INTRODUCTION TO IMAGE PROCESSING AND COMPUTER VISION

LABORATORY PROJECT 2 (LABORATORIES 3 & 4)

Rafał Jóźwiak
R. Jozwiak@mini.pw.edu.pl
Faculty of Mathematics and Information Science
Warsaw University of Technology

REALIZATION

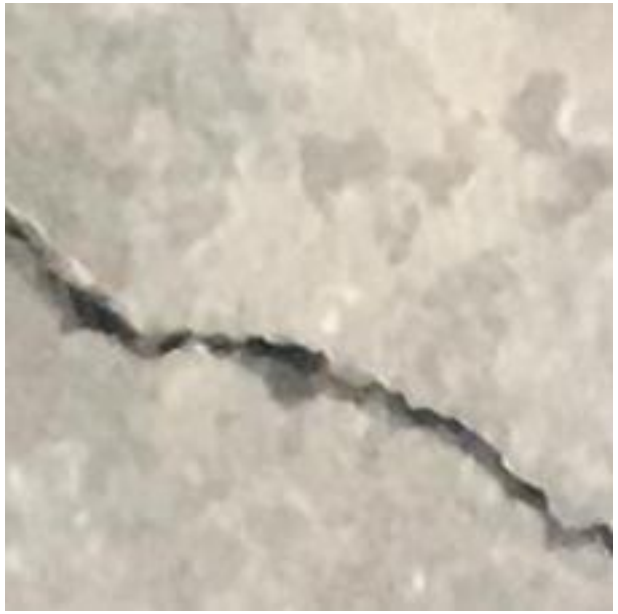
- algorithms elaborated with OpenCV library
  - OpenCV (C++)
  - SimleCV/OpenCV (Python)
  - EmugCV (C#)
- usage of other libraries (for texture characterization and ML) is also allowed
- solution for the laboratory task should contain:
  - source code with description (GUI is not obligatory)
  - documentation (description of solution, testing procedure, results and comments)
- solution should be send up to 26.01.2021

DOCUMENTATION

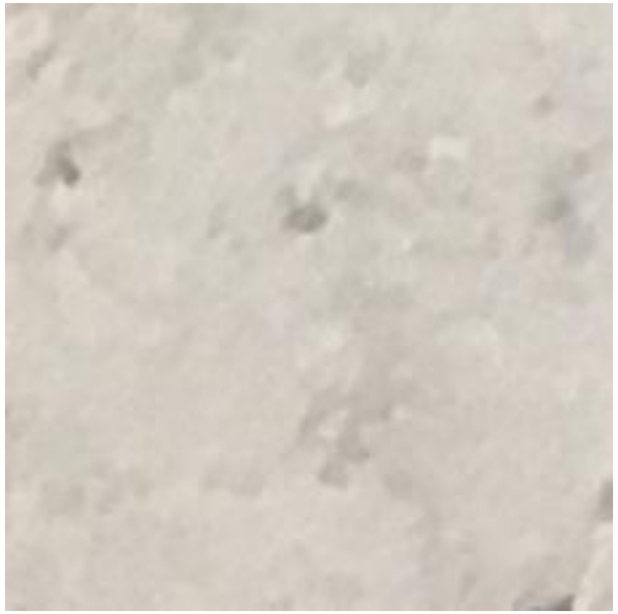
- documentation should contain:
  - short problem definition (task description, data set description)
  - research methodology
  - results (tables)
  - results (visualizations)
  - comments and conclusions

SURFACE CRACK DETECTION

- input: samples of surface (2 classes, 20,000 samples per class)



