Failure Detection for 3D Printing

PREPARED FOR

Computer Vision Course, 2020

PREPARED BY

Alexandru GRIGORAȘ

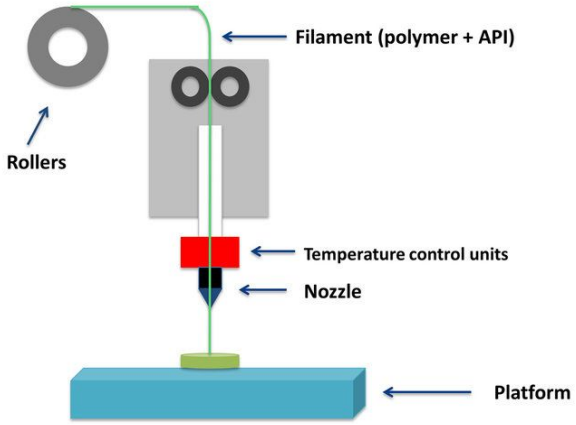
DSWT 1B

CONTENTS

1. Project Overview

3D Printing is a domain that is rising in popularity from the last few years. At the beginning, the printers were only in the industry for creating prototypes or final products and were very expensive. Since the patents for the technology have expired, these have become popular to hobbyists and makers around the world. The most popular models are open-source and everyone has the knowledge to build one.

The technology analyzed is FDM, shown in the image below (taken from "Personalised 3D Printed Medicines: Which Techniques and Polymers Are More Successful?", Andrea Alice Konta, Marta García-Piña and Dolores R. Serrano). Fused deposition modeling is an additive manufacturing (AM) process in which a physical object is created directly from a computer-aided design (CAD) model using layer-by-layer deposition of a feedstock plastic filament material extruded through a nozzle.



Printing time can vary from a few minutes for a small object (50x50mm), to more than a day for larger objects (20x20cm). Computer vision cannot improve the printing time, but it gives the opportunity to have automated tools for detecting failures and save many wasted hours.

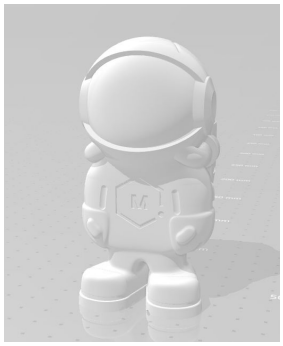
FDM has some drawbacks that can make the experience worse for the average user. These include under extrusion, over extrusion, parts not sticking to bed, layer shifting, clogged extruder, incorrect temperature, dimensional accuracy or layer separation. These can be easily spotted by a specialist, but with constant supervision, which is not an efficient method.

The scope of the document is to alert the user in case of errors occured. An add-on module (for example a raspberry pi and a camera) isn't expensive and can save a lot of wasted filament. In the long term, this application even reduces the cost of supplies and maintenance.

2. Operation mode

Before determining the causes for failing in 3D printing, the steps for printing a model are:

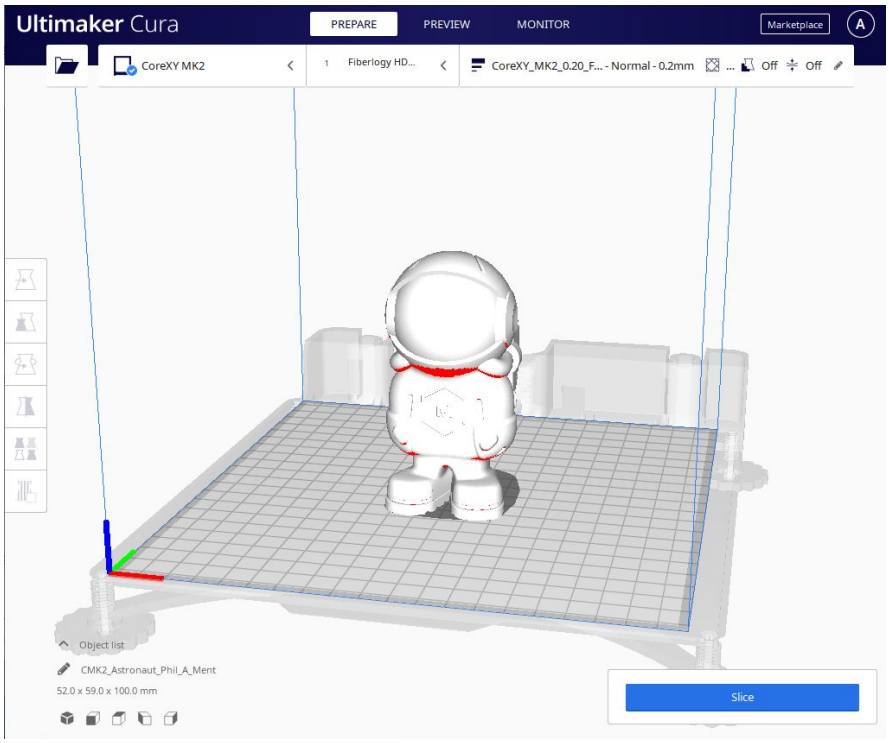
2.1. Getting the model



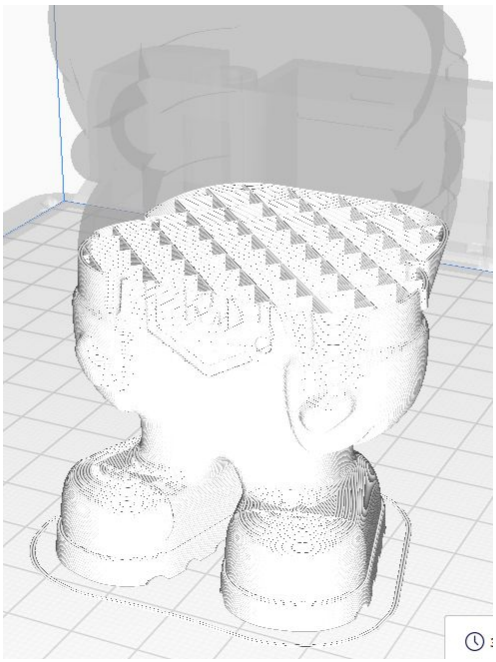
The model has a file format for 3D models (which can be stl or 3mf) and has a solid type. In the following image is a model of the Astronaut Phil. The model is made by Materhackers (and taken from https://www.thingiverse.com/thing:2557603).

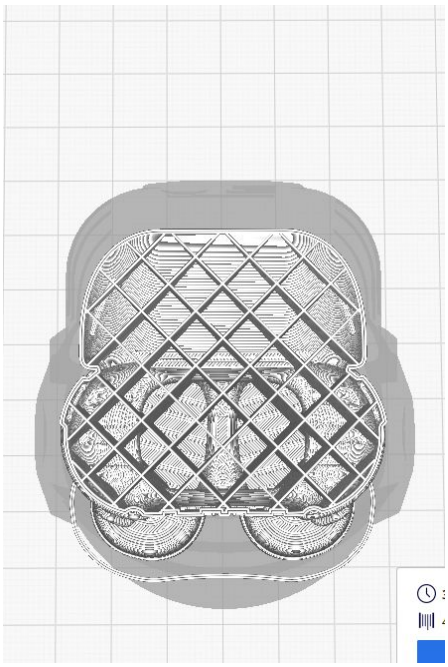
2.2. Slicing

The slicer program converts the model from 3D representation to layers of a specified height (usually 0.2 mm). It then creates commands that can be sent to the printer to make certain actions to print the object. Below is a screenshot of the Cura Slicer by Ultimaker with the selected model.



