the geometrical anomalies are defined and located as the regions in the analyzed image with measured values outside of the admissible ranges. With this approach, our method returns the location, nature and severity of the anomaly. With the analysis of the interlayer lines, we can detect geometrical anomalies; however, other type of anomalies can be identified by also analyzing the raw input image itself.

A monitoring method for 3D concrete printing based on textural characterization and anomaly detection. To the best of our knowledge, we are the first to propose a monitoring method for 3D concrete printing based on texture. This method, based on machine learning for texture classification, is presented in chapter 6.

The principle behind this method is that the water content of the extruded material is an indicator of potential issues (with impact in the structural performance). Since the excess or shortage of water is one of the factors that changes the visible texture of the printed layers, this texture is a good indicator of anomalies during printing. We analyze each layer independently and classify it, window-wise, either as *good* or as one of three defective cases: excessively fluid, excessively dry or containing superficial tearing.

We approached this as a multi-class, single-label, supervised problem. For our experiments, we created a new subset in the 3DCP dataset. This subset is composed of cropped windows, pre-processed to reduce the effect of lighting

and shadows. From this subset, we provide labels for 111 images with two-annotators consensus. Once the windows to classify are extracted, we perform feature extraction to train a classifier. Feature extraction is performed based on gray-level co-occurrence matrices and local binary patterns. The selected classifier was a small neural network with an input convolutional layer and a fully connected layer with 4 output neurons (returning the probability of each of the 4 proposed classes).

Evaluating with 5-fold cross-validation, our method obtained a macro-averaged F-score of 94%. To analyze a raw image, our method consists of: segmenting the independent layers with the help of U-VGG19, pre-processing the image to reduce the effect of lighting and shadows, dividing each layer into windows to classify, extracting the features from each window to create an input vector, classifying the input vector, and assigning to each window in the image both the label and the confidence as provided by the classifier.

The anomalies are then defined as the regions with a class other than good. This setup allows returning the location and nature of the anomaly, as well as the confidence on the decision.

With this method, we close the topic of inline characterization and anomaly detection for construction based on 3D concrete printing. In the next section, we close this manuscript by summarizing diverse lines of research that could extend the work presented in this manuscript.

7.2 Future Research Lines

Regarding the topic of thin object segmentation, either cracks, interlayer lines or other similar objects, one inherent problem is the generation of manual annotations for supervised learning. The labeling task could greatly benefit from active and self-learning approaches, in which models trained from few annotated images are used as labelers to iteratively increase the amount of annotated images. The initial annotations could be obtained from simple methods e.g. threshold segmentation for cracks and border detection for interlayer lines. Such models would be progressively improved by the selection and correction of annotations made by oracles (e.g. human annotators). On this line, rather than focusing on increasing model complexity to reduce the loss on the annotated images, special attention should be given to learning in the presence of inaccurate labels.

In this work, we presented first approaches to the above-mentioned problem. Our efforts were centered around pixel-accurate segmentation from entire images. However, this kind of problem can be easily divided into two consecutive sub-problems: object location (rough segmentation) and pixel-accurate segmentation (fine segmentation). This kind of approach would allow isolating small regions containing the objects of interest for a posterior refinement of the segmentation that would accurately describe the object's geometry.

On the other side, with the help of synthetic image generation, we can avoid the problem of inaccurate annotations prior to training. We showed a first approach through the use of Syncrack. Although the models trained with Syncrack-generated images are transferable to real images, the distributions generated

by Syncrack are different to the ones of real images. In this sense, although the synthetic images are perceptually similar to real ones, the synthetic image generation approach could greatly benefit from more sophisticated generators. One example is the use of generative adversarial networks. Although training this type of networks is complicated, it could help to create photo-realistic background textures, as well as cracks (both in terms of geometry and appearance).

To evaluate the improvement of the predictions in terms of crack width, we proposed a set of unsupervised metrics directly related to the supervised precision. Although these metrics allow measuring if the produced segmentations are wider than the visible crack, they are not very sensitive to crack pixels being mislabeled as background. Because of the heavy class imbalance associated to crack segmentation, even mislabeling all the crack pixels as background will not produce relevant changes in the intensity distribution of pixels segmented as background. Further work is needed to propose or develop new metrics that allow objectively measuring the overall quality of thin object segmentation, without relying on inaccurate annotations.

Regarding the characterization and anomaly detection of 3D concrete printing, the proposed method can be extended to provide quality reports of the whole printing process. This requires a synchronization, either to acquire frames without common regions (providing a single analysis per region) or to evaluate the evolution of the measured values over time (providing an additional dimension for the analysis of the process). The first approach is simple, but it ignores the deformations that occur over time e.g. a progressive crushing of layers under the increasing weight of the layers extruded above. The second approach provides a more complete report, but it implies an additional non-trivial difficulty: providing a line-to-line correspondence between frames under the assumption that the mesh defined by the interlayer lines is prone to change over time. To fully exploit this approach, a global perspective of the printed piece would be more useful than the local approach presented in this manuscript; although it is computationally more expensive, our method can be easily extrapolated to the global perspective.

Regarding the specific topic of textural characterization, the main area of opportunity is the generalization of the proposed method. Currently, feature extraction is performed through gray-level co-occurrence matrices and local binary patterns using user-defined offsets. While these features showed to be effective for texture classification of extruded concrete, the proper offsets will change depending on the camera setup. The proposed method would benefit from automatic offset selection. Nonetheless, this method was developed under the constraint of a limited amount of labeled texture images. Increasing the amount of labeled data leads to the use of deep learning, with strategies such as channel attention modules to directly extend the work presented in this thesis (with each channel corresponding to a single offset). Furthermore, the problem could be approached with alternative strategies such as the use of fully-convolutional networks (skipping the manual feature extraction). The difficulty with the latter is that pixel-wise texture annotation is not really feasible for human annotators. Regarding this, the work on texture classification could be extended to segmentation in presence of inaccurate annotations (starting from window-wise annota-

tions). The annotation process could also be simplified through machine-assisted labeling with active learning loops.

