Failure Detection and Prevention for Additive Manufacturing

Christopher J. Coley* and Judson T. Babcock†
United States Air Force Academy, Colorado, 80840

Additive manufacturing, more commonly referred to as 3d printing, is currently an open-loop process. Even for a perfectly-prepared print, random variations in input conditions can cause failure modes to occur hours or days into a print job. Common failure modes can include separation of the part from the print bed, a clogged print nozzle, or the print head crashing into a malformed piece of the model. Currently, the only way to prevent failures is for a human to close the loop by visually inspecting the print periodically throughout the job. Without human intervention (either in person or through remote monitoring), a failed print can waste hours or days of time and cost the user up to $50 per print job depending on the amount and type of filament wasted. The current work describes a system that automates the detection of print failures and, upon detection of a failure state, automatically pauses the print job and notifies the user. This system has been implemented and tested at the United States Air Force Academy Aeronautics Laboratory and is demonstrated to reduce waste, save time, and improve overall efficiency in the fabrication process. The system is comprised of commercial-of-the-shelf components, making it simple to construct and implement a similar system in other facilities.

I. Introduction

Current additive manufacturing methods require three basic steps: creation of the digital CAD model, preparing the model for printing, and printing. Preparing the model is commonly called "slicing" and relies on various open-source software packages. During this step, the solid CAD model is discretized or "sliced" into 2D layers that will be printed sequentially and layered in the vertical direction to create the 3D part. The user can specify hundreds of parameters that will influence the composition of the final printed object such as layer height, infill percentage, infill pattern, overhang supports, brim, and more. After slicing, the printed part can be viewed at any sliced layer height, giving the user a preview of the part's size and shape at any point during the printing process. Finally, the slicing software combines the geometry, slicing settings, and specific printer settings to create the final g-code which can be sent to the printer.

When the user starts the print job, the 3D printer initializes and begins extruding filament in the shape of the first 2D sliced layer. This step is critical because the distance from the print bed to the nozzle for the first layer typically has to be manually calibrated. The first layer must adhere to the bed properly so the object does not move during the rest of the print. Once the first layer is complete, subsequent layers are printed with the vertical layer height specified by the user. Support materials are also built layer by layer as determined during the slicing process to provide a platform for any overhangs in the part's geometry. In each layer, the outer wall(s) are printed first followed by the infill according to the infill pattern and percentage specified during the slicing process. As the print progresses, a continuous filament of plastic is pulled off a spool and pushed into the hot end by two gears at a carefully regulated feed rate. Finally, as the print head reached the top of the part, plastic is extruded over the entire upper surface to enclose the part completely. Once the printer finished the final layer, the print heads parks in a predetermined position and the user may remove the part.

Common failure states can occur through this process. The most common occurs when the part prematurely separates from the print bed. If the distance of the first layer is too high or low, improper adhesion will cause the part to separate or slide around due to the slight force that the print head imparts as it draws a bead of plastic across a layer. Once that happens, subsequent 2D layers will not have the preceding layer to build upon and the print job will extrude a bead of plastic into the air for the remainder of the print resulting in an unwelcome "spaghetti ball" of plastic. An example of spaghetti in a print is shown in Fig. 1. Considering that some print jobs can last for 72 or 96 hours, this can result in a great deal of wasted time and filament. This failure state can also be caused by an improper support geometry. If the user attempts to print challenging overhang angles without support, the plastic can curl up as it cools and interfere with the print head, causing the print head to forcibly separate the part from the print bed.

Another failure state commonly occurs when the filament becomes clogged in the nozzle. This can result from dust or dirt collecting on the spool of filament, trying to use old filament which has absorbed moisture or become brittle,

*Associate Professor, christopher.coley@afacademy.af.edu, Department of Aeronautics
†Assistant Professor, judson.babcock@afacademy.af.edu, Department of Aeronautics

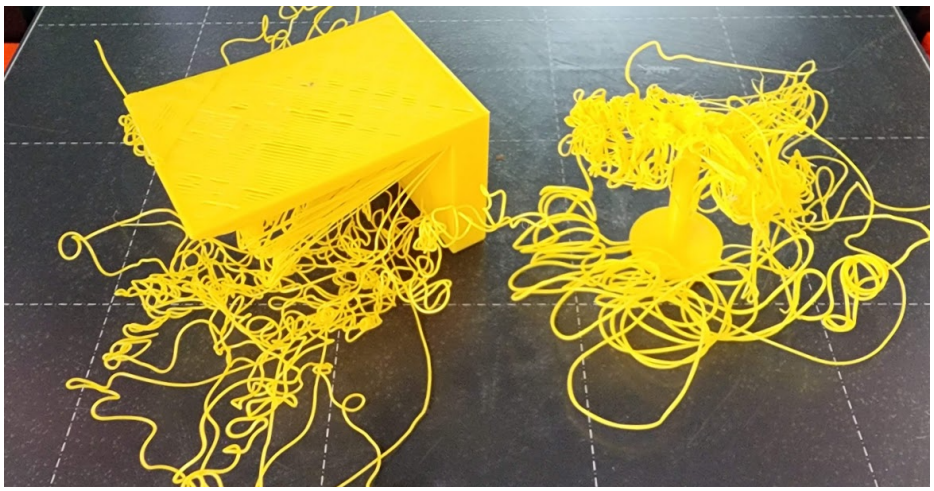


Fig. 1 An example of spaghetti in a 3D print

or from poor printer maintenance. In this failure state, the printer continues to revolve the gears in order to extrude filament but the filament does not advance, instead becoming worn down to the point where the gears no longer contact the filament enough to force the extrusion. The print head continues to advance vertically through the layers but no plastic is extruded, resulting in a "ghost" part which does not exist despite the printer continuing through its open-loop procedures. Although filament waste is not as great as other failure states, this failure can waste hours or days of print time and be very difficult to correct.

The Department of Aeronautics (DFAN) consumes over 100 kg of plastic annually in 500+ print jobs as 400 students progress through various engineering courses which require rapid prototyping and testing. One of these failure states happens approximately every 5 prints especially for students who are just learning. If successful, the proposed method could could prevent 100+ failed prints and save >1,000 hours and $500 of waste filament per year. If applied to stereo-lithography printing which consumes resin costing $3k/gallon, this method could save >$10k/year in material cost in DFAN alone.

The additive manufacturing process currently lacks sufficient automatic feedback to detect failure states which waste time and money. Research is required to develop and implement an automatic sensing technique to visually identify common failure states and automatically stop the print while notifying the user of the failure.

The current work describes a system that automates the detection of print failures and, upon detection of a failure state, automatically pauses the print job and notifies the user. This system has been implemented and tested at the United States Air Force Academy Aeronautics Laboratory and is demonstrated to reduce waste, save time, and improve overall efficiency in the fabrication process. The system is comprised of commercial-of-the-shelf components, making it simple to construct and implement a similar system in other facilities.

An outline of the remainder of the paper is as follows. In Section II, we describe the failure detection system, to include the hardware and software components. Next, in Section III, we present the he methodology used to test and evaluate the failure detection system and present results that demonstrate the system's effectiveness. Finally, in Section V, we make concluding remarks and discuss future work.

II. System Description

The failure detection system consists of the OctoPrint [1] printer controller software running the Obico [2] plugin for 3D print monitoring, AI failure detection, and user notifications. The software runs on a Raspberry Pi controller connected to the 3D printer through USB and relies on a standard webcam for image acquisition. The Obico plugin interacts with a local server connected to the Raspberry Pi through a wireless local-area network (WLAN). This setup is depicted in Fig. 2 and the specifications for the hardware used in the current build are listed in Table 1.

