PrepOCRessor – The QCRI Preprocessing Tool for OCR Version 0.2.1

Qatar Computing Research Institute, HBKU
Felix Stahlberg and Stephan Vogel

2015-09-01

Abstract

This document describes the capabilities of the open source software *PrepOCRessor*. The tool is developed at the Qatar Computing Research Institute for preprocessing document images for optical character recognition. The tool follows the pipeline paradigm in Unix-like operating systems: A set of image processing operations is chained such that the output of each operation serves as input to the next one. The tool supports batch processing for high parallelism and scalability. The OpenCV (Bradski and Kaehler, 2008) library provides efficiently implemented computer vision algorithms and a efficient infrastructure. *PrepOCRessor* is intended to be used in combination with the recognition toolkit *Kaldi* (Povey et al., 2011) and supports file formats for feature sets (.ark,t) and forced-alignments (.al) for a seamless integration. Even though we focus on Arabic script, the tool has been successfully used for other writing systems, e.g. Latin in the ICDAR2015 Competition HTRtS on historic documents.

Contents

1 Introduction

1.1 Preprocessing for Optical Character Recognition

Offline optical character recognition (OCR) refers to the conversion of printed, typewritten, or handwritten text in a scanned image to machine-encoded text. State-of-the-art OCR systems are based on Hidden Markov Models (HMMs) which are statistical models for sequences of feature vectors. The order of the feature vectors within the sequence can represent temporal dependencies. For instance, in automatic speech recognition (ASR), the audio recording is often split into 10-15 ms chunks and a feature vector is extracted for each of these chunks (Huang et al., 2001) (Fig. 1(a)). The input for OCR, however, are images of single text lines. Analogously to ASR, we split the image into chunks with three pixel width and extract a feature vector for each of these chunks. The sequential order of the extracted feature vectors is defined by the reading order of the script (e.g. right-to-left for Arabic in Fig. 1(b)).

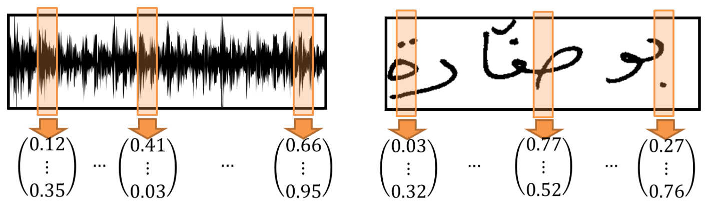
Realizing the similarities between offline OCR and ASR, we suggest to apply the state-of-the art speech recognition toolkit *Kaldi* (Povey et al., 2011) for recognizing text in scanned documents. The *PrepOCRessor* tool bridges the gap between ASR and OCR and prepares images in a way that they can be passed through to Kaldi. PrepOCRessor provides comprehensive functionality to break down the initial document image into text lines, and convert each text line to a sequence of feature vectors for training or decoding with Kaldi. Therefore, the main functions of *PrepOCRessor* can be grouped into one of the following categories:

- **Document layout analysis**: Document rotation, text/non-text segmentation, line-segmentation etc.
- **Text image normalization**: Baseline estimation, slant correction, pen size normalization, letter size normalization etc.
- **Feature extraction**: Methods for feature vector extraction from normalized text line images.

The overall goal of PrepOCRessor is to provide a comprehensive addition to Kaldi for OCR research. The combination of Kaldi and PrepOCRessor results in a fully-fledged scalable OCR framework with state-of-the-art recognition performance.

1.2 PrepOCRessor Design Philosophy

We defined the following non-functional requirements as the main design goals for PrepOCRessor.


$\begin{pmatrix}0.12\\ \vdots\\ 0.35\end{pmatrix} \cdots \begin{pmatrix}0.41\\ \vdots\\ 0.03\end{pmatrix} \cdots \begin{pmatrix}0.66\\ \vdots\\ 0.95\end{pmatrix}$
$\begin{pmatrix}0.03\\ \vdots\\ 0.32\end{pmatrix} \cdots \begin{pmatrix}0.77\\ \vdots\\ 0.52\end{pmatrix} \cdots \begin{pmatrix}0.27\\ \vdots\\ 0.76\end{pmatrix}$

(a) Speech recognition (simplified). (b) Character recognition.

Figure 1: Feature extraction in speech recognition and optical character recognition.

- **Modularity**: In agreement with the Unix philosophy we implement the concept of modularity. PrepOCRessor provides a large number of small, specialized operations which can be composed in a single Unix-like pipeline. The advantage of this approach is that PrepOCRessor is highly customizable and can be applied to a wide range of different tasks – i.e. different document types or writing systems. The disadvantage is the manual effort of composing the pipeline for the task at hand. Section 2.4 lists a number of examples which can serve as starting point for your own experiments.
- **Scalability**: PrepOCRessor supports batch processing. The documents can be distributed to any number of threads. As each document is processed separately, no inter-thread communication is necessary and we achieve nearly a linear (ideal) speed up.
- **Efficiency**: The OpenCV library (Bradski and Kaehler, 2008) provides performance-optimized code for basic computer vision and is widely used both in academia and industry. PrepOCRessor makes heavy use of algorithms and operations provided by this library.

1.3 Licence

PrepOCRessor (Copyright ©2015, QCRI a member of Qatar Foundation. All Rights Reserved) is licensed under the Apache License, Version 2.0 (the "License"); you may not use it except in compliance with the License. You may obtain a copy of the License at

`http://www.apache.org/licenses/LICENSE-2.0`.

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

A copy of the License can be found in the `LICENSE` file in the root directory.

2 Using PrepOCRessor

2.1 Installation

PrepOCRessor is implemented in Java and platform-independent. Therefore, it is possible to run it on iOS, Windows, Linux/Mac, and Android.

2.1.1 Installation on Ubuntu Linux

The following instructions explain the installation on Debian-based systems like Ubuntu but can be easily extended for other platforms. The commands in this guide should work in standard Unix shells like zsh and bash. It was tested on Ubuntu 15.04.

1. **Install the Java runtime environment**. PrepOCRessor was tested with Java 1.7 but should run with other versions as well. On Ubuntu, Java is installed by default. You can check the version number by typing `java -version` into your shell.
2. **Install the OpenCV library**. PrepOCRessor was tested with OpenCV 2.4.10 but other 2.4.x versions are likely to work. Ubuntu

provides out-of-the-box packages which can be installed with the following command:

```
sudo apt-get install libopencv2.4-java
```

If you are not using Ubuntu, you can download the latest OpenCV 2.4.x version from `http://opencv.org/downloads.html` and follow the installation instructions[1].

3. **Download PrepOCRessor**. The easiest way to get started with PrepOCRessor is to download the latest release from `bla` and unzip it wherever you like to install PrepOCRessor. Alternatively, you can compile PrepOCRessor by yourself. The repository at `https://bitbucket.org/fstahlberg/prepocressor` contains an Eclipse project including the source code itself as well as the PRiMA library[2] and Apache Commons Math3[3].