In this direction, it is relevant to notice that the classes proposed in this work are not by any means a complete list of all the possible defects. Consequently, the trained model can exhibit weird behaviors in the presence of textures containing other types of defects. An alternative option could be approaching the task as a one-class classification problem rather than a multi-class one: by learning only the *good* class. This approach still allows detecting anomalies in the extruded layers, albeit without classifying them. This is useful in scenarios where the only requirement is to detect when the printing process does not meet the expected quality requirements.

Nonetheless, a classification of the anomalies provides relevant advantages. Our characterization and anomaly detection method can be used to adjust the printing process in real time. With the current work, the next immediate step is that our monitoring system reports the nature and severity of detected anomalies to a human operator. This allows the expert to make corrective adjustments to the printing parameters. Future work can extend this to a closed-loop control system, in which the nature and severity of the anomalies are used as feedback to an automated decision system. The corrective actions could stem from a rule-based expert system, vector-to-action dictionaries, supervised machine learning models, etc.

List of research communications

Journals

[1] R. Rill-García, E. Dokladalova, and P. Dokládal. Pixel-accurate road crack detection in presence of inaccurate annotations. *Neurocomputing*, 480:1–13, 2022e. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2022.01.051. URL https://www.sciencedirect.com/science/article/pii/S0925231222000728

[2] R. Rill-García, E. Dokladalova, P. Dokládal, J.-F. Caron, R. Mesnil, P. Margerit, and M. Charrier. Inline monitoring of 3d concrete printing using computer vision. *Additive Manufacturing*, 60:103175, 2022g. ISSN 2214-8604. doi: https://doi.org/10.1016/j.addma.2022.103175. URL https://www.sciencedirect.com/science/article/pii/S2214860422005644

International Conferences

[3] R. Rill-García, E. Dokladalova, and P. Dokládal. Syncrack: Improving pavement and concrete crack detection through synthetic data generation. In *Proceedings of the 17th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 4: VISAPP*, pages 147–158. INSTICC, SciTePress, 2022f. ISBN 978-989-758-555-5. doi: 10.5220/0010837300003124. URL https://www.scitepress.org/Link.aspx?doi=10.5220/0010837300003124

[4] R. Rill-García, E. Dokladalova, and P. Dokládal. Poster: Improving pavement and concrete crack detection through synthetic data generation. In *LatinX in CV Research Workshop at CVPR 2022*, New Orleans, United States of America, June 2022d

Local communications

[5] R. Rill-García, E. Dokladalova, and P. Dokládal. Deep crack segmentation on pavement surfaces. In *44ème journée ISS France*, Online, France, February 2021b

[6] R. Rill-García, E. Dokladalova, and P. Dokládal. Detection of cracks in presence of inaccurate labels. In *Intéractions Physique–Images*, Online, France, June 2021a

[7] R. Rill-García, E. Dokladalova, and P. Dokládal. Syncrack: Improving pavement and concrete crack detection through synthetic data generation. In *45ème journée ISS*, Online, France, February 2022c

[8] R. Rill-García, E. Dokladalova, and P. Dokládal. Machine vision for closed loop concrete printing. In *IA et monitoring des bâtiments et construction*, Champs-sur-Marne, France, April 2022a

[9] R. Rill-García, E. Dokladalova, and P. Dokládal. Computer vision methods for 3d concrete printing process monitoring. In *DiXite au FUTURE: Digital Construction Site, Bilan et perspectives*, Champs-sur-Marne, France, October 2022b

Appendix

A. Code repositories

This section contains links to repositories containing the pieces of code used to obtain the results presented in this document and the publications listed in List of research communications.

Code for chapter 2 and chapter 3

- U-VGG19 conceived for crack detection: https://github.com/Sutadasuto/uvgg19_crack_detection
- Crack detection in presence of inaccurate labels, inspired by weakly supervised learning, as presented for Neurocomputing (Rill-García et al. [2022e]): https://github.com/Sutadasuto/weak_supervision_crack_detection
- Crack detection in presence of inaccurate labels, inspired by weakly supervised learning, as used for this document: https://github.com/Sutadasuto/weak_supervision_crack_detection_thesis
- The Syncrack generator: https://github.com/Sutadasuto/syncrack_generator
- Crack detection using Syncrack as presented for VISAPP (Rill-García et al. [2022f]): https://github.com/Sutadasuto/syncrack_crack_detection

Code for chapter 4, chapter 5 and chapter 6

- U-VGG19 conceived for 3D printed concrete interlayer line segmentation: https://github.com/Sutadasuto/concrete_layer_detection
- Code for single-frame analysis as used for Additive Manufacturing (Rill-García et al. [2022g]): https://github.com/Sutadasuto/3dcp_cv_monitoring
- Code for inline monitoring from camera stream (camera can be substituted by image folder). It extends the code presented for Additive Manufacturing (Rill-García et al. [2022g]): https://github.com/Sutadasuto/3dcp_cv_monitoring_inline

- Code for training the texture classification method: https://github.com/Sutadasuto/concrete_texture_analysis
- Code used for hyper-parameter selection prior to final training (for texture classification): https://github.com/Sutadasuto/concrete_texture_analysis_hyperopt

B. Dataset repositories

This section contains links to repositories containing the datasets generated from this work.

- Syncrack, as used for Neurocomputing (Rill-García et al. [2022e]): https://github.com/Sutadasuto/syncrack_generator/tree/demo
- I3DCP, as presented for Additive Manufacturing (Rill-García et al. [2022g]): https://github.com/Sutadasuto/I3DCP/

References

M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL https://www.tensorflow.org/. Software available from tensorflow.org. 27

I. Abdel-Qader, O. Abudayyeh, and M. E. Kelly. Analysis of edge-detection techniques for crack identification in bridges. *Journal of Computing in Civil Engineering*, 17(4):255–263, 2003. 18

J. A. Acosta, J. L. Figueroa, and R. L. Mullen. Low-cost video image processing system for evaluating pavement surface distress. *Transportation research record*, (1348), 1992. 17