At the heart of the failure detection system is the Obico smart 3D printing platform. Obico allows for the monitoring and control of a 3D printer remotely through the use of a web client or smartphone app. Obico also sends notifications to the user, which is used in this work to send alerts related to 3D print failures. The notifications and alerts are sent

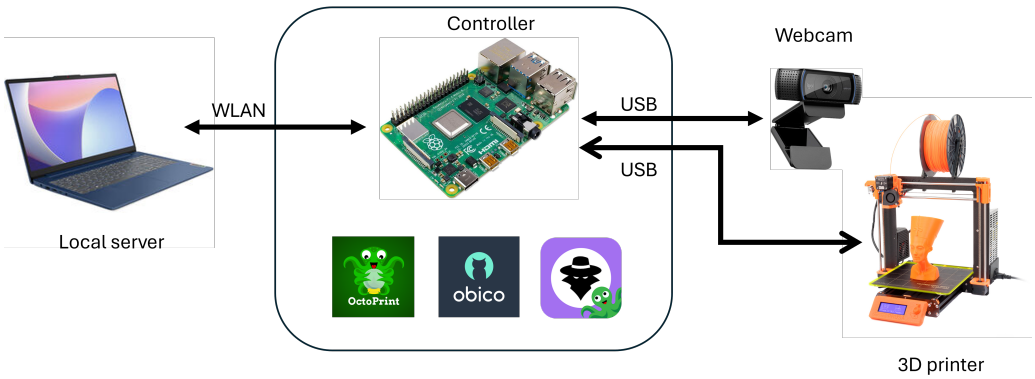


Fig. 2 System diagram

Table 1 Hardware specifications

Component	Model
3D Printer	Prusa i3 Mk3s+
Controller	Raspberry Pi 3, 32GB storage
Webcam	EZCam HD, 1080p resolution
Local server	Windows 11 PC, 32GB Ram, i7 processor

as emails through a GMail account in the current work. Obico incorporates AI-driven failure detection through the Spaghetti Detective, which is described in more detail in the next subsection. Obico is a plugin for the OctoPrint printer controller software.

A. Spaghetti Detective AI Failure Detection

The Spaghetti Detective is a neural network trained to detect 3D print failures in the form of spaghetti. The Spaghetti Detective is built from the You Only Look Once (YOLO) v2 (also known as YOLO9000) object detection network [3] and is implemented using the Darknet [4] neural network framework. YOLOv2 uses a single neural network to predict bounding boxes and class probabilities from images in one single evaluation. This is accomplished by conducting object detection as a regression problem. This is in contrast to other approaches such as R-CNN, which relies on classification within bounding boxes on an image. The details of the YOLOv2 architecture are described in [3].

III. Test and Evaluation

In this section, we detail the methodology used to test and evaluate the failure detection system and present results that demonstrate the system's effectiveness. A series of 7 test prints were designed to intentionally induce print failures and the system response is recorded for each of the failures.

A. Test Print 1: Cone, black filament

The first print is a cone which is designed to intentionally fall over due to a top-heavy part with a small contact area on the bed, resulting in insufficient adhesion. The CAD model for the part is shown in Fig. 3. The model is printed using black filament and the failure sensitivity setting is set to high.

Around layer 210 into the print, the print object fell over due to poor adhesion with the print bed. This in turn caused the printer to start producing spaghetti. At layer 218, the spaghetti is present but the failure detection indicates that there is no problem with the print. By layer 266, there is a considerable amount of spaghetti and the failure detection starts to indicate there might be a problem, but the system still does not reach the threshold for detecting a print failure. At layer 309, the system sends an alert that there might be a possible failure. Eventually, at layer 340, the system detects the presence of spaghetti and responds by pausing the print and sending a print failure alert email. The print evolution and system response is demonstrated in Figs. 4. Additionally, in Fig. 5, the final print object is shown along with the

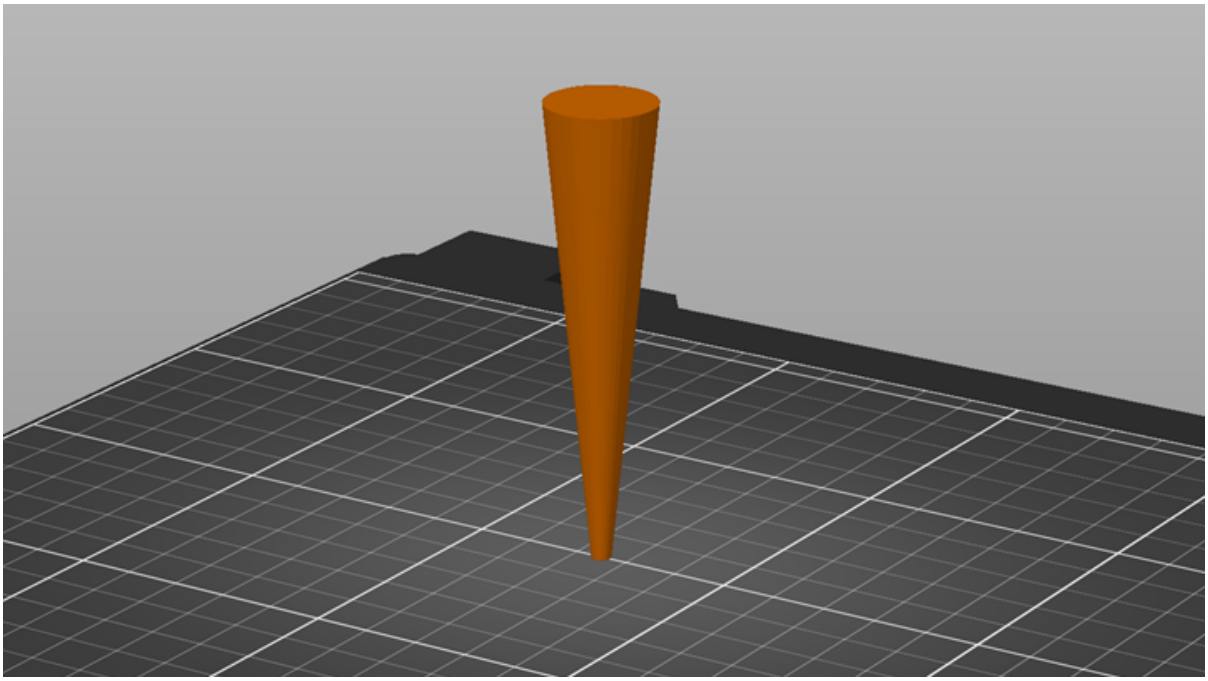
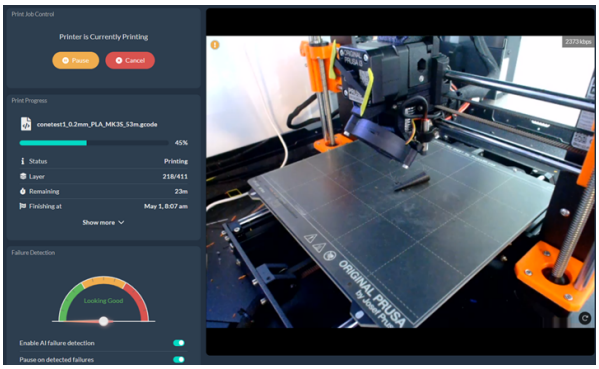
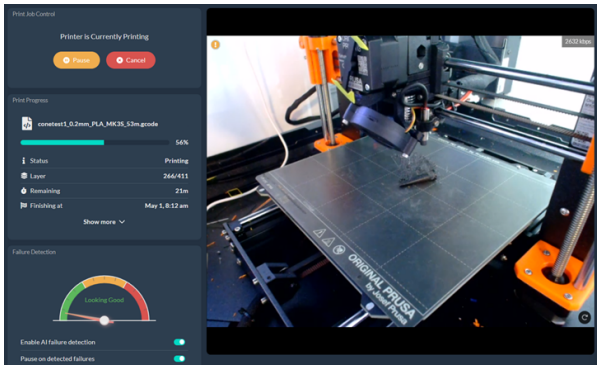


Fig. 3 CAD model for the cone test object

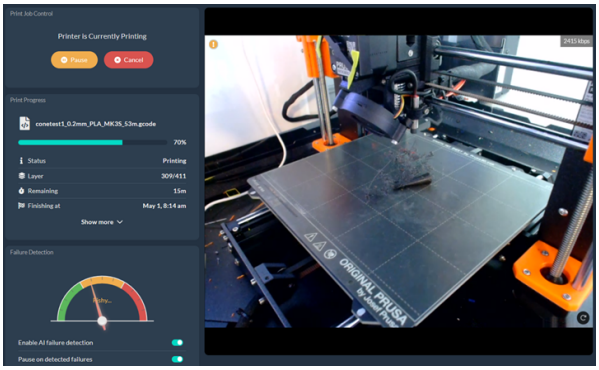
spaghetti detection regions as identified by the spaghetti detective AI.



