

# **Exercise 2 — Image Processing Documentation**

Matthias Rupp, Janik Mayr

November 17, 2019

# 1 Start Instructions

The program can be started with "java -jar <path>imageprocessing.jar <args>". The table 1.1 shows the possible (and for jar-build required) arguments. The arguments can be passed like the following: <name>=<value> Example for minimal required input: java -jar imageprocessing.jar -src=loetstellen.jpg -exptCoords=expectedCentroids.txt When no path is given in the respective arguments for the outputs (see table), the program will create a folder imageprocessing/target in the folder where the jar is located. There, the reports.txt and the disks.png will be put.

Name	Required	Definition	Default
-src	yes	Source image	Path: imageprocessing/src/main/resources/loetstellen.jpg
-acc	/	Accuracy	3
-pull	/	Switch to Pullpipe	false
-exptCoords	yes	File with expected Coords	Path: imageprocessing/src/main/resources/expectedCentroids.txt
-disksImgOut	/	Output for created DiskImg after Erosion	Path: imageprocessing/target/disks.png
-diskReportOut	/	Output for created Report	Path: imageprocessing/target/report.txt
-shapeTypeErode	/	Shape type for erode filter Range[0 - 3]	0
-kernelSizeErode	/	Kernel size for erode filter	2
-kernelSizeMedian	/	Kernel size for median filter	19
-removeFirstN	/	Removes the first n found disks in the given image	1

Table 1.1: Table of command-line arguments

## 2 Workflow Description

The exercise is utilizing the OpenCV framework (Intel and Garage 2019) and the maven artifact `nu.pattern::opencv::2.4.9-4` to execute locally without OpenCV dll files installed. The following workflow description follows the chronological steps of the process, which order of filters are represented by the pull pipe. The source of the pipeline (Class: `ImgSource`) reads the passed file; converts the read image in an OpenCV matrix (Class: `org.opencv.core.Mat`) and exports a `DataTransferObject` DTO (Class: `ImgDTO`) containing the created matrix. The first filter (Class: `ROIFilter`) virtually crops the image by creating a sub-matrix of the passed matrix, the selected area is refereed to as Region of Interest (ROI). Additionally the offset between the origin of the original matrix and the ROI matrix is saved for later. To better distinguish between the disks and the background the second filter, a threshold filter (Class: `ThresholdFilter`) is used; the threshold type is Binary, Inverted. The formula for the type (*Basic Thresholding Operations*) projects every pixel form the src matrix grater then the threshold 30 to white and all other to black.

$$dst(x, y) = \begin{cases} white & \text{if } src(x, y) > 30 \\ black & \text{otherwise} \end{cases}$$

The result still contains noise, which can be minimized by a median blur filter (Class: `MedianFilter`). According to the OpenCV documentation (*OpenCV: Smoothing Images*) each pixel is set to the median of its neighboring pixels. The size in pixels for the neighbourhood is called kernel size. The forth filter, a `ErodeFilter` (Class: `ErodeFilter`) removes the cables to the disk, by removing pixels not part of a structuring element. The fifth filter (Class: `Convert2JAIFilter`) converts the the matrix into a `PlanarImage` and passes it with the ROI offsets to the given filter `CalcCentroidsFilter` and creates a DTO containing the provided coordinates (middle of the disks) and the input `ImgDTO`. The sixth filter (Class: `FilterFirstDisk`) does not modify the images, but removes the fist n found centroids, this was done to match it to the size of expected centroids list. The seventh filter (Class: `QualityCheckFilter`) checks if the detected coordinates are within the given accuracy and creates a report for each centroid with its results. The eighth filter (Class: `FindRadiusFilter`) calculates the minimum and maximum radius of an enclosing circle for a centroid of an report. Finally, the sink creates a file containing all information of the reports.

For the pull pipe, the `SinkImpl` is run and uses the same filters as the push pipe, only that the read method is used and `Readables` are passed through the pipe. The reason for the many casts in the `MainPipes` class is the fact that constructors for both `Readables` and `Writeables` were overwritten, to enable us to create both a push- and a pull-pipe. Because of this, the Compiler can't tell which constructor it should use, requiring the use of casts.

# Bibliography

Documentation, OpenCV 3.4.8. *OpenCV: Smoothing Images*. URL: [https://docs.opencv.org/3.4.8/dc/dd3/tutorial\\_gaussian\\_median\\_blur\\_bilateral\\_filter.html](https://docs.opencv.org/3.4.8/dc/dd3/tutorial_gaussian_median_blur_bilateral_filter.html) (visited on 11/15/2019).

Intel and Willow Garage (Oct. 2019). *OpenCV*. URL: <https://opencv.org/> (visited on 11/15/2019).

OpenCV 2.4.13.7 Documentation. *Basic Thresholding Operations*. URL: <https://docs.opencv.org/2.4/doc/tutorials/imgproc/threshold/threshold.html> (visited on 11/15/2019).