PHYSICS INFORMED SYNTHETIC IMAGE GENERATION FOR DEEP LEARNING BASED DETECTION OF WRINKLES AND FOLDS

Omey M. Manyar
Center for Advanced Manufacturing
University of Southern California
Los Angeles, CA 90007
Email: manyar@usc.edu

Junyan Cheng
Center for Advanced Manufacturing
University of Southern California
Los Angeles, CA 90007
Email: junyanch@usc.edu

Reuben Levine
Center for Advanced Manufacturing
University of Southern California
Los Angeles, CA 90007
Email: reubenle@usc.edu

Vihan Krishnan
Center for Advanced Manufacturing
University of Southern California
Los Angeles, CA 90007
Email: krishnan@usc.edu

Jernej Barbič
Department of Computer Science
University of Southern California
Los Angeles, CA 90007
Email: jnb@usc.edu

Satyandra K. Gupta*
Center for Advanced Manufacturing
University of Southern California
Los Angeles, CA 90007
Email: guptask@usc.edu

ABSTRACT

Deep learning based image segmentation methods have showcased tremendous potential in defect detection applications for several manufacturing processes. Currently, majority of deep learning research for defect detection focuses on manufacturing processes where the defects have well defined features and there is tremendous amount of image data available to learn such a data dense model. This makes deep learning unsuitable for defect detection in high-mix low volume manufacturing applications where data is scarce and the features of defects are not well defined due to the nature of the process. Recently there has been an increased impetus towards automation of high-performance manufacturing processes such as composite prepreg layup. Composite prepreg layup is high-mix low volume in nature and involves manipulation of a sheet-like material. In this work, we propose a deep learning framework to detect wrinkle-like defects during composite prepreg layup process. Our work focuses on three main technological contributions: 1. Generation of physics aware photo-realistic synthetic images with the combination of a thin-shell finite element based sheet simulation and advanced graphics techniques for texture generation, 2. An open-source annotated dataset of 10000 synthetic images and 1000 real process images of carbon fiber sheets with wrinkle-like defects, and 3. An efficient 2-stage methodology for training the deep learning network on this hybrid dataset. Our method can achieve a mean Average Precision (mAP) of 0.98 on actual production data for detecting defects. The code and the entire dataset are available at: https://github.com/RROS-Lab/DeepSynthDefectDetector.

1 INTRODUCTION

Defect detection has been a topic of paramount importance in the field of manufacturing for decades [1–5]. Several processes demand robust defect detection techniques to ensure improvement in part quality and overall process efficiency. Recently there has been a great interest towards automation in the manufacturing community. Several high-performance processes where robotic automation was infeasible earlier have radically pivoted towards extensive use of robots. This increased autonomy in manufacturing has entailed the development of better defect detection methods that are highly accurate, robust, and precise in nature. Hence defect detection has become a crucial element for large-scale adaptation of advanced robotics methods for manufacturing.

Several defect detection applications in manufacturing employ traditional vision-based methods [6–8]. In such applications, a high-resolution camera captures images of the part. These images are then processed with conventional image segmentation methods that are based on pixel filtering and image

*Address all correspondence to this author.

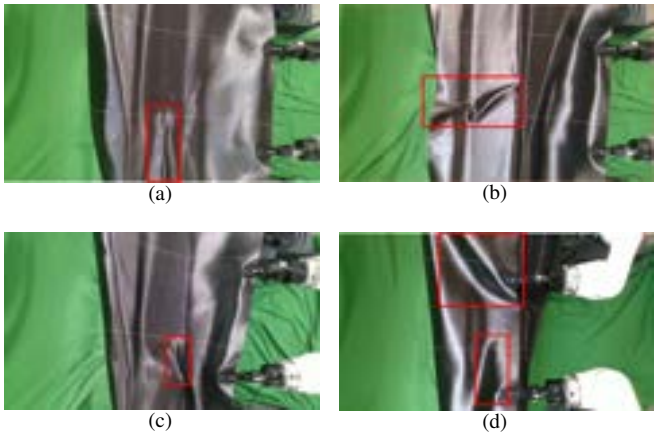


FIGURE 1: Variation in the wrinkles formed during prepreg composite layup

gradients. Such methods are highly sensitive to external factors such as lighting conditions, camera intrinsics, anisotropic interactions of the components, etc. These traditional methods are predominantly used for defect detection related to dimensional errors. However, there exists another class of processes where components might be deformable (e.g. prepreg composite layup) where such methods that depend on well-defined geometrical features of the part might be unable to detect defects. Moreover, these conventional techniques require fine-tuning of certain parameters to ensure robustness and repeatability in detection. Defects generated during processes such as composite prepreg layup have features that vary based on several external factors. Such salient features are mainly a result of the part's interaction with ambient light and the geometrical appearance of the defects.

To overcome these issues, recently, defect detection methods based on deep learning have been gaining a lot of momentum [9–12]. Researchers have explored an implementation of deep learning in applications with extensive availability of process data such as images or thermal signatures from sensors [13]. The major challenge with deep learning methods is their inherent dependency on availability of huge amount of data. Additionally, this data needs to be processed and prepared (e.g., annotated) for utilization in deep learning model training. Such pre-processed data is not readily available for several manufacturing applications. Furthermore, manually collecting and generating such data may not be feasible due to time and cost constraints.

This paper is an extension of our prior conference paper [14]. Processes where collecting online process data becomes infeasible, synthetic data can play a critical role for the deployment of deep learning models. With advancements in generative models and computer graphics, the learning community has pivoted towards use of synthetic data. [15–19]. Synthetic data can be effective in learning segmentation models if the data can exhibit realism not only in texture and appearance but also preserve the innate physics of the original data. With the conventional generative adversarial methods, it is difficult to capture the physics aspect of the real data. Synthetic images that look photorealistic and manifests the physical features of the original data can solve the data generation problem for defect detection in complex manufacturing settings. Therefore, synthetic image generation has a huge potential for enabling large scale implementation of deep learning methods for defect detection in manufacturing applications.

Synthetic image generation as discussed presents its own set of challenges in the context of deep learning. However, when it comes to certain applications that deal with deformable objects e.g. composite prepreg layup processes, the traditional methods employed for synthetic image generation may not be sufficient. Composite prepreg sheets are highly pliable and have a peculiar appearance that is dependent on the material type, weave pattern, etc. Due to the compliant nature of the sheets, it is difficult to synthesize images that emulate defects of the real-world process. Prepreg composite layup is characteristically a low-volume process with the defects exhibiting an irregular pattern, complex deformations of the material, and anisotropic reflections when captured using a 2D camera (Refer Fig. 1). Hence we need a solution that can accurately model the sheet's physical interactions under external constraints to simulate the defects that occur in-process.

In this work, we have explored this class of problems where, with the combination of advanced physics-based simulation and computer graphics rendering, we can generate highly accurate and realistic images for training a deep learning model for defect detection. There are several classes of defects that can be generated during a composite layup [20]. We focus mainly on wrinkles and sheet folding, since these two defects exhibit features and patterns that can be learned using 2D images of the simulated process. We provide a systematic framework for generating configurations of the sheet that are prone to introducing defects. We achieve this by employing a highly accurate physics-based model of the sheet. Additionally, we propose a technique for generating a precise texture of the composite sheet. Using the proposed methodology, we can generate high-quality rendered synthetic images of the composite sheet (Refer Fig. 2). We use this architecture to generate about 10,000 synthetic images encapsulating a wide variety of wrinkles.

Additionally, about 1,000 real in-process images of the wrinkles are collected in our experimental setup using an industrial tool. We use the resulting hybrid dataset of real and synthetic images to train a Mask Region-based Convolutional Neural Network (Mask R-CNN) [21] object detection model by a two-stage training procedure that firstly uses the large synthesized dataset, then fine-tunes with the small real-world dataset. The proposed Mask R-CNN model predicts a bounding box along with a mask

FIGURE 2: Comparing Synthetic Images with the Real Images. **Note**: The images shown here are for representative purpose only. They do not have a correlation in terms of visual appearance.

over pixels in the image that are classified as a defect (Refer Fig. 13). Our trained model was able to achieve an mAP (mean Average Precision) of about 0.98 on test dataset of 200 real images. This model can then be deployed for online defect detection in a prepreg composite layup. We have made the synthetic and real image dataset open-source and accessible to the public.

The rest of the paper is outlined as follows. We give an overview of the entire defect detection pipeline. Consequently, we provide detailed discussions of each system component: Synthetic Image Generation Pipeline, Data Annotation Methods, and Model Generation. We conclude with an in-depth analysis of model performance.

2 Related work

The problem of defect detection has been explored for decades specifically in traditional machining processes. Most of the work in the past has been focused on methods that involve defect detection on post-processed parts. Specifically, in the composite community, researchers have worked with traditional methods such as ultrasonic testing [22] that detects the woven or damaged surfaces for carbon fiber reinforced plastics. There are methods such as magnetic particle detection system [23] used to verify that the products are vacant of metallic magnetic contaminants, eddy current testing [24] which evaluates the health status of conductive materials, and so on. Although these methods are robust and accurate, they cannot be implemented online, making robotic automation an open-loop process. With the gaining momentum for automation in such high-performance applications [20, 25–29], it is important to focus on in-process defect detection applications. Elkington et al. [30] demonstrated the use of tactile shape sensing for detecting defects online, but this method cannot be used to prevent the formation of defects in the first place. Our method can predict anomalous regions during the layup that can potentially avoid the onset of a defect.