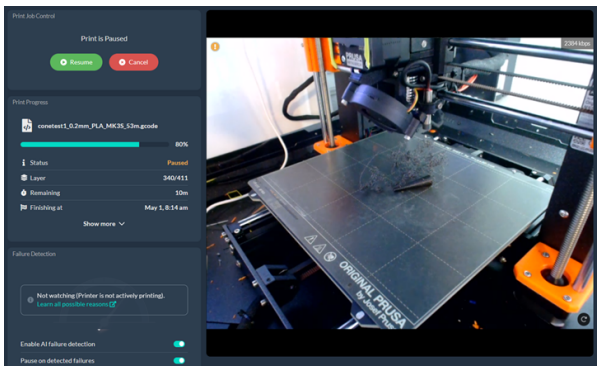
(a) Black filament test object at layer 218



(b) Black filament test object at layer 266



(c) Black filament test object at layer 309



(d) Black filament test object at layer 340

Fig. 4 Print evolution and failure detection system response for the black filament cone

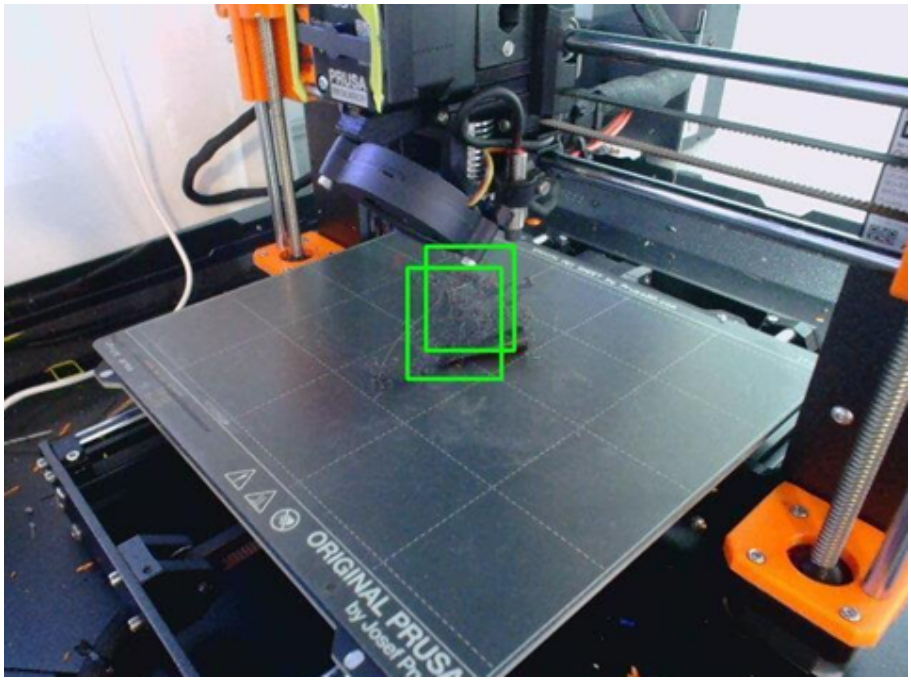


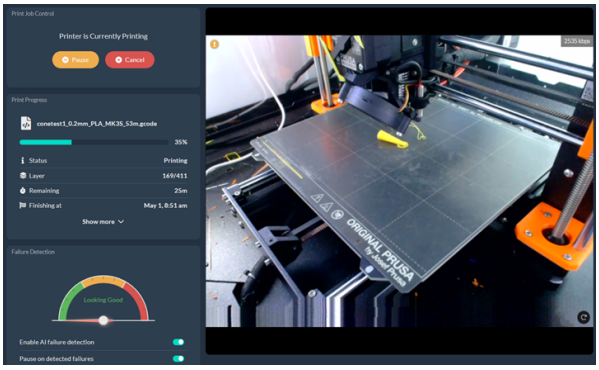
Fig. 5 Final print object with identified spaghetti detection regions for the black filament cone

This was an example of a failure due to poor adhesion to the print bed because of the part geometry, resulting in the part falling over and the printer creating spaghetti. It took a surprisingly long time for the spaghetti detection to pause the print (130 layers of printing continued after the part fell over and the first spaghetti formed). Even after the print was labeled "fishy" and the spaghetti was clearly detected, the print continued for about 31 more layers before automatically pausing. This was with the spaghetti detection in the "high" setting. The most likely reason for the delayed detection is the color of the filament resulting in low contrast between the filament and the background, making it difficult for the image processing to see the spaghetti.

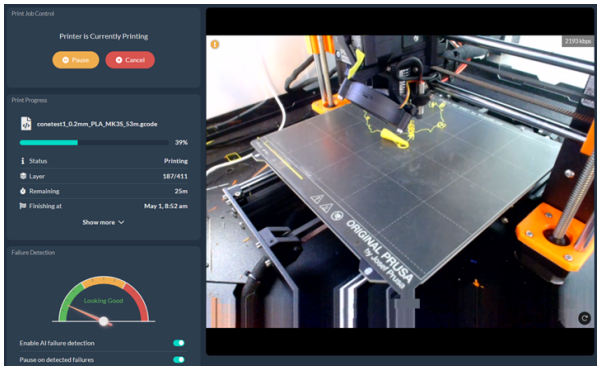
B. Test Print 2: Cone, yellow filament

The second print is the same cone geometry as the first test print, but is printed in yellow filament in order to provide better contrast between the printed part and the print bed.

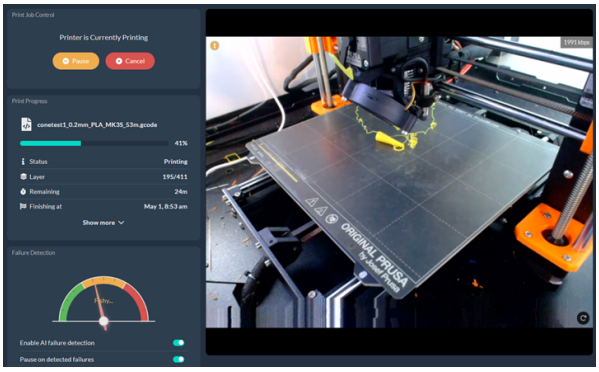
Similar to the black filament cone object, this object falls over at an early time during the print job. At layer 169, spaghetti is present but the failure detection indicates that there is no problem with the print. By layer 187, there is a considerable amount of spaghetti and the failure detection starts to indicate there might be a problem, but the system still does not reach the threshold for detecting a print failure. At layer 195, the system starts to indicate that there is something wrong. Eventually, at layer 216, the system detects the presence of spaghetti and responds by pausing the print and sending a print failure alert email. The print evolution and system response is demonstrated in Figs. 6. Additionally, in Fig. 7, the final print object is shown along with the spaghetti detection regions as identified by the spaghetti detective AI.



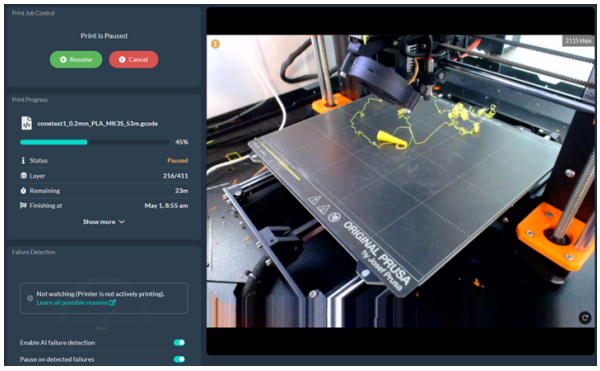
(a) Yellow filament test object at layer 169



(b) Yellow filament test object at layer 187



(c) Yellow filament test object at layer 195



(d) Yellow filament test object at layer 216

Fig. 6 Print evolution and failure detection system response for the yellow filament cone

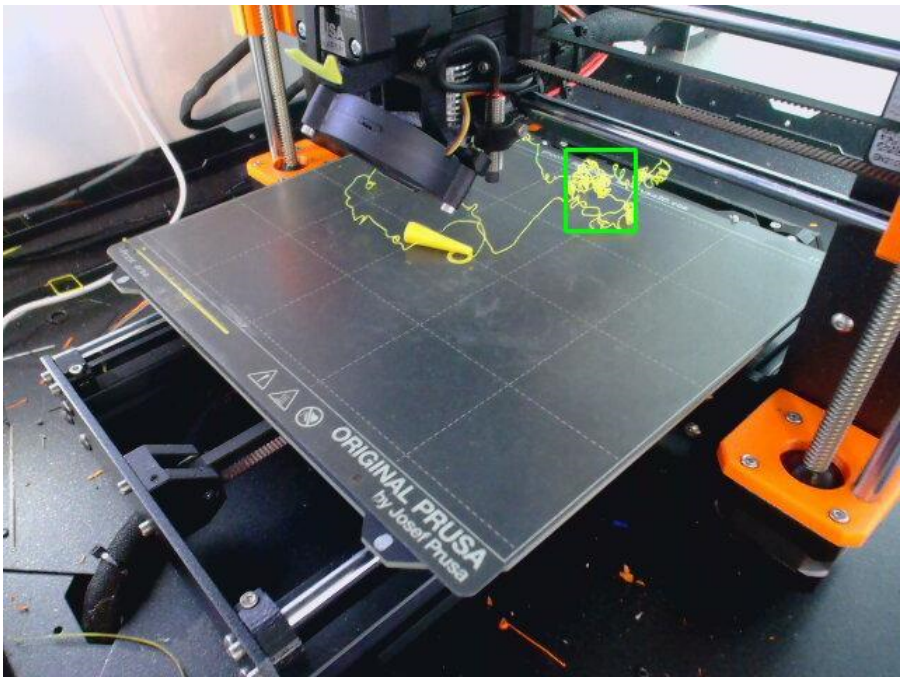


Fig. 7 Final print object with identified spaghetti detection regions for the yellow filament cone

This print failure was successfully detected and paused 47 layers after the failure occurred. Compared to test print 1, the yellow filament aided the image processing by providing a higher contrast between the filament and the background. This resulted in the AI failure detection pausing the print after 47 layers instead of 130 layers in test 1.

