Data provisioning for 3D printing defect (spaghetti) detection

Full version

Michał Raczkowski

15-01-2024

Contents

1 Introduction

This document outlines the process of data collection, processing, and labeling for training a YOLO algorithm in detecting defects in 3D printing, with a focus on the "spaghetti" issue.

2 Data Requirements

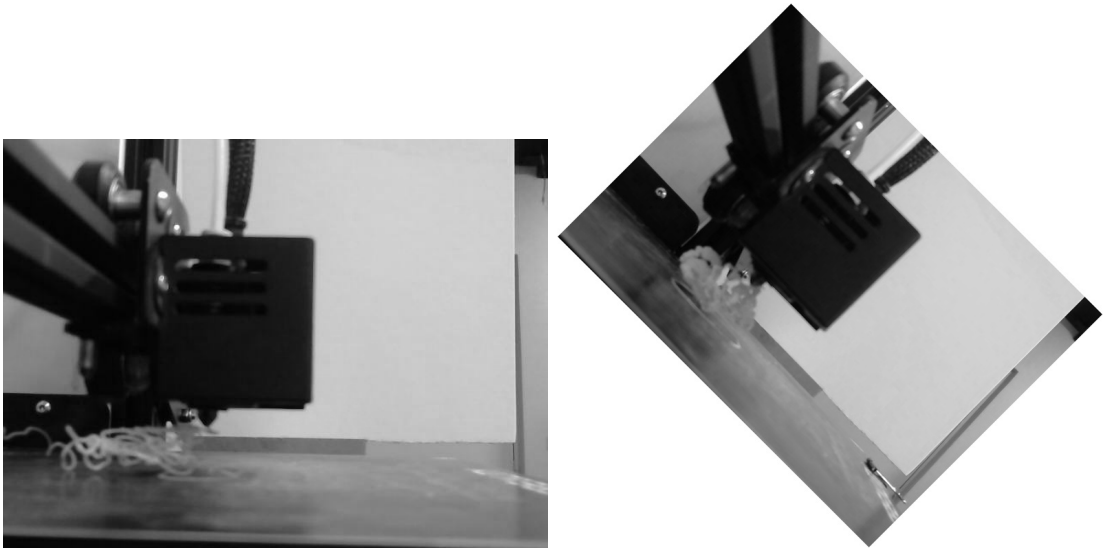
YOLO algorithm require set of labeled images in YOLO format representing the object we wish the algorithm to recognize.

Image Data

- Data Type: Image
- Minimum resolution: 640 x 480 pixels
- Color: Grayscale (black and white)
- Format: `.jpg` or `.png`
- Content: The image should feature a 3D printer's nozzle and a partially printed object with "spaghetti" defect on the print.
- Composition: The composition is such that the print head is in the upper part of the image, and the printed object is clearly visible with "spaghetti" defect.
- Orientation: The orientation should be with the printer oriented for a side view.
- Lighting: The lighting should be sufficient and flat, with no harsh shadows.
- Dimension: The aspect ratio should be standard, possibly 4:3 or similar.
- File Size: The file size should be under one megabyte.
- Legal Considerations: Ensure the image is free of copyright or privacy issues.
- Metadata: Include capture date, time and settings.
- Variety: Some images should be from different angles to provide variety for model

Example: Image of "spaghetti" defect from side Figure 5a and 5b

- Data Type: Image
- Resolution: 640x480 pixels
- Color: Grayscale (black and white)
- Format: `.jpg`
- Rotated: 45 degree only Figure 5b



(a) "Spaghetti" issue[1] (b) "Spaghetti" issue rotated[1]

Figure 1: Example of images

3 Data Collection

Data is collected from homemade pictures displaying failing prints with certain defect and from online open sources[1],[2].

4 Data Understanding

In our datasets, we possess a variety of photographs that require filtering to select those most suitable for our model. It's essential to clearly understand our selection criteria while choosing these photos.