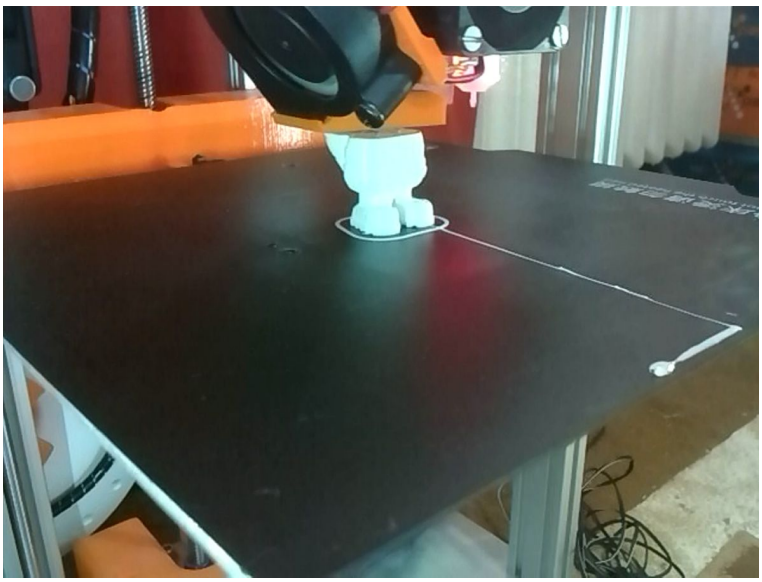
The slicer also offers a layer representation, that can be seen in the following images. The model is split in half and viewed from different angles. These can be used for comparison with the real printed object and verify for similarity.





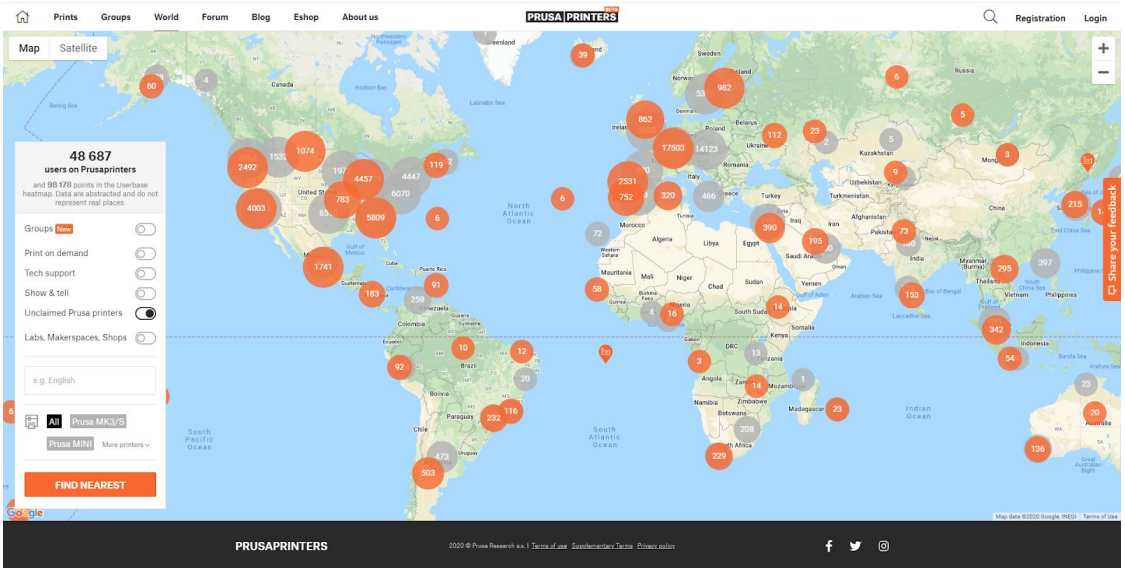
2.3. Printing

The sliced model is sent to the printer and the machine converts the code (named G-codes) into actions (such as heating the elements, turning on the fans) and print moves (e.g. move the print head in the X axis direction with 10mm). The printing process and the final product are seen in the images below.

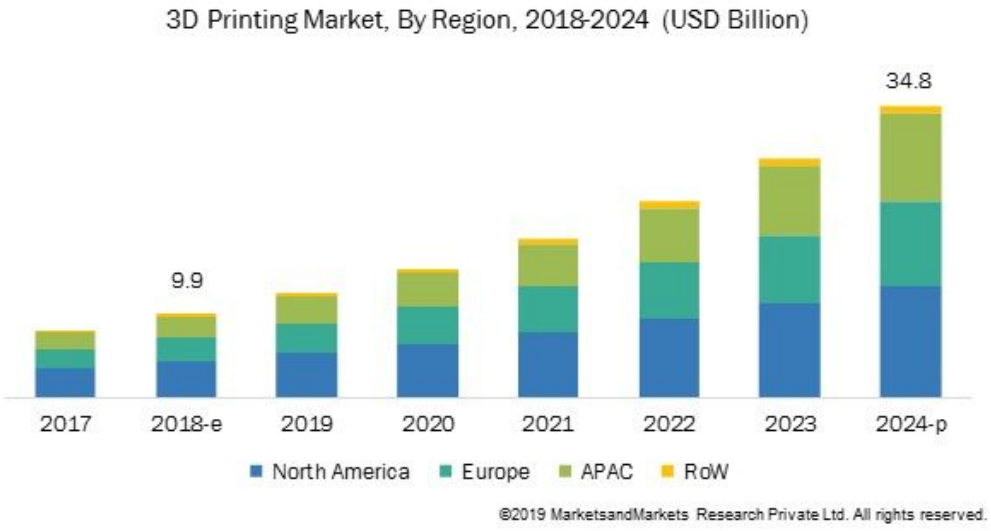


3. Usage

There is a massive usage of 3D printing around the world. Prusa design is the most known open source printer design in the world. According to the founder, Josef Prusa, they sold over 130,000 3D printers, gaining a significant 10% market share in 2018. Below is the map of printers sold in the world. Besides Prusa, there are many other printers used by makers and industry. Below is the map of Prusa printers sold.



The market of 3D printing is increasing, so there is a need for a software to reduce the failures and improve the technology. The next image is taken from https://www.marketsandmarkets.com/Market-Reports/3d-printing-market-1276.html and represents market share of the 3D printing from 2017 to 2024.

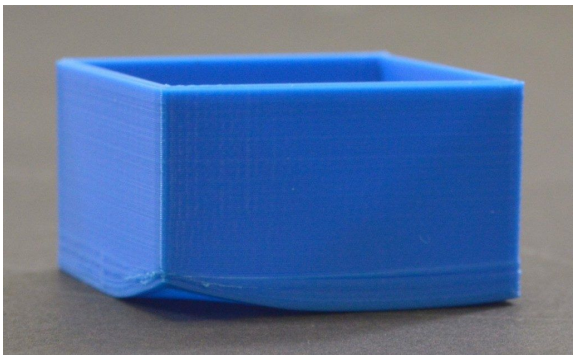


4. Printing Failures

These are some of the most common reasons for failed 3D prints. In this document are proposed some methods for detecting these automatically, without a human operator.

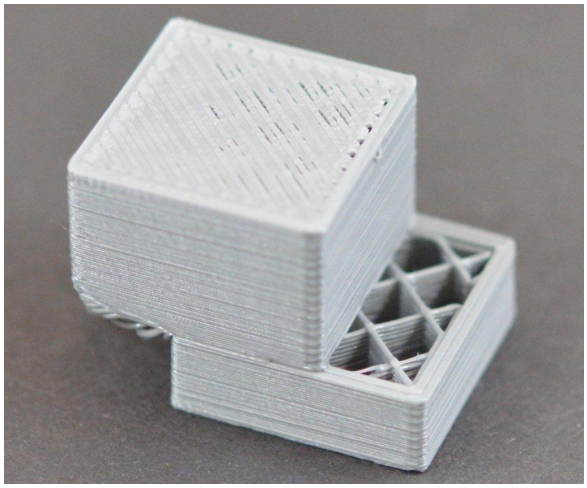
4.1. Print warping

Warping is caused by the plastic filament not sticking to the print bed. Usually, it depends on the surface of the print bed and the temperature to ensure a good first layer. The image taken from https://www.simplify3d.com/support/print-quality-troubleshooting/warping/.