C. Test Print 3: Front overhang

The CAD model for the third test print is shown in Fig. 8. The goal of this test is to induce a spaghetti failure caused by a lack of support under an extreme overhang. The base of the model should stay attached to the bed (unlike in tests 1-2), and the spaghetti should form under the overhang. The overhang is oriented to the front of the plate and the camera is in the front corner.

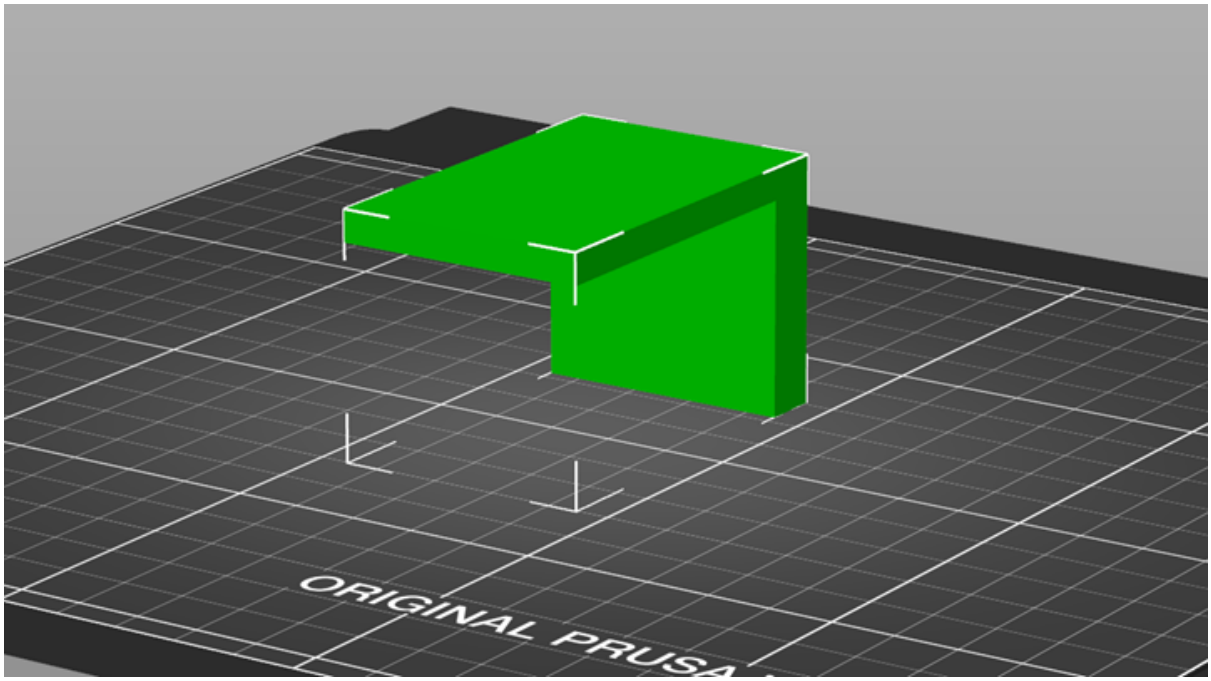
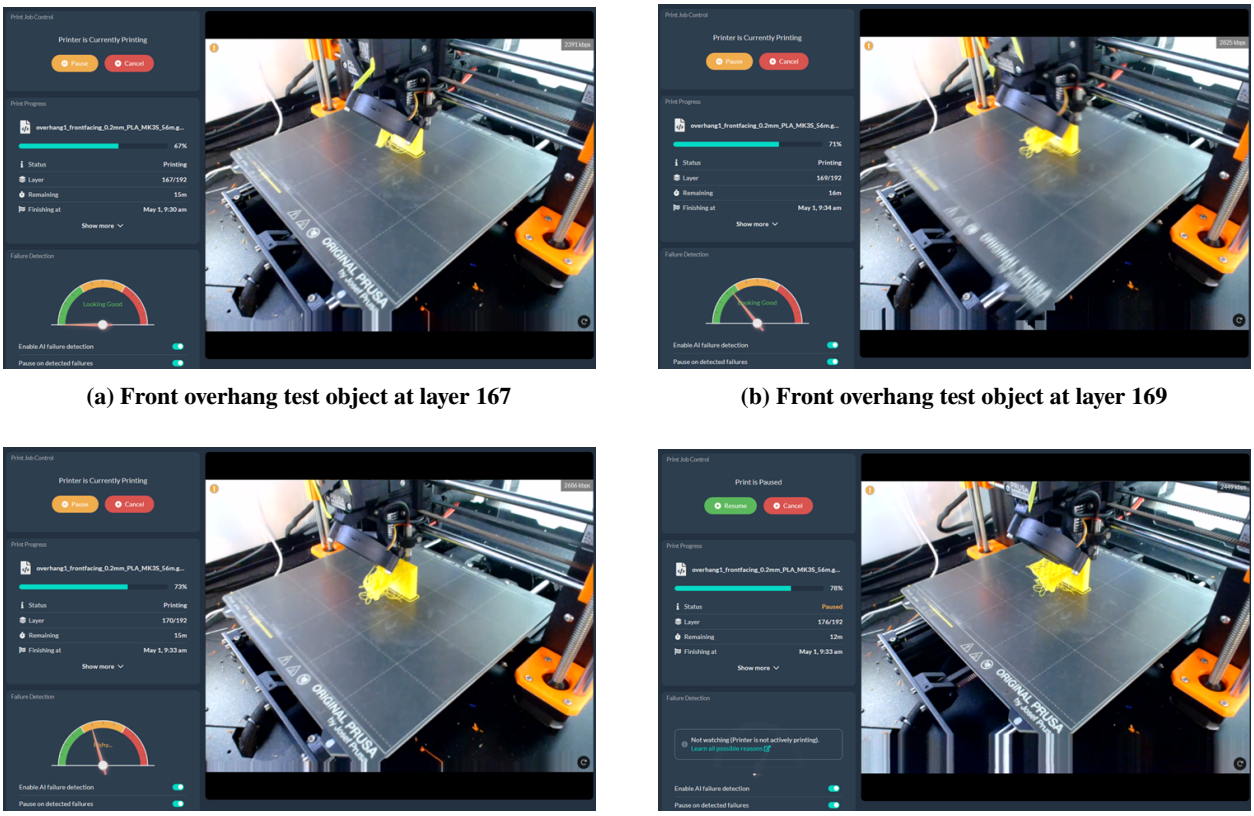


Fig. 8 CAD model for the front overhang test object

For the front overhang test object, the failure beings at layer 167, but the failure detection does not immediately indicate that there is a problem with the print. By layer 169, there is a considerable amount of spaghetti and the failure detection starts to indicate there might be a problem, but the system still does not reach the threshold for detecting a print failure. At layer 170, the failure indication increases. Finally, at layer 176, the system detects the presence of spaghetti and responds by pausing the print and sending a print failure alert. The print evolution and system response is demonstrated in Figs. 9. Additionally, in Fig. 10, the final print object is shown along with the spaghetti detection regions as identified by the spaghetti detective AI.