In [31], the authors survey various classical computer vision techniques for defect detection in a fabric. Although deep learning is the obvious solution to overcome the hurdles of traditional methods, past research has demonstrated that there are several challenges with data generation, rendering deep learning methods to be infeasible. Hu et al. [32] proposed to use Convolutional Neural Networks (CNN) with object-level attention mechanism enhanced by bilinear class activation maps (CAM) to detect casting defects on radiography images. This achieved a high accuracy; however, their method can only perform binary classification: defected or defect-free images. Wang et al. [33] introduce a CNN-based model to classify the defected and defect-free regions in cloth while using a sliding window which enables them to detect the location of the defect; such a sliding window-based method, however, is slow. Tabernik et al. [34] proposed a segmentation-based method that first uses a segmentation network to identify the surface-crack region in the metal surface, and then uses a decision network to classify the defect. While it can provide good results with a few training data samples, like many other previous methods [35,36], their method applied clean high-quality surface image with the surface fixed. Also, the defect patterns are more obvious: input images are usually all fixed under the same orthogonal viewpoint, the surfaces are usually flat, images are usually clean and high-resolution with no foreign objects, and the defects are usually clear of wrinkles or wear. Several deep learning methods used for defect detection are surveyed in [9, 37]. Unlike our method, most of these methods are not designed to be used in-process. Additionally, the layup is performed in an open environment where the surface continuously deforms and the pattern of the defects is less constrained. This makes the data scarcity problem even more salient.

Multiple methods have been proposed to handle the data scarcity problem in defect detection. Xu et al. [38] proposed to use the self-supervised data augmentation method to expand dataset size. However, their method can only be applied for the binary classification of data. Zhao et al. [39] introduced GANs to destruct positive samples to construct defect regions. However, their method is suitable for clean organized data with simple defect patterns. This is not feasible in our case as our process contains complex defects that are difficult for generative models to reconstruct. Chow et al. [40] utilized unsupervised learning with a convolutional autoencoder on defect-free samples, in order to learn by reconstructing input samples. The regions with high reconstruction error are classified as defects. However, like GAN, such a generative model is difficult to apply to complex defects and open environment images. Li et. al. [41] proposed a Fisher criterion-based stacked denoising autoencoders for differentiating between defective and defect-free samples. In their work, a thresholding based technique is used for locating the defect, making the method prone to errors due varying lighting conditions.

Meister et al. [42] demonstrated methods to perform data augmentation and explored GANs to generate the synthetic datasets. Although their method focuses on different types of defects, its application is meant for Automated Fiber Placement. Additionally, the aforementioned method is meant for detecting defects post fiber placement, as opposed to in-process like in our proposed method.

In this work, we sample photo-realistic synthesized data with the aid of a physics-based simulator and high-quality texture & ray-tracing methods. We then train a deep learning model to perform defect detection, overcoming the majority of the challenges of previous work in this domain.

3 Overview of Approach

In this section, we will briefly outline the components of our defect detection system. The entire system can be subdivided into four main components: Real Image Collection, Synthetic Image Generation, Data Preparation, and the Deep Learning Model. Fig. 4 gives an overview of these subsystems and the individual elements.

Real Image Collection: Although collecting real data for the layup process might be cumbersome, it is important that we capture the features embodying wrinkles that are not simulated. To achieve this, we perform a step-by-step layup and try to recreate scenarios that may lead to defect formation, as well as capturing the ones that lead to actual defects. We collect about 1,000 images using the approach described in Section 4.

Synthetic Image Generation: The synthetic imaging component constitutes of a FEM thin-shell simulator and the CGI module for computer graphics rendering. Our simulator here is based on the previous work by the authors [28] that builds an accurate mechanical model of the composite sheet. This simulator helps in accurately predicting the sheet deformation under varied external constraints. We emulate the possible configurations of the sheet that can lead to formation of anomalies on the sheet. Such anomalies signify the onset of a defect. Furthermore, we also simulate defects in the form of wrinkles for an already conformed sheet on the composite tool. The simulator outputs a triangulated mesh of the sheet. This mesh then passes through a CGI pipeline that uses ray tracing to render a photo-realistic image. The rendering is improved by applying the custom texture that we generate for the composite sheet. The entire process is discussed in detail in Section 5.

Data Preparation: Once we have developed a hybrid dataset of real and synthetic images, we describe methods to annotate the data and provide a formal definition of what constitutes as a wrinkle or a defect. We describe the data preparation method in detail in Section 6.

Deep Learning Model: After the data preparation and processing is completed, the properly annotated dataset can be used to train a deep learning model for predicting defects. As mentioned in Section 1, we use the Mask R-CNN architecture that outputs a mask of the predicted defect within a bounding box. The model architecture and framework are discussed precisely in Section 7.1.

Using the methodology presented in our work, we can design a robust system that can be deployed for detecting defects in the composite layup process. Such a system can effectively detect wrinkles formed during composite layup as well as detect anomalous configurations of the sheet that may lead to defect formation. This obviates the need to remove the defect altogether. We will describe each aforementioned components comprehensively in the following sections.

4 Real Image Collection

(a) (b)

FIGURE 3: The two types of wrinkles witnessed during composite layup. (a) the wrinkles formed on a conformed/draped portion of the sheet; (b) an anomalous region on the undraped portion of the sheet that signifies an onset of a wrinkle.

In order to generate a dataset comprising of images of the defects formed during the actual process, we used an experimental setup with an industrial tool and two sheet grasping robots as shown in Fig. 1 and Fig. 2. We use a set of industrial grippers with custom 3D printed attachments to grasp the sheet appropriately. The objective of the setup is to emulate a robotic composite layup process and capture various types of wrinkles and anomalies that may emerge during the process.

In this work, we have focused only on wrinkles as the class of defects for detection. Defining a wrinkle on a cross-section of the sheet that has already conformed to the tool is straightforward. These wrinkles have clear well-defined features as can be seen in Fig. 3(a). They are formed due to improper flexing of the sheet, leading to the formation of ridges.

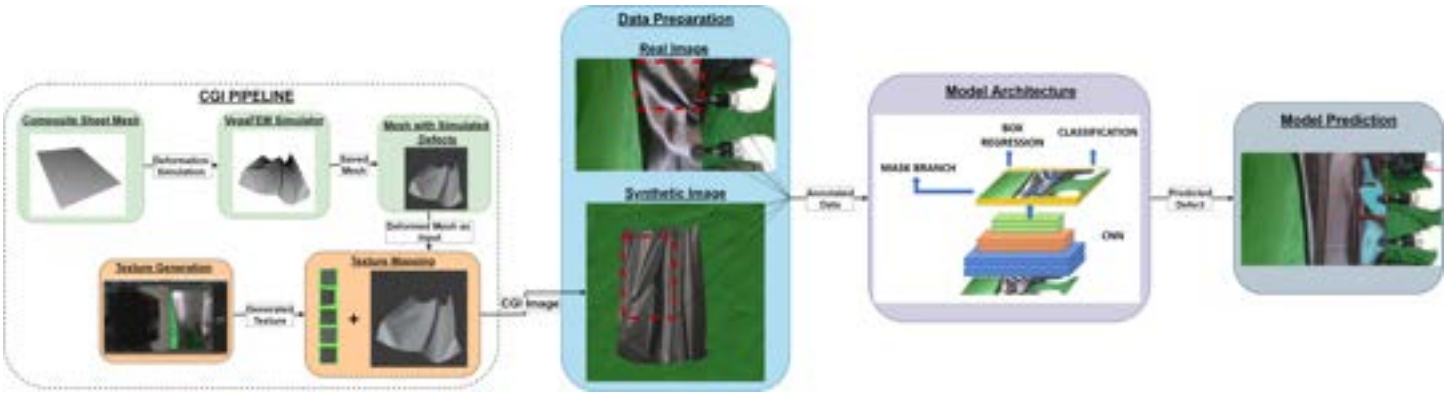


FIGURE 4: Process flow describing the system.

A robust defect detection system should not only be able to detect already formed wrinkles, but also predict if a certain configuration of the sheet is likely to form a wrinkle. This helps in avoiding defect formation altogether. To achieve this functionality, we try to flex and hold the composite sheet in configurations as depicted in Fig. 3(b). These anomalous configurations are the ones where if a robot or an operator were to conform the sheet to the tool, a wrinkle will form. These configurations are commonly encountered and are the main cause of wrinkle formation in the composite layup process. Hence, we capture features of such anomalous regions and classify them as defects.

The majority of parts in the composite industry have tubular (or tubular-like) cross-sections. This entails the tool to be rotated about an axis to wrap the composite sheet around the tool's geometry. To incorporate this variation, we capture images of the wrinkles at different configurations of the tool (Refer Fig. 3).