4. **Configure PrepOCRessor**. If you don't use Ubuntu Linux or you compiled OpenCV by yourself without using the Ubuntu packages, you need to tell PrepOCRessor where to find the OpenCV library. Open the `prepocressor` file in the installation root directory in your favourite text editor. You need to set the variables `OPENCV_JAR_PATH` and `OPENCV_NATIVE_LIB`. The variable `OPENCV_JAR_PATH` should point to the OpenCV .jar file. For example, in OpenCV 2.4.10 this file can be found within the OpenCV installation in `<opencv-install-dir>/bin/opencv-2410.jar`. If you don't find it, you may have compiled OpenCV without Java support. The `OPENCV_NATIVE_LIB` variable needs to contain the native library directory path (usually `<opencv-install-dir>/lib`). This directory should contain a file called `libopencv_java2410.so` or similar.

5. **Test PrepOCRessor installation**. You can start PrepOCRessor by changing into the installation directory and type the following command into the shell:

```
./prepocressor -help
```

This should output a list of global parameters together with a description for each of them. To test if the OpenCV library is installed and configured correctly, type

```
./prepocressor
```

(i.e. without arguments). The output should be similar to this:

```
13:33:14 INFO: Configuration loaded...
13:33:14 FATAL: Input file 'imageList.txt' reading
error: imageList.txt (No such file or directory)
```

→ Section 3

If you get a significantly different output, consult Section 3 for troubleshooting.

[1] For more information about the Java support of OpenCV, check `http://docs.opencv.org/doc/tutorials/introduction/desktop_java/java_dev_intro.html`
[2] Original available at `http://primaresearch.org/tools/PAGELibraries`
[3] Available at `https://commons.apache.org/proper/commons-math/`. PrepOCRessor was tested with Apache Commons Math 3.4.1

6. **Make your shell aware of PrepOCRessor**. This manual assumes that you have included PrepOCRessor in your `$PATH` environment variable so that you can start it with typing `prepocressor` into your shell. You can to this by writing the following line at the end of your `~/.bashrc`:

   ```
   export PATH=$PATH:<prepocressor-install-dir>
   ```

   Alternatively, you can create a symlink to PrepOCRessor in a directory which is already in your `$PATH` variable.

   ```
   sudo ln -s /usr/local/bin/prepocressor
           <prepocressor-install-dir>/prepocressor
   ```

2.1.2 Installation on Other Platforms

As PrepOCResoor is written in Java, the application is platform-independent and can run on a wide range of operating systems. Modifying the instructions in the previous section for other Linux distributions should be straight-forward. For Windows, however, the `prepocressor` script in the PrepOCRessor root directory needs to be adjusted to Windows syntax.

2.2 Getting Help

The `-help` parameter in PrepOCRessor displays all available global parameters together with a short description.

```
prepocressor -help
```

Detailed help texts for specific operations can be displayed by adding the names of the operations. For example, the following command shows detailed descriptions for the `log` and `tee` operations.

```
prepocressor -help log tee
```

For help with modifying and extending the code (Section 2.3.4) you can find the JavaDoc in the `javadoc/` subdirectory.

2.3 Tutorials

2.3.1 Specifying Input and Output Files

The most important parameters for specifying the in- and output of the PrepOCRessor pipeline are `-inputFile` and `-outputPath`. Following formats are supported:

- Windows bitmaps - *.bmp, *.dib
- JPEG files - *.jpeg, *.jpg, *.jpe
- JPEG 2000 files - *.jp2
- Portable Network Graphics - *.png
- Portable image format - *.pbm, *.pgm, *.ppm
- Sun rasters - *.sr, *.ras
- TIFF files - *.tiff, *.tif
- Comma-separated values - *.csv

The following command loads an image in JPEG format (`test.jpg`) and stores it without modification in PNG format.

```
prepocressor −inputFile test.jpg −outputPath test.png
```

PrepOCRessor also supports a batch mode which is particularly useful in combination with the `-nThreads` parameter. In the following example, `test.txt` is a text file containing a newline-separated list of paths to image files. PrepOCRessor converts each of these files to a grayscale image using 8 threads. Per default, the output images are stored with a time and user name encoding prefix.

```
prepocressor −inputFile test.txt
        −nThreads 8 −pipeline "grayscale"
```

In batch mode, a simple `-outputPath` like "test.png" is not useful because the results for all images in the batch are written to `test.png` – the same file gets overridden multiple times, and `test.png` just contains the processed image for the last entry in the batch. Therefore, you can use special placeholders in the `-outputPath` parameter:

- `%dir`: Name of the directory containing the input image.
- `%base`: Base name of the input image file name (without file extension).
- `%-base`: Same as `%base`, but cut after first minus hyphn ('-')
- `%idx`: Some operators split up input images into smaller pieces. The pieces are stored subsituting
- `%ext`: File extension (given by the input file).

The following command is similar to the previous example but stores the generated files in a dedicated folder called `blacknwhite`.

```
prepocressor −inputFile test.txt −pipeline "grayscale"
        −outputPath "blacknwhite/%base%ext"
```

2.3.2 Basic Image Manipulation

PrepOCRessor offers a number of basic image transformation operations which are not necessarily related to OCR or document image processing. They are a great way to get used to the pipeline architecture of PrepOCRessor.

The following command first stretches the image along the x-axis and then transposes it. This results in an image which is stretched horizontally and then flipped such that the height of the resulting image is two times the width of the original image.

```
prepocressor −inputFile test.png −outputPath out.png
        −pipeline "scale␣−xScale␣2|transpose"
```

If we switch the order of the `scale` and `transpose` operation, we produce an image which is stretched vertically and turned on its side.

```
prepocressor −inputFile test.png −outputPath out.png
        −pipeline "transpose|scale␣−xScale␣2"
```

To reproduce the same result as in the first command, we need to stretch around the y-axis after transposing the image:

```
prepocressor -inputFile test.png -outputPath out.png
        -pipeline "transpose|scale␣-yScale␣2"
```

2.3.3 Feature Extraction

→ Section 4.2.20

The `featExtract` operation (Section 4.2.20) is the main interface to the Kaldi toolkit. The input is expected to be a single channel image of a single normalized text line. The `-extractors` parameter specifies which feature extractors are used. If you specify multiple extractors, the features are stacked on each other. The `featExtract` operation accepts a number of extractor specific parameters. By convention, `-featXyz*` parameters are specific to the `xyz` extractor. Following extraction methods are implemented:

- *raw* – Raw pixel values.
- *directional* – Directional features.
- *snake* – Snake feature extraction method (also called segment-based method in (Stahberg and Vogel, 2015))
- *runlengths* – Use pixel-wise runlengths in 4 directions
- *anhdf* – ANHDF features (El-Mahallawy, 2008)
- *distribution* – Distribution features as defined by (Likforman-Sulem et. al., 2012)
- *concavity* – Concavity features as defined by (Likforman-Sulem et. al., 2012)

The following hints help you working with the `featExtract` operation.

