Honeywell contribution to PILOTING

Document Information

This document provides information about the work performed by Honeywell in 2020 as part of the H2020 PILOTING project. The intended readership of this report are the PILOTING consortium members, especially those who will continue in the work originally planned for Honeywell.

Table 1 summarizes authorship of this document. The issue date of this document is December 21, 2020.

Name	Role	e-mail
Petr Talla	Author	petr.talla@honeywell.com
Miroslav Štrba	Author	miroslav.strba@honeywell.com
Silvie Luisa Brázdilová	Reviewer	N/A
Jan Kubalčík	Reviewer	jan.kubalcik@honeywell.com

Table 1: Document authorship

Honeywell contribution scope

Honeywell contribution to the PILOTING project was AI detection of defects in building structures. This is obviously a very broad task which, in general, may include also detection of missing or broken parts. However, our scope was limited to assessment of material health conditions regardless of the type of building made of such material.

Defects can be classified according to various aspects:

- **Material** – steel, concrete, al-alloys, zn electroforming steel cover, zn hot-dip galvanized steel, stainless steel, brickwork, wood, watering sediments, aerosol presence, corrosion protected steel etc.
- **Defect category** – corrosion, delamination, crack, cover delamination, watering, watering sediments, aerosol presence, dirt etc.
  The same defect category can behave differently in various materials.
- **Structure category** – bridge, tunnel, piping etc.
  Some defect categories are emphasized in specific structures (cracks and water in tunnels)

The most important classification is material/defect category matrix (see Figure 1).

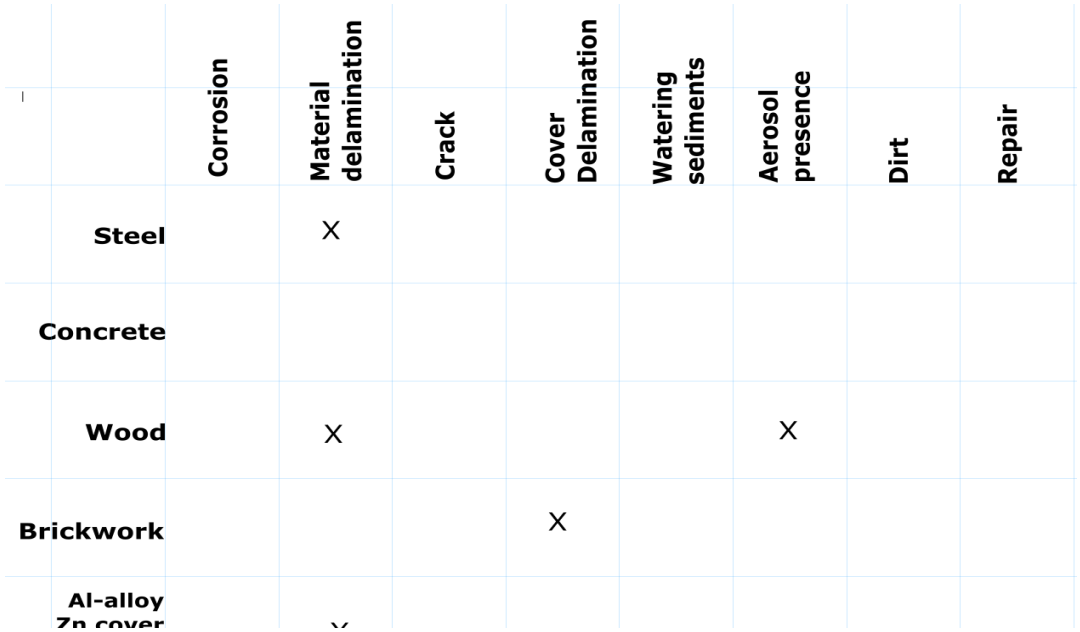

Figure 1: Classification matrix

Each cross section in this matrix represents a separate knowledge task. For each task, specific samples must be collected and annotated. In the early times of development, we formulated questionnaire to cover most of the needs of participants (i.e. end users) with acceptable effort. The questionnaire result is documented in Figure 2.

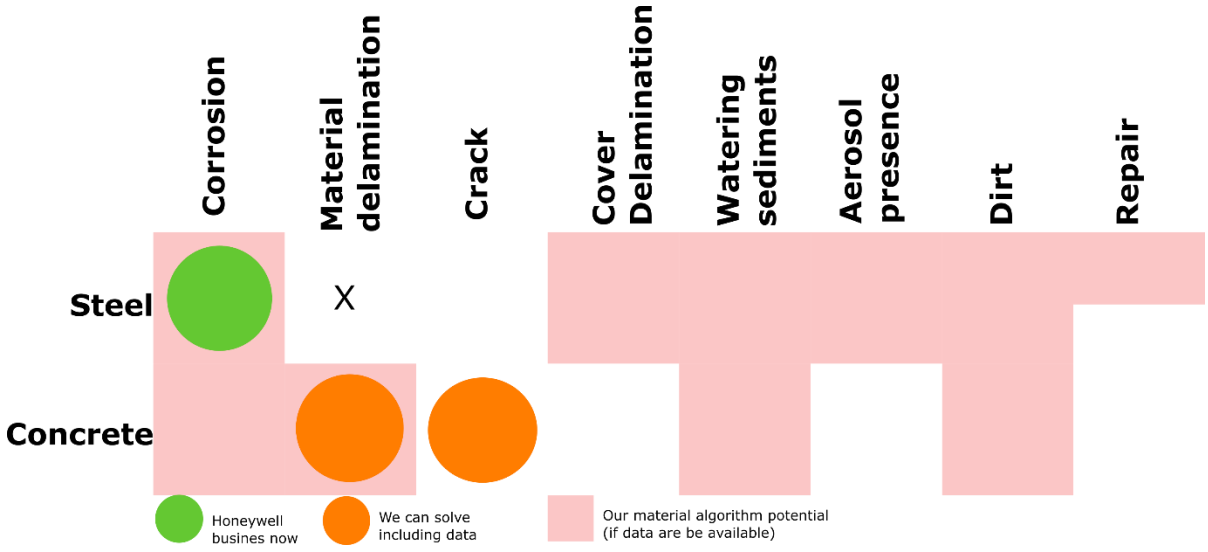

Figure 2: Questionnaire result

We agreed with the PILOTING consortium to target on the following combinations:

1.) **Steel/corrosion** – planed realization JUN 2021 (samples provided by Honeywell)
2.) **Concrete/material delamination + rebar exposure** – planned realization JUN 2021 (if samples are available from PILOTTING consortium partners)
3.) **Concrete/crack** – planned realization DEC 2022

4.) **Concrete/watering sediments** – planned realization DEC 2022

The abovementioned combinations satisfy 2/3 of the relevant requirements found in the questionnaire.

While completing each material/defect cross section needs separate knowledge analysis and annotated sample collection, the core algorithm can be widely reused.

We planned to realize two algorithms:

1.) **Texture recognizer** – AI based algorithm which identifies texture of material. Different texture than texture of perfect material marks failure and even its type and quality. Planned realization JUN 2021
2.) **Crack locator** – AI/image processing-based algorithm which identifies material cracks. Planned realization DEC 2022

Both algorithms cover more than single material/defect crossing. The texture recognizer covers all the crossings marked by pink. Crack locator covers whole crack failure. Therefore, for each task (i.e. material/defect combination), only problem analysis, sample collection/annotation, and training need to be performed, but the algorithm itself can be reused.

Regarding to early project termination (DEC 2020), only steel/corrosion problem was partially solved, so the next text will cover only the steel/corrosion case. There is available the problem analysis, a collection of samples from 129 independent buildings, of which 205 annotated images are created. Moreover, a basic texture samples generator was developed, and a proof-of-concept of neural texture recognition performed. Because the AI proof and annotation run in parallel, we have also prepared a collection of easy-to-reach general texture samples.

Figure 3 illustrates the complete solution. Its components will be in detail described in the next sections.

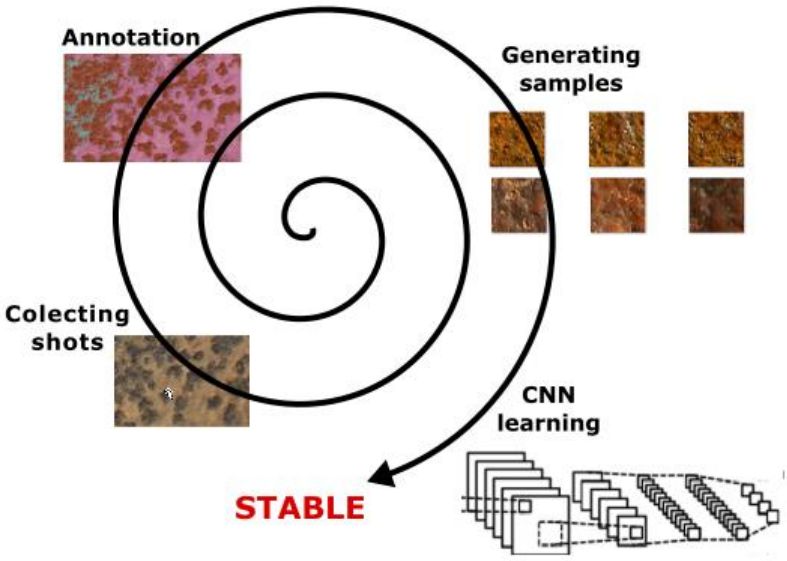

Figure 3: Iterative process of neural network training.

Data acquisition

Based on scale, images of building structures can be divided into three categories – near, medium, far. This is illustrated in Figure 4.

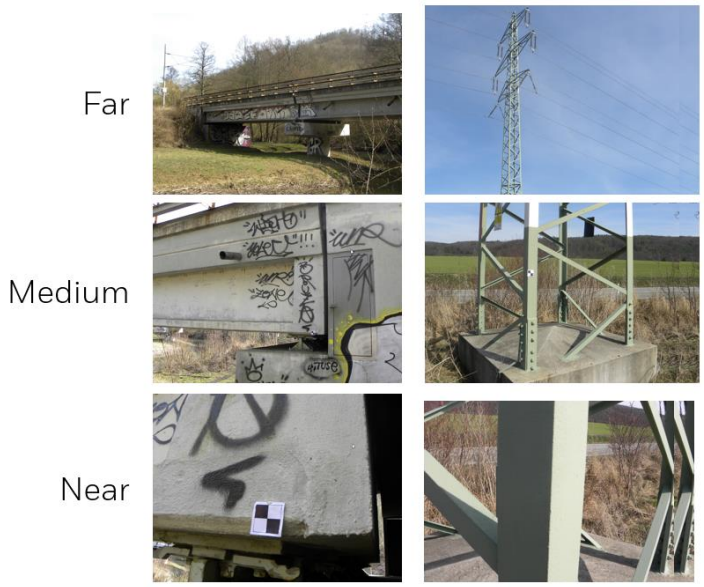

Figure 4: Far, medium and near scale image examples.