We used an overhead 2D camera with a 1920 x 1080 resolution for capturing the images. During the data collection process, there were slight variations in the lighting conditions. Furthermore, we also changed the position of the grasping robots to avoid introducing bias into the images. In this manner, we generate a dataset of 1,000 real images. This dataset consists of 800 images with a combination of Type 1 an d Type 2 Defects, and 200 images with no visible defects.

5 Synthetic Image Generation

The real image dataset generated using the methodology of Section 4 provides a good benchmark for describing a wrinkle. Since generating large amounts of real image data can be infeasible from the perspective of time and material costs, we need to rely on synthetic images for training a robust and accurate deep learning model. In this section, we will describe synthetic image generation pipeline in detail. The key feature of our proposed pipeline is the blend of accurate physics simulation that can emulate the Type 1 and Type 2 defects, and a realistic texture of the composite prepreg material, capturing the anisotropic parameters to generate photo-realistic images. We use the sheet configurations that generated most of defects as the basis for simulating and replicating the Type 1 and Type 2 defects. We also generated synthetic images with no perceivable defects.

We will now describe every element of our synthetic image generation pipeline and present a methodology to generate accurate and photorealistic images of defects in the composite layup process.

5.1 Physics Based Simulator

In order to generate accurate synthetic images with realistic wrinkles, it is crucial to simulate the sheet precisely. Traditional FEM simulators are infeasible due to their slow convergence. In this work, we have employed a physics-based simulator that is based on the previous work done by authors in [28], and that is built on top of the VegaFEM simulation library [43].

In order to generate synthetic imagery, we recreate the setup of Section 4 in the simulation environment. A triangulated mesh with the same size as the sheet is created, and four constraints are added replicating the sheet grasping points. The aforementioned simulator has the capability to handle dynamic constraints. We move these constraints so as to replicate the anomalous configurations experienced during the real image generation process. We discuss this process in detail in the next section.

5.2 Data sampling

In this section, we discuss our methodology to simulate the anomalous configurations. We generate uniform data by randomly sampling the thin-shell locations and world-coordinate trajectories of the four constraints located on the perimeter of the sheet mesh. The constrained vertices of the mesh are selected close to the perimeter of the rectangular sheet, to replicate the scenario of Section 4. These constrained points are denoted by P_1, P_2, P_3, P_4. For each trajectory, the converged simulated shape

Parameter	Distribution
P_1	$[0, U(0.7, 1.0)]$
P_2	$[1, U(0.7, 1.0)]$
P_3	$[0, U(0, 0.3)]$
P_4	$[1, U(0, 0.3)]$
τ_1	$[U(0.1, 1.4), U(0, -0.4), 1]$
τ_2	$[U(-2.5, -1.2), U(0, -0.4), 1]$
τ_3	$[U(0.1, 1.4), U(0, 0.4), 1]$
τ_4	$[U(-2.5, -1.2), U(0, 0.4), 1]$
V_1, V_2, V_3, V_4	$U(0.1, 0.3)$
n	$\mathcal{N}(0, 0.1)$

TABLE 1: Parameters and their distribution in data sampling.

of the sheet is only stored at the endpoint of the trajectory, regardless of the intermediate trajectory positions. Therefore, we simplify the trajectory of the constrained vertices into a straight line from the starting to the ending point. We represent the vectors joining starting and ending points by $\tau_1, \tau_2, \tau_3, \tau_4$, where each τ_i is sampled randomly as given in Table 1. The velocities of the constraint vertices are represented by V_1, V_2, V_3, V_4, and are also sampled randomly (Table 1). The simulation timestep t is a constant fixed at 10 milliseconds in our sampling process. This enables the constrained vertices to reach the end points faster, while still being able to span the desired search space.

A trajectory is discretized into multiple simulation timesteps. In each timestep, the four holding points move along their velocity vectors. To model the noise from the robot actuators, the planner, the sensors, etc., we applied a simple Gaussian noise n as a perturbation to the moving distance at each time step. The sampling parameters can be found in Table 1.

The data sampling process can be executed in parallel. After the completion of trajectory execution, we stop the simulator after 55 timesteps, which is sufficient for the simulator to converge and output the thin shell shape. With a simple multi-core implementation, the proposed pipeline can generate about 360 shapes per hour, using a Core i7-9750H processor with 6 cores. By employing the proposed pipeline, we generated 10,000 mesh shapes, capturing varying wrinkling features. These shapes also capture some of the anomalous configurations that we encountered during real image data collection.

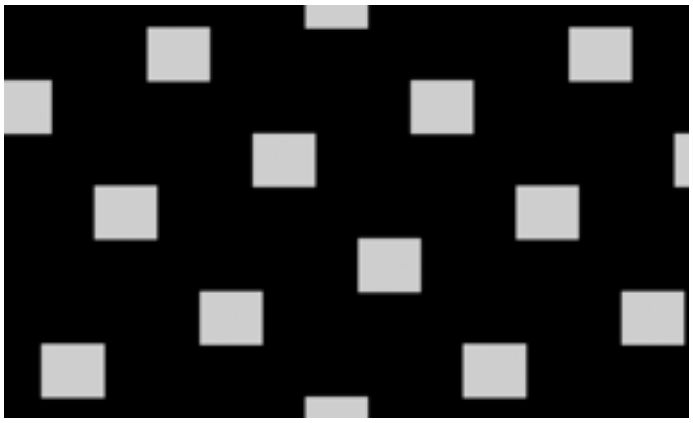


FIGURE 5: The black-and-white tiled matte used to indicate fiber direction of the synthetic texture.

5.3 CGI Pipeline

Our physics-based simulator and the data sampling methodology produce meshes that contain realistic wrinkles. However, to generate photo-realistic images that can be used for training the model, we need to also realistically render the produced sheet meshes. To do so, a high-quality texture of the sheet is of paramount importance. We employed CGI technology (as described below) to generate a detailed procedural texture of the actual composite sheet used in the real data collection. This texture can then be applied to the 10,000 synthetically generated meshes, to generate high-quality renders close to the actual real images. In this work, initially, we experimented with generating an "Image Texture" by taking several images of the carbon fiber sheet under varying lighting conditions and camera angles. However, the generated image texture could not precisely capture the light interactions of the sheet, rendering the images to look distinctively different from the actual image. Hence, we adopted a methodology to generate a "procedural texture" of the carbon fiber prepreg sheet rather than an "Image Texture".

We used Blender [44], an open-source animation and rendering software, to build the carbon fiber texture and render the images. A texture was constructed to imitate the real carbon fiber. We developed a custom CG texture in Blender using its node editing system. There were three main components to the texture: The first was mimicking the texture of the parallel-running fibers in the carbon fiber. This was accomplished by stretching a procedurally-generated noise map in one dimension. The second step was to find a way to re-orient these fibers at 90^o angles in certain areas since the carbon fiber sheet is made of interwoven vertical and horizontal strips of carbon fiber. This was accomplished by procedurally generating an orientation map in different software (Matlab), which can be seen in Fig. 5. The Blender texture generator then oriented the virtual fibers of the texture

horizontally where the "orientation map" showed a pixel value of 0 (black) and vertically where the "orientation map" showed a pixel value of 255 (white). The result is a close emulation of the interwoven appearance of the carbon fiber sheet. A close-up view of the CG texture is shown in Fig. 6. Although to the naked eye, the texture showcased in Fig. 6 might not look exactly similar in appearance but on a macro level, the objective of the texture is to capture the salient features such as shape of the wrinkle, appearance, etc. of the prepreg sheet. These features are sufficiently good for training the deep learning model.

The generated texture can then be applied to the 10,000 synthetically generated meshes. When virtually lit and rendered, these objects look close to photorealistic (see Fig. 2). We now describe this process.

First, we need to match the lighting of the virtual environment to the lighting of the real environment. This entails capturing a spherical panorama of our manufacturing space. We do this by photographing a mirror sphere and then wrapping this 2D image onto a 3D photo-sphere in the computer. We photographed a standard $3''$ chrome ball commonly used in CGI applications from two different angles, and at several different levels of camera exposure, to capture the entire spectrum of light present in the workroom. These images were then wrapped to create the 360° high dynamic range image (HDRI) shown in Fig. 7.

Next, we apply the texture and the lighting setup to produce renders that closely replicate real-world images. The synthetic images generated using this framework can be seen juxtaposed against real images in Fig. 2. We rendered the 10,000 meshes to produce 10,000 images of the composite sheet in varying configurations. The rendering was performed on a core i7-11700 (8 core, 2.5 to 4.9 GHz) processor boosted with an NVIDIA GeForce GTX 1660 Ti 6GB GPU. With this configuration, the average time to render a single image was about 5 secs. Using these methods, a digital rendering environment can then be created with optical materials and lighting conditions that match the real manufacturing environment; see Fig. 8. We achieved this by importing the HDRI images into blender and consequently performing ray tracing.