(a) Front overhang test object at layer 167

(b) Front overhang test object at layer 169

(c) Front overhang test object at layer 170

(d) Front overhang test object at layer 176

Fig. 9 Print evolution and failure detection system response for the front overhang object

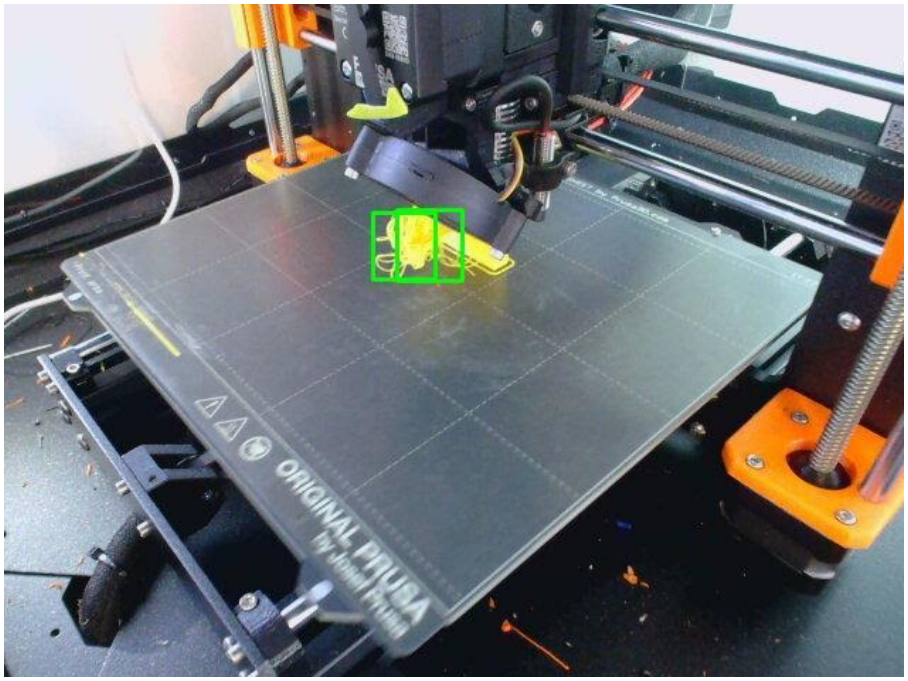


Fig. 10 Final print object with identified spaghetti detection regions for the front overhang object

A possible failure was detected 2 layers after the actual failure occurred, although the print was not automatically paused until 9 layers after the failure occurred. Overall the test was successful and the failure detection paused the print very soon after the failure started.

D. Test Print 4: Rear overhang

The fourth test print is the same geometry as test print 3, but the part is rotated so that the overhang is at the rear of the print bed. This test measures how the change in perspective for the camera affects the spaghetti detection. The CAD for this test print is shown in Fig. 11.

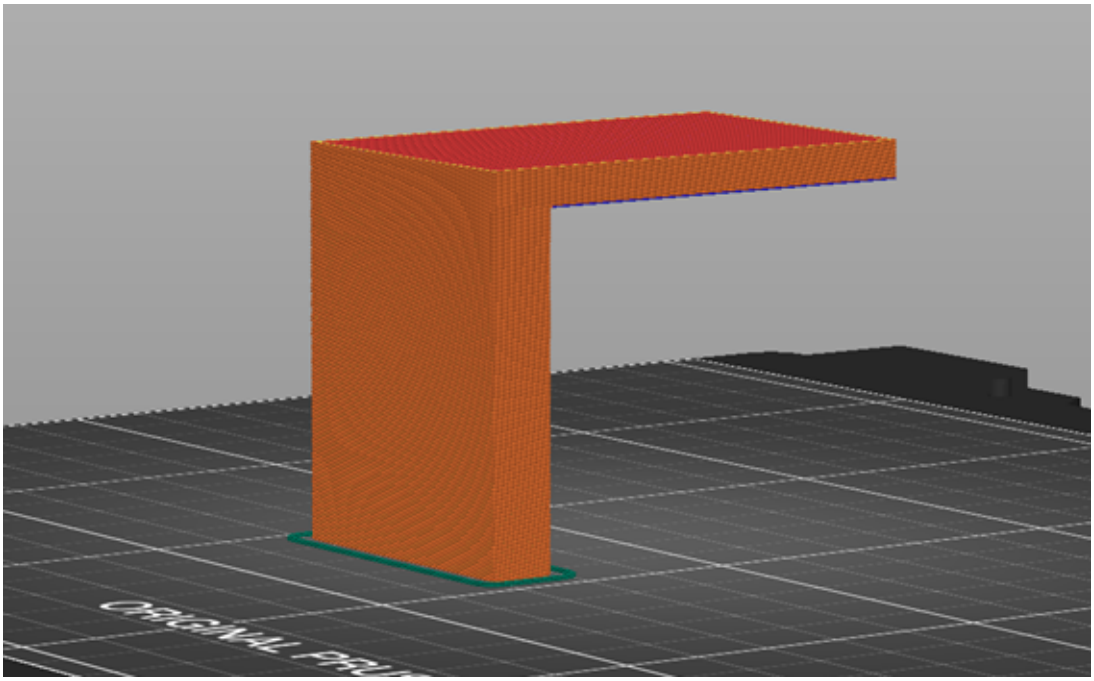
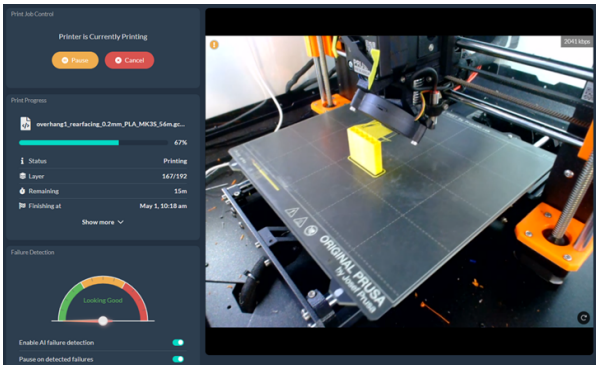
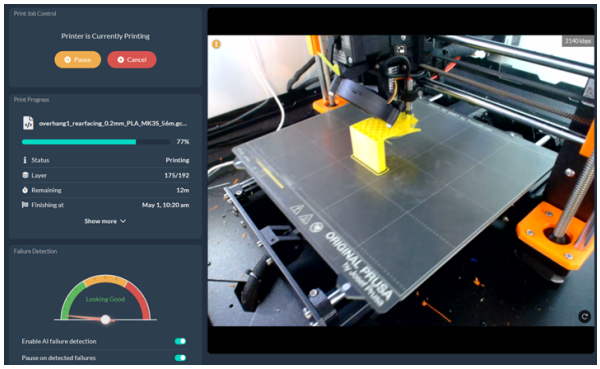


Fig. 11 CAD model for the rear overhang test object

For the rear overhang test object, there are obvious print quality issues as early as layer 167, but there is no spaghetti present yet and the print has not obviously failed. By layer 175, there is some visible spaghetti and the failure detection starts to indicate there might be a problem. However, the failure detection does not significantly increase until layer 187, after the object has separated from the print bed. Immediately after, while still on layer 187, the system confirms the presence of spaghetti and responds by pausing the print and sending a print failure alert. The print evolution and system response is demonstrated in Figs. 12. Additionally, in Fig. 13, the final print object is shown along with the spaghetti detection regions as identified by the spaghetti detective AI.



(a) Rear overhang test object at layer 167



(b) Rear overhang test object at layer 175

(c) Rear overhang test object at layer 187 (earlier time)

(d) Rear overhang test object at layer 187 (later time)

Fig. 12 Print evolution and failure detection system response for the rear overhang object

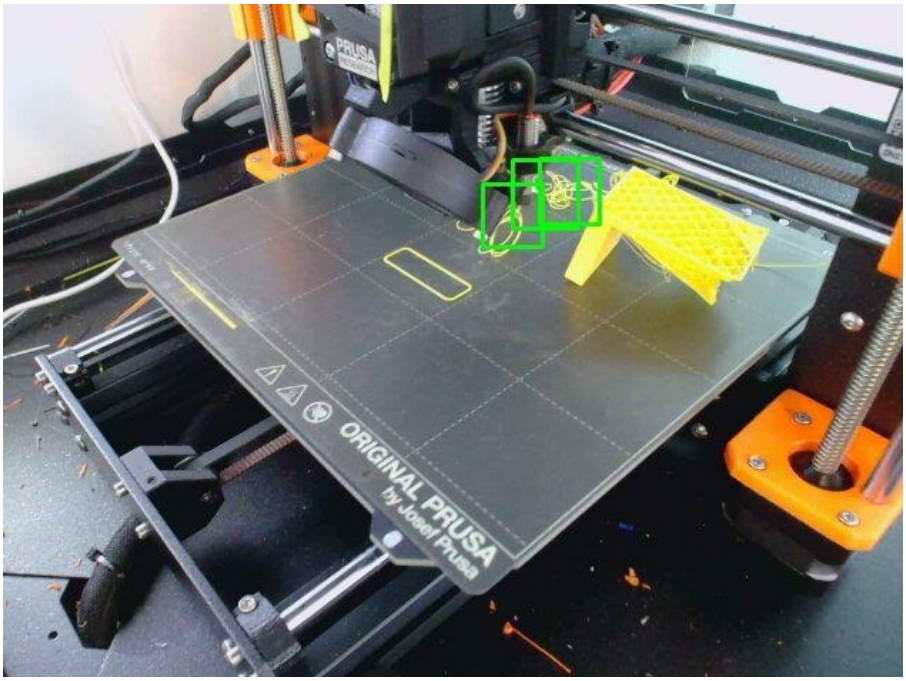


Fig. 13 Final print object with identified spaghetti detection regions for the rear overhang object

The first detection of a possible failure did not occur until layer 175, 8 layers after the failure began, and the print was not automatically paused until the object separated from the print bed. If the part had not separated, it is likely that the failure detection would not have identified the sagging layers on the rear side as a failure. The rear facing failure was harder to detect because less of the failure was visible to the camera.