Texture recognition is possible only in near scale. Only the near images enable to identify the quality of rust, salt accelerating the corrosion and that there is quick coat repair without previous coat removal. This fact is illustrated in the following comparison:

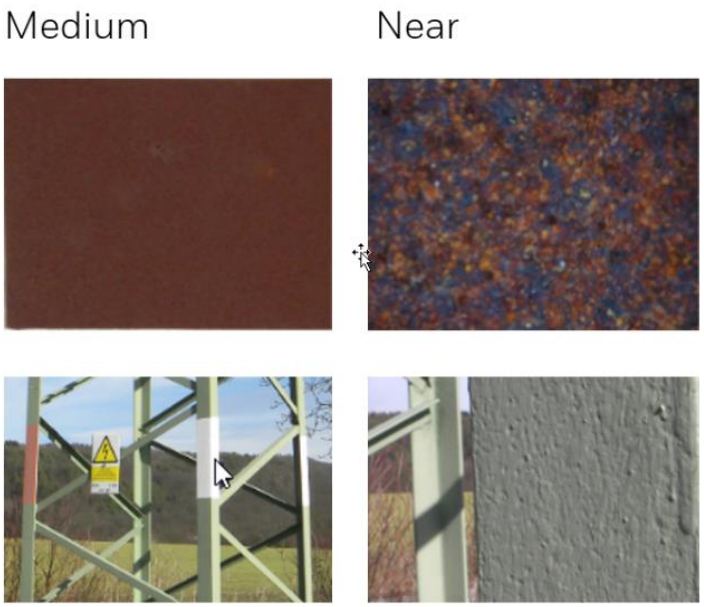

Figure 5: Details are captured only in near scale images.

For steel there are expected near scale images with resolution at least **8 pixels/mm**. The precision of the resolution information is at least ±20 %. For training samples this can be satisfied by a mark of known dimensions (50 mm) positioned in the image, as in the bottom left image in Figure 4. In real process this can be satisfied by lidar, radar or single laser beam data, associated with the image.

Data requirements for neural network training

Images must be proportionally distributed per all potential states of material. No more than three near scale images can be taken per individual building instance (defined by homogeneity of building year, material, microclimate and maintenance). If we count with 300 near scale images, so we can obtain 100 m^2 of material for the first iteration. **It can be expected that such sample set covers all failure aspects of the material worldwide.** It is expected that the solution can start converging with such data amount, but it is not guaranteed. In such a case the sample set must be completed, based on results of each iteration.

Data annotation

It is expected that each sample will be annotated on site when the image is taken. Otherwise it is not possible to annotate with adequate precision: the human annotator must be at hand touch distance to the imaged object to recognize the features correctly. We used textual annotation, which is stored in a file of the same base name as the image, but it has extension *.txt. This is shown in the example in Figure 6.

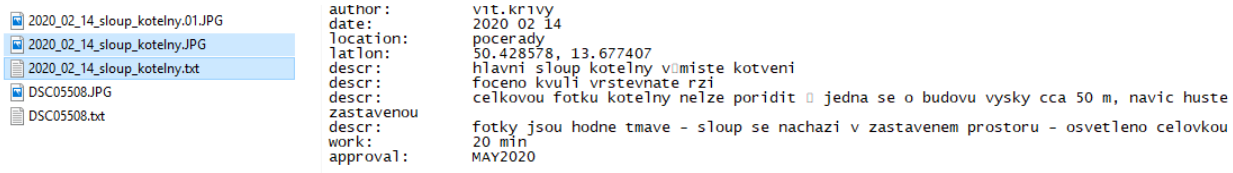

Figure 6: Example of textual annotation associated with JPG image.

The description is in the file 2020_02_14_sloup_kotelny.txt, and it is associated with the file 2020_02_14_sloup_kotelny.jpg. Note that a descriptive file can be associated with a whole directory. In such a case the information is valid for all files in the directory. Note that the description is in Czech language, because we did not expect the primary usage out of Honeywell in Brno, Czech Republic.

Note that another approach for in-situ quick annotation is the annotation application, as it is shown in Figure 7, but we have not used this approach in our data.

Figure 7: Result of in-situ annotation using a dedicated application.

Each near scale image ought to have associated also a medium scale image, but it is not necessary. The medium scale image is expected to be used for recognition results displaying, as it is shown in Figure 8.

Figure 8: Medium scale image is suitable for results presentation.

Feature Set

Each separate problem, such as steel/corrosion, starts with deep analysis. This means that the types of defect (in this case corrosion) are identified, for example, based on corrosion depth. Together with other artifacts potentially available on the image (such as dirt, painting etc.) a feature set is identified, as it is illustrated in Figure 9. It is expected that whole image will be exclusively annotated on pixel basis with these features.