→ Section 4.2.23

- PrepOCRessor was initially designed for Arabic script with a right-to-left reading direction. Therefore, the `featExtract` method reads the image from right to left to extract the feature vector sequences. If you deal with a left-to-right writing system (e.g. Latin), insert a `flip` operation (Section 4.2.23) prior to `featExtract`.
- The produced feature files are in .ark,t format (i.e. text) and therefore very large. You should compress them with Kaldi's `copy-feats` command:

  ```
  copy-feats ark,t:data/feats/pixel_test.ark,t
            ark,scp:data/feats/pixel_test.ark,
            data/test/feats.scp
  ```

- The feature vector dimension of many feature extractors is dependent on the image height. Therefore, it is important that all images in the pipeline have the same height. Also, keep in mind the curse of dimensionality and that a high dimensionality leads to huge feature files. In our experiments, we use image heights of 40-70 pixels.
- Set `-nThreads` to 1 when using the `featExtract` operation. This ensures that the initial order of images is preserved. Kaldi can be bitchy when it comes to the order of the entries.

→ Section 2.4

You can find some example pipelines for feature extraction in Section 2.4.

2.3.4 Modifying the Code

Instructions for accessing the PrepOCRessor repository can be found at following URL:

`https://bitbucket.org/fstahlberg/prepocressor`

This section outlines the high level design of the project. It consists of six packages:

- `qa.qcri.prepocressor.datastructures` – This package contains basic data structures like images or image lists that are used to transfer data through the pipeline.
- `qa.qcri.prepocressor.imageprocessing` – This package contains tools for image processing.
- `qa.qcri.prepocressor.io` – This package contains classes for I/O handling, i.e. loading data sets, logging, storing results.
- `qa.qcri.prepocressor.operations` – This package contains all available operations, i.e. all commands which can be used within the pipeline.
- `qa.qcri.prepocressor.operations.feat` – The feat package contains feature extractors that work together with the featExtract operation
- `qa.qcri.prepocressor.ui` – This package contains classes for the user interaction.

The main runner class is `qa.qcri.prepocressor.ui.Main`. If you want to implement a new operation, you need to inherit from the `Operation` class in the `operations` package. Take a look at the `FlipOperation` class as a basic example. Operations are required to implement at least two methods:

- `createConfiguration()` – This allows you to define the possible parameters for the operation and insert a description. The `flip` operations allows one integer parameter called `flipCode` which decides the axis along which to flip the image. Integer, String, and Float parameters are supported. The type is derived from the type of the second argument of `addParameter` (the default value).
- `processIndividual()` – This method contains the actual implementation of the operation.

Images in the pipeline are represented as `datastructures.Individual` instances. The `Individual` class stores the image itself (see `Individual.getContent()`) together with some meta information. The `processIndividual()` returns a list of individuals because operations can split up images in the pipeline into smaller parts. As `flip` does no such thing, the method returns a list with a single entry holding the input individual. This is possible because the OpenCVs flip implementation is implemented in a in-place manner.

Open the `ScaleOperation` class for a more complex example. In this operation, a new image is created (`dst`) and the result is written to that image. The returned list contains a freshly created `Individual` instance referring to `dst`. Note that Java's garbage collector does not apply to OpenCV matrices. Therefore, you need to release `Mat` instances after usage to prevent memory leaks. Of course, do not release matrices which are passed through the pipeline.

You should store new operation classes in the `operations` package. The naming convention is `XyzOperation`. You can call your operation in the PrepOCRessor pipeline with `xyz`. Please add your new operation to the list in `GlobalConfiguration` in the `ui` package to make it visible in the documentation.

2.4 Example Pipelines

2.4.1 QCRI Submission for the ICDAR2015 Competition HTRtS

This section describes the PrepOCRessor for the QCRI submission to the *ICDAR2015 Competition HTRtS: Handwritten Text Recognition on the tranScriptorium Dataset* Sanchez et al. (2015). For all but the 2ndBatch set, the following pipeline has been used to generate binarized text line images:

```
invert |
multiChannelOtsu
        -xmlPath <page-dir>/PAGE/%base%idx.xml
        -blackDiscount 0.1
        -normalizeRegionChannels |
morph
        -operation close
        -kernelSize 2
        -kernelShape ellipse |
cutWithPageXml
        -xmlPath <page-dir>/%base%idx.xml
        -extractRegions 0
        -extractTextObjects -usePageIds |
normalize
```

For the 2ndBatch, no line segmentation was given, so we applied our line segmentation algorithm based on fitting a sinus function to the vertical projection profile.

```
invert |
multiChannelOtsu
        -xmlPath <page-dir>/%base%idx.xml
        -blackDiscount 0.2
        -normalizeRegionChannels
        -maxForegroundFraction 0.1
        -extractRegions 1
        -extractTextObjects 0 |
morph
        -operation close
        -kernelSize 2
        -kernelShape ellipse |
cutWithPageXml
        -xmlPath <page-dir>/%base%idx.xml
        -extractRegions 1
        -extractTextObjects 0 |
projectionLineSegmentation
        -minLineHeight 100
        -maxLineHeight 320
        -analysisMode 0 |
vertTextSegmentation
        -minWidth 0.0005
        -minMargin 0.05
        -minSlope 0.000001
        -morph closeFirst
        -concatChildren |
normalize
```

The binarized images were then normalized using the following pipeline:

```
grayscale |
transpose |
removeUnderline
        -minRelWidth 0.2
        -minWidth 10
        -foregroundThreshold 20
        -maxVariation 5
        -maxHeight 150 |
flip -flipCode 0 |
removeUnderline
        -minRelWidth 0.2
        -minWidth 10
        -foregroundThreshold 20
        -maxVariation 5
        -maxHeight 150 |
flip -flipCode 0 |
transpose |
houghTextLine
        -resolution 60
        -noTextLineOperation bottom
        -startLambdaBandWidth 0.65
        -endLambdaBandWidth 0.75
        -startLambdaBandMin 0.1
        -endLambdaBandMin 0.0 |
textSkewCorrection
        -maxDegree 52
        -fromDegree 0
        -toDegree 50 |
removeUnderline
        -minWidth 45
        -foregroundThreshold 20
        -maxRelHeight 0.6
        -maxVariation 5 |
flip -flipCode 0 |
removeUnderline
        -minWidth 75
        -foregroundThreshold 20
        -maxRelHeight 0.5
        -maxVariation 5 |
flip -flipCode 0 |
polynomialTextLine
        -operation align
        -order 3
        -outlierFactor 1.0 |
houghTextLine
        -resolution 100
        -noTextLineOperation original
        -startLambdaBandWidth 0.65
        -endLambdaBandWidth 0.75
        -startLambdaBandMin 0.1
        -endLambdaBandMin 0.0
        -operation align
        -deleteAboveAscenders
        -deleteBelowDescenders |
normalizeText
        -belowBaseline 30
        -aboveBaseline 48
        -minCroppedAboveRatio 0.0
        -minCroppedBelowRatio 0.0 |
normalize
```