4.1 "Spaghetti" defect recognition

Because datasets are made out of photos which displays various types of 3D printed defects we have to choose images which displays relevant issue (spaghetti defect) to our use case

(a) "Spaghetti" issue[1] (b) Layers split issue[2]

Figure 2: Two types of 3D printing issues

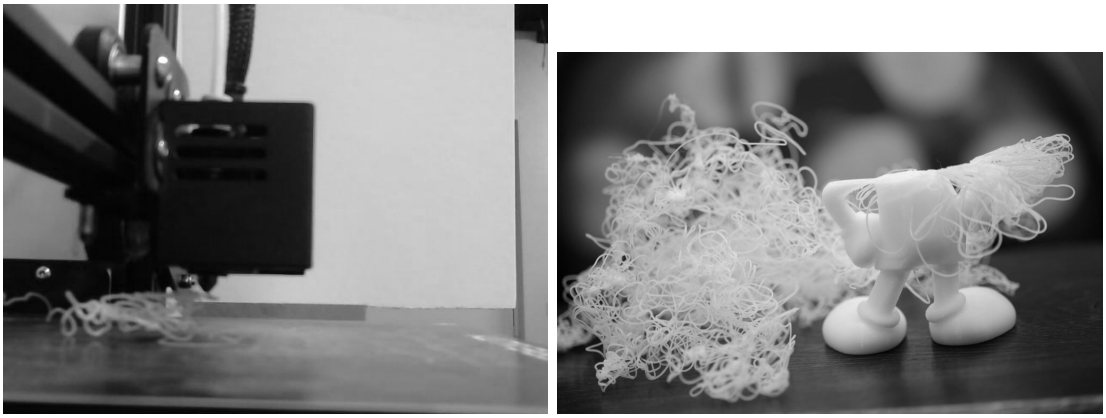
In example images above we can see two types of 3d printing issues.

- Figure 6a we can see issue we are interested in which is "spaghetti" issue it characterize by disorganized, tangled layers of filament, resembling spaghetti, resulting from print errors.
- Figure 6b we can see issue which we are not interested because it doesn't display "spaghetti" issue which we are focusing on, it displays layers split issue which characterize by distinct gaps and separations between layers, compromising print integrity and appearance

From looking on pictures and understanding displayed issues we can come to conclusion that Figure 6a will be relevant to our dataset, because it displays "spaghetti" issue which we are focusing on. This kind of filtering have to be done by hand, or we can try to find dataset containing already described images.

4.2 Correct Composition

Upon analyzing the dataset, it becomes apparent that numerous images exhibit the "spaghetti" defect. Our focus should be on photos showcasing a specific composition - a view of the printer bed with the nozzle above it look Figure 3a This is essential since the software's aim is to detect spaghetti defects during printing. Consequently, we've narrowed down the selection to photos that match the composition outlined in the data requirements. Additionally, we can include a few extra images that solely display the "spaghetti" defect Figure 3b to teach the model about the general shape of this specific defect.



(a) Desirable composition [2] (b) Undesirable composition[1]

Figure 3: Comparison of image without and with label

5 Data Processing

5.1 Dataset structure

Dataset we are using is composed of main two important parts:

- **Images**: Display issue of our interest (spaghetti issue)
- **Labels**: Set of annotations on an image that define the location (through bounding box coordinates) and type (class information) of each object within the image.

For better understanding here is structure of directories:

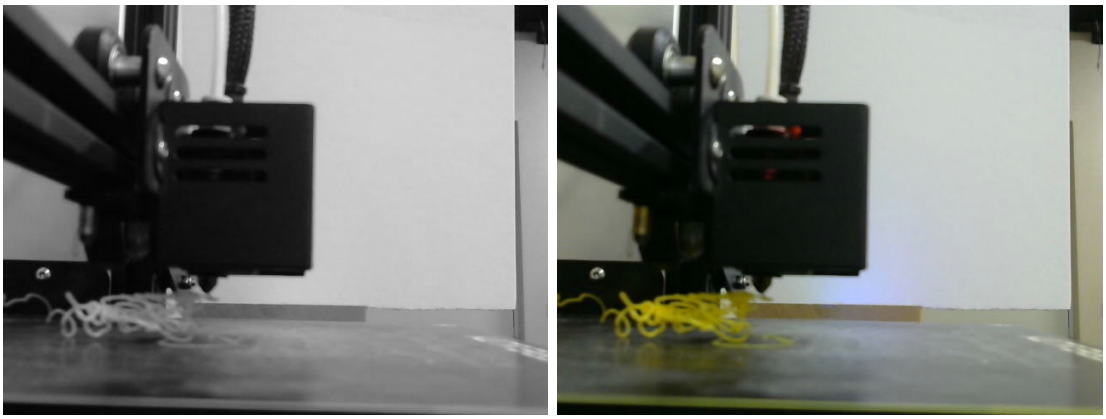
```
data/
├── images/
│   ├── train/
│   │   └── ...
│   └── val/
│       └── ...
└── labels/
    ├── train/
    │   └── ...
    └── val/
        └── ...
```