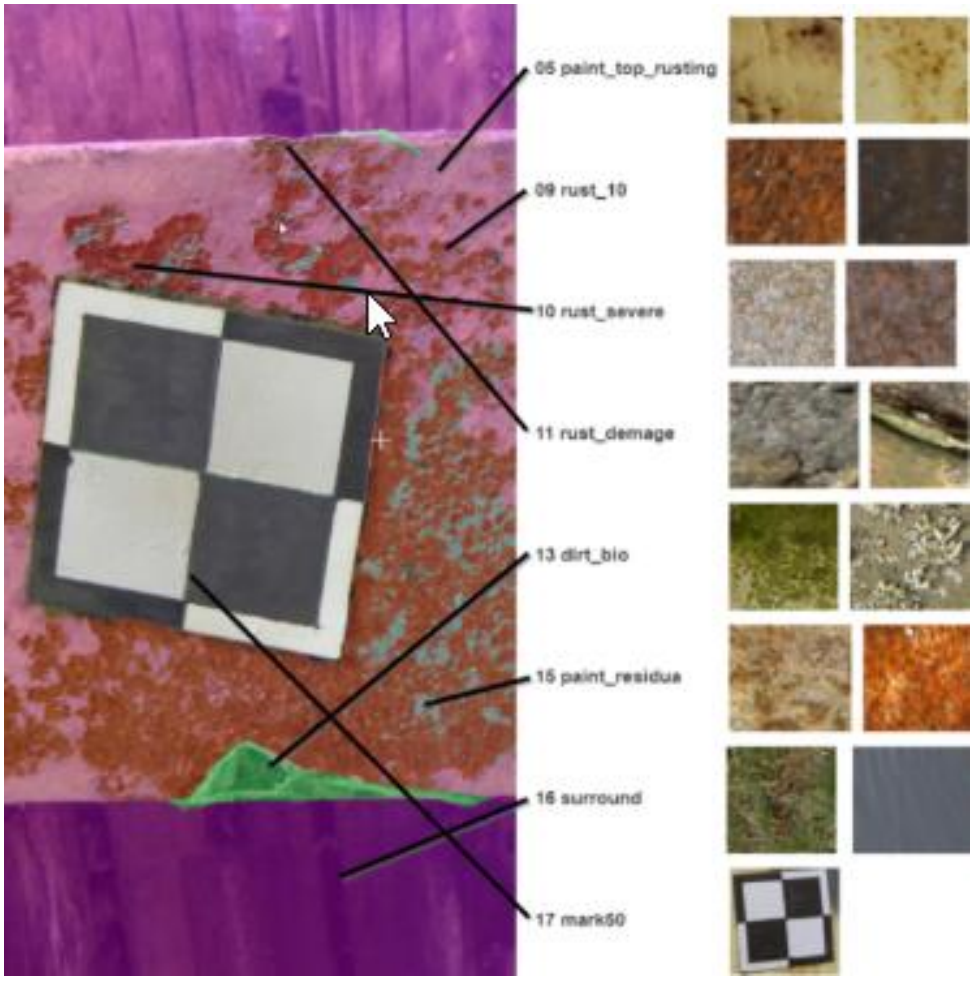

Figure 9: An example of several features found in an image.

The annotated features are stored in file with the same base name, as the annotated image, but it has extension *.00.anno.png. Vector drawing (mark50 feature or another vector annotation) is stored in file with extension *.t2l. Each annotated file has also associated feature list in the file *.features.t2l.

The file *.00anno.png is palette-based image, where palette index is feature index. Palette color has no significance and it can be freely changed.

We have defined a complete feature set for the problem of steel/corrosion.

Annotation Tool

The annotation application is primarily tool for annotating images. It can be used for handling the image file viewer, as it is shown in Figure 10.

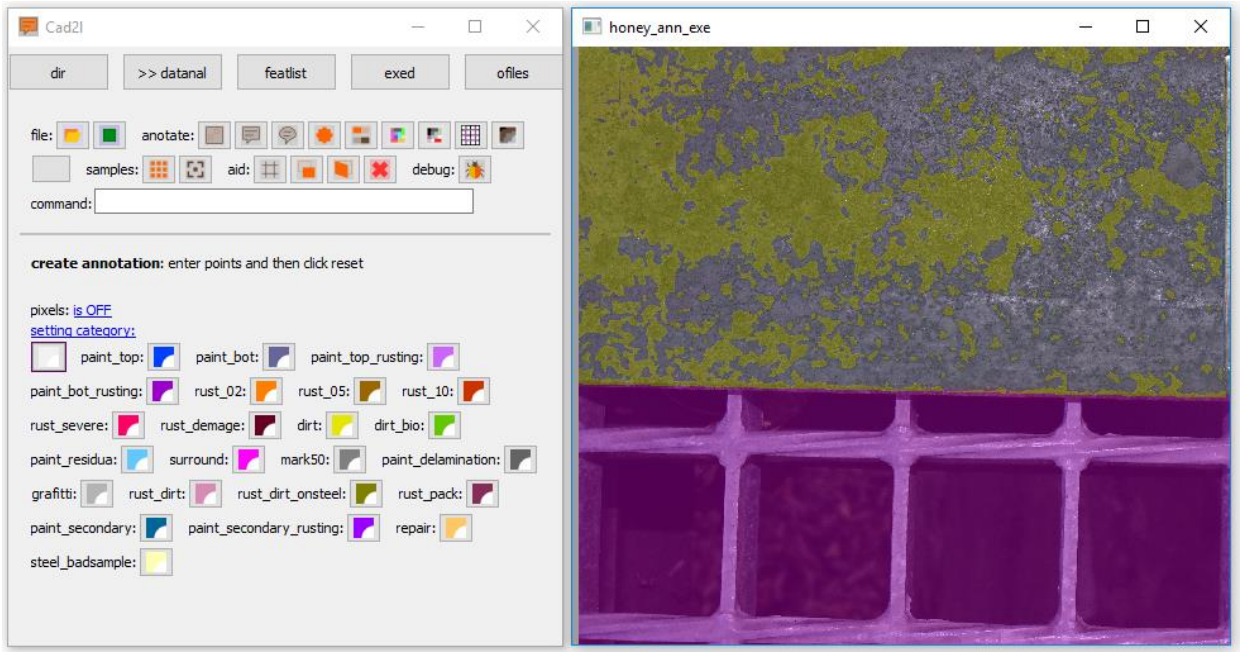

Figure 10: Annotation tool.

The application also enables easy view of data using all the rules above in simple form, as it is shown in Figure 11:

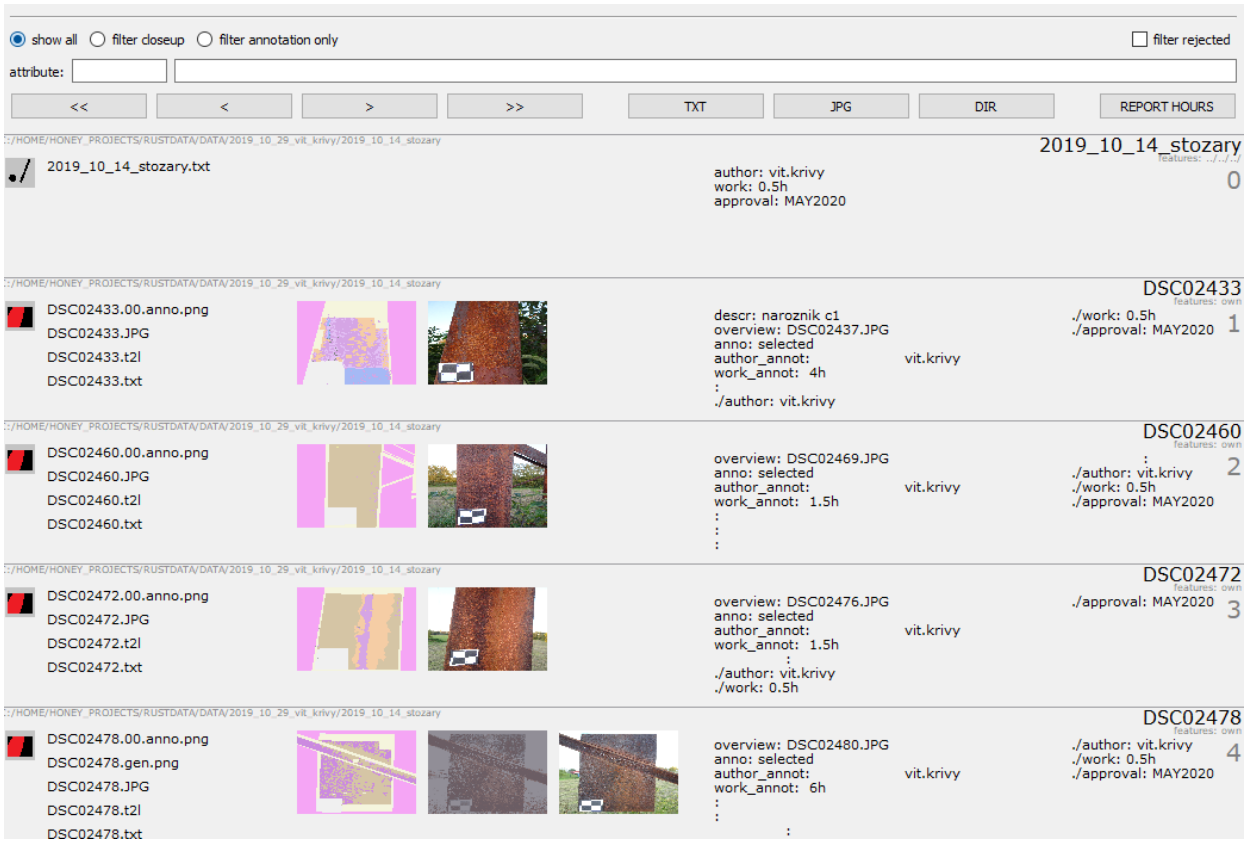

Figure 11: View of associated files.

The basic functionality of the annotation application is presented in a documentation file [3] (also provided with the application).

Samples Generator

The samples are not used directly, but texture samples are generated first. It is expected, that 30 samples are generated based on one image at most, so about 50 000 samples can be estimated in the first iteration. Such amount could be suitable for intended converging solution of the NN, especially using AI techniques such as pre-learning.

The sample data are generated using query. Query is a set of rules constructed to extract samples from annotated images with a specific features composition. There are 2 main types of rules. First one is simple and corresponds to all relevant sample characteristics. More precisely, the simple query enquires samples about annotated features, it means whereas the samples contain *less/greater than* or *equal to* measure of area with some annotated feature. Second type is complex and joins rules above (the simple queries) together. The simple queries could be combined by AND/OR conjunctions. So, based on these rules complex expressions for groups are created (for instance a group where at least 30 % of paint and maximum 25 % of minor rust or 30 % of paint and maximum 5 % of minor rust and maximum 5 % of major rust are present – see Figure 12).

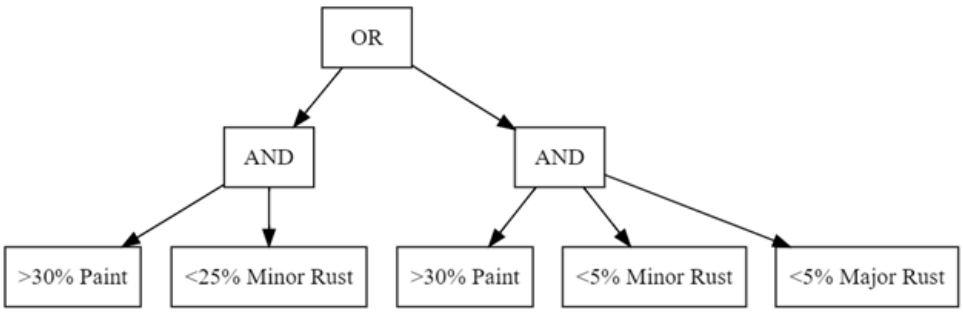

Figure 12: An example of a complex query.

The query generator randomly passes through images and checks if a (typically) rectangular region satisfies the constructed query. If the query conditions are met, the region is recorded as a training/testing sample.

The example of query “samples of size 1.5 cm where precision >= 100 px and rust_10 > 75 % and paint_residua < 20 %” on annotated image can be shown in Figure 13. Result of the query algorithm is schematically depicted for this example: 0 means that the query is not complied, 1 means that the sample is accepted.