D. Ai, G. Jiang, L. Siew Kei, and C. Li. Automatic Pixel-Level Pavement Crack Detection Using Information of Multi-Scale Neighborhoods. *IEEE Access*, 6: 24452–24463, 2018. ISSN 2169-3536. doi: 10.1109/ACCESS.2018.2829347. Conference Name: IEEE Access. 22, 34

A. Akagic, E. Buza, S. Omanovic, and A. Karabegovic. Pavement crack detection using otsu thresholding for image segmentation. In *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 1092–1097. IEEE, 2018. 17

E. Aldea and S. L. Hégarat-Mascle. Robust crack detection for unmanned aerial vehicles inspection in an a-contrario decision framework. *Journal of Electronic Imaging*, 24(6):061119, 2015. doi: 10.1117/1.JEI.24.6.061119. URL https://doi.org/10.1117/1.JEI.24.6.061119. 18

All3DP Pro. 8 Biggest Companies Building 3D Printed Houses. *All3DP Pro*, Sept. 2021. URL https://all3dp.com/2/best-companies-building-3d-printed-houses/. Available at https://all3dp.com/2/best-companies-building-3d-printed-houses/. 2

R. Amhaz, S. Chambon, J. Idier, and V. Baltazart. Automatic Crack Detection on Two-Dimensional Pavement Images: An Algorithm Based on Minimal Path Selection. *IEEE Transactions on Intelligent Transportation Systems*, 17(10):

2718–2729, Oct. 2016. ISSN 1558-0016. doi: 10.1109/TITS.2015.2477675. Conference Name: IEEE Transactions on Intelligent Transportation Systems. 18, 25, 83

J. Archez, S. Maitenaz, L. Demont, M. Charrier, R. Mesnil, N. Texier-Mandoki, X. Bourbon, S. Rossignol, and J.-F. Caron. Strategy to shape, on a half-meter scale, a geopolymer composite structure by additive manufacturing. *Open Ceramics*, 5:100071, 2021. 87

S. Bang, S. Park, H. Kim, and H. Kim. Encoder–decoder network for pixel-level road crack detection in black-box images. *Computer-Aided Civil and Infrastructure Engineering*, 34(8):713–727, 2019. ISSN 1467-8667. doi: https://doi.org/10.1111/mice.12440. URL http://onlinelibrary.wiley.com/doi/abs/10.1111/mice.12440. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/mice.12440. 23

F. Barbosa, J. Woetzel, and J. Mischke. Reinventing construction: A route of higher productivity. Technical report, McKinsey Global Institute, 2017. 1

J. L. Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517, sep 1975. ISSN 0001-0782. doi: 10.1145/361002.361007. URL https://doi.org/10.1145/361002.361007. 49

J. Bergstra, D. Yamins, and D. Cox. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In S. Dasgupta and D. McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 115–123, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR. URL https://proceedings.mlr.press/v28/bergstra13.html. 115

S. Bhat, S. Naik, M. Gaonkar, P. Sawant, S. Aswale, and P. Shetgaonkar. A Survey On Road Crack Detection Techniques. In *2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE)*, pages 1–6, Feb. 2020. doi: 10.1109/ic-ETITE47903.2020.67. 33

R. S. Boyer and J. S. Moore. Mjrty—a fast majority vote algorithm. In *Automated Reasoning*, pages 105–117. Springer, 1991. doi: 10.1007/978-94-011-3488-0_5. 49

C. E. Brodley and M. A. Friedl. Identifying Mislabeled Training Data. *Journal of Artificial Intelligence Research*, 11:131–167, Aug. 1999. ISSN 1076-9757. doi: 10.1613/jair.606. URL https://www.jair.org/index.php/jair/article/view/10238. 36, 49, 50

R. Buswell, W. Leal de Silva, S. Jones, and J. Dirrenberger. 3D printing using concrete extrusion: A roadmap for research. *Cement and Concrete Research*, 112:37–49, Oct. 2018. ISSN 00088846. doi: 10.1016/j.cemconres.2018.05.006. URL https://linkinghub.elsevier.com/retrieve/pii/S0008884617311924. 3, 4

R. Buswell, P. Kinnell, J. Xu, N. Hack, H. Kloft, M. Maboudi, M. Gerke, P. Massin, G. Grasser, R. Wolfs, and F. Bos. Inspection Methods for 3D Concrete Printing. In F. P. Bos, S. S. Lucas, R. J. Wolfs, and T. A. Salet, editors, *Second RILEM International Conference on Concrete and Digital Fabrication*, RILEM Bookseries, pages 790–803, Cham, 2020. Springer International Publishing. ISBN 978-3-030-49916-7. doi: 10.1007/978-3-030-49916-7_78. 1

T. I. Cannings, Y. Fan, and R. J. Samworth. Classification with imperfect training labels. *arXiv preprint arXiv:1805.11505*, 2018. 36, 37, 48

P. Carneau, R. Mesnil, O. Baverel, and N. Roussel. Layer pressing in concrete extrusion-based 3d-printing: Experiments and analysis. *Cement and Concrete Research*, page 106741, 2022. ISSN 0008-8846. doi: https://doi.org/10.1016/j.cemconres.2022.106741. URL https://www.sciencedirect.com/science/article/pii/S0008884622000321. 87

H. Cheng, J.-R. Chen, C. Glazier, and Y. Hu. Novel approach to pavement cracking detection based on fuzzy set theory. *Journal of Computing in Civil Engineering*, 13(4):270–280, 1999. 17

R. D. Cook. Detection of influential observation in linear regression. *Technometrics*, 19(1):15–18, 1977. doi: 10.1080/00401706.1977.10489493. URL https://doi.org/10.1080/00401706.1977.10489493. 36

R. D. Cook and S. Weisberg. *Residuals and influence in regression*. New York: Chapman and Hall, 1982. 36

E. Coquelle, J.-L. Gautier, and P. Dokládal. Automatic assessment of a road surface condition. In *7th Symposium on Pavement Surface Characteristics, Surf*, Norfolk, Virginia, 2012. 15

A. Cord and S. Chambon. Automatic road defect detection by textural pattern recognition based on adaboost. *Computer-Aided Civil and Infrastructure Engineering*, 27(4):244–259, 2012. 19