FIGURE 6: Left: zoom on the CG texture. Right: photo of real carbon fiber under a microscope (Visual Appearance may differ due to the scale).

FIGURE 7: A collection of images at different exposures (top) were compiled to create a 360° panoramic HDRI (bottom).

The proposed CGI pipeline helped us in augmenting our real dataset of 1,000 images. This hybrid dataset with 11,000 images is sufficient for feature extraction, and is employed for training the deep learning model proposed in Section 7. The synthetic image generation approach is summarized in Fig. 9.

6 Data Preparation

The hybrid dataset generated using the methodology described in Sections 4 and 5 needs to be appropriately annotated for model training. We described the key features of wrinkles in Section 4 and categorized them into Type 1 and Type 2 defects. We used the same description for generating the annotations for the dataset. We classify both Type 1 and Type 2 defects as one class called "wrinkle". Fig. 10 shows example annotations on a real and a synthetic image.

For performing the annotations, we used an online open source tool called makesense.ai. This tool helped us generate polygon annotations that engulf the defected region ("wrinkle")

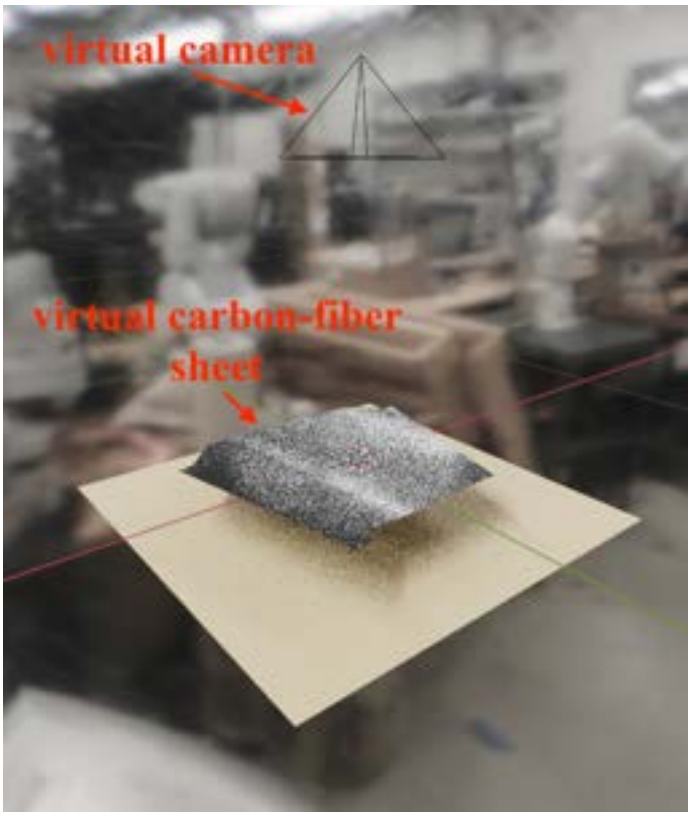


FIGURE 8: The virtual environment.

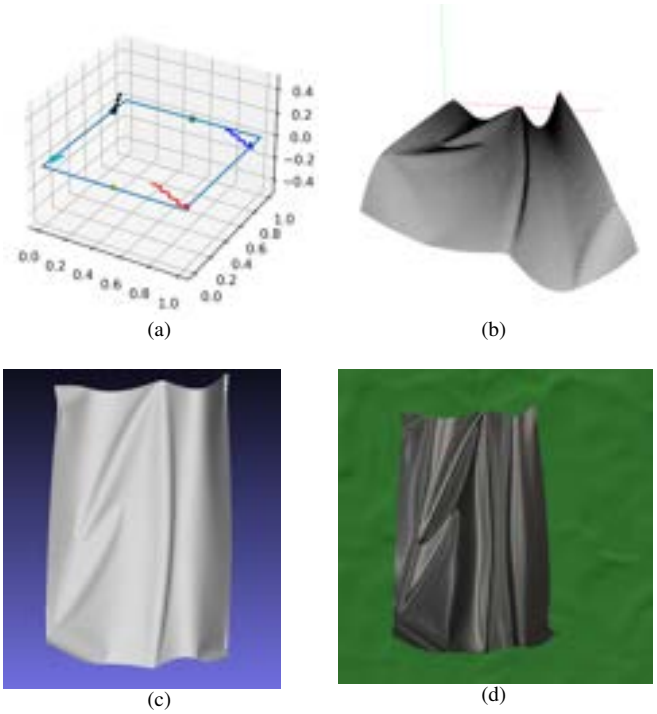


FIGURE 9: (a) Trajectory of the four holding points; (b) the shape after executing the trajectory in our physical-based simulator; (c) output mesh before rendering; (d) the CGI-rendered mesh.

in the image and can be saved as json files. All the images were annotated using only one class per annotation called "wrinkle". Images with no visible defects received no annotation. Once we complete the data preparation phase, we can then use the annotated data for training our proposed deep learning network.

7 Deep Learning

Our modeling methodology is based on the image segmentation deep learning model Mask R-CNN [21]. We use a two-stage training procedure to train the image segmentation model to detect defects in the composite sheet. The first stage is a pre-training stage that uses a large synthesized dataset. The second stage is a fine-tuning stage that utilizes a small real-world dataset. We further introduce a scaled mixing technique to the fine-tuning stage, so that the synthesized dataset is also employed in the fine-tuning stage; this avoids catastrophic forgetting. Our deep learning model architecture and training details will be described in Sections 7.1 and 7.2, respectively.

FIGURE 10: Annotations depicting the Type 1 and Type 2 defects. Kindly note that both Type 1 and Type 2 Defects are annotated as one single class "Wrinkle"

7.1 Model Architecture and Settings

We applied the Mask R-CNN shown in Fig. 11 as our base model for the image segmentation task. After trying typical backbones including VGG [45], Inception [46], DenseNet [47], We select a ResNet-50 for its best performance among them as the backbone network which extracts features from the input image. Feature

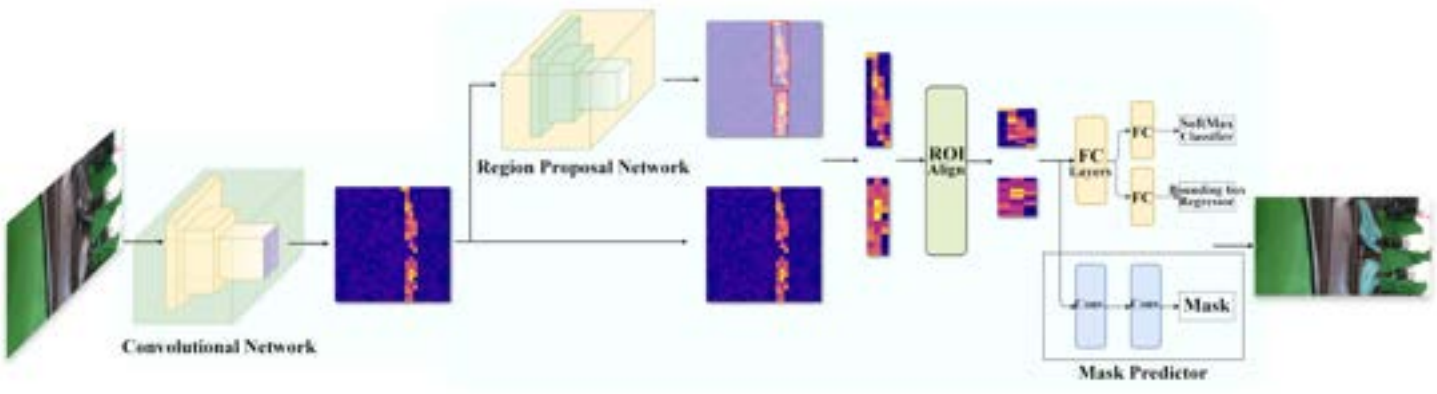


FIGURE 11: Mask R-CNN architecture.

Pyramid Network (FPN) [48] is used as the neck, along with the backbone network. This architecture better utilizes the extracted features in a multi-scale manner.

Ultimately, we use two heads applied as the output of the model, one for outputting the bounding box and another one for predicting the mask. The two heads share a Region-of-Interest (RoI) network that proposes the potential RoIs on the top of the neck network.

The Mask R-CNN we use has been pre-trained on the Microsoft Common Objects in Context (MS COCO) [49] dataset. This dataset consists of 1.5 million labeled images from 80 categories. Our pre-trained model learns rich common visual patterns of images, after performing transfer learning by iteratively adjusting modeling parameters through in-domain images of the carbon fiber sheet. The proposed model will be able to detect task-specific patterns (i.e. defects) quickly. Compared to training a model from scratch, we need less redundant data for letting the model start from learning the basic common patterns.

Our implementation is based on the MMDetection framework [50]. The model is trained on one single NVIDIA GeForce RTX 2070 GPU. For the model that uses ResNet-50 as the backbone, the entire training process takes about 12.5 hours. The memory use is approximately 4.4 GB, and the inference runs at about 14.5 fps. We also explored variants of the model architecture that improve the model performance; they will be discussed in Section 8.3.