4.2. Layer shifting

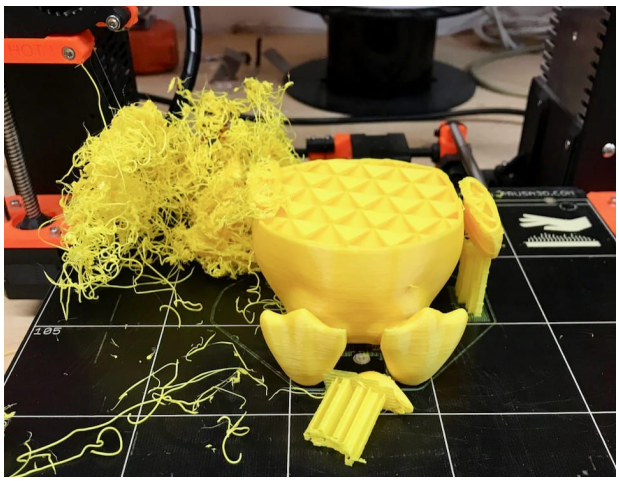
It has a mechanical cause (belt not tighten, print head hitting a printed part), or an electrical cause (motor driver skipping steps because of the speed or lower input voltage).



https://www.simplify3d.com/support/print-quality-troubleshooting/layer-shifting/ is the source of the image.

4.3. Spaghetti printing

It is caused by the extruder printing in air. Some parts of the printed object are snapping off the object or print bed and are moved away or the whole object is separated from the bed. The extruded filament is then falling and by moving the print head at the same time it curls and creates this effect. The image is taken from https://www.fabbaloo.com/blog/2017/11/3/how-to-persistent-3d-print-failure-heres-what-to-look-for.



4.4. Broken parts

Mechanical failures can also occur. The belts can break because of the tension and wear from the movement (presented in the image below). The printed parts can also break because of the tension and vibrations. These can lead to a failed print.

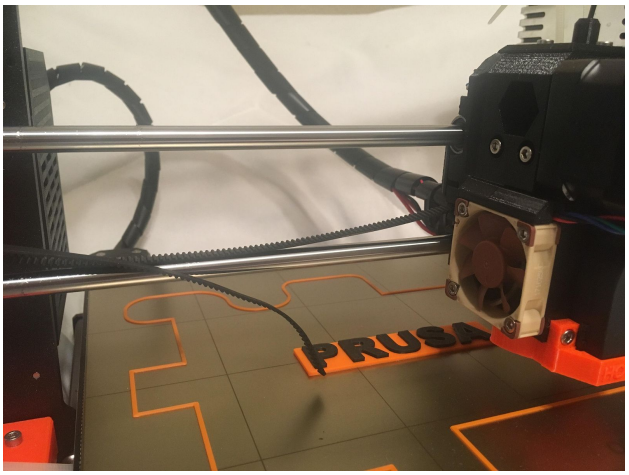


Image taken from https://www.reddit.com/r/3Dprinting/comments/7tkrdf/broken_belt/.

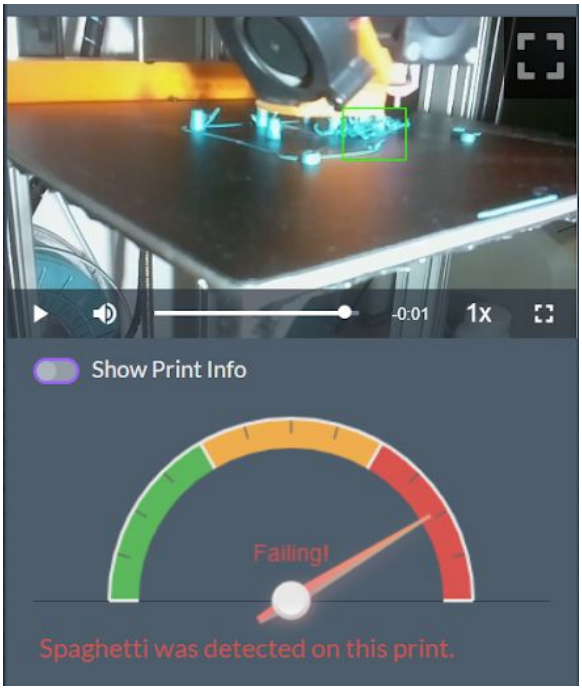
5. Existing applications

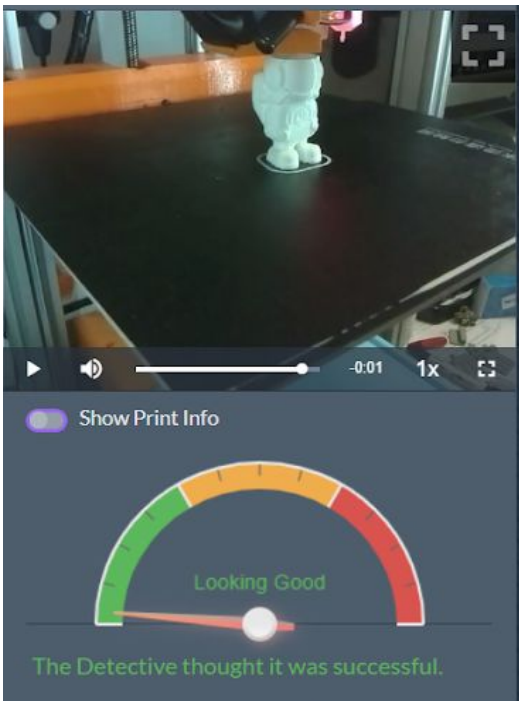
The Spaghetti Detective (TSD) is a commercial open-source software for detecting 3D printing failures.

THE SPAGHETTI DETECTIVE

It detects the spaghetti failures using Deep Learning. Using Convolutional Neural Networks, it efficiently trains a model to detect the failures, no matter the model of the printer, material, camera position or illumination (extreme cases may fail).

These screenshots are taken from the TSD user interface. Timelapses are from the 3D printer used for experiments on this document. On the left is a failed 3D print and in the right a successful one.





TSD detects spaghetti with high accuracy. It can detect layer shifting (if the shift is significant, over 20-30% of the object size) and broken parts (because if one axis or both are not moving, there is a possibility to print in air and create spaghetti), but it cannot detect warping or other failure types.

6. Technical Obstacles

The challenge is to detect as many printing failures as possible using Computer Vision techniques. There isn't a solution that can solve all the failures. In this article, there are three proposed solutions to solve some of the failures. The challenges are split into hardware and software.

6.1. Hardware challenges

6.1.1. Camera position

Ideally, the camera needs to cover the whole object, but this cannot be done using only one camera. I selected the close upper left corner view and the top view.

6.1.2. Camera focus

The objects in the first view can be closer or further away from the camera lens. The camera cannot focus on all objects at the same time. For this, the camera focus is set on the middle of the print volume, sacrificing the quality at extremities.

6.1.3. Color similarity between print bed and filament color

Having a contrast between the print bed and the filament color is preferable. But in cases that are similar, the solution is to have a different print bed color. This can be achieved by using a magnetic heatbed with swappable surfaces that have different colors.

6.1.4. The lighting

The lighting needs to be uniform spread and remain the same for the whole printing process. There is a problem with reflection that can influence the images taken.

6.2. Software challenges

6.2.1. Processing time

The application needs to be executed in real time to spot the failures. In this way, any problem spotted is reported and the printing process is paused. The user can then resume the print if the fault can be solved or cancel the print if it's a complete failure. For this faster algorithms are used at the expense of accuracy. More processing power can be added by changing the hardware, but the problem is the cost.