Directories `train` contains data on which model is trained

Directories `val` contains data on which model is validated

5.2 Change of color

Since the images in our dataset are colored, we need to convert them into Grayscale. Various tools such as GIMP, Photoshop, and ImageMagick can be employed for this purpose. Alternatively, online tools like ResizePixel [6] are also an option for this conversion task.

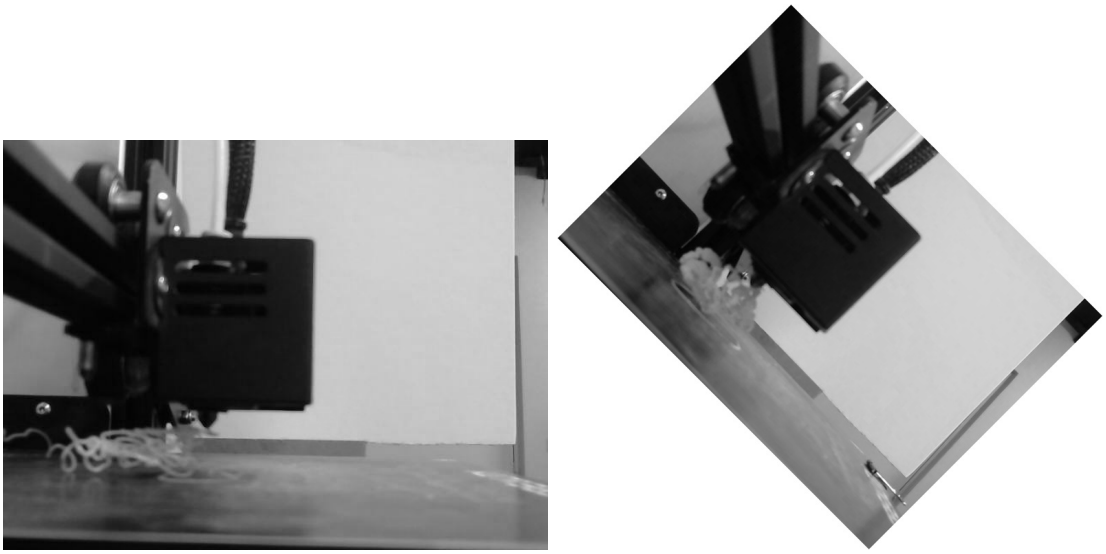


(a) Grayscale image [2] (b) Color image (labeled)[2]

Figure 4: Comparison of image without and with label

5.3 Change of orientation

To enhance the diversity of our dataset and improve the accuracy of our model, we should consider rotating a portion of our dataset images. Rotating them by 45 degrees is ideal as it significantly alters the image. For this task, tools such as GIMP, Photoshop, and ImageMagick are suitable options. Additionally, online tools like ResizePixel [6] can also be used for rotating the images.