Positive

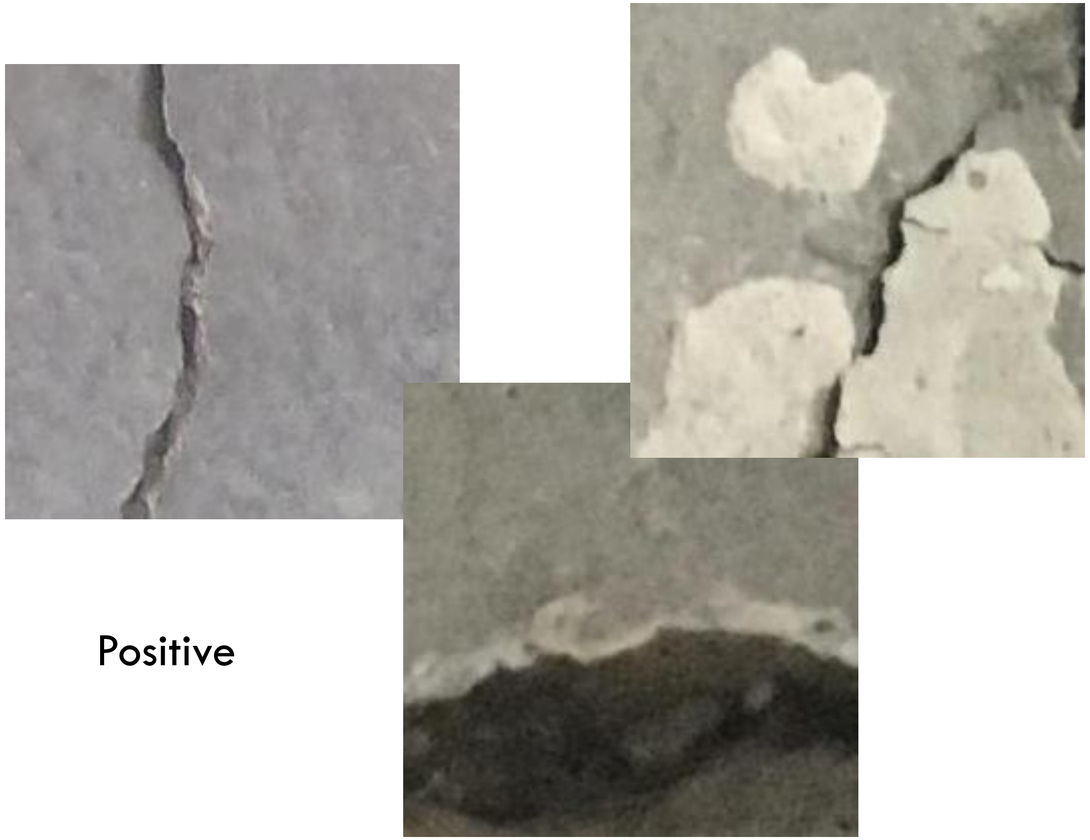


Negative

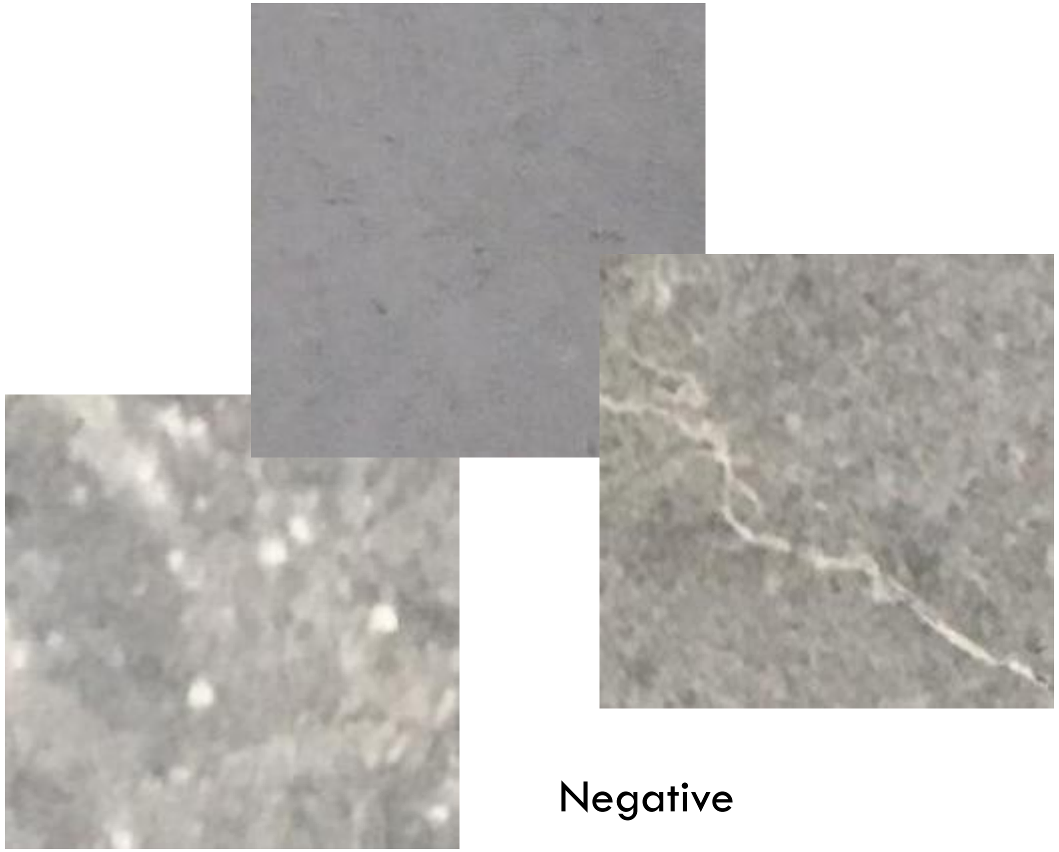
https://www.kaggle.com/arunrk7/surface-crack-detection

SURFACE CRACK DETECTION

- input: samples of surface (2 classes, 20,000 samples per class)