7.2 Training

Despite the high quality of our synthetic images, the sim2real gap [51, 52] still exists. For example, the real carbon fiber sheet may have a slightly worn surface, which causes the reflectance to be uneven. The physical specifications of a real sheet may also change over time. There may be temperature variations. The complex lighting conditions in our production environment is also difficult to reconstruct perfectly. The movements of the real robots are computationally expensive to be completely reproduced in the simulator. Finally, the error from the physical simulator itself and the imperfectness of the rendering engine should also be considered. These minor errors accumulate and cause a gap between the images of real and virtual sheets. The time and processing cost of pursuing a higher degree of realism in the synthetic data increases exponentially, even higher than the cost of collecting real data. Thus, a better solution is to compensate for the gap through real images.

However, directly mixing the synthesized data with real data and then using the mixed dataset to train the model would cause an inductive bias. This might cause the model to overly to rely on the features learned from virtual images while ignoring real images. What we desire is to learn the physics of the sheet from the synthetic data, but then use the real data to compensate for the details in the real production environment. The problem occurs due to the data imbalance, given that synthetic images outnumber real ones by 10 to 1 in our dataset.

One approach to solve this problem is to use a **2-stage training** method. Sajjan et al. [53] proposed to use a two-stage training procedure that first pre-trains the network on a large out-of-domain real-world dataset, then fine-tune it on a smaller in-domain synthetic dataset for the robotic grasp of transparent objects. Inspired by their method of learning basic patterns in the first stage and then transferring knowledge into the target real domain in the second stage, we applied a two-stage training method that first pre-trains on a large synthetic dataset, and then fine-tunes the model on the smaller real dataset.

In the pre-training stage, the model learns typical patterns of the defects formed in the composite sheet, and the deformation of the sheet caused by the movement of robots. By using our physics informed simulator to simulate the deformation of the composite sheet, we implicitly introduce the physical knowledge as a bias into a deep learning model. This helps the model understand the generating process of a defect. In the fine-tuning

stage, the model learns task-specific knowledge on how the defects look in the real layup, which includes additional noise originating from lighting conditions, robot actuation, occlusions, etc.

Another way to mitigate the gap between the real and synthesized datasets is to regard it as a multi-task problem where the model has to learn both the physics-based behavior through the synthetic dataset and the expert knowledge about the defects in the real world through the real dataset. We use **scaled mixing** that entails training with each real image several times, whereas training only once for each synthesized image. This is a technique that has succeeded in training multilingual language models where there is a huge disparity between the data set sizes of larger languages vs smaller languages; each language is a subtask to learn. Instead of directly mixing real and synthetic data, we replicate the real data k times before mixing (we use $k = 5$). Furthermore, we combine this approach with 2-stage training, by applying scaled mixing in the second stage. The performance and analysis of these training techniques in our experiments will be discussed in Section 8.1.

8 Results

We evaluated our method on images acquired from a robotic composite layup cell. The experimental setting will be discussed in Section 8.1. We will also discuss techniques to enhance our model (Section 8.2). The experimental results are analyzed in Section 8.3. Finally, we also discuss failure cases where our model failed to produce a successful prediction (Section 8.4).

8.1 Training settings

The real dataset consists of 1,000 images and the synthetic dataset has 10,000 images. We split the real dataset to 6:2:2 for training, validation, and test sets, respectively. For models trained in 2 stages, we use the entire synthetic dataset for training in the first stage, whereas we use the validation set of the real dataset for model selection.

In both stages, we apply data augmentation techniques comprising random shifts, rotations, and scalings of the images. Hence, before an image enters our training pipeline, it will initially pass through the data augmentation pipeline. With a probability of 0.5, the image is randomly transformed by first shifting it by a pixel amount uniformly sampled from $U[-0.0625, 0.0625]$ for the x and y axis, respectively. Subsequently, the image is rotated by an angle uniformly sampled from $[-45^o, 45^o]$, with the probability of 0.5. Finally, with a 0.5 probability, we scale the image by a factor uniformly sampled from $[0.9, 1.1]$. These data augmentation methods largely expand the dataset size and introduce an inductive bias whereby the patterns of the defects are invariant to the transformations from the camera. This in turn helps the model to learn more robust representations. While our data augmentation pipeline is randomized, by fixing the random seed in our experiment, we ensure the reproducibility of the images when re-running the experiment.

FIGURE 12: Data augmentation methods. (a) Original image, (b) random shift, rotation or scale, (c) random brightness and contrast, (d) random hue/saturation/value, (e) randomly shifted RGB values, and (f) random blur.

We employ the Intersection-over-Unification (IoU) area metric to evaluate the accuracy of our deep network. This metric divides the area of intersection of the predicted defect region and the ground truth region, with the area of the unified region. To define whether an output is considered accurate, we use the common IoU threshold of 0.5, based on work done in the PASCAL Visual Object Classes (VOC) challenge [54]. This means that a predicted defect with IoU larger than the threshold 0.5 is considered positive, whereas IoU below 0.5 is regarded as negative.

We implemented our deep learning pipeline using the PyTorch framework [55]. We trained 12 epochs for each stage and applied Stochastic Gradient Descent (SGD) with a momentum

of 0.9 and weight decay of $1e-4$ to optimize our model. The learning rate is $2e-3$. A step learning rate scheduler is employed, hence in the 8th and 11th epochs, the learning rate will be scaled down by $\times 0.1$. We also applied a linear learning rate warm-up for 5000 training steps with a ratio of $1e-5$.

8.2 Model enhancement

Apart from the experiments that explore the effectiveness of the training techniques and the use of synthetic data, we performed additional experiments to enhance our model. This helped us devise auxiliary schemes that are effective for our application, as described next.

Advanced backbone network: We explored the use of state-of-the-art backbone network ResNeSt-50 [56] which is an advanced version of the ResNet backbone we used as the base model. ResNeSt-50 introduces channel-wise attention to the different network branches of ResNet, to better capture cross-feature interactions and learn diverse representations. In order to analyze the trade-off between the cost and accuracy improvement, we replace the ResNet-50 network with ResNeSt-50.

Deeper model: We use a deeper backbone network ResNet-101 to replace the ResNet-50 network in this experiment. This helped us examine the cost vs accuracy trade-off of using a deeper model.

Advanced model architecture: We apply a Cascade Mask R-CNN [57] which improves the Mask R-CNN architecture by introducing a multi-stage object detection architecture. This consists of a sequence of detectors trained with increasing IoU thresholds to replace our basic Mask R-CNN architecture. This helped us explore the trade-off between a more complex and advanced architecture and the accuracy improvement.

Additional data augmentation: We introduce additional data augmentation techniques into the pipeline to explore the influence on model performance due to this subsequent inductive bias. We include randomly changing brightness and contrast with a factor uniformly sampled from $[-0.2, 0.2]$ with a probability of 0.2; randomly shifting RGB channel by pixels uniformly sampled from $[-10, 10]$ for each channel with a probability of 0.1; randomly changing the hue, saturation and value of the input image with a quantity uniformly sampled from $[-20, 20]$, $[-30, 30]$, and $[-20, 20]$, respectively, by a probability of 0.1; randomly shuffling the RGB channels by a probability of 0.1; and randomly imposing blur with the kernel size uniformly sampled from $[3, 7]$ with a probability of 0.1. These data augmentation methods introduce inductive biases such that the patterns of defects are invariant to the color of the carbon fiber sheet and the camera parameters. The data augmentation methods are shown in Fig. 12.

8.3 Analysis of results

The results of our experiments are shown in Table 2. We compare the performance of the mask R-CNN-based base model, and the four model enhancements discussed in Section 8.3.

For each of the five variants, we first train them without synthetic data and then demonstrate the improvement with synthetic data based on the best-observed training techniques. The best training technique is a combination of Scaled mixing and 2-Stage training, denoted as "+ Scaled 2-S.". The results show that for all variants, the synthetic data gives a gain of about 8 to 10%.

In the first variant, we experimented with the base model denoted as "Base" and performed ablation studies for training techniques. "+ Syn. data" denotes a direct mixing of the real and synthetic data set. It provides only a nominal increment of 2.4% compared to model trained without synthetic data. By introducing "2-Stages" training, the data imbalance problem is moderated, however this approach is still limited by the "catastrophic forgetting" problem. This is reflected in the model trained with "Scaled" mixing where the gain increases from 5.02% to 8.02%. Finally, by combining the two approaches, the model achieved an overall gain of 9.81%.

The advanced data augmentation ("More aug.") increased average mAP by 1.24% and also slightly improved the trained model on synthesized data. Advanced backbone model ResNeSt-50 ("ResNeSt") achieved the highest 10.08% improvement, but the memory use increased to 5.5 GB while the inference speed decreased to 13.3 fps, and training time increased to 25 hours. Cascade architecture ("Cascade") also achieved a higher mAP but increased the memory use to 6.0 GB. The inference speed decreased to 9.6 fps while training time increased to 15 hours. However, using this deeper model ResNet-101 with 101 layers ("More layers"), the average test mAP gain for the model with synthetic data decreased from 9.81% to 8.39%. This may be due to the over-complexity of the model for our application. The memory usage increased to 6.4 GB while the inference speed decreased to 11.7 fps, and the training time increased to 18 hours.