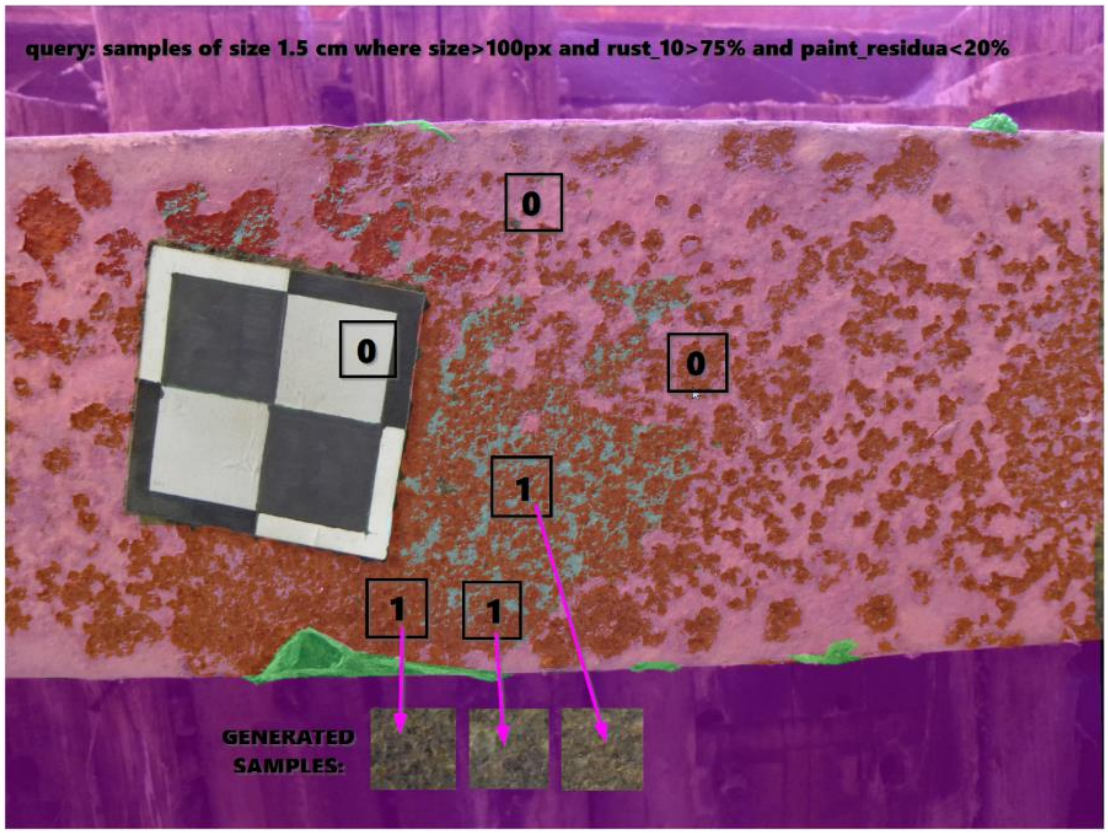

Figure 13: Examples of rectangular areas that satisfy (value 1) and do not satisfy (value 0) given query.

Figure 14 shows another example: samples that all satisfy query “1 cm sample size and rust from 0.2 mm and at least 85 % of area and paint under 10 %”.

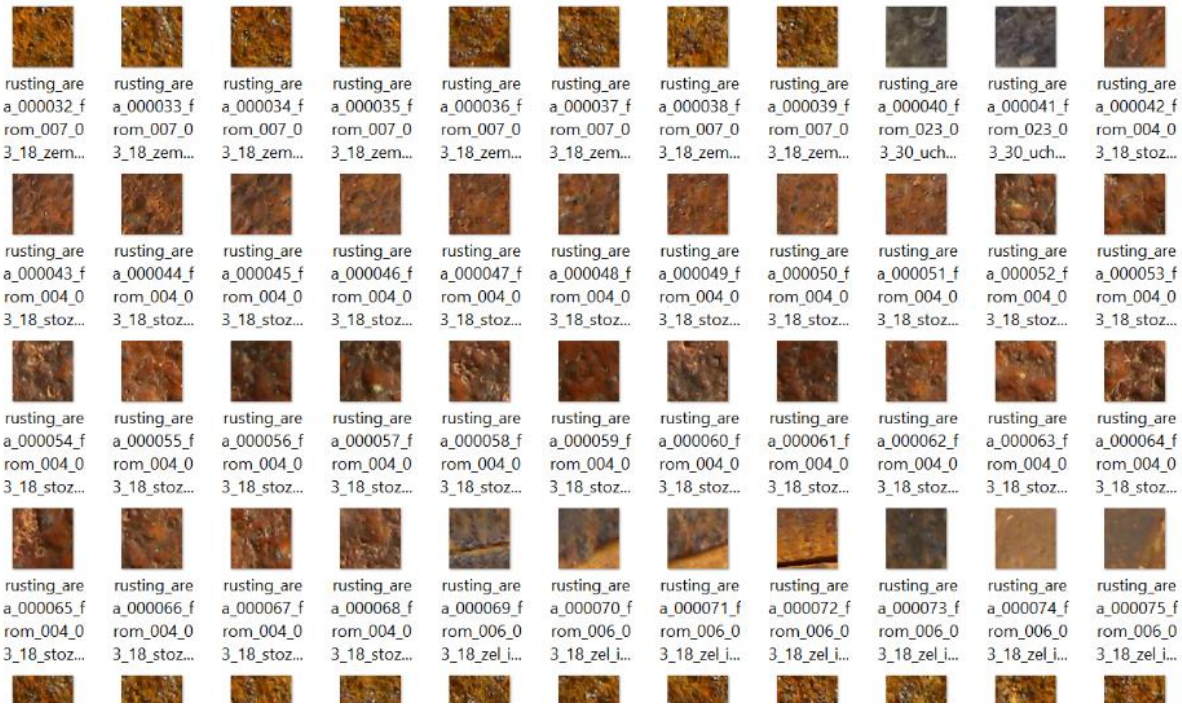

Figure 14: Example of samples that satisfy a common query.