Positive

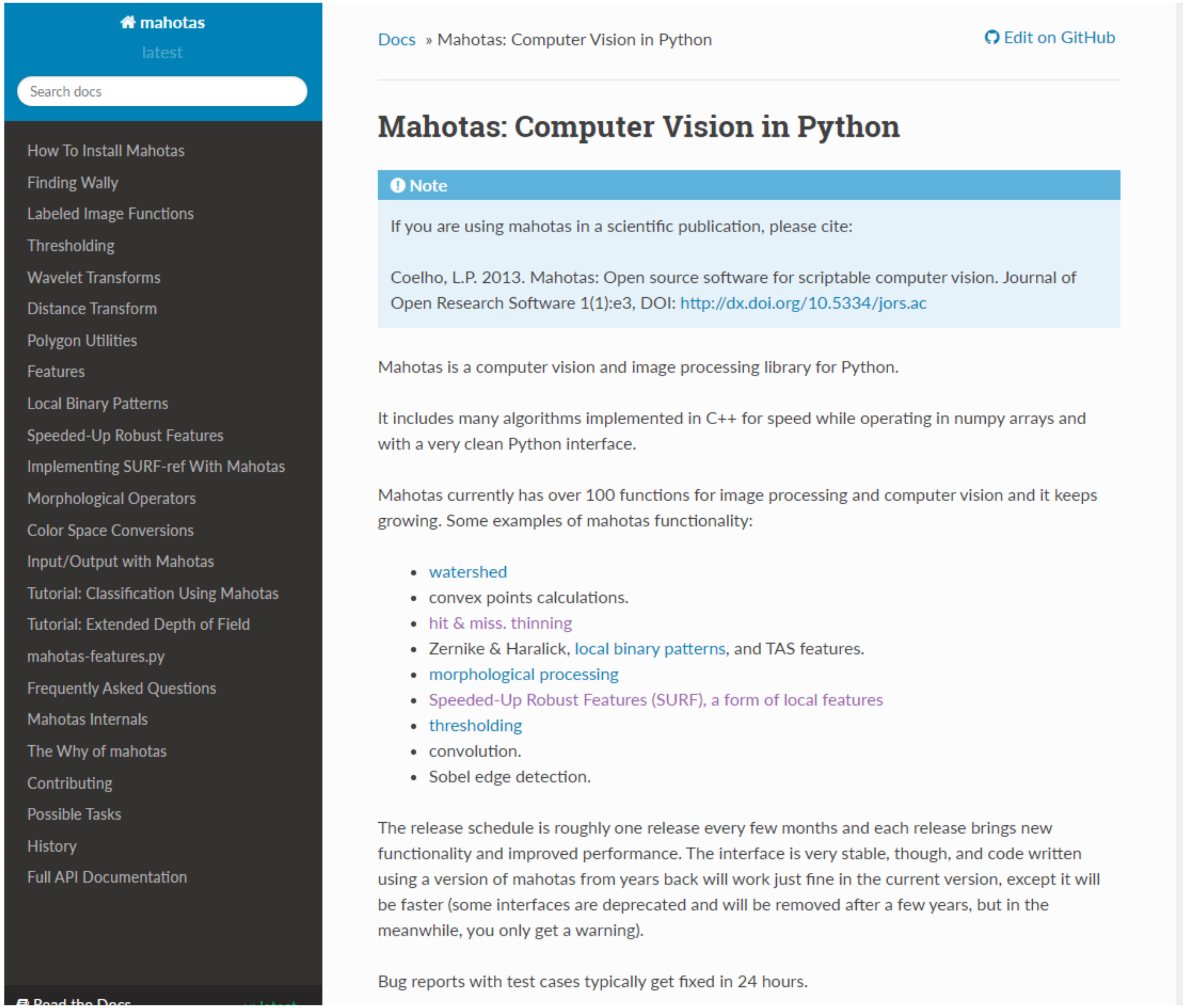


Negative

SURFACE CRACK DETECTION

- aim: elaboration of discriminative feature vector (for classification purpose), to assess quality of different features (assuming different research scenario)
- input: samples (227 x 227)
- output: sample class -> classification accuracy -> features quality
- try to use different features (texture features, LBP, local features descriptors, etc.)
- for the purpose of classification use any classifier (usage of different environments is allowed e.g. R, Python, Matlab etc.; try to use quite simply classifier) – DO NOT CHANGE DURING EXPERIMENTS!!!
- try to prepare analysis for:
  - different types of texture features (initially independently) e.g. statistical, GLCM, LBP, etc.
  - combining different groups of features together
- for learning and classification evaluation use:
  - divide into training and testing set (80% training / 20% testing)
- try to assess impact of set size (accuracy as a function of samples number)
  - start using 1000 P + 1000N, increasing by 1000 every time
- usage of CNN (as a magic black box) is NOT ALLOWED !!!

TOOLS



Mahotas: Computer Vision in Python

Note

If you are using mahotas in a scientific publication, please cite:

Coelho, L.P. 2013. Mahotas: Open source software for scriptable computer vision. Journal of Open Research Software 1(1):e3, DOI: http://dx.doi.org/10.5334/jors.ac

Mahotas is a computer vision and image processing library for Python.

It includes many algorithms implemented in C++ for speed while operating in numpy arrays and with a very clean Python interface.

Mahotas currently has over 100 functions for image processing and computer vision and it keeps growing. Some examples of mahotas functionality:

- watershed
- convex points calculations.
- hit & miss. thinning
- Zernike & Haralick, local binary patterns, and TAS features.
- morphological processing
- Speeded-Up Robust Features (SURF), a form of local features
- thresholding
- convolution.
- Sobel edge detection.

The release schedule is roughly one release every few months and each release brings new functionality and improved performance. The interface is very stable, though, and code written using a version of mahotas from years back will work just fine in the current version, except it will be faster (some interfaces are deprecated and will be removed after a few years, but in the meanwhile, you only get a warning).

Bug reports with test cases typically get fixed in 24 hours.

https://mahotas.readthedocs.io/en/latest/index.html