6.2.2. Accuracy

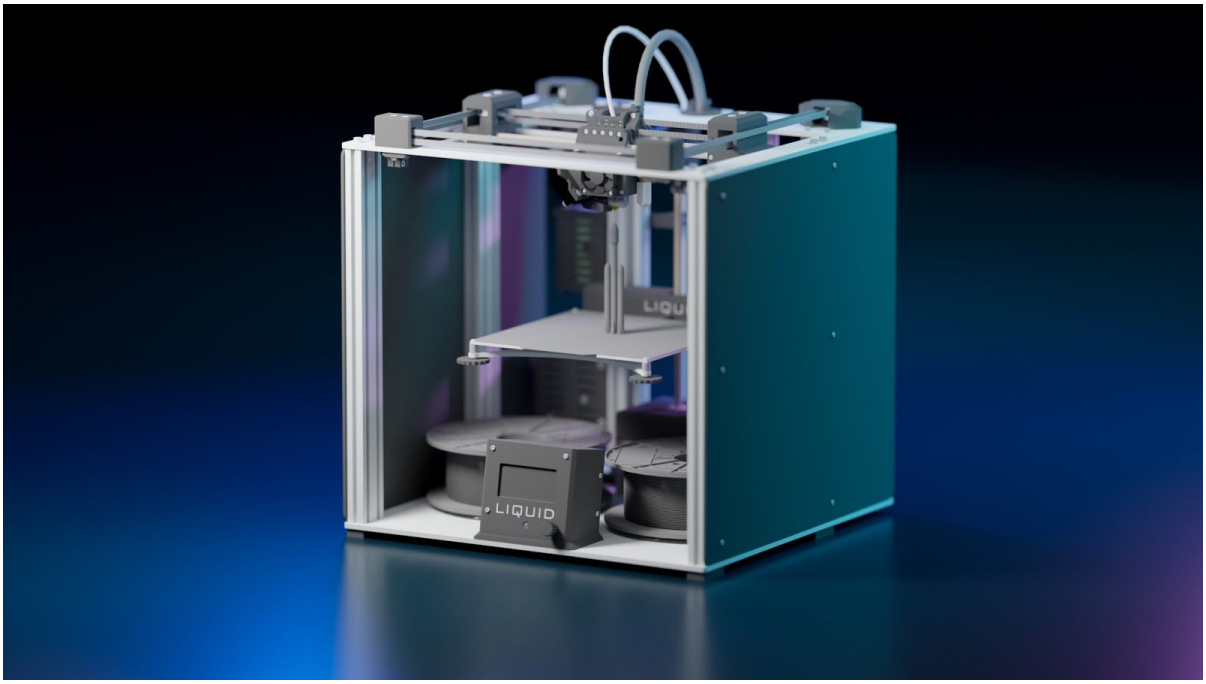
Depends on the accuracy error that is needed. Faster algorithms can be used for faster printing but with expense of accuracy. If a high accuracy is needed, algorithms with better accuracy and lower processing time are used, but with expense of a lower printing speed to keep up with processing of the images. The cost also affects these options.

6.2.3. Generalization

The applications should work on different printers and filament colors. Some of these can be compensated in software (e.g. camera angle and focal length) and others mechanically (e.g. print bed color and filament color needs to be different).

7. Hardware

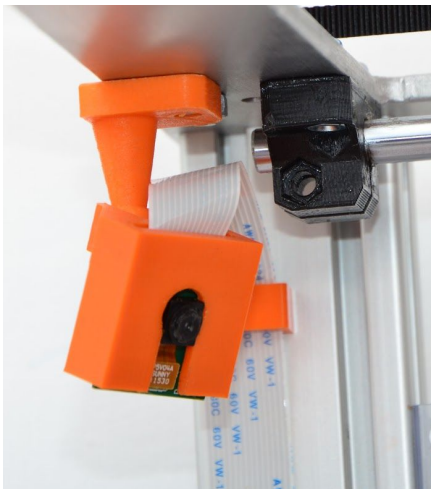
The 3D printer used for experiments is Liquid 3D printer with CoreXY kinematics the print head is moving in X and Y coordinates and the print bed is moving in Z coordinates. Below is a rendered image of the developed printer from the CAD model.



The hardware is divided into the acquisition and processing of the images.

7.1. Acquisition of images

A common way to capture images of a 3D printer is to mount a camera in a position where the print bed can be seen. In this case in the close left corner. For this, a Raspberry Pi 2B was used and a Raspberry Pi camera. During printing, the camera captures images at a specified interval, creating a timelapse.



The disadvantage of the RPi camera is the low quality of the images, where small imperfections cannot be distinguished. To get better quality of images, a DSLR camera was used. The camera is a Nikon D5100 and the shots were taken at 18mm focal length for tracking the head and 50mm for print bed images. The shutter speed is set to 1/50 of a second, matching the main light source frequency to prevent bands in the images.

Multiple light sources were used to ensure a good lighting across the whole print bed.

7.2. Processing the images

Processing images in the experiment is done using a Notebook with the following specification: i7-8750H CPU, GTX 1060 GPU and 16GB of RAM.

Ideally, the processing should be done by creating a processing server and a client using a Raspberry Pi (or a similar board) with a camera that sends the images to the server and it responds with the result of the processing (fault or success). The server can be used for multiple instances (in this case multiple printers).

Another version is to use a dedicated board for processing images. This needs to have dedicated GPU cores. For example, Nvidia Jetson Nano has a good compromise between processing power and cost.

8. Software

Three experiments are created. These are used to determine if the methods offer good results in spotting failures. The first is to compare the images generated by the slicer software to the images taken from the printer. The second is tracking the print head to determine the X, Y coordinates in cm (in this experiment X and Y have values between 0 and 20 cm). The third is to detect spaghetti plastic in the printed parts.

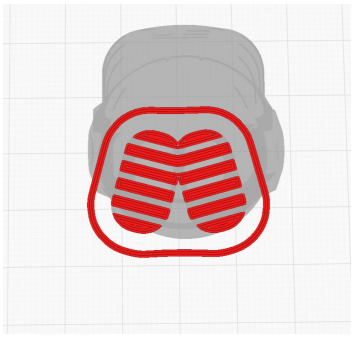
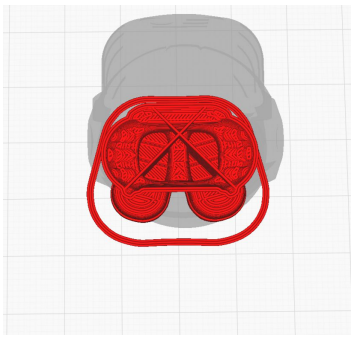
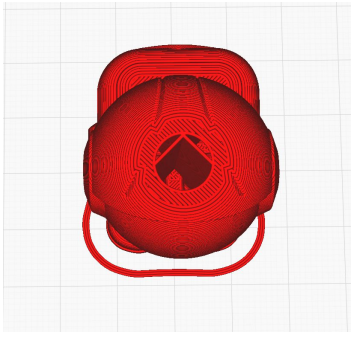
These experiments are detailed in the following subchapters.

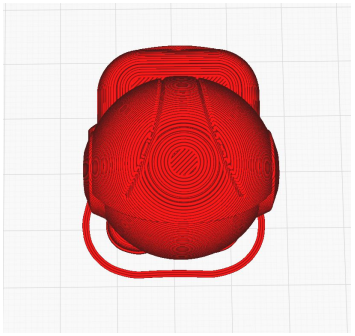
8.1. Sliced and printed object similarity

From the slicer software with the model loaded and sliced, pictures of layers are taken from the top view. For the experiment, images from the layers 2, 68, 148, 242 and 249 are selected.

With the camera positioned on top, images for the same layers are taken from the printed object. This is done by sending commands to the printer to pause and take a picture with the camera on the selected layers.

Below are the raw images taken from slicer and printer.

Raw Images		
Layer	Slicer image	Printed image
2		
68		
148		
242		

249		

These are preprocessed before the matching.