O. Davtalab, A. Kazemian, and B. Khoshnevis. Perspectives on a BIM-integrated software platform for robotic construction through Contour Crafting. *Automation in Construction*, 89:13–23, May 2018. ISSN 09265805. doi: 10.1016/j.autcon.2018.01.006. URL https://linkinghub.elsevier.com/retrieve/pii/S0926580517307975. 6

O. Davtalab, A. Kazemian, X. Yuan, and B. Khoshnevis. Automated inspection in robotic additive manufacturing using deep learning for layer deformation detection. *Journal of Intelligent Manufacturing*, Oct. 2020. ISSN 1572-8145. doi: 10.1007/s10845-020-01684-w. URL https://doi.org/10.1007/s10845-020-01684-w. 6, 7, 74, 83, 85, 88, 90, 91

C. De Boor and C. De Boor. *A practical guide to splines*, volume 27. springer-verlag New York, 1978. 92

P. Dokládal. Statistical threshold selection for path openings to detect cracks. In *Mathematical Morphology and Its Applications to Signal and Image Processing*, volume 10225, pages 369–380. Springer International Publishing, May 2017. doi: 10.1007/978-3-319-57240-6_30. URL https://hal-mines-paristech.archives-ouvertes.fr/hal-01478089. 18

C. Duncan. noise. https://github.com/caseman/noise, 2018. 38

M. Eisenbach, R. Stricker, D. Seichter, K. Amende, K. Debes, M. Sesselmann, D. Ebersbach, U. Stoeckert, and H. Gross. How to get pavement distress detection ready for deep learning? A systematic approach. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 2039–2047, May 2017. doi: 10.1109/IJCNN.2017.7966101. ISSN: 2161-4407. 19

U. Escalona, F. Arce, E. Zamora, and J. H. Sossa Azuela. Fully Convolutional Networks for Automatic Pavement Crack Segmentation. *Computación y Sistemas*, 23(2):451–460–460, June 2019. ISSN 2007-9737. doi: 10.13053/cys-23-2-3047. URL https://www.cys.cic.ipn.mx/ojs/index.php/CyS/article/view/3047. Number: 2. 20, 23, 33

V. Esnault, A. Labyad, M. Chantin, and F. Toussaint. Experience in online modification of rheology and strength acquisition of 3d printable mortars. In *RILEM International Conference on Concrete and Digital Fabrication*, pages 24–38. Springer, 2018. 72

Z. Fan, C. Li, Y. Chen, P. D. Mascio, X. Chen, G. Zhu, and G. Loprencipe. Ensemble of Deep Convolutional Neural Networks for Automatic Pavement Crack Detection and Measurement. *Coatings*, 10(2):152, Feb. 2020a. doi: 10.3390/coatings10020152. URL https://www.mdpi.com/2079-6412/10/2/152. Number: 2 Publisher: Multidisciplinary Digital Publishing Institute. 33

Z. Fan, C. Li, Y. Chen, J. Wei, G. Loprencipe, X. Chen, and P. Di Mascio. Automatic Crack Detection on Road Pavements Using Encoder-Decoder Architecture. *Materials*, 13(13):2960, Jan. 2020b. doi: 10.3390/ma13132960. URL https://www.mdpi.com/1996-1944/13/13/2960. Number: 13 Publisher: Multidisciplinary Digital Publishing Institute. 21

J. Fang, B. Qu, and Y. Yuan. Distribution equalization learning mechanism for road crack detection. *Neurocomputing*, 424:193–204, Feb. 2021. ISSN 09252312. doi: 10.1016/j.neucom.2019.12.057. URL https://linkinghub.elsevier.com/retrieve/pii/S0925231219317667. 21, 29

J. L. Fleiss. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378, 1971. 120

B. Frenay and M. Verleysen. Classification in the Presence of Label Noise: A Survey. *IEEE Transactions on Neural Networks and Learning Systems*, 25(5):845–869, May 2014. ISSN 2162-2388. doi: 10.1109/TNNLS.2013.2292894. Conference Name: IEEE Transactions on Neural Networks and Learning Systems. 35, 37, 48

Y. Gao and K. M. Mosalam. Deep transfer learning for image-based structural damage recognition. *Computer-Aided Civil and Infrastructure Engineering*, 33 (9):748–768, 2018. 15

Z. Gao, B. Peng, T. Li, and C. Gou. Generative Adversarial Networks for Road Crack Image Segmentation. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, July 2019. doi: 10.1109/IJCNN.2019.8851910. ISSN: 2161-4407. 21, 29, 33

L. Gatys, A. S. Ecker, and M. Bethge. Texture Synthesis Using Convolutional Neural Networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 262–270. Curran Associates, Inc., 2015. URL http://papers.nips.cc/paper/5633-texture-synthesis-using-convolutional-neural-networks.pdf. 23

Z. Guo and R. W. Hall. Parallel thinning with two-subiteration algorithms. *Commun. ACM*, 32(3):359–373, mar 1989. ISSN 0001-0782. doi: 10.1145/62065.62074. URL https://doi.org/10.1145/62065.62074. 91

I. Hager, A. Golonka, and R. Putanowicz. 3D Printing of Buildings and Building Components as the Future of Sustainable Construction? *Procedia Engineering*, 151:292–299, Jan. 2016. ISSN 1877-7058. doi: 10.1016/j.proeng.2016.07.357. URL https://www.sciencedirect.com/science/article/pii/S1877705816317453. 1, 2

D. Hao, L. Zhang, J. Sumkin, A. Mohamed, and S. Wu. Inaccurate labels in weakly supervised deep learning: Automatic identification and correction and their impact on classification performance. *IEEE Journal of Biomedical and Health Informatics*, pages 1–1, 2020. ISSN 2168-2208. doi: 10.1109/JBHI.2020.2974425. Conference Name: IEEE Journal of Biomedical and Health Informatics. 36

R. M. Haralick, K. Shanmugam, and I. Dinstein. Textural features for image classification. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-3 (6):610–621, 1973. doi: 10.1109/TSMC.1973.4309314. 45, 112