This pipeline was used for pixel-based feature extraction:

```
grayscale |
flip |
convertToFloat |
normalize -newMax 1 |
featExtract
        -winWidth 3
        -winShift 2
        -featRawCellHeight 1
        -featRawCellWidth 1
        -featRawCellShift 1
        -kaldiFile data/feats/fsushi.ark,t |
devNull
```

The segment-based features were generated with the following pipeline.

```
grayscale |
flip |
convertToFloat |
normalize -newMax 1 |
featExtract
        -extractors snake
        -winWidth 3
        -winShift 2
        -kaldiFile data/feats/fslytherin.ark,t |
devNull
```

2.4.2 Document Skew Detection Based on Hough Space Derivatives

We presented in Stahlberg and Vogel (2015a) a novel method for document skew estimation using gradients in the Hough transformed image. The `exactOrientationCorrection` operation in PrepOCRessor is an implementation of the described procedure. The following pipeline has been used in Stahlberg and Vogel (2015a):

```
grayscale |
threshold -type BINARY_INV,OTSU |
extendForHough -maxAngle 15.0 |
exactOrientationCorrection
        -maxAngle 17.0
        -resolution 60
        -eps 0.001
        -criterion sum
        -noCorrection
```

2.4.3 Detecting Dense Foreground Stripes in Arabic Handwriting for Accurate Baseline Positioning

Stahlberg and Vogel (2015b) describes our approach for detecting the baseline in Arabic handwritings. The following pipeline corresponds to the best result on the IFN/ENIT database reported in this paper:

```
grayscale |
threshold -type BINARY_INV |
removeDiacritics |
houghTextLine
        -startLambdaBandWidth 0.25
        -endLambdaBandWidth 0.45
        -startLambdaBandMin 0.2
        -endLambdaBandMin 0.4
        -combination lowest
        -resolution 130
```

For the KHATT corpus, we have to deal with curved an discontinous baselines. We use the following pipeline for the KHATT corpus. It also deals with background borders around the text.

```
grayscale |
threshold -type BINARY,OTSU |
vertTextSegmentation
        -minSlope 0.00000001
        -minMargin 0.2
        -concatChildren |
transpose |
vertTextSegmentation
        -minSlope 0.00000001
        -minMargin 0.2
        -concatChildren |
transpose |
houghTextLine |
transpose |
vertTextSegmentation
        -minSlope 0.00000001
        -minMargin 0.2
        -concatChildren |
transpose |
splitTextLines -minWidth 2.5 |
houghTextLine |
concat |
transpose |
vertTextSegmentation
        -minSlope 0.00001
        -minMargin 0.2
        -concatChildren |
transpose |
houghTextLine
```

3 Troubleshooting

This section contains some of the most common error messages and their solutions.

Exception in thread "main" java.lang.NoClassDefFoundError: org/opencv/core/Core This error usually occurs if the path to the OpenCV jar file is not set correctly. Check the `OPENCV_JAR_PATH` variable in the `prepocressor` file in the root directory of your PrepOCRessor installation. You can find detailed installation instructions in Section 2.1.

→ Section 2.1

Exception in thread "main" java.lang.UnsatisfiedLinkError: no opencv_java2410 in java.library.path This error usually occurs if the path to the OpenCV native library is not set correctly. Check the `OPENCV_NATIVE_LIB` variable in the `prepocressor` file in the root directory of your PrepOCRessor installation. The variable must point to a directory containing a file called `opencv_java24x.so` where `x` corresponds to your OpenCV version (2.4.x). You can find detailed installation instructions in Section 2.1.

→ Section 2.1

OpenCV Error: Assertion failed (src.type() == CV_XY) XY stands for a certain data type (like 8UC1, 32FC3, see the OpenCV documentation for more information). This error indicates that two consecutive operations in the pipeline do not fit together. For example, the following command usually results in such a type error.

```
prepocressor -inputFile test.jpg -pipeline "threshold"
```

The reason is that the input image usually consists of three channels, and the `threshold` operation expects a single channel image. A preceding `grayscale` operation (i.e. the pipeline `"grayscale|threshold"`) results in the expected behaviour. Useful operations for resolving this type of error are the `grayscale` operation (Section 4.2.24) and the `convertToFloat` operation (Section 4.2.9).

→ Section 4.2.24
→ Section 4.2.9

Memory consumption explodes with large batch sizes Early PrepOCRessor versions had problems with memory leaks resulting in a linear increase of required RAM in time. However, as long as the images in the batch have approximately the same size/complexity, the memory requirement should be constant after the initial start-up phase. If you have problems with memory, please report the pipeline to us and we try to fix the issue. In the meantime, it helps to split the batch input file into multiple smaller files (e.g. using the Linux `split` command) and call PrepOCRessor for each split file separately.

```
man split
```

4 Operation and Parameter Reference

4.1 Global Parameters

PREPOCRESSOR 0.2.1 is a tool for preprocessing images and feature extraction for OCR developed at the Qatar Computing Research Institute. Configuration via command line arguments: -<name> <value> Configuration via file (per line): <name> <value> Following parameters are available:

`configDumpFileName` *String, Default: <not set>* – This can be used to write a file containing all parameters of the used configuration.

`configFile` *String, Default: <not set>* – Configuration file (format: <key> <val>). Parameters in this file override values in the default configuration file. Command line arguments override all other settings. If this parameter is used within a config file, it has 'include once' semantics.

`idLength` *Integer, Default: 3* – Length of the numerical IDs that are inserted when one image in the pipeline produces multiple children. Fill up with trailing 0s if the number is shorter. Set to 0 to switch off trailing 0s.

`inputFile` *String, Default: imageList.txt* – Text file containing paths to the input images. The paths should be separated by line breaks. This parameter can also point directly to an image file if only one image is to be processed. Following input formats are supported (provided by OpenCVs imread function):

- Windows bitmaps - *.bmp, *.dib
- JPEG files - *.jpeg, *.jpg, *.jpe
- JPEG 2000 files - *.jp2
- Portable Network Graphics - *.png
- Portable image format - *.pbm, *.pgm, *.ppm
- Sun rasters - *.sr, *.ras
- TIFF files - *.tiff, *.tif
- Additionally, the CSV file format is supported by prepocressor.

`logLevel` *String, Default: INFO* – Controls the amount of output.