For the slicer images the following steps are applied:

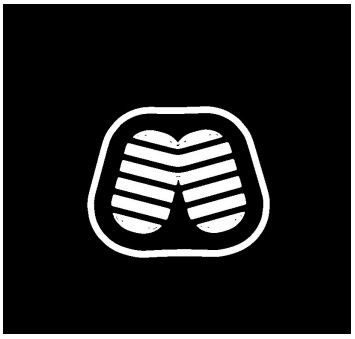
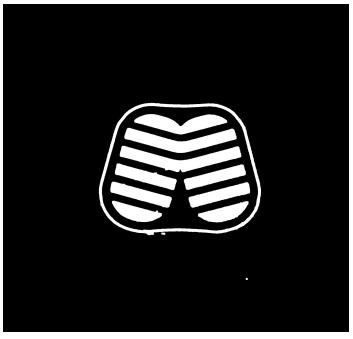
- Convert image to HSV
- Create a mask with a lower and upper range of red color and combine them
- Apply mask to image

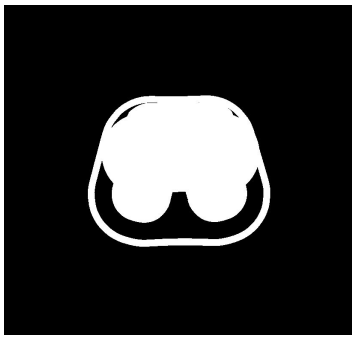
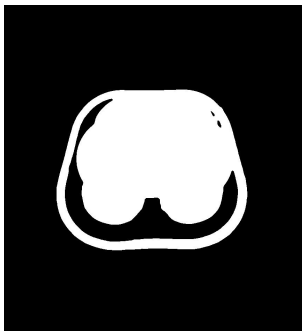
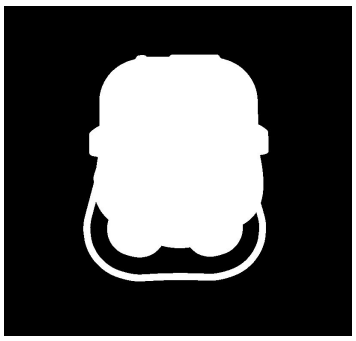
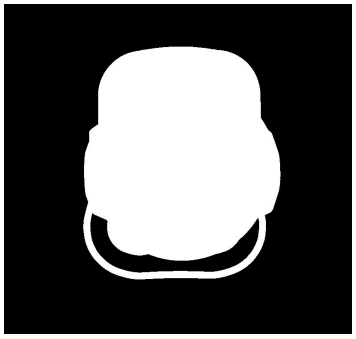
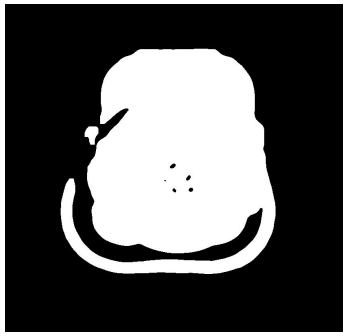
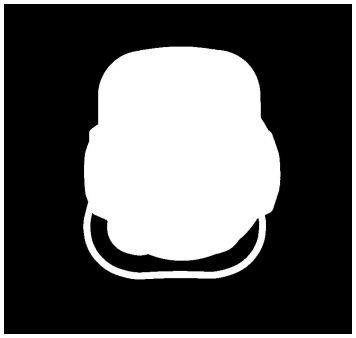
This puts the red pixels to 1 (white normalized value) and others to 0 (black). The result is the image with a black background and white object.

For the printer images the following steps are applied:

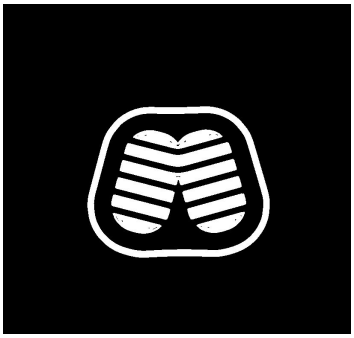
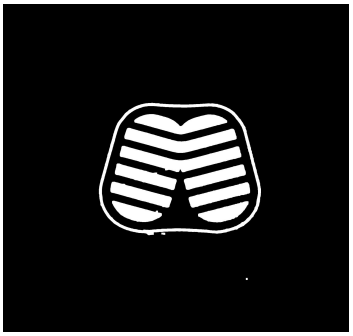
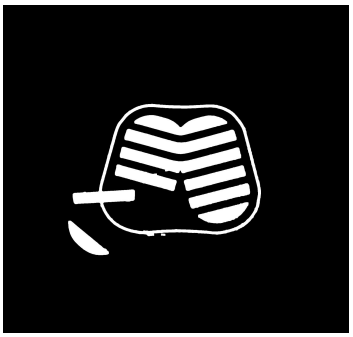
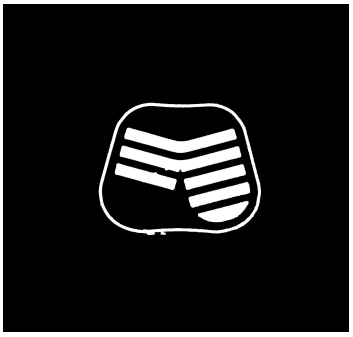
- Convert image to grayscale
- Filter image using median filter with kernel size 11
- Apply Otsu threshold
- Apply Opening (morphological transformation, erosion followed by dilation) with kernel size (11, 11)

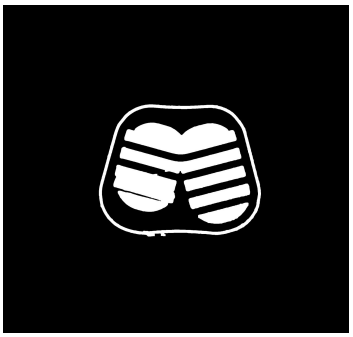
This results in an image similar to the slicer image.

Processed Slicer and Successful printed images			
Layer	Slicer image	Printed image	Difference
2			8.756732907136597e-05

68			1.94200696267 1823e-06
148			1.41989283334 83607e-05
242			3.61604249109 6255e-05
249			1.72896767052 69897e-05

To simulate a failure, at layer 2, some parts are taken away from the print bed or repositioned. In other situations, the object can deform or can increase in size if layer shifting occurs. The same filters are applied.

Processed Slicer and Failed printed images			
Layer	Slicer image	Printed image	Difference
2			8.75673290713 6597e-05
			0.00030144793 42604065
			0.00111981310 62109707
			0.00024960829 226874524

			0.00018676242831490724

To compute the similarity between slicer and printer images, Hu moments are calculated. The difference between the Hu moments of different images are calculated using Euclidean distance. The distances are displayed in the tables above.

It can be noticed that the difference between similar images are in the range of 10^{-5} or below and for different images are in the range of 10^{-4} or above.

8.2. Print head tracking

Detecting mechanical defects can be done by tracking the actual machine and detecting movement. Below are two frames from a video of the 3D printer used for testing. In the lower left corner are displayed the details for tracking:

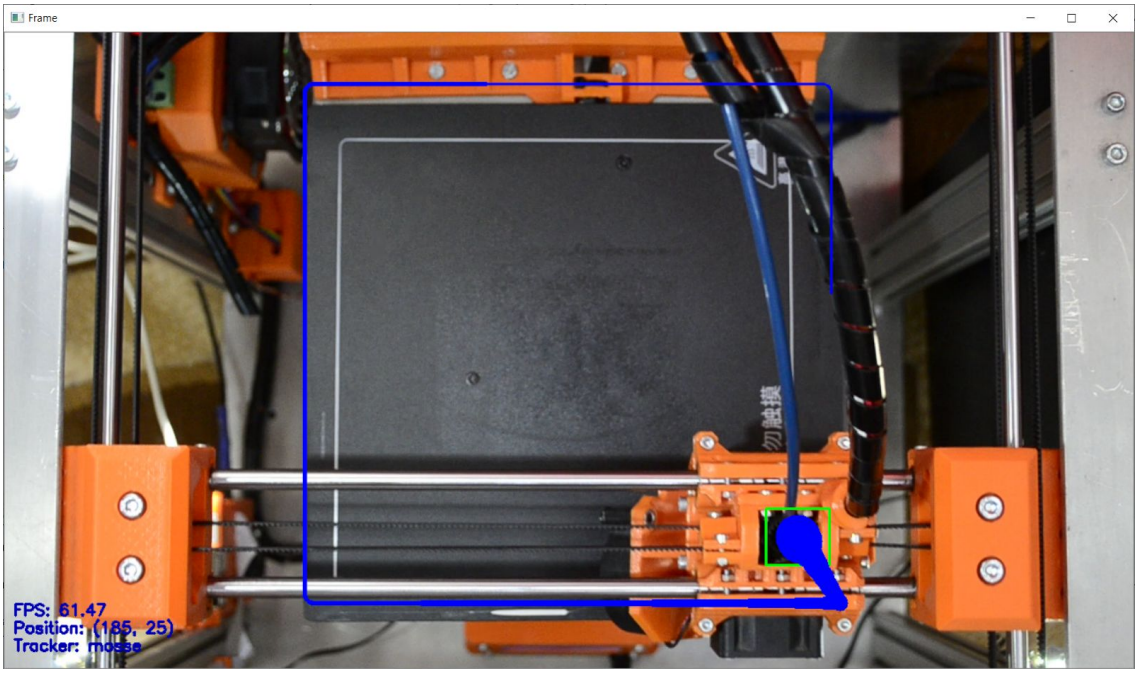
- FPS: the number of frames that can be processed in a second, ideally should be above 60 to achieve a real time, but it depends on the printing speed (which can be from 10mm/s to 200mm/s)
- Position: the actual coordinates of the print head (computed from the pixel values and known machine limits)
- Tracker: the algorithm used for tracking the print head

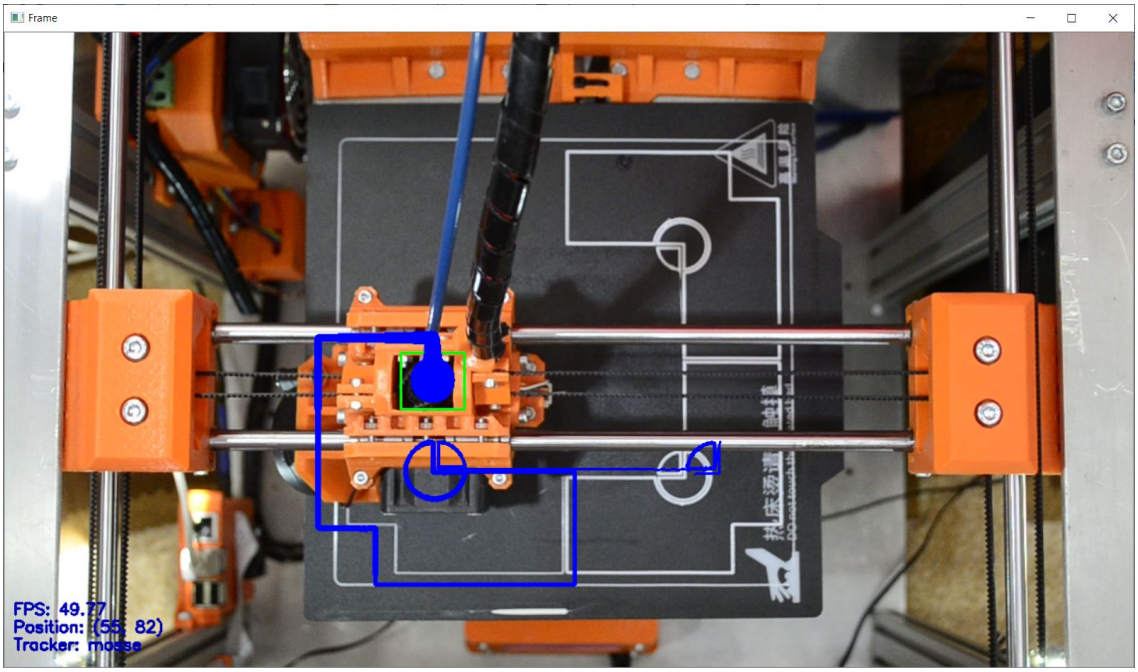
The software tracks the print head (in this case the black part of the print head). The coordinates are in pixels (the resolution of the video is 1280x720). For viewing the path of the print head, a tail is displayed, The pixel coordinates are converted to millimeters. The machine dimensions are 20x20x20cm (for x, y, z coordinates))and the origin is on the bottom left.

The file that is sent to the printer is GCODE. The file contains a set of commands. For example:

G1 X55 Y82 -> moves the print head to position 90mm on the X axis and 82mm on the Y axis.

By parsing the GCODE file, the movement can be compared in real-time to the actual travel of the print head. The results will probably be different, but it is important to have the error under a certain threshold value.





The tracking methods used are:

- **Minimum Output Sum of Squared Error (MOSSE):** uses adaptive correlation for object tracking which produces stable correlation filters when initialized using a single frame. Is robust to variations in lighting, scale, pose, and non-rigid deformations. It also detects occlusion based upon the peak-to-sidelobe ratio, which enables the tracker to pause and resume where it left off when the object reappears. It operates at a higher fps(450 fps and even more). It is easy to implement, as accurate as other complex trackers and much faster. But, on a performance scale, it lags behind the deep learning based trackers.
- **Discriminative Correlation Filter with Channel and Spatial Reliability (DCF-CSR):** uses the spatial reliability map for adjusting the filter support to the part of the selected region from the frame for tracking. This ensures enlarging and localization of the selected region and improved tracking of the non-rectangular regions or objects. It uses only 2 standard features (HoGs and Colornames). It also operates at a comparatively lower fps (25 fps) but gives higher accuracy for object tracking.
- **Kernelized Correlation Filters (KCF):** It is based on BOOSTING and MIL tracker. Utilizes that fact that the multiple positive samples used in the MIL tracker have large overlapping regions. This overlapping data leads to some nice mathematical properties that are exploited by this tracker to make tracking faster and more accurate at the same time. Accuracy and speed are both better than MIL and it reports tracking failure better than BOOSTING and MIL. Does not recover from full occlusion.

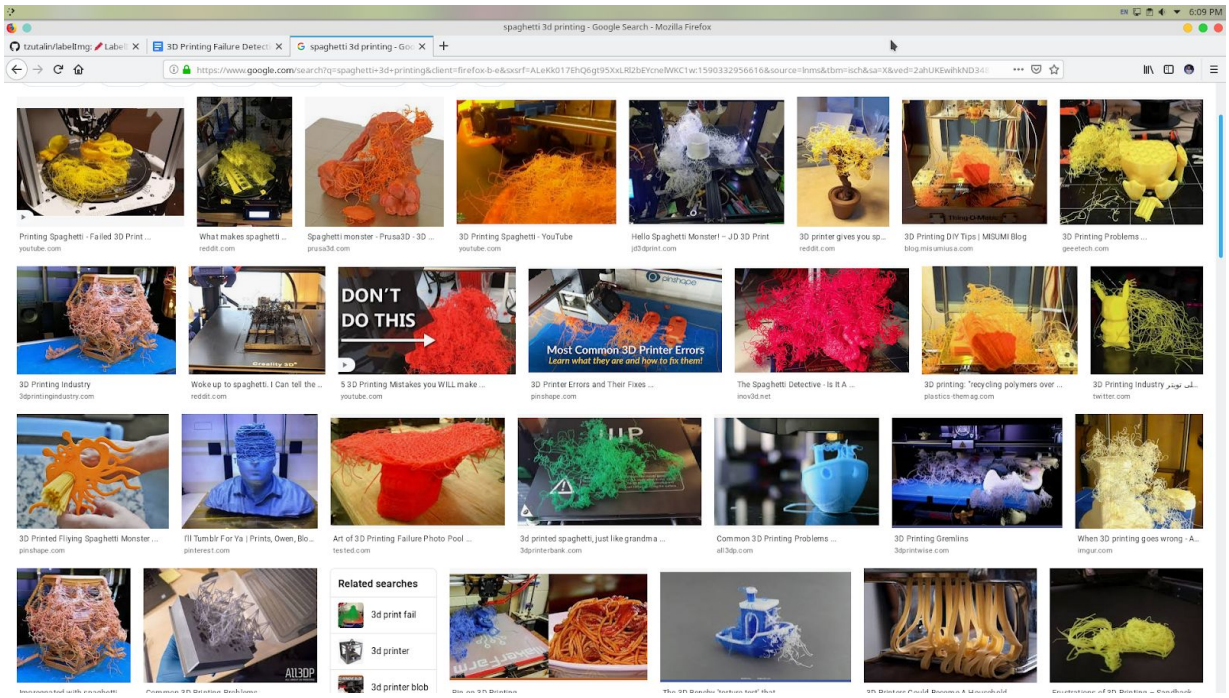
Best overall method chosen is MOSSE. It has a higher fps, which ensures high fps for real-time tracking. The experiment shows 40-80 fps, which is enough for real time processing for a normal printing speed (60mm/sec).