Z. Jin, Z. Zhang, and G. X. Gu. Autonomous in-situ correction of fused deposition modeling printers using computer vision and deep learning. *Manufacturing Letters*, 22:11–15, 2019. ISSN 2213-8463. doi: https://doi.org/10.1016/j.mfglet.2019.09.005. URL https://www.sciencedirect.com/science/article/pii/S2213846319300847. 108, 109

A. Kazemian, X. Yuan, O. Davtalab, and B. Khoshnevis. Computer vision for real-time extrusion quality monitoring and control in robotic construction. *Automation in Construction*, 101:92–98, May 2019. ISSN 09265805. doi: 10.1016/j.autcon.2019.01.022. URL https://linkinghub.elsevier.com/retrieve/pii/S0926580518307751. 6, 7, 88, 90

B. Khoshnevis and D. Hwang. Contour crafting. In *Rapid Prototyping*, pages 221–251. Springer, 2006. 1

B. Kim and S. Cho. Automated Vision-Based Detection of Cracks on Concrete Surfaces Using a Deep Learning Technique. *Sensors*, 18(10):3452, Oct. 2018. doi: 10.3390/s18103452. URL https://www.mdpi.com/1424-8220/18/10/3452. Number: 10 Publisher: Multidisciplinary Digital Publishing Institute. 19

K. Kirschke and S. Velinsky. Histogram-based approach for automated pavement-crack sensing. *Journal of Transportation Engineering*, 118(5):700–710, 1992. 17

P. W. Koh and P. Liang. Understanding black-box predictions via influence functions. *arXiv preprint arXiv:1703.04730*, 2017. 36

M. Koziarski, B. Krawczyk, and M. Woźniak. Radial-Based oversampling for noisy imbalanced data classification. *Neurocomputing*, 343:19–33, May 2019. ISSN 09252312. doi: 10.1016/j.neucom.2018.04.089. URL https://linkinghub.elsevier.com/retrieve/pii/S0925231219301596. 37

J. König, M. David Jenkins, P. Barrie, M. Mannion, and G. Morison. A Convolutional Neural Network for Pavement Surface Crack Segmentation Using Residual Connections and Attention Gating. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 1460–1464, Sept. 2019. doi: 10.1109/ICIP.2019.8803060. ISSN: 2381-8549. 21

W. Li, M. Zhang, and D. Chen. Fundus retinal blood vessel segmentation based on active learning. In *2020 International Conference on Computer Information and Big Data Applications (CIBDA)*, pages 264–268, 2020. doi: 10.1109/CIBDA50819.2020.00066. 75

G. Liang, X. Wang, Y. Zhang, and N. Jacobs. Weakly-Supervised Self-Training for Breast Cancer Localization*. In *2020 42nd Annual International Conference of the IEEE Engineering in Medicine Biology Society (EMBC)*, pages 1124–1127, July 2020. doi: 10.1109/EMBC44109.2020.9176617. ISSN: 2694-0604. 36, 49

H. Lindemann, R. Gerbers, S. Ibrahim, F. Dietrich, E. Herrmann, K. Dröder, A. Raatz, and H. Kloft. Development of a Shotcrete 3D-Printing (SC3DP) Technology for Additive Manufacturing of Reinforced Freeform Concrete Structures. In T. Wangler and R. J. Flatt, editors, *First RILEM International Conference on Concrete and Digital Fabrication – Digital Concrete 2018*, volume 19, pages 287–298. Springer International Publishing, Cham, 2019. ISBN 978-3-319-99518-2 978-3-319-99519-9. doi: 10.1007/978-3-319-99519-9_27. URL http://link.springer.com/10.1007/978-3-319-99519-9_27. Series Title: RILEM Bookseries. 6, 7, 88, 90, 94

C. Liu, A. C. C. Law, D. Roberson, and Z. J. Kong. Image analysis-based closed loop quality control for additive manufacturing with fused filament fabrication. *Journal of Manufacturing Systems*, 51:75–86, Apr. 2019a. ISSN 0278-6125. doi: 10.1016/j.jmsy.2019.04.002. URL https://www.sciencedirect.com/science/article/pii/S0278612518304060. 109

C. Liu, A. C. C. Law, D. Roberson, and Z. J. Kong. Image analysis-based closed loop quality control for additive manufacturing with fused filament fabrication. *Journal of Manufacturing Systems*, 51:75–86, 2019b. ISSN 0278-6125. doi: https://doi.org/10.1016/j.jmsy.2019.04.002. URL https://www.sciencedirect.com/science/article/pii/S0278612518304060. 108, 112

Z. Liu, Y. Cao, Y. Wang, and W. Wang. Computer vision-based concrete crack detection using U-net fully convolutional networks. *Automation in Construction*, 104:129–139, Aug. 2019c. ISSN 0926-5805. doi: 10.1016/j.autcon.2019.04.005. URL https://www.sciencedirect.com/science/article/pii/S0926580519301244. 21

Z. Liu, M. Li, Y. W. D. Tay, Y. Weng, T. N. Wong, and M. J. Tan. Rotation nozzle and numerical simulation of mass distribution at corners in 3d cementitious material printing. *Additive Manufacturing*, 34:101190, 2020. 71

E. Lloret, A. R. Shahab, M. Linus, R. J. Flatt, F. Gramazio, M. Kohler, and S. Langenberg. Complex concrete structures: Merging existing casting techniques with digital fabrication. *Computer-Aided Design*, 60:40–49, Mar. 2015. ISSN 0010-4485. doi: 10.1016/j.cad.2014.02.011. URL https://www.sciencedirect.com/science/article/pii/S001044851400044X. 6

V. Mandal, L. Uong, and Y. Adu-Gyamfi. Automated Road Crack Detection Using Deep Convolutional Neural Networks. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 5212–5215, Dec. 2018. doi: 10.1109/BigData.2018.8622327. 19

C. R. Maurer, R. Qi, and V. Raghavan. A linear time algorithm for computing exact euclidean distance transforms of binary images in arbitrary dimensions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(2):265–270, 2003. 93

