SURFACE DEFECT DETECTION BASED ON DEEP LEARNING APPROACH

II International Scientific Symposium
“Intelligence Solutions”
September 28-30, 2021

MYKOLA ROBOTYSHYN, MARIANNA SHARKADI, MYKOLA MALYAR

Uzhhorod National University

OUTLINE

1. Problem definition
2. Reed straws domain (очеретяні трубки)
3. Segmentation model -> Ensemble of segmentation models
4. Dataset, labeling options, images quality
5. Results

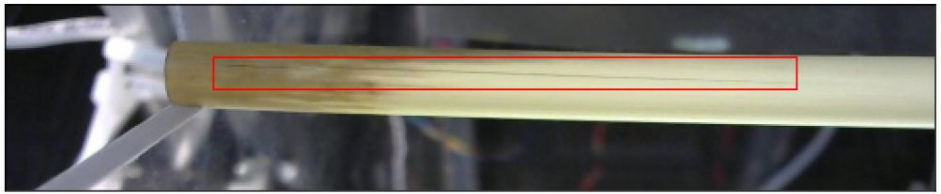
Problem definition

Surface defect detection, also known as quality inspection, is one of the most essential parts of any manufacturing process that helps businesses to ensure the quality of their product by detecting defects. Often, surface quality control is carried out manually and workers are trained to identify complex surface defects. Such control is, however, very time consuming, inefficient, and can contribute to a serious limitation of the production capacity.

Deep-learning methods have become the most promising approaches to solve this problem. The advantage of using deep learning is that it can detect the defects that cannot be detected by traditional machine vision algorithms.

Our paper proposed a way to build defect detection system from scratch based on deep learning algorithms, specifically applying deep segmentation neural network.

Reed straws domain (очеретяні трубки)



(a) Defect class: crack

(b) Defect class: circle holes

(c) Defect class: cut

(d) Defect class: black spots

(e) Defect class: brown spots

(f) No defects

To get access to dataset contact authors by email: *mykolarobotyshyn@gmail.com*

Main difficulties

1. Very often it is impossible to “see” whole object’s surface from 1 image thus we need more images from different angles -> more cameras -> optimal way to place cameras.
2. Very often to clearly “see” defects on image, image quality must be decent -> requires custom settings on web cameras.
3. Needed high-quality labeled dataset.
4. GPU’s.
5. Error-analysis, determining how good the system is.

Input images to the system

Binary segmentation network

1. Input data:

   X - rgb image;

   Y - 2d matrix, where each value can be 1 (pixel defected) or 0 (pixel not defected).

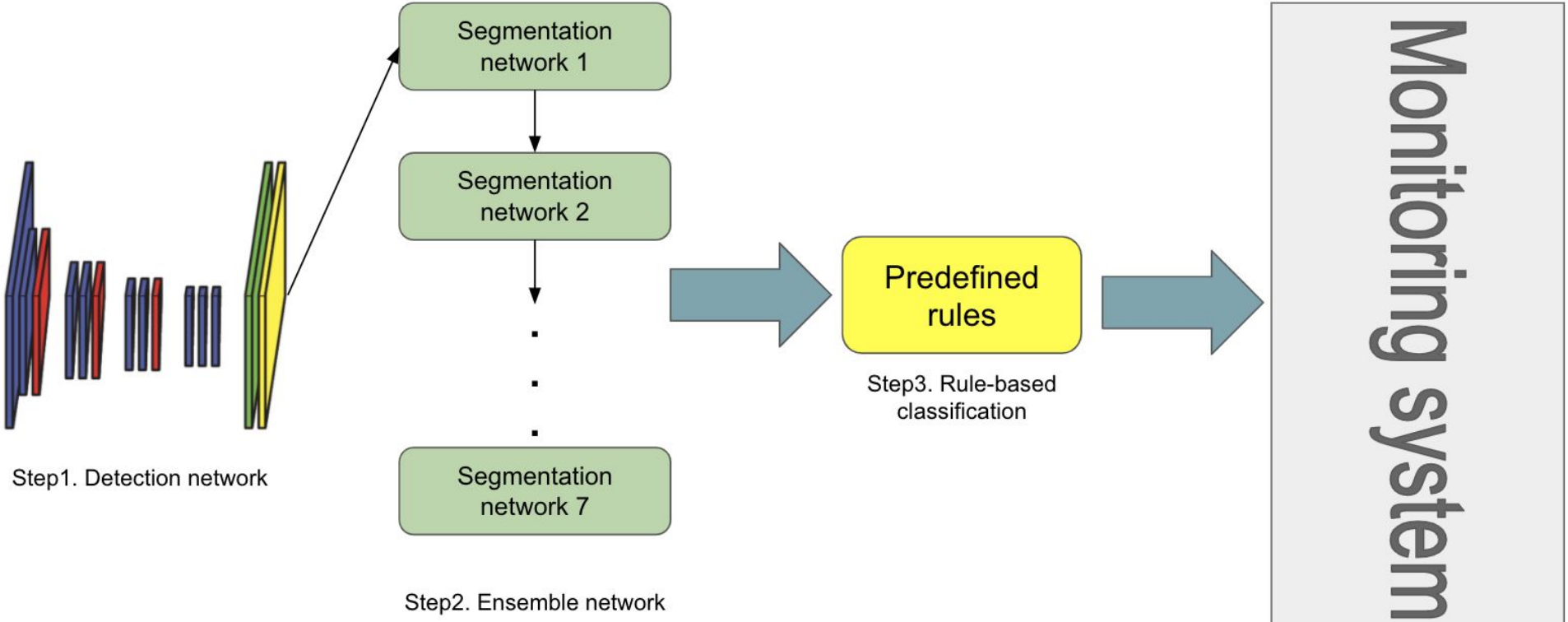
2. Model's output is two-dimension matrix termed as **mask** where each value corresponds to a pixel and has **probability in range [0,1]** for being defected. As a closer value to 1 is more likely the pixel is defected.
3. Then we apply threshold operation -> 2d matrix with values either 1 or 0.
4. Architecture: U-Net with mobilenet-v2 encoder + pretrained weights on ImageNet.