The “all in one” sample generator principle is shown in Figure 15. In this example is also the query written in real query language:

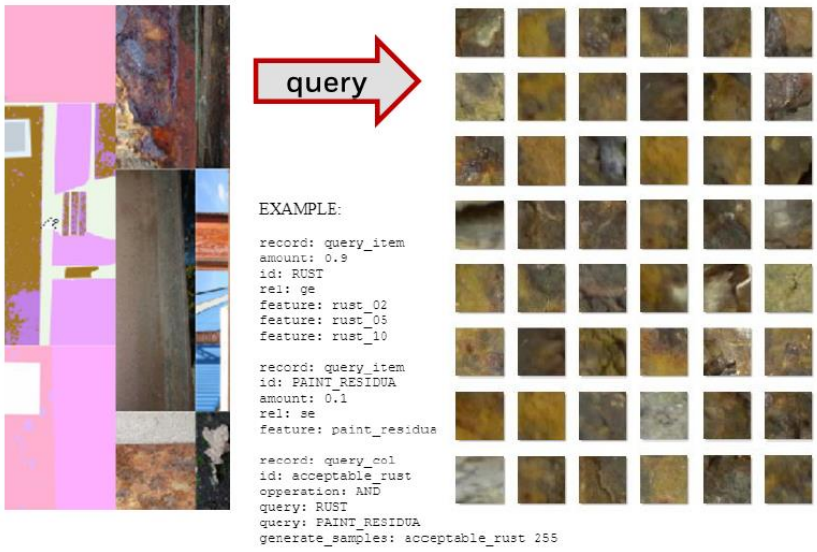

Figure 15: Another query example.

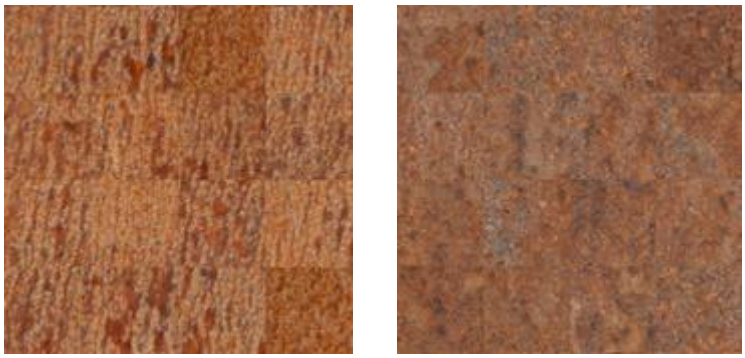

Figure 16: Rust samples with rust depth less than 2 mm (left) and less than 10 mm (right).

show examples of rust with different rust depth.

Simplified NN prototype

Our participation in the project was limited till DEC 2020, while delivering the first prototype of NN was planned in JUN 2021. The first annotated data availability was planned to DEC 2020, since this required not only capturing the data (photos from various buildings), but also development of the Annotation tool and providing training to the annotators before annotations could even start. Nevertheless, we decided to check at least the suitability of this approach.

For this reason, we collected 262 textures available freely on the Internet. Selected images are big enough to use one texture image for one category. Textures should be as much uniform as possible to be suitable to extract samples. Samples are small areas with same features, however different enough to cover within-category variability. Recommended size of sample is 150 x 150 pixels. Every texture should contain at least 50 (optimally at least 75) non-overlapping samples.

The original approach we planned to implement is based on [1]. In this modified version we implemented and tested the approach published in [2].

The planned and simplified solution relationship is illustrated in Figure 17.

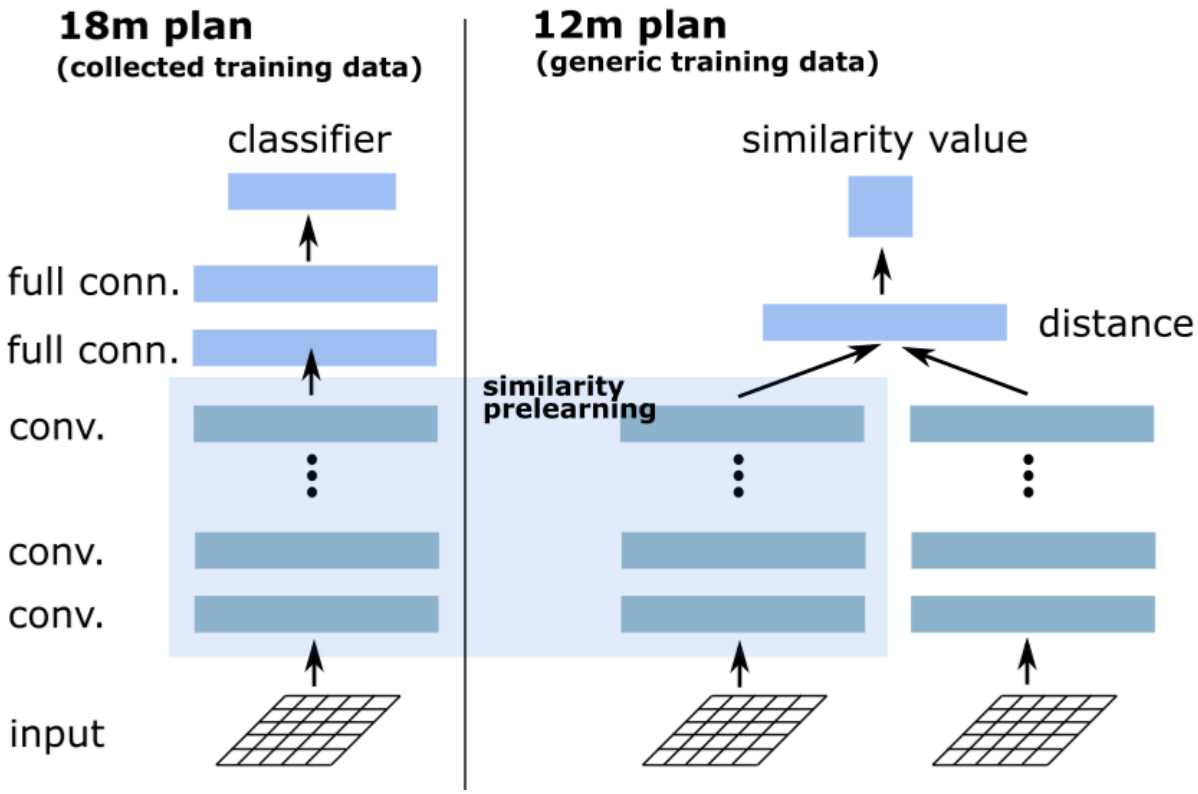

Figure 17: Originally planned (left) and tested (right) NN architectures.