- FATAL: Only fatal errors,
- ERROR: All errors,
- WARN: Warnings and errors,
- INFO: Notices, warnings and errors,
- DEBUG: Debug mode

`nThreads` *Integer, Default: 1* – Number of threads.

`outputPath` *String, Default: 15-09-01.110225.felix-%base%idx%ext* – This parameter controls where the output files are stored. The file format is determined by the file extension defined by this template. For available file formats, see the -inputFile parameter. Following placeholders can be used:

- '%dir': Name of the directory containing the input image.
- '%base': Base name of the input image file name (without file extension).
- '%-base': Same as %base, but cut after first minus hyphn ('-')
- '%idx': Some operators split up input images into smaller pieces. The pieces are stored subsituting %idx with '_1', '_2'...
- '%unqatip': Assume base name of the input image file name (without file extension) is a QATIP style ID. Then the %unqatip placeholder stands for the original corpus id. Breaks if original speaker or utterance id starts with 'x'

- '%ext': File extension (given by the input file).

`pipeline` *String, Default: <not set>* – This parameter defines the operations to be executed on the input images. The syntax is similar to the linux shell pipeline: Operations are separated by '—' und parameterized with the common-<arg> <val> syntax. For details regarding the operations, try -help <operation-name>. Available operations are:

- adaptiveThreshold
- ascendersTextLine
- axisAlignedHough
- bbq
- blur
- col2graph
- componentDensity
- concat
- convertToFloat
- cutWithAltecXml
- cutWithHadaraXml
- cutWithPageXml
- devNull
- drawChildren
- drawKaldiAlignment
- drawTextLines
- exactOrientationCorrection
- extractConstantRegions
- extend
- extendForHoughsquare
- featExtract
- fillTransparency
- filter
- flip
- grayscale
- hough
- houghTextLine
- invert
- log
- morph
- multiChannelOtsu
- normalize
- normalizeText

- normalizeUpperBaseline
- orientationCorrection
- outlierRemove
- polynomialTextLine
- potatoPeelingTextLine
- printMax
- projectionLineSegmentation
- rectSum
- reduce
- reducedAlcmTransform
- renderPageXmlTranscriptions
- removeDiacritics
- removeLargeComponents
- removeSmallComponents
- removeUnderline
- removeVertTextMargin
- scale
- sobel
- splitTextLines
- subtractMean
- tee
- textSkewCorrection
- thinning
- threshold
- transpose
- vertTextSegmentation
- visualizePageXml
- writeRects

`silentOverwrite`	*Integer, Default: 1* – No files are overridden if this is set to 0.
`singlePopulation`	*Integer, Default: 0* – If this parameter is set to 1, only a single population is used for all input images. Otherwise, a population is created for each of the input files separately. If this option is set the -nThreads parameter is not used. Note that the file name of the population is set to the first input file

4.2 Operations

4.2.1 `adaptiveThreshold` Operation

The adaptiveThreshold command creates binary images. Itis based on the OpenCV function adaptiveThreshold(). Following parameters are available:

`C`	*Float, Default: 2.0* – C constant (passed through to OpenCV).
`adaptiveType`	*String, Default: MEAN_C* – Adaptive thresholding method. See OpenCVs documentation for the adaptiveThreshold function. Available values are MEAN_C or GAUSSIAN_C
`blockSize`	*Integer, Default: 12* – Block size (passed through to OpenCV).
`configDumpFileName`	*String, Default: <not set>* – This can be used to write a file containing all parameters of the used configuration.
`configFile`	*String, Default: <not set>* – Configuration file (format: <key> <val>). Parameters in this file override values in the default configuration file. Command line arguments override all other settings. If this parameter is used within a config file, it has 'include once' semantics.
`maxVal`	*Integer, Default: 255* – Lowest possible value (passed through to OpenCV).
`type`	*String, Default: BINARY* – Thresholding type. See OpenCVs documentation for the threshold function. Connect options with ','. Available options are: BINARY, BINARY_INV, TRUNC, TOZERO, TOZERO_INV, OTSU

4.2.2 `ascendersTextLine` Operation

ascendersTextLine aims to rectify baselines in Arabic script based on the height of ascenders: We fit a polynomial to the top ends of the ascenders end shift each column according the value of the estimated polynomial. Note that this operation might shift the baseline vertically, so it is recommended to apply a houghTextLine operation with low resolution afterwards. Following parameters are available:

`blurSize`	*Integer, Default: 5* – Size of blurring kernel for the projection profile.
`configDumpFileName`	*String, Default: <not set>* – This can be used to write a file containing all parameters of the used configuration.
`configFile`	*String, Default: <not set>* – Configuration file (format: <key> <val>). Parameters in this file override values in the default configuration file. Command line arguments override all other settings. If this parameter is used within a config file, it has 'include once' semantics.
`descenderArea`	*Float, Default: 0.6* – Bottom part of the image is discarded for ascender recognition. Set this to 0.5 if you've already estimated a linear baseline with houghTextLine to reduce the chance of descenders wrongly being recognized as ascenders. Set to 0 to disable this feature. Must be between 0 and 1.
`flatBorders`	*Integer, Default: 1* – If this is set to 1, the baseline left and right from all data points is set to a straight horizontal line at the height of the curve at the last data point. This usually reduces artefacts due to large slopes in the curves outside the data point range.
`minDataPoints`	*Integer, Default: 8* – Minimum number of data points.
`minScore`	*Float, Default: 0.4* – Minimal score for ascender recognition. Between 0 and 1 (the higher the more restrictive). Score is composed of the skyline value and the vertical projection profile value.
`minScoreOffset`	*Float, Default: 0.2* – Minimal score for ascender recognition. Between 0 and 1 (the higher the more restrictive). Score is composed of the skyline value and the vertical projection profile value.

`operation`	*String, Default: align* – This parameter decides what is passed through the pipeline. Available values are: • 'none': Leave the images as they are. • 'draw': Draw baseline. • 'align': Create a new image where the baseline is horizontal and at image height/2
`order`	*Integer, Default: 4* – Order of the polynom
`outlierFactor`	*Float, Default: -1.0* – Remove data points this factor times stdDev from median. Set to negative value to disable outlier detection
`skylineWeight`	*Float, Default: 0.33* – Weight of the skyline criterion. Weight of the projection profile criterion is 1 minus this value

4.2.3 `axisAlignedHough` Operation

This is a specialized and modified version of the Hough transformation. In contrast to the Hough space, rho is always on the x-axis. Rho ranges from 0 to image width. The range and resolution for theta can be specified. The advantage of this implementation is that there are no quantization errors for rho since the resolution is exactly one pixel. The disadvantage is that only lines crossing the x axis between 0 and image width are considered. The returned image contains the counts where the y coordinate represents theta. Following parameters are available:

`configDumpFileName`	*String, Default: <not set>* – This can be used to write a file containing all parameters of the used configuration.
`configFile`	*String, Default: <not set>* – Configuration file (format: <key> <val>). Parameters in this file override values in the default configuration file. Command line arguments override all other settings. If this parameter is used within a config file, it has 'include once' semantics.
`fromTheta`	*Float, Default: 45.0* – Minimum value for theta.
`thetaResolution`	*Integer, Default: 90* – Number of theta quantization steps.
`toTheta`	*Float, Default: 135.0* – Maximum value for theta.