D. Mazzini, P. Napoletano, F. Piccoli, and R. Schettini. A Novel Approach to Data Augmentation for Pavement Distress Segmentation. *Computers in Industry*, 121:103225, Oct. 2020. ISSN 0166-3615. doi: 10.1016/j.compind.2020.103225. URL http://www.sciencedirect.com/science/article/pii/S0166361519310516. 23

S. A. O. Nair, G. Sant, and N. Neithalath. Mathematical morphology-based point cloud analysis techniques for geometry assessment of 3D printed concrete elements. *Additive Manufacturing*, page 102499, Nov. 2021. ISSN 2214-8604. URL https://www.sciencedirect.com/science/article/pii/S2214860421006461. 7, 90

T. Ojala, M. Pietikainen, and D. Harwood. Performance evaluation of texture measures with classification based on kullback discrimination of distributions. In *Proceedings of 12th International Conference on Pattern Recognition*, volume 1, pages 582–585 vol.1, 1994. doi: 10.1109/ICPR.1994.576366. 114

A. Oleff, B. Küster, M. Stonis, and L. Overmeyer. Process monitoring for material extrusion additive manufacturing: a state-of-the-art review. *Progress in Additive Manufacturing*, pages 1–26, 2021. 8, 108

H. Oliveira and P. L. Correia. Automatic road crack segmentation using entropy and image dynamic thresholding. In *2009 17th European Signal Processing Conference*, pages 622–626. IEEE, 2009. 17

H. Oliveira and P. L. Correia. CrackIT — An image processing toolbox for crack detection and characterization. In *2014 IEEE International Conference on Image Processing (ICIP)*, pages 798–802, 2014. doi: 10.1109/ICIP.2014.7025160. 17, 18

S. M. Omohundro. *Five balltree construction algorithms*. International Computer Science Institute Berkeley, 1989. 49

N. R. Pal and D. Bhandari. Image thresholding: Some new techniques. *Signal Processing*, 33(2):139–158, Aug. 1993. ISSN 01651684. doi: 10.1016/0165-1684(93)90107-L. URL https://linkinghub.elsevier.com/retrieve/pii/016516849390107L. 44

B. Panda, J. H. Lim, N. A. Noor Mohamed, S. C. Paul, Y. W. D. Tay, and M. J. Tan. Automation of robotic concrete printing using feedback control system. In *Proceedings of the 34th International Symposium on Automation and Robotics in Construction (ISARC)*, pages 276–280, Taipei, Taiwan, July 2017. Tribun EU, s.r.o., Brno. ISBN 978-80-263-1371-7. doi: 10.22260/ISARC2017/0037. 6

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. 49

Peri. 3d construction printing, 2020. URL https://www.peri.com/en/business-segments/3d-construction-printing.html. Available at https://www.peri.com/en/business-segments/3d-construction-printing.html. 2

A. L. Petsiuk and J. M. Pearce. Open source computer vision-based layer-wise 3d printing analysis. *Additive Manufacturing*, 36:101473, 2020. ISSN 2214-8604. doi: https://doi.org/10.1016/j.addma.2020.101473. URL https://www.sciencedirect.com/science/article/pii/S2214860420308459. 108, 110

R. Rill-García, E. Dokladalova, and P. Dokládal. Detection of cracks in presence of inaccurate labels. In *Intéractions Physique–Images*, Online, France, June 2021a.

R. Rill-García, E. Dokladalova, and P. Dokládal. Deep crack segmentation on pavement surfaces. In *44ème journée ISS France*, Online, France, February 2021b.

R. Rill-García, E. Dokladalova, and P. Dokládal. Machine vision for closed loop concrete printing. In *IA et monitoring des bâtiments et construction*, Champs-sur-Marne, France, April 2022a.

R. Rill-García, E. Dokladalova, and P. Dokládal. Computer vision methods for 3d concrete printing process monitoring. In *DiXite au FUTURE: Digital Construction Site, Bilan et perspectives*, Champs-sur-Marne, France, October 2022b.

R. Rill-García, E. Dokladalova, and P. Dokládal. Syncrack: Improving pavement and concrete crack detection through synthetic data generation. In *45ème journée ISS*, Online, France, February 2022c.

R. Rill-García, E. Dokladalova, and P. Dokládal. Poster: Improving pavement and concrete crack detection through synthetic data generation. In *LatinX in CV Research Workshop at CVPR 2022*, New Orleans, United States of America, June 2022d.

R. Rill-García, E. Dokladalova, and P. Dokládal. Pixel-accurate road crack detection in presence of inaccurate annotations. *Neurocomputing*, 480:1–13, 2022e. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2022.01.051. URL https://www.sciencedirect.com/science/article/pii/S0925231222000728. 9, 10, 135, 136

R. Rill-García, E. Dokladalova, and P. Dokládal. Syncrack: Improving pavement and concrete crack detection through synthetic data generation. In *Proceedings of the 17th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 4: VISAPP*, pages 147–158. INSTICC, SciTePress, 2022f. ISBN 978-989-758-555-5. doi: 10.5220/0010837300003124. URL https://www.scitepress.org/Link.aspx?doi=10.5220/0010837300003124. 9, 10, 135

R. Rill-García, E. Dokladalova, P. Dokládal, J.-F. Caron, R. Mesnil, P. Margerit, and M. Charrier. Inline monitoring of 3d concrete printing using computer vision. *Additive Manufacturing*, 60:103175, 2022g. ISSN 2214-8604. doi: https://doi.org/10.1016/j.addma.2022.103175. URL https://www.sciencedirect.com/science/article/pii/S2214860422005644. 11, 135, 136

O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015. 20

B. Settles. Active learning literature survey. 2009. 75

H. Shen, W. Sun, and J. Fu. Multi-view online vision detection based on robot fused deposit modeling 3d printing technology. *Rapid Prototyping Journal*, 2018. 112

V. S. Sheng, F. Provost, and P. G. Ipeirotis. Get another label? improving data quality and data mining using multiple, noisy labelers. In *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data*

mining - KDD 08, page 614, Las Vegas, Nevada, USA, 2008. ACM Press. ISBN 978-1-60558-193-4. doi: 10.1145/1401890.1401965. URL http://dl.acm.org/citation.cfm?doid=1401890.1401965. 36, 49