(a) "Spaghetti" issue[1] (b) "Spaghetti" issue rotated[1]

Figure 5: Example of images

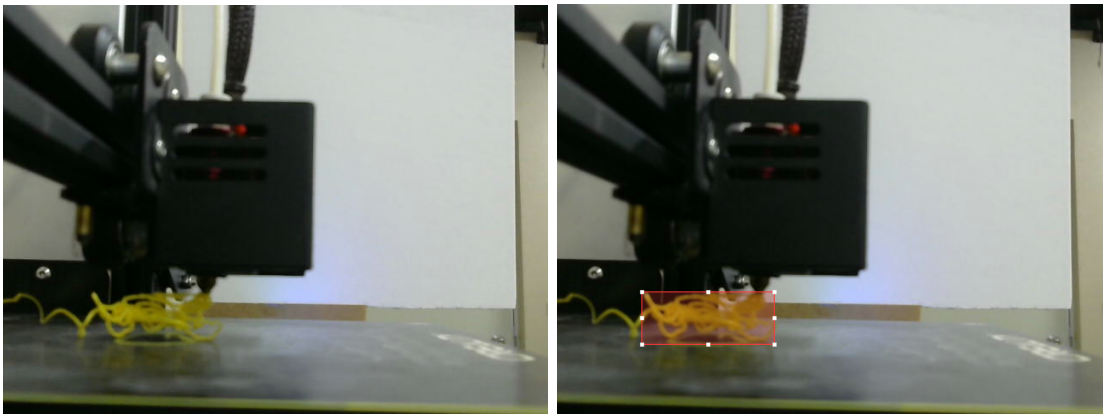
5.4 Why we need to add labels to images?

Labeling images is essential in training the YOLO (You Only Look Once) object detection algorithm for several reasons:

- **Training Supervision**: Labels provide the algorithm with examples to learn from, indicating what objects are present in the image and where they are.
- **Bounding Box Coordinates**: Labels include bounding box coordinates to teach the model how to locate and size objects in an image.
- **Class Identification**: Labels assign a class to each object, helping the model to classify objects correctly.
- **Model Accuracy and Evaluation**: Good quality labels improve the accuracy of the model and allow for effective evaluation of its performance.
- **Algorithm Optimization**:Accurate labels are crucial for minimizing the loss during training and optimizing the algorithm's performance.

5.5 Process of labeling

There are various tools we can use to label data (e.g. CVAT [3], VoTT [4]) but to keep simplicity and be able of rapid prototyping we are using free open source online tool called **MakeSense** [5] which is capable enough for our purposes



(a) no_support_62.jpeg [2] (b) no_support_62.jpeg (labeled)[2]

Figure 6: Comparison of image without and with label

Labeling involves identifying and marking the coordinates on an image to pinpoint the location of the defect we aim to detect in our case "spaghetti" defect. Once we have labeled all the images, we can export our labels in the YOLO format, which is specifically designed for the YOLO algorithm.

5.6 Dataset final structure

The final dataset should have this directory structure:

```
data/
├── images/
│   ├── train/
│   │   ├── no_support_62.jpeg
│   │   └── ...
│   └── val/
│       ├── exampleValidationImage.jpeg
│       └── ...
└── labels/
    ├── train/
    │   ├── no_support_62.txt
    │   └── ...
    └── val/
        ├── exampleValidationImage.txt
        └── ...
```

We can observe that labels are stored in separate directories, yet they share the same names as their corresponding images. This correlation is essential, as it enables the algorithm to accurately pair each image file with its respective label file. Additionally, it's important to note that the final dataset will contain many images along with their corresponding labels.

5.7 Label explained

Here is an example of what the contents of a label file might look like:

```
0 0.281250 0.764286 0.244643 0.130952
```

The meaning of sentence above is as follows:

- `0`: number of a class
- `0.281250`: X coordinate of center of the box
- `0.764286`: Y coordinate of center of the box
- `0.244643`: width of the box
- `0.130952`: height of the box

6 Conclusion

After preforming all those task and reading about importance of labeling in YOLO algorithm we should be capable to prepare our own dataset for YOLO algorithm to detect "spaghetti" issue in 3D printing

References

[1] Dataset: 3D-Printer Defected Dataset:
https://www.kaggle.com/datasets/justin900429/3d-printer-defected-dataset

[2] Dataset: 3D printing errors:
https://www.kaggle.com/datasets/mikulhe/3d-printing-errors

[3] Computer Vision Annotation Tool: CVAT:
https://www.cvat.ai/

[4] Visual Object Tagging Tool: VoTT:
https://github.com/microsoft/VoTT

[5] MakeSense: MakeSense:
https://www.makesense.ai/

[6] Resizepixel: Resizepixel:
https://www.resizepixel.com