E. Test Print 5: Spaghetti object

This test print is an object specifically designed to create spaghetti on every side of the print. The CAD for this test print is shown in Fig. 14.

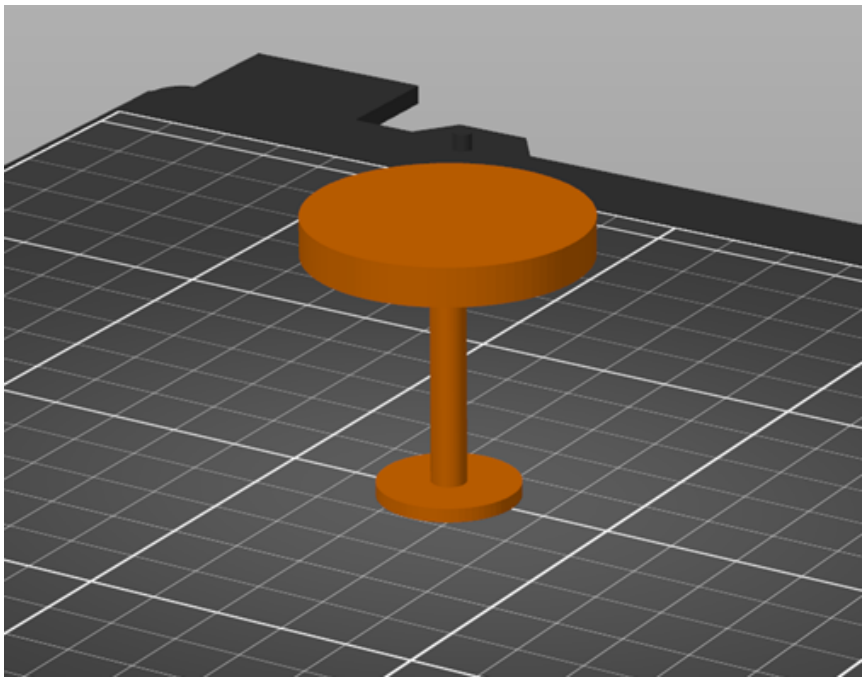


Fig. 14 CAD model for the spaghetti object

For the spaghetti object, the failure began on layer 175, but the failure detection system did not start to detect any issues until layer 198. This is shown in Figs. 15. Even though a possible failure was detected, the failure detection system did not pause the print job and it finished to completion despite the lack of support under the overhang.

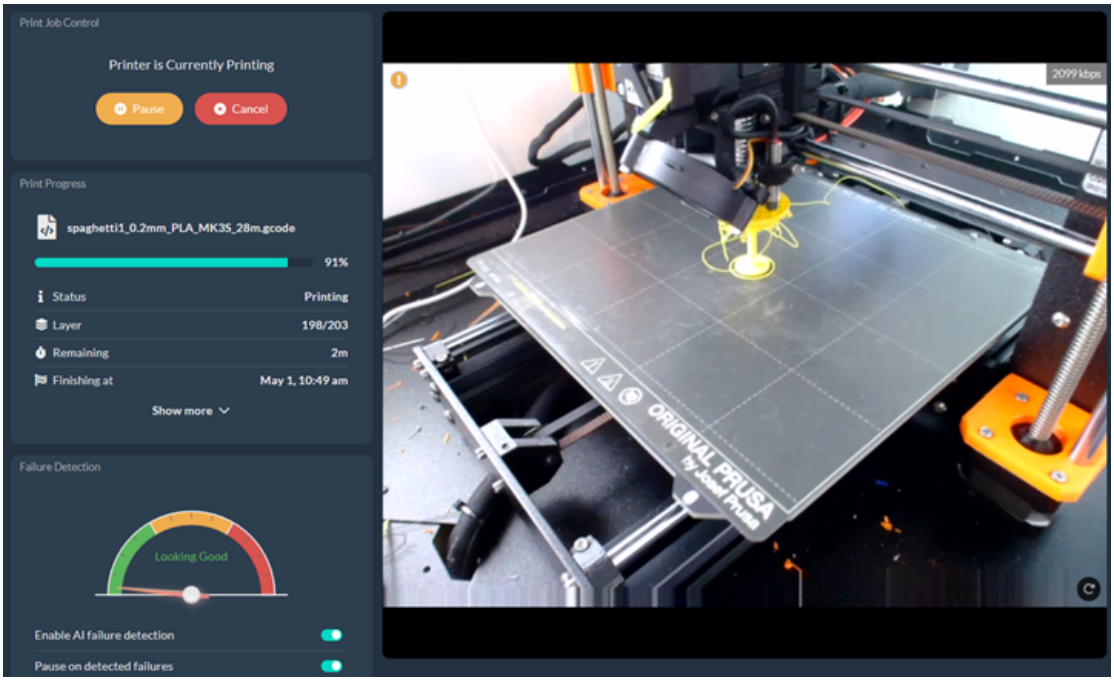


Fig. 15 Spaghetti object at layer 198

F. Test Print 6: Wider spaghetti object

This test print is similar to test print 5 but the diameter of the unsupported overhang is increased to increase the amount of spaghetti generated. The CAD for this test print is shown in Fig. 16.

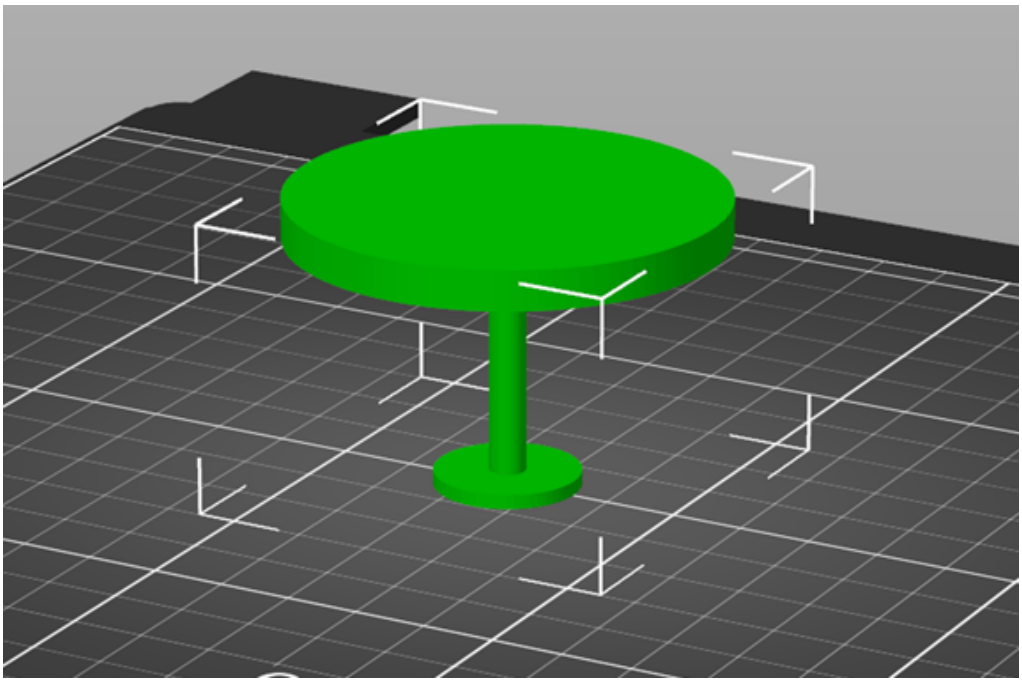
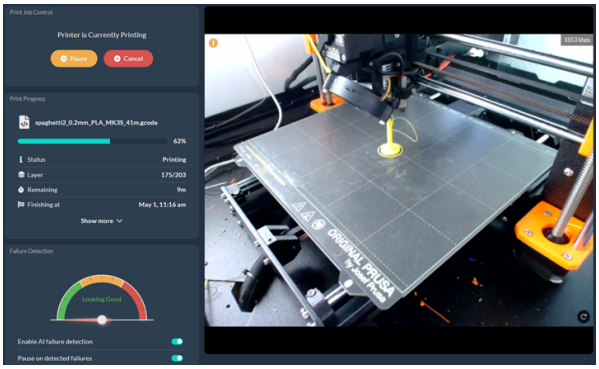
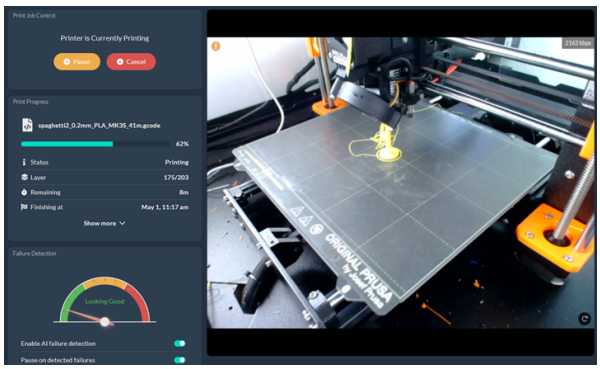