T. Shi, N. Boutry, Y. Xu, and T. Géraud. Local intensity order transformation for robust curvilinear object segmentation. *IEEE Transactions on Image Processing*, 31:2557–2569, 2022. doi: 10.1109/TIP.2022.3155954. 22

Y. Shi, L. Cui, Z. Qi, F. Meng, and Z. Chen. Automatic Road Crack Detection Using Random Structured Forests. *IEEE Transactions on Intelligent Transportation Systems*, 17(12):3434–3445, Dec. 2016. ISSN 1558-0016. doi: 10.1109/TITS.2016.2552248. Conference Name: IEEE Transactions on Intelligent Transportation Systems. 16, 19, 25, 83

E. Shojaei Barjuei, E. Courteille, D. Rangeard, F. Marie, and A. Perrot. Real-time vision-based control of industrial manipulators for layer-width setting in concrete 3d printing applications. *Advances in Industrial and Manufacturing Engineering*, 5:100094, 2022. ISSN 2666-9129. doi: https://doi.org/10.1016/j.aime.2022.100094. URL https://www.sciencedirect.com/science/article/pii/S2666912922000228. 88, 90

K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 23

N. V. Smirnov. On the estimation of the discrepancy between empirical curves of distribution for two independent samples. *Bull. Math. Univ. Moscou*, 2(2): 3–14, 1939. 45

P. Soille. *Erosion and Dilation*, pages 63–103. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004. ISBN 978-3-662-05088-0. doi: 10.1007/978-3-662-05088-0_3. 41

M. Sun, R. Guo, J. Zhu, and W. Fan. Roadway Crack Segmentation Based on an Encoder-decoder Deep Network with Multi-scale Convolutional Blocks. In *2020 10th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 0869–0874, Jan. 2020. doi: 10.1109/CCWC47524.2020.9031213. 21, 25, 26, 29, 33

J. A. Sáez, J. Luengo, J. Stefanowski, and F. Herrera. Smote–ipf: Addressing the noisy and borderline examples problem in imbalanced classification by a re-sampling method with filtering. *Information Sciences*, 291:184–203, 2015. ISSN 0020-0255. doi: https://doi.org/10.1016/j.ins.2014.08.051. URL https://www.sciencedirect.com/science/article/pii/S0020025514008561. 37

N. Tanaka and K. Uematsu. A crack detection method in road surface images using morphology. *MVA*, 98:17–19, 1998. 18

J. Tang and Y. Gu. Automatic crack detection and segmentation using a hybrid algorithm for road distress analysis. In *2013 IEEE international conference on systems, man, and cybernetics*, pages 3026–3030. IEEE, 2013. 18

J. Vandoni, S. Le Hégarat-Mascle, and E. Aldea. Crack detection based on a marked point process model. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 3933–3938, 2016. doi: 10.1109/ICPR.2016.7900249. 18

R. J. M. Wolfs, F. P. Bos, E. C. F. van Strien, and T. A. M. Salet. A Real-Time Height Measurement and Feedback System for 3D Concrete Printing. In D. Hordijk and M. Luković, editors, *High Tech Concrete: Where Technology and Engineering Meet*, pages 2474–2483. Springer International Publishing, Cham, 2018. ISBN 978-3-319-59470-5 978-3-319-59471-2. doi: 10.1007/978-3-319-59471-2_282. URL http://link.springer.com/10.1007/978-3-319-59471-2_282. 6, 7, 88, 90, 94

S. Woo, J. Park, J.-Y. Lee, and I. S. Kweon. Cbam: Convolutional block attention module. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018. 120

L. Wu, S. Mokhtari, A. Nazef, B. H. Nam, and H.-B. Yun. Improvement of Crack Detection Accuracy Using a Novel Crack De-fragmentation Technique in Image-Based Road Assessment. *Journal of Computing in Civil Engineering*, Nov. 2014. 19

F. Yang, L. Zhang, S. Yu, D. Prokhorov, X. Mei, and H. Ling. Feature Pyramid and Hierarchical Boosting Network for Pavement Crack Detection. *IEEE Transactions on Intelligent Transportation Systems*, 21(4):1525–1535, Apr. 2020. ISSN 1558-0016. doi: 10.1109/TITS.2019.2910595. Conference Name: IEEE Transactions on Intelligent Transportation Systems. 21, 25, 29, 33

X. Yang, H. Li, Y. Yu, X. Luo, T. Huang, and X. Yang. Automatic Pixel-Level Crack Detection and Measurement Using Fully Convolutional Network. *Computer-Aided Civil and Infrastructure Engineering*, 33(12):1090–1109, 2018. ISSN 1467-8667. doi: https://doi.org/10.1111/mice.12412. URL http://onlinelibrary.wiley.com/doi/abs/10.1111/mice.12412. https://onlinelibrary.wiley.com/doi/pdf/10.1111/mice.12412. 15

J. Young, J. Ashburner, and S. Ourselin. Wrapper Methods to Correct Mislabelled Training Data. In *2013 International Workshop on Pattern Recognition in Neuroimaging*, pages 170–173, June 2013. doi: 10.1109/PRNI.2013.51. 36, 49

E. Zalama, J. Gómez-García-Bermejo, R. Medina, and J. Llamas. Road crack detection using visual features extracted by gabor filters. *Computer-Aided Civil and Infrastructure Engineering*, 29(5):342–358, 2014. 19

H. Zhang, J. E. Fritts, and S. A. Goldman. Image segmentation evaluation: A survey of unsupervised methods. *Computer Vision and Image Understanding*, 110(2):260–280, May 2008. ISSN 10773142. doi: 10.1016/j.cviu.2007.08.003. URL https://linkinghub.elsevier.com/retrieve/pii/S1077314207001294. 45

J. Zhang, G. Wang, H. Xie, S. Zhang, N. Huang, S. Zhang, and L. Gu. Weakly supervised vessel segmentation in X-ray angiograms by self-paced learning from noisy labels with suggestive annotation. *Neurocomputing*, 417:114–127, Dec. 2020a. ISSN 0925-2312. doi: 10.1016/j.neucom.2020.06.122. URL https://www.sciencedirect.com/science/article/pii/S0925231220312261. 37