It can be seen that the core of NN is identical in classifier and Siamese network, so the results from Siamese training could indicate the capability of texture recognition. It is also possible to apply the trained network on textures from steel/corrosion detection problem.

Figure 18 shows result of training on 22 categories from collected textures. 64 samples from each texture were extracted. 10 of them were used for testing.

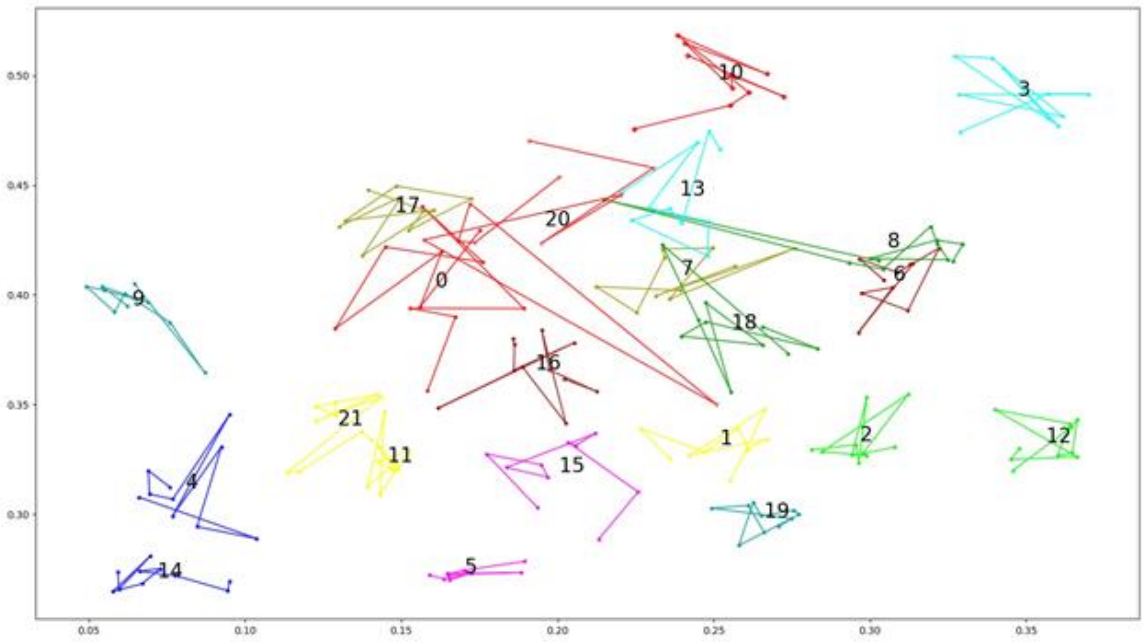

Figure 18: Each numbered group of points connected by a piecewise direct line represents one group of texture samples.

Piecewise direct lines are joining samples that belong to the same category for better visualization of the results. Distance between categories indicate how similar the categories are. Distance between dot symbol and category label indicate within-group variability.

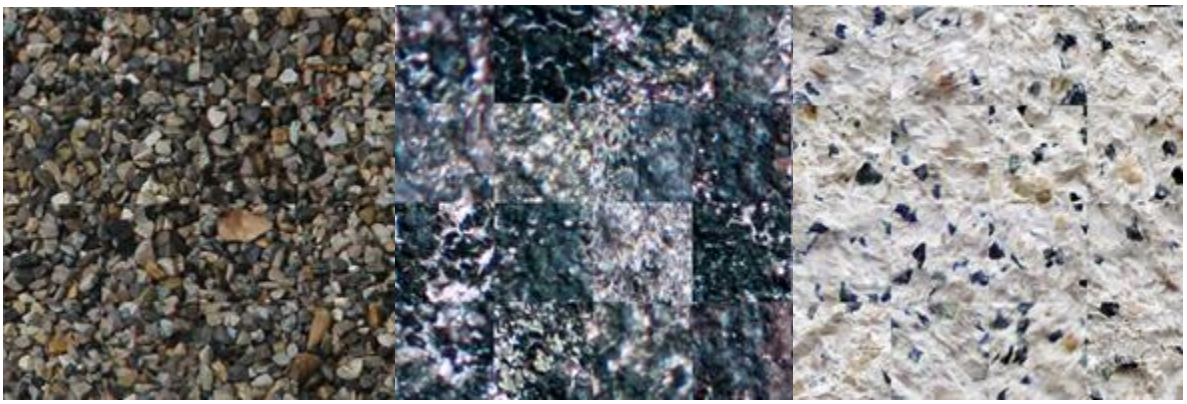

Figure 19: From left to right: Category 6 – dark gravel, category 8 – graffiti rock wall, category 21 – white wall with black pubbles.

From Figure 18 we can see that categories 6 and 8 are quite similar. However, samples from category 21 are easy to distinguish from these two categories. Samples of these textures can be seen on Figure 19.

Conclusion

References

[1] Andrearczyk Vincent, “Deep learning for texture and dynamic texture analysis”, 2017, PhD Thesis. Available online: http://doras.dcu.ie/22040/1/Vincent_Andrearczyk_Final_PhD_thesis.pdf.

[2] L. Hudec and W. Bencsova, "Texture Similarity Evaluation via Siamese Convolutional Neural Network," 2018, 25th International Conference on Systems, Signals and Image Processing (IWSSIP), Maribor, 2018, pp. 1-5, doi: 10.1109/IWSSIP.2018.8439387.

[3] Documentation to annotation tool:

readme.pdf