Fig. 16 CAD model for the wider spaghetti object

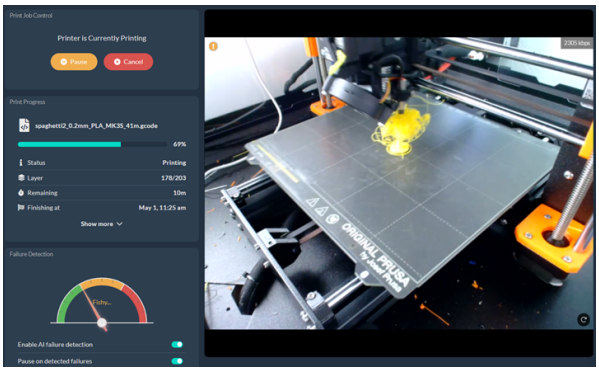
For the wider spaghetti object, the failure again starts at layer 175, but the failure detection does not immediately identify any problems with the print. Later, during the same layer, the spaghetti has increased and the failure detection starts to detect the possibility of a failure. The system does not indicate that there is a problem until layer 178, after the spaghetti has become quite considerable. Eventually, on layer 194 and after the object separates from the print bed, the system detects the presence of spaghetti and responds by pausing the print and sending a print failure alert. The print evolution and system response is demonstrated in Figs. 17. Additionally, in Fig. 18, the final print object is shown along with the spaghetti detection regions as identified by the spaghetti detective AI.



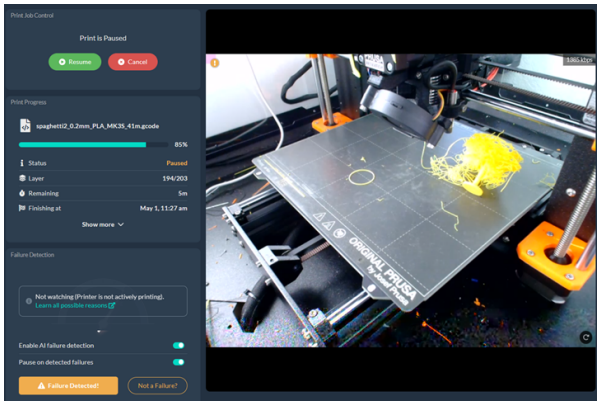
(a) Wider spaghetti object at layer 175 (earlier time)



(b) Wider spaghetti object at layer 175 (later time)



(c) Wider spaghetti object at layer 178



(d) Wider spaghetti object at layer 194

Fig. 17 Print evolution and failure detection system response for the wider spaghetti object

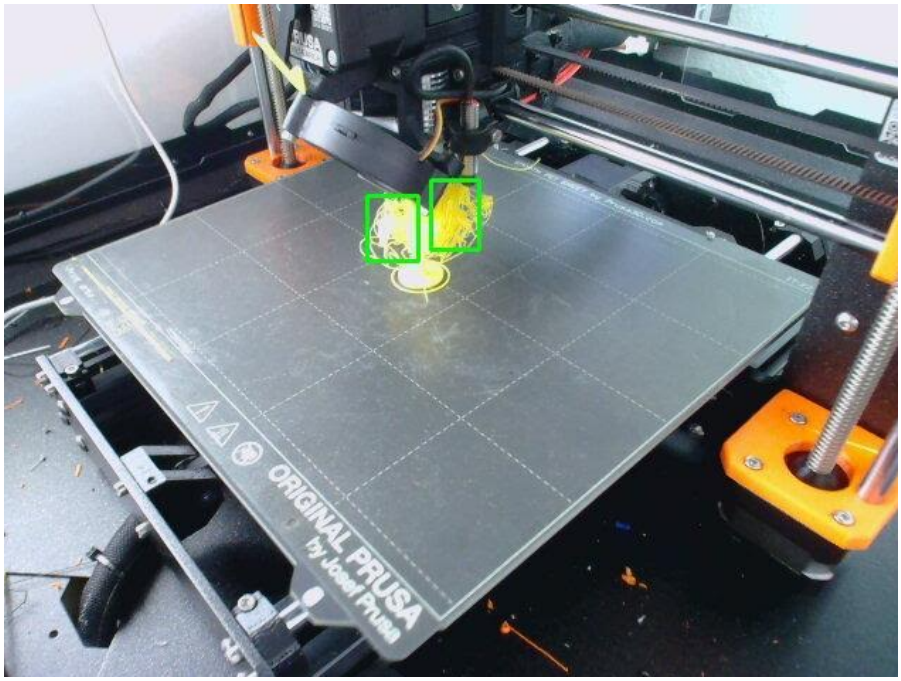


Fig. 18 Final print object with identified spaghetti detection regions for the wider spaghetti object

This test object created a lot of spaghetti and the failure detection identified the possible problem but was not sensitive enough to stop it until the object separated from the bed. This was with the failure sensitivity in the highest setting.

G. Test Print 7: Aircraft

This test is meant to simulate a poor student print. It does not have enough supports, especially around the tail in the beginning layers. The CAD for this test print is shown in Fig. 19.

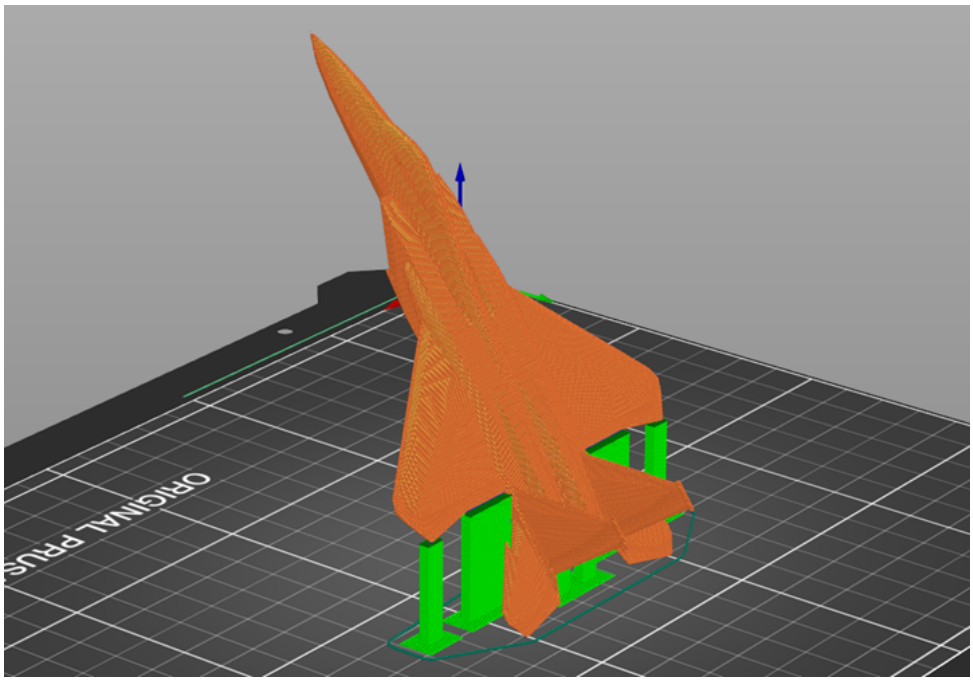
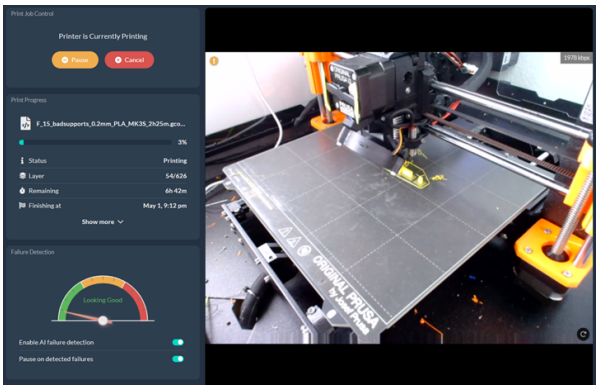
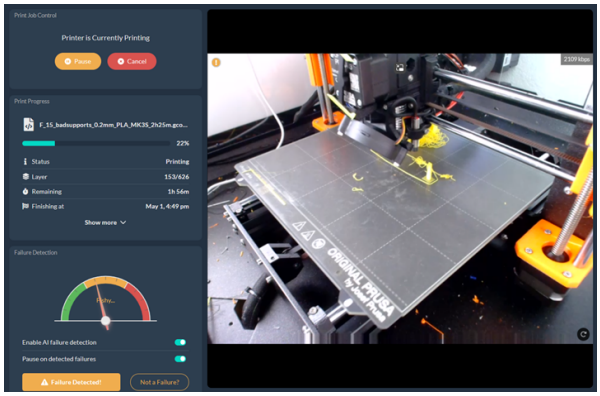