The results show that our training techniques are effective. The augmentation methods do improve the performance of model architectures such as the Cascade architecture or the advanced backbone model. Additionally, advanced data augmentation helps the model to better generalize the dataset. That said, an upgrade to the model architecture incurs a cost of lowering the computational efficiency. Due to the relatively lower complexity of our problem, as compared to a general image segmentation model which may contain millions of samples, a highly complex model with too many layers eventually deteriorates the performance.

Overall, our method achieved the highest average test mAP of 0.98. A proper choice of model complexity and the trade-off between higher performance due to advanced model architecture vs efficiency is important in real-world applications. Moreover, additional data and training "tricks" also require more training time.

	Test mAP					Train mAP							
	Def.	**Pre.**	**Und.**	**Avg.**	**Gain**	**Def.**	**Pre.**	**Und.**	**Avg.**	**Gain**	**Mem.**	**Inf.**	**Train.**
Base	0.938	0.832	0.9	0.89	-	0.938	0.821	0.875	0.878	-	4.4 GB	14.5 fps	0.5 hrs
+ Syn. data	0.944	0.865	0.925	0.911	2.4%	0.938	0.844	0.9	0.894	1.82%	4.4 GB	14.5 fps	5.5 hrs
+ 2-Stages	0.956	0.898	0.95	0.935	5.02%	0.949	0.868	0.917	0.911	3.80%	4.4 GB	14.5 fps	5.5 hrs
+ Scaled	0.977	0.932	0.975	0.961	8.02%	0.956	0.892	0.945	0.931	6.04%	4.4 GB	14.5 fps	7.5 hrs
+ Scaled 2-S.	0.987	0.945	1.0	0.977	9.81%	0.961	0.903	0.958	0.941	7.14%	4.4 GB	14.5 fps	12.5 hrs
More aug.	0.940	0.838	0.925	0.901	1.24%	0.939	0.830	0.942	0.904	2.92%	4.4 GB	14.5 fps	0.5 hrs
+ Scaled 2-S.	0.989	0.948	1.0	0.979	10.0%	0.966	0.912	0.967	0.948	8.01%	4.4 GB	14.5 fps	12.5 hrs
More layers	0.936	0.838	0.9	0.891	0.15%	0.942	0.833	0.875	0.883	1.01%	6.4 GB	11.7 fps	0.72 hrs
+ Scaled 2-S.	0.980	0.939	0.975	0.965	8.39%	0.960	0.9	0.942	0.934	6.38%	6.4 GB	11.7 fps	18 hrs
Cascade	0.942	0.847	0.925	0.905	1.65%	0.947	0.848	0.95	0.915	4.21%	6.0 GB	9.6 fps	0.6 hrs
+ Scaled 2-S.	0.990	0.947	1.0	0.979	10.0%	0.963	0.918	0.967	0.949	8.13%	6.0 GB	9.6 fps	15 hrs
ResNeSt	0.947	0.841	0.95	0.913	2.55%	0.950	0.845	0.95	0.915	4.21%	5.5 GB	13.3 fps	1 hrs
+ Scaled 2-S.	0.989	0.950	1.0	0.98	10.08%	0.965	0.913	0.975	0.951	8.31%	5.5 GB	13.3 fps	25 hrs

TABLE 2: Results of experiments. All gains are compared to the "Base" model. Detailed interpretation is available in Section 8.3.

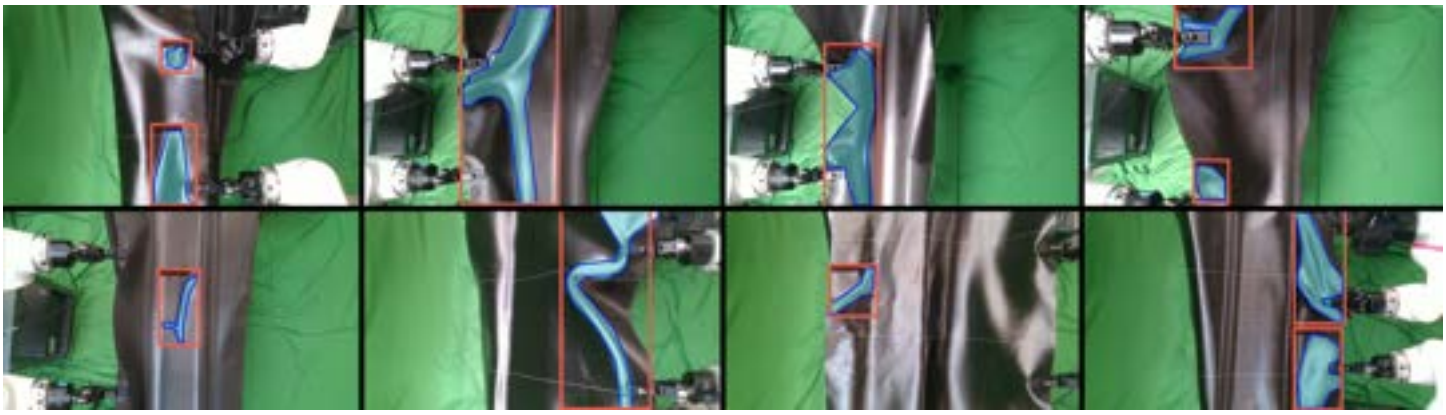
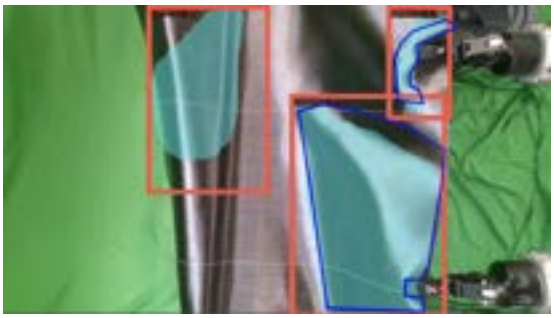


FIGURE 13: Model prediction examples. Orange rectangle marks the predicted bounding box, transparent cyan region represents the predicted mask, and dark blue polygons denote the ground-truth.

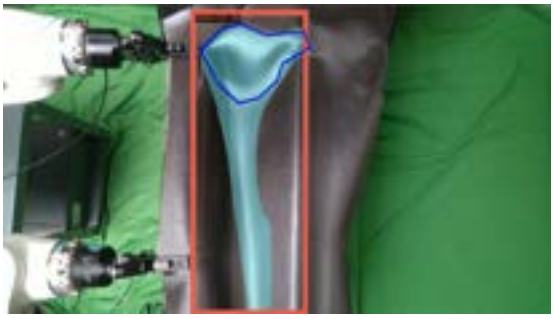
8.4 Analysis of failure cases

We further analyzed the test samples where the model failed to make satisfactory predictions in the test set. We summarize these instances in the following four cases, and recommend possible solutions.

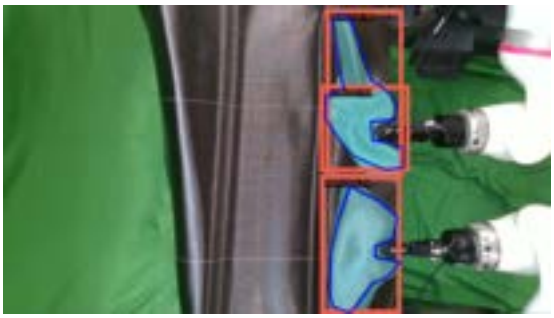
Failure case 1: False Positives The model sometimes predicts an unexpected region; this happens very rarely. An example of this case is shown in Fig. 14(a) where the region on the left side predicted by the model is unexpected. This can be avoided simply by using model ensembling [58], i.e., training multiple models and making predictions by voting. Although deep learning models always have the risk of behaving unexpectedly, these be-



(a)



(b)



(c)

(d)

FIGURE 14: Failure cases. The orange rectangle and transparent cyan regions mark the bounding box and mask predicted by the model, respectively; the dark blue polygons denote human annotation. (a) False positive (Unexpected region), (b) Confused region, (c) Redundant prediction, (d) Annotation mistakes.

haviors usually vary between models. Hence, by ensembling, the results, the weight of each unexpected prediction will be largely reduced and thus the risk of predicting unexpected regions will be largely avoided.

Failure case 2: Confused Region The edge of the composite tool overlaps with the wrinkle, making it hard to distinguish the defect and defect-free region as shown in Fig. 14(b). Such local features of the tool sometimes makes it difficult to locate the wrinkle when the defect lies in the vicinity of these features. This is a less obvious case where the boundary between defected and defect-free region is overlapping. This could be attributed to a high sensitivity of the model where the model would try to catch all potential defects. However, such sensitivity is preferred in our intended application where a false negative that potentially causes layup failure is more detrimental than a false positive. The model may implicitly be imposed with such inductive biases by the data annotation process. A high sensitivity may also be a possible cause of the failure case 2. Like case 1, this case could also be solved by model ensemble methods.