K. Zhang, Y. Zhang, and H.-D. Cheng. CrackGAN: Pavement Crack Detection Using Partially Accurate Ground Truths Based on Generative Adversarial Learning. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–14, 2020b. ISSN 1558-0016. doi: 10.1109/TITS.2020.2990703. Conference Name: IEEE Transactions on Intelligent Transportation Systems. 22, 25

L. Zhang, J. Shen, and B. Zhu. A research on an improved Unet-based concrete crack detection algorithm. *Structural Health Monitoring*, page 1475921720940068, July 2020c. ISSN 1475-9217. doi: 10.1177/1475921720940068. URL https://doi.org/10.1177/1475921720940068. Publisher: SAGE Publications. 23

Q. Zhang, F. Lee, Y.-g. Wang, R. Miao, L. Chen, and Q. Chen. An improved noise loss correction algorithm for learning from noisy labels. *Journal of Visual Communication and Image Representation*, 72:102930, Oct. 2020d. ISSN 1047-3203. doi: 10.1016/j.jvcir.2020.102930. URL https://www.sciencedirect.com/science/article/pii/S1047320320301619. 36

Y. Zhou, H. Yu, and H. Shi. Study group learning: Improving retinal vessel segmentation trained with noisy labels. In M. de Bruijne, P. C. Cattin, S. Cotin, N. Padoy, S. Speidel, Y. Zheng, and C. Essert, editors, *Medical Image Computing and Computer Assisted Intervention – MICCAI 2021*, pages 57–67, Cham, 2021. Springer International Publishing. ISBN 978-3-030-87193-2. doi: 10.1007/978-3-030-87193-2_6. 49

Z.-H. Zhou. A brief introduction to weakly supervised learning. *National Science Review*, 5(1):44–53, 08 2017. ISSN 2095-5138. doi: 10.1093/nsr/nwx106. URL https://doi.org/10.1093/nsr/nwx106. 35, 37, 49, 75

Q. Zou, Y. Cao, Q. Li, Q. Mao, and S. Wang. CrackTree: Automatic crack detection from pavement images. *Pattern Recognition Letters*, 33(3):227–238, Feb. 2012. ISSN 0167-8655. doi: 10.1016/j.patrec.2011.11.004. URL http://www.sciencedirect.com/science/article/pii/S0167865511003795. 18, 33

Q. Zou, Z. Zhang, Q. Li, X. Qi, Q. Wang, and S. Wang. DeepCrack: Learning Hierarchical Convolutional Features for Crack Detection. *IEEE Transactions on Image Processing*, 28(3):1498–1512, Mar. 2019. ISSN 1941-0042. doi: 10.1109/TIP.2018.2878966. Conference Name: IEEE Transactions on Image Processing. 21

Abstract

Construction based on 3D concrete printing has become an attractive technology, infusing digital technology to the sector. Towards this digitalization, we propose methods based on computer vision for the monitoring of constructions built using contour crafting (a 3D concrete printing method). Our methods can be divided into two types: damage and quality monitoring.

For damage monitoring, we work on crack segmentation. Given that cracks are difficult to annotate accurately, an accurate segmentation is difficult using supervised machine learning. With a method inspired by weakly supervised classification, we improved the accuracy of the predicted cracks' width in synthetic data: the precision increases up to 26% with respect to training with raw inaccurate annotations; this method applies to real data straightforwardly. With a method based on transfer learning, we show that models trained solely on our synthetic images are able to segment cracks in real images. In both cases, the predicted crack width accuracy improves when measured with unsupervised metrics.

For quality monitoring, we work on inline anomaly detection in 3D printed pieces. First, we segment the interlayer lines in images acquired during printing (F-score = 91%). We analyze these lines by measuring geometrical properties at pixel level; we define as anomalies the regions with values outside the ranges of user-admissible values. We also classify the located layers based on their texture: either as good or as one of 3 defective cases (macro-averaged F-score = 94%).

Keywords: Computer vision, 3D concrete printing, Weakly supervised segmentation, Texture classification, Anomaly detection.

Résumé

La construction basée sur l'impression 3D de béton est devenue une technologie attractive, infusant la technologie numérique au secteur. Vers cette numérisation, nous proposons des méthodes basées sur la vision par ordinateur pour la surveillance des constructions construites à l'aide du *contour crafting* (une méthode d'impression 3D du béton). Nos méthodes peuvent être divisées en deux types : détection des défauts et contrôle qualité.

Pour la détection des défauts, nous travaillons sur la segmentation des fissures. Étant donné que les fissures sont difficiles à annoter avec précision, une segmentation précise est difficile en utilisant l'apprentissage automatique supervisé. Avec une méthode inspirée de la classification faiblement supervisée, nous avons amélioré l'exactitude de la largeur des fissures prédites dans des données synthétiques : la précision augmente jusqu'à 26% par rapport à l'apprentissage avec des annotations brutes imprécises; cette méthode est applicable directement à des données réelles. Avec une méthode basée sur l'apprentissage par transfert, nous montrons que des modèles entraînés uniquement sur nos images synthétiques sont capables de segmenter des fissures dans des images réelles. Dans les deux cas, l'exactitude de la largeur des fissures prédites s'améliore lorsqu'elle est mesurée avec des mesures non supervisées.

Pour le contrôle qualité, nous travaillons sur la détection d'anomalies en ligne dans les pièces imprimées en 3D. D'abord, nous segmentons les lignes intercouches dans les images acquises lors de l'impression (F-mesure = 91%). Nous analysons ces lignes en mesurant des propriétés géométriques au niveau du pixel; nous définissons comme anomalies les régions avec des valeurs en dehors des intervalles de valeurs admissibles par l'utilisateur. Nous classons également les couches localisées en fonction de leur texture : soit comme bonne, soit comme un de 3 cas défectueux (F-mesure macro-moyenne = 94%).

Mots clés: Vision par ordinateur, Impression 3D de béton, Segmentation faiblement supervisée, Classement des textures, Détection d'anomalies.