Fig. 19 CAD model for the aircraft

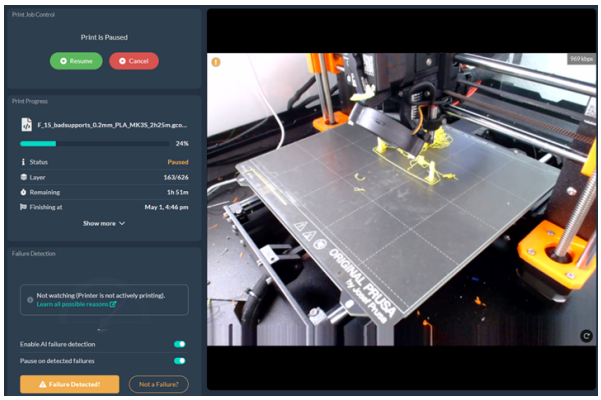
The lack of support on the tail results in stray pieces of filament very early in the print. By layer 54, the failure detection system has detected an abnormality. However, the print continues until layer 153 before the failure detection starts to detect there might be a problem. Finally, the detection system paused the print at layer 163, 109 layers after the first detection of the possible failure. At this time, the system responds by pausing the print and sending a print failure alert email. The print evolution and system response is demonstrated in Figs. 20. Additionally, in Fig. 21, the final print object is shown along with the spaghetti detection regions as identified by the spaghetti detective AI.



(a) Aircraft at layer 54



(b) Aircraft at layer 153



(c) Aircraft at layer 163

Fig. 20 Print evolution and failure detection system response for the aircraft

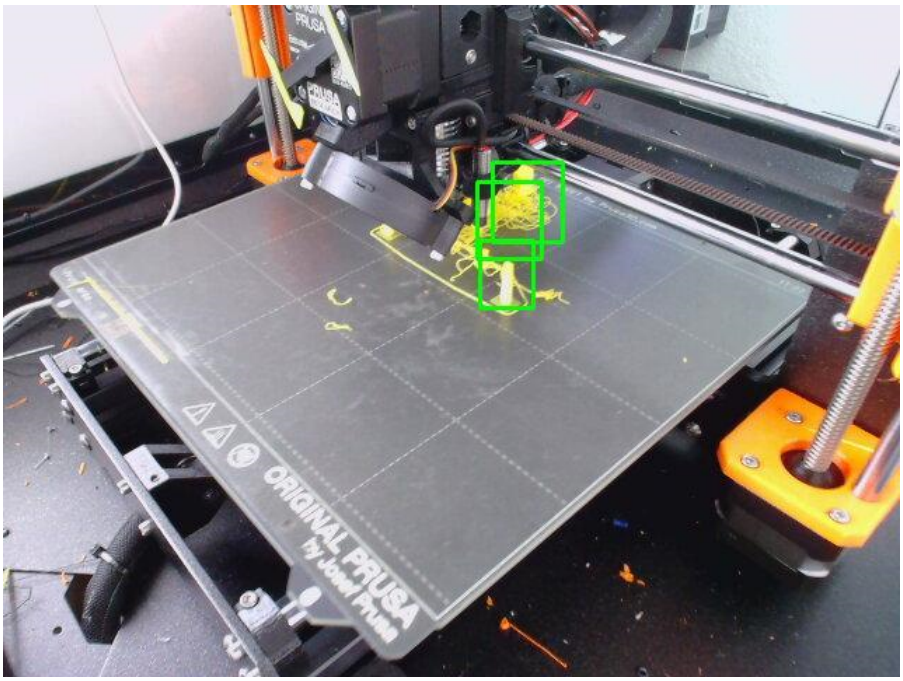


Fig. 21 Final print object with identified spaghetti detection regions for the aircraft

Most of the spaghetti generated in this failure seemed to stay behind the print head, from the camera's perspective. Perhaps this was caused by the print fan blowing in that direction. Unfortunately, that delayed the automatic failure detection from pausing the print until much later in the print. However, the print was still paused about 25% of the way through the job, saving about 75% of the print's potential waste. Overall the test was a success and this typical student error was prevented from printing completely through the job.

IV. Summary of Results

In summary, the failure detection system is demonstrated to perform admirably in the detection of spaghetti and taking appropriate intervening action to prevent filament waste. In all of the test cases provided, the Spaghetti Detective AI detected the presence of spaghetti when it was present. In a majority of the cases, the system not only detected the presence of spaghetti, but correctly recognized the need to pause the print and send alerts to the user. There are areas for improvement, however. The system has more difficulty detecting spaghetti when certain (dark) filament colors are used. Additionally, in some cases the spaghetti detective did not detect the failure until after the part separated from the bed, despite the fact that spaghetti was present for many layers prior to the separation. Additionally, the relative location of the camera to the spaghetti is a key factor which can affect the sensitivity of the detector (for example is most of the spaghetti is obscured by the moving print head). Overall, with the proper camera placement, lighting, focus, and filament color, the spaghetti detective is very effective at detecting and pausing failure prints.

V. Conclusions and Future Work

In this work, a 3D printing failure detection system is described, implemented, and evaluated. The system automates the detection of print failures and, upon detection of a failure state, automatically pauses the print job and notifies the user. This system has been implemented and tested at the United States Air Force Academy Aeronautics Laboratory and is demonstrated to reduce waste, save time, and improve overall efficiency in the fabrication process. It is estimated that this system will save the Department of Aeronautics 1000 hours of lost productivity and $10k in wasted material annually.

While the current system is demonstrated to be effective, there is future work which will enhance the capability of the system. First and foremost, the failure detection AI can be enhanced. This can take the form of enhanced training of the existing neural network, adoption of a new neural network (e.g. YOLOv3 [5], or development of an enhanced neural

network. Secondly, software updates to the Obico notification system will make it better suited for use in a facility such as the Aeronautics Laboratory. While the current software can monitor several 3D printers at once, it can only send notifications to a single user. Additional features to allow for a multi-user paradigm will improve the usage in the Aeronautics Laboratory. Finally, as the system developed here is comprised of commercial-of-the-shelf components, it is a simple affair to construct and implement a similar system in other facilities across the Air Force and Space Force. It would be highly desireable to take the system described here and scale it out across the entire Department of Air Force enterprise.

Acknowledgments

This material is based upon work supported by the Air Force Research Laboratory Edison Grant program.

References

[1] Haußge, G., "OctoPrint: The snappy web interface for your 3D printer," `http://www.octoprint.org`, 2024.

[2] Jiang, K., "Obico," `http://www.obico.io`, 2024.

[3] Redmon, J., and Farhadi, A., "YOLO9000: better, faster, stronger," *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7263–7271.

[4] Redmon, J., "Darknet: Open Source Neural Networks in C," `http://pjreddie.com/darknet/`, 2013–2016.

[5] Redmon, J., and Farhadi, A., "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.