8.3. Spaghetti detection

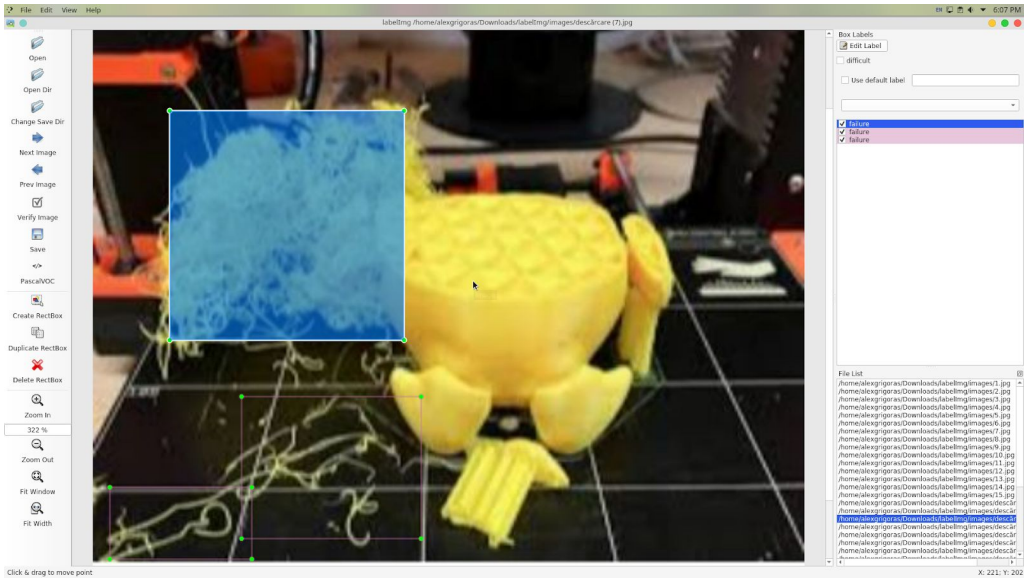
To detect the spaghetti plastic in images, a deep neural network model is chosen. It is a convolutional neural network, similar to YOLO v2. The architecture is taken from TSD.

To create an object detector for spaghetti plastic, the first step is to train a model (or use a pre-trained model) and the second is to test it to real images. Darknet is used to define the model, train using the dataset and test with images from the used printer.

Getting images from Google or other sources to create a dataset.



Creating the annotated dataset using Labellmg, by Tzutalin. Labellmg. Git code (2015). https://github.com/tzutalin/labellmg.



The **Model Structure** is composed of 32 layers. It has only one output class (failure).

layer	filters	size/strd(dil)	input	output

```
 0 conv     32      3 x 3/ 1    416 x 416 x   3 ->  416 x 416 x  32 0.299 BF
 1 max           2x 2/ 2    416 x 416 x  32 ->  208 x 208 x  32 0.006 BF
 2 conv     64      3 x 3/ 1    208 x 208 x  32 ->  208 x 208 x  64 1.595 BF
 3 max           2x 2/ 2    208 x 208 x  64 ->  104 x 104 x  64 0.003 BF
 4 conv    128      3 x 3/ 1    104 x 104 x  64 ->  104 x 104 x 128 1.595 BF
 5 conv     64      1 x 1/ 1    104 x 104 x 128 ->  104 x 104 x  64 0.177 BF
 6 conv    128      3 x 3/ 1    104 x 104 x  64 ->  104 x 104 x 128 1.595 BF
 7 max           2x 2/ 2    104 x 104 x 128 ->   52 x  52 x 128 0.001 BF
 8 conv    256      3 x 3/ 1     52 x  52 x 128 ->   52 x  52 x 256 1.595 BF
 9 conv    128      1 x 1/ 1     52 x  52 x 256 ->   52 x  52 x 128 0.177 BF
10 conv    256      3 x 3/ 1     52 x  52 x 128 ->   52 x  52 x 256 1.595 BF
11 max           2x 2/ 2     52 x  52 x 256 ->   26 x  26 x 256 0.001 BF
12 conv    512      3 x 3/ 1     26 x  26 x 256 ->   26 x  26 x 512 1.595 BF
13 conv    256      1 x 1/ 1     26 x  26 x 512 ->   26 x  26 x 256 0.177 BF
14 conv    512      3 x 3/ 1     26 x  26 x 256 ->   26 x  26 x 512 1.595 BF
15 conv    256      1 x 1/ 1     26 x  26 x 512 ->   26 x  26 x 256 0.177 BF
16 conv    512      3 x 3/ 1     26 x  26 x 256 ->   26 x  26 x 512 1.595 BF
17 max           2x 2/ 2     26 x  26 x 512 ->   13 x  13 x 512 0.000 BF
18 conv   1024      3 x 3/ 1     13 x  13 x 512 ->   13 x  13 x 1024 1.595 BF
19 conv    512      1 x 1/ 1     13 x  13 x 1024 ->   13 x  13 x 512 0.177 BF
20 conv   1024      3 x 3/ 1     13 x  13 x 512 ->   13 x  13 x 1024 1.595 BF
21 conv    512      1 x 1/ 1     13 x  13 x 1024 ->   13 x  13 x 512 0.177 BF
22 conv   1024      3 x 3/ 1     13 x  13 x 512 ->   13 x  13 x 1024 1.595 BF
23 conv   1024      3 x 3/ 1     13 x  13 x 1024 ->   13 x  13 x 1024 3.190 BF
```

24 conv 1024 3 x 3/ 1 13 x 13 x 1024 -> 13 x 13 x 1024 3.190 BF

25 route 16 -> 26 x 26 x 512

26 conv 64 1 x 1/ 1 26 x 26 x 512 -> 26 x 26 x 64 0.044 BF

27 reorg_old / 2 26 x 26 x 64 -> 13 x 13 x 256

28 route 27 24 -> 13 x 13 x1280

29 conv 1024 3 x 3/ 1 13 x 13 x 1280 -> 13 x 13 x 1024 3.987 BF

30 conv 30 1 x 1/ 1 13 x 13 x1024 -> 13 x 13 x 30 0.010 BF

31 detection

mask_scale: Using default '1.000000'

Total BFLOPS 29.338

avg_outputs = 607364

Training:

Learning Rate: 0.001, Momentum: 0.9, Decay: 0.0005

If error occurs - run training with flag: -dont_show

Resizing, random_coef = 1.40

608 x 608

Loaded: 0.000000 seconds

Region Avg IOU: 0.371475, Class: 1.000000, Obj: 0.531537, No Obj: 0.542987, Avg Recall: 0.150000, count: 20

Region Avg IOU: 0.295104, Class: 1.000000, Obj: 0.587007, No Obj: 0.542839, Avg Recall: 0.142857, count: 14

Region Avg IOU: 0.393452, Class: 1.000000, Obj: 0.659053, No Obj: 0.542041, Avg Recall: 0.250000, count: 16