Failure case 3: Redundant Prediction In this case, the same wrinkle region is predicted multiple times, as shown in Fig. 14(c). The upper wrinkle has been marked by the model twice. Although reducing the test accuracy, this case does not influence real-world applications, and we can simply merge the predictions.

Failure case 4: Annotation mistakes Fig. 14(d) shows two examples of such mistakes. The top annotation in the image is very small. Therefore, a small perturbation of the model prediction will cause the IoU metric recommend it as a failed prediction. However, the prediction is completely acceptable in the actual application. The bottom annotation in the image is a weak wrinkle missed by the model. It is an unexpected defect due to reusing the same composite sheet several times during the real image collection process. Furthermore, there might be some ambiguity amongst the human annotators as per whether to annotate such weak wrinkles as a defect or not. These annotation errors could be easily handled by a improved annotation and data collection procedure.

9 Conclusions

Our work showcases how synthetic data alongside sparse real process data can be used to effectively train a deep learning model for defect detection task. We have presented a deep learning framework for detecting defects in manufacturing applications that deal with sheet-like deformable materials. We demonstrated our approach for composite prepreg applications, using an actual industrial tool. This work illustrates how a physics-based simulation can aid in developing 3D meshes of the sheet with features inherent to actual wrinkles. Using these 3D meshes we demonstrated how we can employ CGI techniques to develop photo-realistic synthetic images of the composite sheet defects.

We also elaborated on the challenges encountered while implementing the synthetic image generation framework for a process such as composite prepreg layup. A detailed CGI procedure for generating a precise and realistic texture of the composite sheet is prescribed. We have investigated different modeling variants and presented an ablation study to identify the best modeling and training methodology. The 2-Stage training process with our hybrid dataset using the mask R-CNN architecture helped us achieve 0.98 mAP on our test dataset. The entire image generation pipeline and model training can be accomplished within a short time enabling fast deployment.

We have also analyzed the failure cases of our model and recommended potential techniques to improve the performance. The developed model can then be deployed online in a production setting to detect defects during a composite layup process. Such a robust defect detection method can aid in the adaptive control of robots, and significantly reduce defect introduction, boosting the overall process quality.

There are several fronts in which this work can be extended to further improve defect detection capabilities using deep learning. We will discuss them in detail here.

Synthetic Image Generation: The idea of combining physics based simulation and advanced CGI to generate synthetic data can be further extended such that independent generative models can be implemented to achieve the proposed capabilities. Although GANs in its conventional form cannot be directly used for this application, we can use ensemble of networks where multiple networks can be deployed. We can have subgroups of networks capturing the physics of the sheet as well as the sheet's texture and physical appearance. Such a physics aware generative model can further improve the photo-realism of the synthetic data without incurring extremely high computational cost. Ideas from physics aware image restoration can be utilized to achieve this capability [59].

Data Preparation: Currently, since the training process is completely supervised, data preparation becomes a strenuous task. For scalability of deep learning models, self-supervision is an important attribute. Recently there has been significant progress in self-supervised image segmentation [**?**,60,61]. Inspiration can be taken from similar work to execute learning in a self/semi-supervised fashion hence resolving the data preparation issue. Additionally, there is a potential to identify a wrinkle based defect based on energy calculations from the simulator as the nodes of the mesh belonging to a wrinkled region might posses different elastic energy. This information can be used for annotation of synthetic data.

Transfer Learning: Currently the defect detection framework has been tested for a single type of composite prepreg sheet with a specific texture. Although high level features such as color, surface texture, etc. are similar for several carbon fiber sheets, finer features such as fiber angles, weave pattern, etc. still differ significantly across applications and type of carbon fiber used. Hence, a transfer learning methodology can be adapted where the model can be trained to be indifferent to these finer details and can be easily used even if the type of carbon fiber sheet changes. Additionally, sheet like deformable material is used across several other manufacturing process, a truly scalable model should be able to generalize itself across different processes and material properties.

References

[1] Jie, L., Siwei, L., Qingyong, L., Hanqing, Z., and Shengwei, R., 2009. "Real-time rail head surface defect detection: A geometrical approach". In *IEEE International Symposium on Industrial Electronics*, pp. 769–774.

[2] Cui, Y., Jin, J. S., Luo, S., Park, M., and Au, S. S., 2009. "Automated pattern recognition and defect inspection system". In *Fifth International Conference on Image and Graphics*.

[3] Iivarinen, J., 2000. "Surface defect detection with histogram-based texture features". In *SPIE Optics East*.

[4] Xue-wu, Z., Yan-qiong, D., Yan-yun, L., Ai-ye, S., and Rui-yu, L., 2011. "A vision inspection system for the surface defects of strongly reflected metal based on multi-class svm". *Expert Syst. Appl.,* ***38***(5), May, p. 5930–5939.

[5] Masci, J., Meier, U., Ciresan, D., Schmidhuber, J., and Fricout, G., 2012. "Steel defect classification with max-pooling convolutional neural networks". In *The 2012 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–6.

[6] Yun, J. P., Choi, S., ju Jeon, Y., chul Choi, D., and Kim, S. W., 2008. "Detection of line defects in steel billets using undecimated wavelet transform". In *2008 International Conference on Control, Automation and Systems*, pp. 1725–1728.

[7] Aziz, M. A., Haggag, A. S., and Sayed, M. S., 2013. "Fabric defect detection algorithm using morphological processing and dct". In *2013 1st International Conference on Communications, Signal Processing, and their Applications (ICCSPA)*, pp. 1–4.

[8] Mei, H., Jiang, H., Yin, F., Wang, L., and Farzaneh, M., 2021. "Terahertz imaging method for composite insulator defects based on edge detection algorithm". *IEEE Transactions on Instrumentation and Measurement,* ***70***, pp. 1–10.

[9] Bhatt, P. M., Malhan, R. K., Rajendran, P., Shah, B. C., Thakar, S., Yoon, Y. J., and Gupta, S. K., 2021. "Image-based surface defect detection using deep learning: A review". *Journal of Computing and Information Science in Engineering,* ***21***(4), Feb.

[10] He, Y., Song, K., Meng, Q., and Yan, Y., 2020. "An end-to-end steel surface defect detection approach via fusing multiple hierarchical features". *IEEE Transactions on Instrumentation and Measurement,* ***69***(4), pp. 1493–1504.

[11] Li, F., Li, F., and Xi, Q., 2021. "Defectnet: Toward fast and effective defect detection". *IEEE Transactions on Instrumentation and Measurement,* ***70***, pp. 1–9.

[12] Xu, L., Lv, S., Deng, Y., and Li, X., 2020. "A weakly supervised surface defect detection based on convolutional neural network". *IEEE Access,* ***8***, pp. 42285–42296.

[13] Sreedhar, U., Krishnamurthy, C., Balasubramaniam, K., Raghupathy, V., and Ravisankar, S., 2012. "Automatic defect identification using thermal image analysis for online weld quality monitoring". *Journal of Materials Processing Technology,* ***212***(7), pp. 1557–1566.

[14] Manyar, O. M., Junyan, C., Levine, R., Krishnan, V., Barbic, J., and Gupta, S. K., 2022. "A synthetic image assisted deep learning framework for detecting defects during composite prepreg layup". *ASMEs 2022 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference. St. Louis, MO, USA*.

[15] Li, Y., Cheng, Y., Gan, Z., Yu, L., Wang, L., and Liu, J., 2020. "Bachgan: High-resolution image synthesis from salient object layout". In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

[16] Tremblay, J., Prakash, A., Acuna, D., Brophy, M., Jampani, V., Anil, C., To, T., Cameracci, E., Boochoon, S., and Birchfield, S., 2018. "Training deep networks with synthetic data: Bridging the reality gap by domain randomization". In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pp. 969–977.

[17] Lin, J., Zhang, R., Ganz, F., Han, S., and Zhu, J.-Y., 2021. "Anycost gans for interactive image synthesis and editing". In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 14986–14996.

[18] Schraml, D., 2019. "Physically based synthetic image generation for machine learning: a review of pertinent literature". In Photonics and Education in Measurement Science 2019, B. Zagar, P. Mazurek, M. Rosenberger, and P.-G. Dittrich, eds., SPIE.

[19] Figueira, A., and Vaz, B., 2022. "Survey on synthetic data generation, evaluation methods and gans". *Mathematics,* ***10***(15).

[20] Malhan, R. K., Shembekar, A. V., Kabir, A. M., Bhatt, P. M., Shah, B., Zanio, S., Nutt, S., and Gupta, S. K. "Automated planning for robotic layup of composite prepreg". *Robotics and Computer-Integrated Manufacturing,* ***67***, p. 102020.

[21] He, K., Gkioxari, G., Dollár, P., and Girshick, R., 2017. "Mask r-cnn". In *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2980–2988.

[22] Kusano, M., Hatano, H., Watanabe, M., Takekawa, S., Yamawaki, H., Oguchi, K., and Enoki, M., 2018. "Mid-infrared pulsed laser ultrasonic testing for carbon fiber reinforced plastics". *Ultrasonics,* ***84***, pp. 310–318.