Binary vs Multi-class approach

The most time-consuming process of creating a dataset is not taking images but labeling them. Segmentation approach requires to label each defectable area on the image. To simplify and speed up the labeling process we created a separate dataset for each type of defect thus it required to label only one corresponding type of defect on the image.

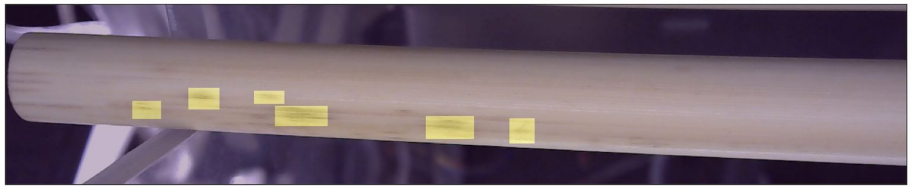
Better accuracy.

Main components of the system



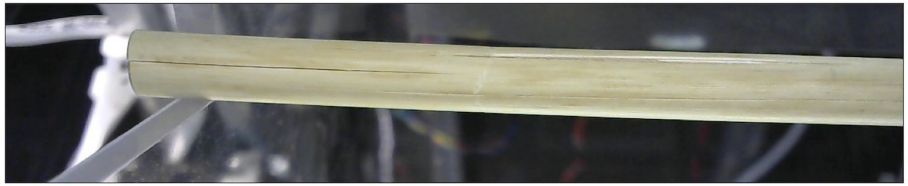
Different strategies of labeling images

Input image

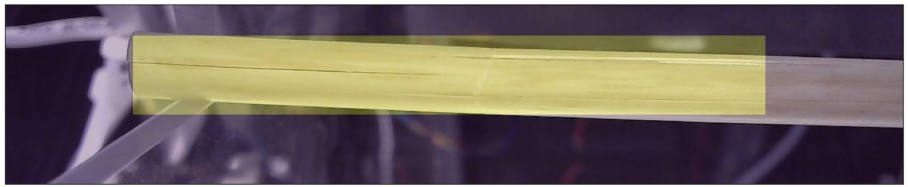


(a) Not strict labeling

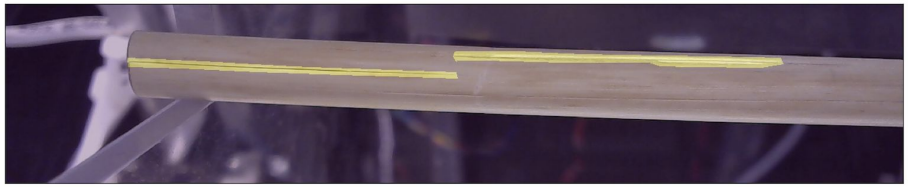
(b) Strict labeling



Input image



(a) Soft labeling



(b) Hard labeling

Results/error-analysis

To apply a surface defect detection system to production it is crucial to understand how accurate the system is. On the other hand, **it is difficult to evaluate the accuracy of the system with just a one number** due to:

- the large number of different defects;
- the different proportion of these defects in real-time;
- the impossibility in some cases to say unambiguously to which defect class the object belongs.

Compare to Human Level Performance

Table 2
System and human-level evaluation metric (only Recall)

	Defect class	System (%)	Human-level on images (%)	Human-level on real objects (%)
	Hammered stuff inside	90 % - 96 %	90+ %	95%
	Cut	97 %	99 %	99%
	Brown spots	97 %	95+ %	95+%
	Circle holes	90 % - 95 %	90+ %	95+ %
	Dark spots	90 % +	95+ %	95+ %
!!!	Cracks	60 % - 70 %	65 % - 73 %	85 %-90 %

What need to be changed to improve accuracy: **quality of input data or model** ?

Implementation details

- Proportion of positive and negative samples in the training dataset was not balanced and to fix this problem when it necessary we adjusted the class weight parameter in the cross-entropy loss function;
- Nvidia GTX 1070;
- 256x1248;
- To process 6 images with binary segmentation model 100 ms is needed thus ~17 ms for one image;
- To upload one binary model to RAM the server used approximately 600 mb and the whole ensemble network utilized up to 5gb;
- Python, PyTorch, Via labeling tool, OpenCV.

Discussion and Conclusion

1. Quality and diverse dataset rather than just “big”.
2. A method to compare results with human level results can give two main conclusions: whether system accuracy is acceptable for deploying it to production and whether data or models need to be changed to improve results.
3. Accuracy metrics close to human level performance were achieved on datasets with an amount of images within a range of 1.5k - 3k that are relatively small and can be collected in any domain.
4. For the reed straw domain, we have proven that the standard web camera Logitech C270 along with specific camera placement are suitable to assure high quality images.

Idea to improve results

We are thinking about a methodology to label defectable areas not as binary values 1 or 0 but rather as probability values in range of [0,1]. In our opinion, if we design a loss function in a way to penalize it more for mistakes where the probability value is close to 1 then a network will predict nondisputed areas more correctly than now and errors in disputed areas will be reasonable due to defect ambiguous definition. Such an approach requires spending more time on data labeling but it looks more natural and closer to a human's way to solve the problem.

Thanks you for your attention!