Region Avg IOU: 0.394552, Class: 1.000000, Obj: 0.528770, No Obj: 0.541172, Avg Recall: 0.363636, count: 11

Region Avg IOU: 0.336160, Class: 1.000000, Obj: 0.555715, No Obj: 0.540708, Avg Recall: 0.100000, count: 10

Region Avg IOU: 0.411765, Class: 1.000000, Obj: 0.604610, No Obj: 0.542922, Avg Recall: 0.333333, count: 15

Region Avg IOU: 0.402684, Class: 1.000000, Obj: 0.595708, No Obj: 0.543325, Avg Recall: 0.428571, count: 14

Region Avg IOU: 0.293618, Class: 1.000000, Obj: 0.466598, No Obj: 0.542356, Avg Recall: 0.300000, count: 10

1: 31.201496, 31.201496 avg loss, 0.000000 rate, 518.629000 seconds, 64 images

Loaded: 0.000000 seconds

Region Avg IOU: 0.278009, Class: 1.000000, Obj: 0.510910, No Obj: 0.540460, Avg Recall: 0.090909, count: 11

Region Avg IOU: 0.452195, Class: 1.000000, Obj: 0.569742, No Obj: 0.541567, Avg Recall: 0.250000, count: 12

Region Avg IOU: 0.356026, Class: 1.000000, Obj: 0.579426, No Obj: 0.541403, Avg Recall: 0.272727, count: 11

Region Avg IOU: 0.397630, Class: 1.000000, Obj: 0.602071, No Obj: 0.542702, Avg Recall: 0.333333, count: 12

Region Avg IOU: 0.315875, Class: 1.000000, Obj: 0.578367, No Obj: 0.543047, Avg Recall: 0.187500, count: 16

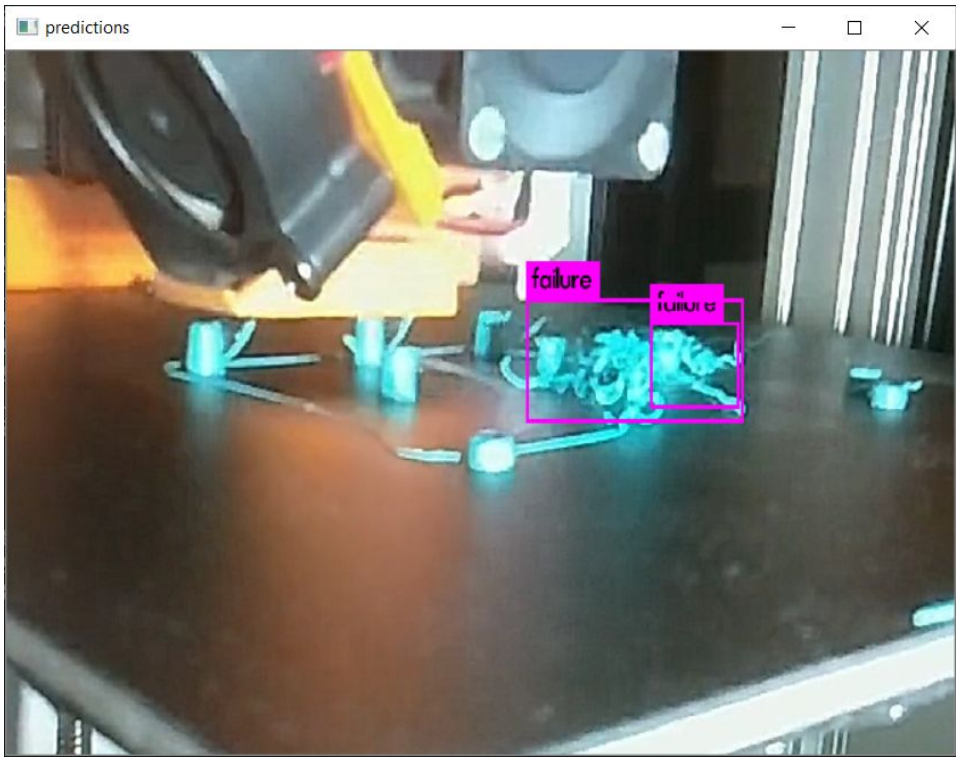
Region Avg IOU: 0.424426, Class: 1.000000, Obj: 0.567349, No Obj: 0.542483, Avg Recall: 0.333333, count: 15

Region Avg IOU: 0.398777, Class: 1.000000, Obj: 0.596936, No Obj: 0.541774, Avg Recall: 0.400000, count: 15

Region Avg IOU: 0.329390, Class: 1.000000, Obj: 0.549246, No Obj: 0.540833, Avg Recall: 0.250000, count: 12

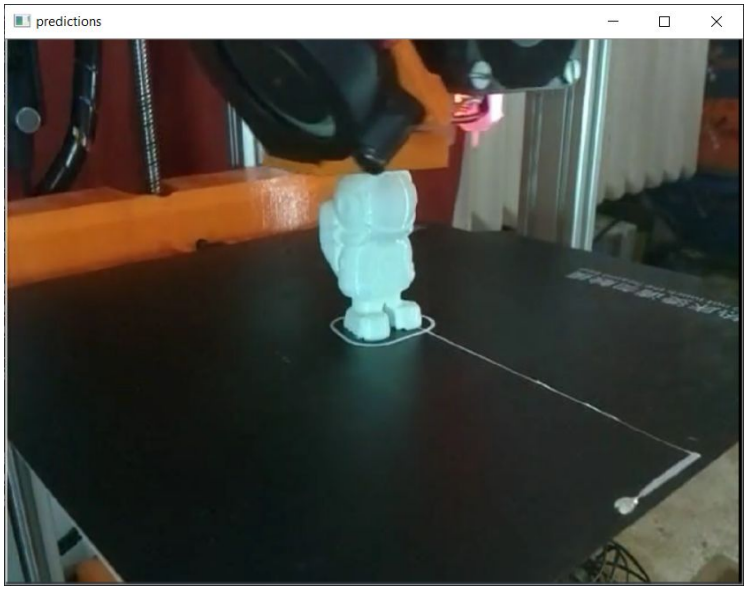
. . .

Testing with images from the printer. Example for a failed print is shown below:



failure: 38% failure: 35%

Example for a successful print is shown below:



9. Conclusion

The 3D printing technology allows users to rapidly make prototypes or models by making complex objects. The FDM technology is the most popular because of the ease of use. The market is still increasing exponentially.
3D printing has many sources of failure that can be solved using mechanical, hardware or software solutions. This paper was focused on solving them using the software, especially using computer vision techniques.

The first proposed method was image matching. It consisted in taking images from the slicer software and comparing it to the printed object from the printer. Before matching, the images are preprocessed and binarized. The matching algorithm uses HU moments and computes the euclidean distance to determine similarity. From the experiment, there is a difference between the successful prints and failing prints. This can detect warping, layer-shifting, parts separating from print bed and some mechanical defects. The disadvantage is the exact positioning of the camera and the filament color too similar to the print bed.

The second method consists in tracking the print head. The algorithms used are CSRT, KCF, MOSSE. CSRT has a higher object tracking accuracy and can tolerate slower FPS throughput, KCF has faster FPS throughput but can handle slightly lower object tracking accuracy and MOSSE is the fastest, but with a lower accuracy. The experiment shows the correlations between the coordinates from the GCODE file and the real coordinates. A difference higher than the threshold (selected experimentally) indicates a failure. In this case, only mechanical failures can be detected.

The third method is the one that has the best results overall. It uses deep learning to detect the spaghetti plastic created by the extruder. It has the highest accuracy and detects parts that get separated from the print bed, layer shifting with a higher degree and some mechanical failures. The positioning of the camera is not that strict, compared to the first method. The disadvantage is the necessity of the training with a big dataset.

These methods prove that failures can be detected using computer vision techniques and improve the reliability of the 3d printing. Not all the errors can be detected, but the highest occurring failures can be detected, which makes the experiments successful.