[23] Elrefai, A. L., and Sasada, I., 2016. "Magnetic particle detection system using fluxgate gradiometer on a permalloy shielding disk". *IEEE Magnetics Letters,* ***7***, pp. 1–4.

[24] D'Angelo, G., Laracca, M., Rampone, S., and Betta, G., 2018. "Fast eddy current testing defect classification using lissajous figures". *IEEE Transactions on Instrumentation and Measurement,* ***67***(4), pp. 821–830.

[25] Malhan, R. K., Kabir, A. M., Shah, B., Centea, T., and Gupta, S. K., 2018. "Hybrid cells for multi-layer prepreg composite sheet layup.". *IEEE International Conference on Automation Science and Engineering (CASE). Munich, Germany.*

[26] Malhan, R. K., Kabir, A. M., Shah, B. C., and Gupta, S. K., 2019. "Identifying feasible workpiece placement with respect to redundant manipulator for complex manufacturing tasks.". *IEEE International Conference on Robotics and Automation (ICRA). Montreal, Canada.*

[27] Malhan, R. K., Kabir, A. M., Shah, B., Centea, T., and Gupta, S. K., 2019. "Determining feasible robot placements in robotic cells for composite prepreg sheet layup". *ASMEs 14th Manufacturing Science and Engineering Conference. Erie, PA, USA*.

[28] Chen, Y., Joseph, R. J., Kanyuck, A., Khan, S., Malhan, R. K., Manyar, O. M., McNulty, Z., Wang, B., Barbic, J., and Gupta, S. K., 2021. "A digital twin for automated layup of prepreg composite sheets". *ASMEs 16th Manufacturing Science and Engineering Conference. Virtual, Online*.

[29] Manyar, O. M., Desai, J., Deogaonkar, N., Joesph, R. J., Malhan, R., McNulty, Z., Wang, B., Barbič, J., and Gupta, S. K., 2021. "A simulation-based grasp planner for enabling robotic grasping during composite sheet layup". In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 930–937.

[30] Elkington, M., Almas, E., Ward-Cherrier, B., Pestell, N., Lloyd, J., Ward, C., and Lepora, N., 2021. "Real time defect detection during composite layup via tactile shape sensing". *Science and Engineering of Composite Materials,* ***28***(1), Jan., pp. 1–10.

[31] Kumar, A., 2008. "Computer-vision-based fabric defect detection: A survey". *IEEE Transactions on Industrial Electronics,* ***55***(1), pp. 348–363.

[32] Hu, C., and Wang, Y., 2019. "An efficient convolutional neural network model based on object-level attention mechanism for casting defect detection on radiography images". *IEEE Transactions on Industrial Electronics,* ***67***, 12, pp. 10922 – 10930.

[33] Wang, T., Chen, Y., Qiao, M., and Snoussi, H., 2018. "A fast and robust convolutional neural network-based defect detection model in product quality control". *The International Journal of Advanced Manufacturing Technology,* ***94***(9), pp. 3465–3471.

[34] Tabernik, D., Šela, S., Skvarč, J., and Skočaj, D., 2020. "Segmentation-based deep-learning approach for surface-defect detection". *Journal of Intelligent Manufacturing,* ***31***(3), pp. 759–776.

[35] Wang, S., Xia, X., Ye, L., and Yang, B., 2021. "Automatic detection and classification of steel surface defect using deep convolutional neural networks". *Metals,* ***11***(3), Feb., p. 388.

[36] Rudolph, M., Wandt, B., and Rosenhahn, B., 2020. "Same same but differnet: Semi-supervised defect detection with normalizing flows". *CoRR,* ***abs/2008.12577***.

[37] Yang, J., Li, S., Wang, Z., Dong, H., Wang, J., and Tang, S., 2020. "Using deep learning to detect defects in manufacturing: A comprehensive survey and current challenges". *Materials,* ***13***(24).

[38] Xu, X., Zheng, H., Guo, Z., Wu, X., and Zheng, Z., 2019. "Sdd-cnn: Small data-driven convolution neural networks for subtle roller defect inspection". *Applied Sciences,* ***9***(7), p. 1364.

[39] Zhao, Z., Li, B., Dong, R., and Zhao, P., 2018. "A surface defect detection method based on positive samples". In *Lecture Notes in Computer Science*. Springer International Publishing, pp. 473–481.

[40] Chow, J., Su, Z., Wu, J., Tan, P., Mao, X., and Wang, Y., 2020. "Anomaly detection of defects on concrete structures with the convolutional autoencoder". *Advanced Engineering Informatics,* ***45***, p. 101105.

[41] Li, Y., Zhao, W., and Pan, J., 2017. "Deformable patterned fabric defect detection with fisher criterion-based deep learning". *IEEE Transactions on Automation Science and Engineering,* ***14***(2), pp. 1256–1264.

[42] Meister, S., Möller, N., Stüve, J., and Groves, R. M., 2021. "Synthetic image data augmentation for fibre layup inspection processes: Techniques to enhance the data set". *Journal of Intelligent Manufacturing,* ***32***(6), Feb., pp. 1767–1789.

[43] Sin, F. S., Schroeder, D., and Barbič, J., 2013. "Vega: Nonlinear FEM deformable object simulator". *Computer Graphics,* ***32***(1), pp. 38–50.

[44] Community, B. O., 2018. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam.

[45] Simonyan, K., and Zisserman, A., 2015. "Very deep convolutional networks for large-scale image recognition". In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, eds.

[46] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A., 2015. "Going deeper with convolutions". In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–9.

[47] Huang, G., Liu, Z., van der Maaten, L., and Weinberger, K. Q., 2017. "Densely connected convolutional networks". In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[48] Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., and Belongie, S., 2017. "Feature pyramid networks for object detection". In 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 936–944.

[49] Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L., 2014. "Microsoft coco: Common objects in context". In *Computer Vision – ECCV 2014*, Springer International Publishing, pp. 740–755.

[50] Chen, K., Wang, J., Pang, J., Cao, Y., Xiong, Y., Li, X., Sun, S., Feng, W., Liu, Z., Xu, J., Zhang, Z., Cheng, D., Zhu, C., Cheng, T., Zhao, Q., Li, B., Lu, X., Zhu, R., Wu, Y., Dai, J., Wang, J., Shi, J., Ouyang, W., Loy, C. C., and Lin, D., 2019. "MMDetection: Open mmlab detection toolbox and benchmark". *arXiv preprint arXiv:1906.07155*.

[51] Tzeng, E., Devin, C., Hoffman, J., Finn, C., Abbeel, P., Levine, S., Saenko, K., and Darrell, T., 2020. "Adapting deep visuomotor representations with weak pairwise constraints". In *Algorithmic Foundations of Robotics XII*. Springer, pp. 688–703.

[52] Pashevich, A., Strudel, R., Kalevatykh, I., Laptev, I., and Schmid, C., 2019. "Learning to augment synthetic images for sim2real policy transfer". *CoRR,* ***abs/1903.07740***.

[53] Sajjan, S., Moore, M., Pan, M., Nagaraja, G., Lee, J., Zeng, A., and Song, S., 2020. "Clear grasp: 3d shape estimation of transparent objects for manipulation". In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, pp. 3634–3642.

[54] Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A., 2010. "The pascal visual object classes (voc) challenge". *International Journal of Computer Vision,* ***88***(2), June, pp. 303–338.

[55] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S., 2019. "Pytorch: An imperative style, high-performance deep learning library". In *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, eds. Curran Associates, Inc., pp. 8024–8035.

[56] Zhang, H., Wu, C., Zhang, Z., Zhu, Y., Zhang, Z., Lin, H., Sun, Y., He, T., Muller, J., Manmatha, R., Li, M., and Smola, A., 2020. "Resnest: Split-attention networks". *arXiv preprint arXiv:2004.08955*.

[57] Cai, Z., and Vasconcelos, N., 2021. "Cascade r-cnn: High quality object detection and instance segmentation". *IEEE Transactions on Pattern Analysis and Machine Intelligence,*

43(5), pp. 1483–1498.

[58] Ganaie, M. A., Hu, M., Tanveer, M., and Suganthan, P. N., 2021. "Ensemble deep learning: A review". *CoRR,* ***abs/2104.02395***.

[59] Pan, J., Dong, J., Liu, Y., Zhang, J., Ren, J., Tang, J., Tai, Y.-W., and Yang, M.-H., 2021. "Physics-based generative adversarial models for image restoration and beyond". *IEEE Transactions on Pattern Analysis and Machine Intelligence,* ***43***(7), pp. 2449–2462.

[60] Hoyer, L., Dai, D., Chen, Y., Koring, A., Saha, S., and Van Gool, L., 2021. "Three ways to improve semantic segmentation with self-supervised depth estimation". In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11130–11140.

[61] Ibrahim, M. S., Vahdat, A., Ranjbar, M., and Macready, W. G., 2020. "Semi-supervised semantic image segmentation with self-correcting networks". In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12715–12725.