4.2.4 `bbq` Operation

Keep only the lowest point in each column. The lowest point is detected by comparing the first channel with the threshold parameter. Following parameters are available:

`configDumpFileName`	*String, Default: <not set>* – This can be used to write a file containing all parameters of the used configuration.
`configFile`	*String, Default: <not set>* – Configuration file (format: <key> <val>). Parameters in this file override values in the default configuration file. Command line arguments override all other settings. If this parameter is used within a config file, it has 'include once' semantics.
`threshold`	*Float, Default: 0.5* – Threshold for detecting the lowest point

4.2.5 `blur` Operation

Blurs the images. Following parameters are available:

`configDumpFileName`	*String, Default: <not set>* – This can be used to write a file containing

all parameters of the used configuration.

`configFile`	*String, Default: <not set>* – Configuration file (format: <key> <val>). Parameters in this file override values in the default configuration file. Command line arguments override all other settings. If this parameter is used within a config file, it has 'include once' semantics.
`mode`	*String, Default: mean* – Blur mode. Available: mean, gaussian, median. For the median filter, -xSize is used for both dimensions.
`xSize`	*Integer, Default: 5* – Kernel size in x direction.
`ySize`	*Integer, Default: 5* – Kernel size in y direction.

4.2.6 `col2graph` Operation

Converts the first column of the image to a graph. Following parameters are available:

`configDumpFileName`	*String, Default: <not set>* – This can be used to write a file containing all parameters of the used configuration.
`configFile`	*String, Default: <not set>* – Configuration file (format: <key> <val>). Parameters in this file override values in the default configuration file. Command line arguments override all other settings. If this parameter is used within a config file, it has 'include once' semantics.
`graphWidth`	*Integer, Default: 100* – Width of the generated image.

4.2.7 `componentDensity` Operation

Calculates a map of connected component density. Each connected component adds 1/area to all pixels in its bounding box where area is the size of the bounding box. Following parameters are available:

`configDumpFileName`	*String, Default: <not set>* – This can be used to write a file containing all parameters of the used configuration.
`configFile`	*String, Default: <not set>* – Configuration file (format: <key> <val>). Parameters in this file override values in the default configuration file. Command line arguments override all other settings. If this parameter is used within a config file, it has 'include once' semantics.
`minSize`	*Integer, Default: 4* – Bounding boxes below this value are ignored.

4.2.8 `concat` Operation

Concatinate all images of one population horizontally. Following parameters are available:

`center`	*Integer, Default: 1* – Set to 0 if the images should not be centered in case of different heights
`configDumpFileName`	*String, Default: <not set>* – This can be used to write a file containing all parameters of the used configuration.
`configFile`	*String, Default: <not set>* – Configuration file (format: <key> <val>). Parameters in this file override values in the default configuration file. Command line arguments override all other settings. If this parameter is used within a config file, it has 'include once' semantics.

4.2.9 `convertToFloat` Operation

Convert Matrix to CV_32FC1. Following parameters are available:

`configDumpFileName`	*String, Default: <not set>* – This can be used to write a file containing all parameters of the used configuration.
`configFile`	*String, Default: <not set>* – Configuration file (format: <key> <val>). Parameters in this file override values in the default configuration file. Command line arguments override all other settings. If this parameter is used within a config file, it has 'include once' semantics.

4.2.10 `cutWithAltecXml` Operation

This operation corresponds to cutWithPageXml but reads xml files in the format used by the ALTEC corpus. This format specifies line tags on the first level and word tags on the second level. See http://ALTEC-Center.org/xsd/ocr-annotation-1-0.xsd for a specification. Following parameters are available:

`configDumpFileName`	*String, Default: <not set>* – This can be used to write a file containing all parameters of the used configuration.
`configFile`	*String, Default: <not set>* – Configuration file (format: <key> <val>). Parameters in this file override values in the default configuration file. Command line arguments override all other settings. If this parameter is used within a config file, it has 'include once' semantics.
`cutLevel`	*String, Default: line* – Splitting level. Either 'line' or 'word'
`useIndexAttributes`	*Integer, Default: 0* – Set the prepocressor index to the value of the index attribute of the corresponding node in the xml file.Note that this can lead to problems in combination with cutLevel=word because the same word level index might be used multiple times in a single xml file.
`xmlPath`	*String, Default: %base%idx.xml* – Path to the xml files in ALTEC format. The same placeholders as in the global outputPath can be used.

4.2.11 `cutWithPageXml` Operation

Cut a page image using an XML file in PAGE format.Note: The used PAGE library may break with multiple threads! Following parameters are available:

`border`	*Integer, Default: 0* – If -border equals 0 we cut the text regions accurately. Use a value greater than zero if the extracted regions are used for human inspections. It will add a padding to the image with decreased brightness and draw a red rectangle around the region.
`configDumpFileName`	*String, Default: <not set>* – This can be used to write a file containing all parameters of the used configuration.
`configFile`	*String, Default: <not set>* – Configuration file (format: <key> <val>). Parameters in this file override values in the default configuration file. Command line arguments override all other settings. If this parameter is used within a config file, it has 'include once' semantics.
`extractRegions`	*Integer, Default: 1* – Extract regions.
`extractTextObjects`	*Integer, Default: 0* – Extract text lines.
`minLevel`	*Integer, Default: 0* – Minimum level in layout of region to be extracted.

`usePageIds` *Integer, Default: 0* – Use id attributes in xml file for naming. Otherwise, use consecutive numbering (see global idLength parameter)

`xmlPath` *String, Default: %base%idx.xml* – Path to the xml files in PAGE format. The same placeholders as in the global outputPath can be used.

4.2.12 `devNull` Operation

Deletes all images in the pipeline. Equivalent to '> /dev/null' in the unix shell. Following parameters are available:

`configDumpFileName` *String, Default: <not set>* – This can be used to write a file containing all parameters of the used configuration.

`configFile` *String, Default: <not set>* – Configuration file (format: <key> <val>). Parameters in this file override values in the default configuration file. Command line arguments override all other settings. If this parameter is used within a config file, it has 'include once' semantics.

4.2.13 `drawChildren` Operation

Reloads the input image of the given population and draws all children in the population with rectangles. Following parameters are available:

`configDumpFileName` *String, Default: <not set>* – This can be used to write a file containing all parameters of the used configuration.

`configFile` *String, Default: <not set>* – Configuration file (format: <key> <val>). Parameters in this file override values in the default configuration file. Command line arguments override all other settings. If this parameter is used within a config file, it has 'include once' semantics.

`thickness` *Integer, Default: 2* – Thickness of the rectangle. Negative for filled rectangles.

`transpose` *Integer, Default: 0* – Transpose original.

4.2.14 `drawKaldiAlignment` Operation

Reads a Kaldi alignment file and draws the forced alignment into the image. Assumes left-to-right topology for nonsilence phones and whatever for silence. Following parameters are available:

`alignmentFile` *String, Default: kaldi.al* – Path to the Kaldi alignment file in text format

`borderHeight` *Integer, Default: 30* – Height of the border for annotations.

`configDumpFileName` *String, Default: <not set>* – This can be used to write a file containing all parameters of the used configuration.

`configFile` *String, Default: <not set>* – Configuration file (format: <key> <val>). Parameters in this file override values in the default configuration file. Command line arguments override all other settings. If this parameter is used within a config file, it has 'include once' semantics.

`kaldiId` *String, Default: %base* – This string specifies how the kaldi ID is generated. You can use the same placeholders as in outputPath.

`offset` *Integer, Default: 0* – Offset from the right image border.

4.2.15 `drawTextLines` Operation

Reloads the original input images and draws text lines in it for visual verification. The text lines must be in the pipeline, e.g. produced by the houghTextLine operation. Following parameters are available:

`configDumpFileName`	*String, Default: <not set>* – This can be used to write a file containing all parameters of the used configuration.
`configFile`	*String, Default: <not set>* – Configuration file (format: <key> <val>). Parameters in this file override values in the default configuration file. Command line arguments override all other settings. If this parameter is used within a config file, it has 'include once' semantics.
`drawSkewLine`	*Integer, Default: 1* – Draw the skew line. Set to 0 to ignore the text skew information

4.2.16 `exactOrientationCorrection` Operation

Brings rotated text documents in an upright position. This is done by finding the maximum squared variance angle in the Hough transformed image. An iterative algorithm is applied to increase the accuracy of the skew angle estimation. Following parameters are available:

`configDumpFileName`	*String, Default: <not set>* – This can be used to write a file containing all parameters of the used configuration.
`configFile`	*String, Default: <not set>* – Configuration file (format: <key> <val>). Parameters in this file override values in the default configuration file. Command line arguments override all other settings. If this parameter is used within a config file, it has 'include once' semantics.
`criterion`	*String, Default: horiz* – Maximization criterion. Available values: • 'horiz': Horizontal estimation • 'vert': Vertical estimation • 'sum': Sum of horizontal and vertical estimation
`eps`	*Float, Default: 0.1* – Accuracy in degree.
`horizWeight`	*Float, Default: 0.5* – If the sum criterion is used, the horizontal profile is weighted with horizWeight and the vertical profile is weighted with (1-horizWeight)
`houghLineMode`	*String, Default: scaling* – Method for line definition in Hough space. Available values: • 'scaling': Gradually increase scaling factor of line definition • 'bresenham': Use Bresenhams line drawing algorithm • 'exact': Take fractional counts for pixels into account
`maxAngle`	*Float, Default: 45.0* – Maximum skew angle.
`noCorrection`	*Integer, Default: 0* – Pass thru image without modification.
`refine`	*Integer, Default: 0* – Set to 1 to enable refinement. If this parameter is set an additional search in degree +- 0.5 with resolution 100 is added assuring that the tested values are multipliers of 0.01
`reloadOriginal`	*Integer, Default: 0* – Reload the original image and rotate it. Otherwise use image in the pipeline.

`resolution` *Integer, Default: 90* – Resolution of the Hough transform.

`sobelKSize` *Integer, Default: 3* – Size of the sobel kernel.

4.2.17 `extractConstantRegions` Operation

This operations assumes that the images in the pipeline are frames of a video. Note that the fps video filter in ffmpeg can be used to extract an image every X seconds of the video. The operation searches for regions which do not change during a certain time period. In news videos, these regions usually correspond to overlays embedded in the video displaying additional information. In the documentation for this operation we refer a single image in the pipeline as 'frame'. Following parameters are available:

`closeSize` *Integer, Default: 30* – Kernel size of closing operation

`configDumpFileName` *String, Default: <not set>* – This can be used to write a file containing all parameters of the used configuration.

`configFile` *String, Default: <not set>* – Configuration file (format: <key> <val>). Parameters in this file override values in the default configuration file. Command line arguments override all other settings. If this parameter is used within a config file, it has 'include once' semantics.

`minLength` *Integer, Default: 5* – Minimal time period for a region in number of frames

`openSize` *Integer, Default: 150* – Kernel size of open operation

`threshold` *Float, Default: 20.0* – Maximal sum of differences in channels for a pixel to be considered as constant.

4.2.18 `extend` Operation

Extend image canvas. Following parameters are available:

`bottom` *Integer, Default: 20* – Extend image at bottom border (in pixels).

`configDumpFileName` *String, Default: <not set>* – This can be used to write a file containing all parameters of the used configuration.

`configFile` *String, Default: <not set>* – Configuration file (format: <key> <val>). Parameters in this file override values in the default configuration file. Command line arguments override all other settings. If this parameter is used within a config file, it has 'include once' semantics.

`left` *Integer, Default: 20* – Extend image at left border (in pixels).

`right` *Integer, Default: 20* – Extend image at right border (in pixels).

`top` *Integer, Default: 20* – Extend image at top border (in pixels).

4.2.19 `extendForHoughsquare` Operation

11:02:33 FATAL: Operation 'extendForHoughsquare' is not implemented

4.2.20 `featExtract` Operation

Feature extraction for Kaldi. The feat* parameters are extractor specific. NOTE: Feature extraction is based on a sliding window in right-to-left direction as this tool was initially developed for Arabic. If you wish to change direction, apply the flip operation first. Following parameters are available:

`baselineHeight` *Integer, Default: 32* – Height of the baseline for baseline dependent features.

`centerFrames` *Integer, Default: 0* – Set this to 1 to enable vertically repositioned windows based on the center of gravity. This can help to better handle vertically translated letters. See (Doetsch et. al., 2012) for more information. Additionally, the vertical shift of each window is added as feature to the feature vectors if centerFrames=1.

`configDumpFileName` *String, Default: <not set>* – This can be used to write a file containing all parameters of the used configuration.

`configFile` *String, Default: <not set>* – Configuration file (format: <key> <val>). Parameters in this file override values in the default configuration file. Command line arguments override all other settings. If this parameter is used within a config file, it has 'include once' semantics.

`delayDelta` *Integer, Default: 1* – Set to positive value to add deltas of feature vectors according -delays

`delayRaw` *Integer, Default: 0* – Set to positive value to add raw feature vectors according -delays (feature staking)

`delays` *String, Default: <not set>* – Comma-separated list of integers specifying the deltas to add. The integers are the delta distances, i.e. '1' stands for standard deltas, '2' calculates deltas to second last feature vector.

`extractors` *String, Default: raw* – Comma separated list of feature extractors. Available:

- 'raw': Raw pixel values.
- 'directional': Directional features.
- 'snake': Snake feature extraction method (also called segment-based method in (Stahlberg and Vogel, 2015))
- 'runlengths': Use pixel-wise runlengths in 4 directions
- 'anhdf': ANHDF features (El-Mahallawy, 2008)
- 'distribution': Distribution features as defined by (Likforman-Sulem et. al., 2012)
- 'concavity': Concavity features as defined by (Likforman-Sulem et. al., 2012)

`featAnhdfConnecticityTolerance` *Integer, Default: 4* – Tolerance for segments in the ANHDF feature to be connected. See (El-Mahallawy, 2008) PhD thesis for more information.

`featAnhdfReductionMode` *String, Default: max* – ANHDF features are defined for windows with 1 pixel width. Wider windows a reduced according to this method:

- 'max': Take the maximum of each row.
- 'min': Take the minimum of each row.
- 'average': Take the average of each row.
- 'firstAndLast': Use right most column for slice i and left most column of previous slice as i-1.

`featAnhdfSegmentNum`	*Integer, Default: 4* – Number of segments in ANHDF features. 4 is also used by (El-Mahallawy, 2008) and reasonable for Arabic.
`featConcavityBaselineDependence`	*Integer, Default: 1* – Extract also concavity separately for above and below baseline.
`featDirectionalRadius`	*Integer, Default: 10* – Radius for directional feature extractor (maximum feature value)
`featRawCellHeight`	*Integer, Default: 1* – Height of the cell for the raw feature extractor.
`featRawCellShift`	*Integer, Default: 1* – Vertical cell shift for the raw feature extractor.
`featRawCellWidth`	*Integer, Default: 1* – Height of the cell for the raw feature extractor.
`featRunlengthsNonNegative`	*Integer, Default: 1* – Set to positive value to avoid using negative values for background pixel runlengths (use 0 instead)
`featRunlengthsRadius`	*Integer, Default: 10* – Radius for runlength feature extractor (maximum feature value)
`featSnakeAddCenterDistances`	*Integer, Default: 0* – Add distance between consecutive segment centers as features.
`featSnakeAddRelativeFeats`	*Integer, Default: 0* – Add snake features divided by height of entire slice
`featSnakeBackground`	*Integer, Default: 0* – Set to positive value to use background snakes.
`featSnakeDefaultHeight`	*Integer, Default: 0* – Default value for height features which is used in silence.
`featSnakeForeground`	*Integer, Default: 1* – Set to positive value to use foreground snakes.
`featSnakeNumber`	*Integer, Default: 6* – Number of snakes.
`foregroundThreshold`	*Float, Default: 0.5* – Pixel values above this threshold are considered as foreground.
`kaldiFile`	*String, Default: kaldi.ark,t* – Path to the kaldi feature file to generate. This is a feature table in text format. See kaldis copy-feat tool with ark,t specifiers for more information.
`kaldiId`	*String, Default: %base* – This string specifies how the kaldi ID is generated. You can use the same placeholders as in outputPath.
`winShift`	*Integer, Default: 2* – Shift of the sliding window.
`winWidth`	*Integer, Default: 3* – Width of the sliding window.

4.2.21 `fillTransparency` Operation

Loads the original image, fetches the alpha channel and fills transparent areas in the current image in the pipeline with zero. Following parameters are available:

`configDumpFileName`	*String, Default: <not set>* – This can be used to write a file containing all parameters of the used configuration.
`configFile`	*String, Default: <not set>* – Configuration file (format: <key> <val>). Parameters in this file override values in the default configuration file. Command line arguments override all other settings. If this parameter is used within a config file, it has 'include once' semantics.

4.2.22 `filter` Operation

Removes images that are likely to be no text lines. Following parameters are available:

`configDumpFileName`	*String, Default: <not set>* – This can be used to write a file containing all parameters of the used configuration.
`configFile`	*String, Default: <not set>* – Configuration file (format: <key> <val>). Parameters in this file override values in the default configuration file. Command line arguments override all other settings. If this parameter is used within a config file, it has 'include once' semantics.
`minAspectRatio`	*Float, Default: 2.0* – Images with width/height<minAspectRatio are removed.
`minHeight`	*Integer, Default: 10* – Images smaller height (in pixel) are removed.

4.2.23 `flip` Operation

Flips the image around x or y axis. Following parameters are available:

`configDumpFileName`	*String, Default: <not set>* – This can be used to write a file containing all parameters of the used configuration.
`configFile`	*String, Default: <not set>* – Configuration file (format: <key> <val>). Parameters in this file override values in the default configuration file. Command line arguments override all other settings. If this parameter is used within a config file, it has 'include once' semantics.
`flipCode`	*Integer, Default: 1* – Except from OpenCV docu: Specifies how to flip the array: 0 means flipping around the x-axis, positive (e.g., 1) means flipping around y-axis, and negative (e.g., -1) means flipping around both axes.

4.2.24 `grayscale` Operation

Converts the images in the pipeline to grayscale. Following parameters are available:

`configDumpFileName`	*String, Default: <not set>* – This can be used to write a file containing all parameters of the used configuration.
`configFile`	*String, Default: <not set>* – Configuration file (format: <key> <val>). Parameters in this file override values in the default configuration file. Command line arguments override all other settings. If this parameter is used within a config file, it has 'include once' semantics.

4.2.25 `hough` Operation

The hough command performs a Hough transformation on the input images. This is the original OpenCV implementation. Consider using axisAlignedHough if this is not the last operation in the pipeline or you want to obtain a meaningful image. This operation is useful for directly redirecting the output to a CSV file. Following parameters are available:

`angleResolution`	*Integer, Default: 360* – Angle resolution (number of different values for theta)
`angleSamplingFactor`	*Integer, Default: 200* – HoughLines is called with resolution 1/angleSamplingFactor. A higher value reduces the noise in the Hough transform, but needs longer execution time.

`configDumpFileName` *String, Default: <not set>* – This can be used to write a file containing all parameters of the used configuration.

`configFile` *String, Default: <not set>* – Configuration file (format: <key> <val>). Parameters in this file override values in the default configuration file. Command line arguments override all other settings. If this parameter is used within a config file, it has 'include once' semantics.

`toMatrix` *Integer, Default: 0* – Transform the hough transformation to a matrix. The rho values are shifted so that the minimum value corresponds to x=0, and x=width/2 to rho=0. The theta values are scaled by angleResolution. If toMatrix is not set, the operation passes an array of 3 dimensional vectors storing [rho, theta, voteCount] ordered descending by voteCount.

4.2.26 `houghTextLine` Operation

Calculates the base line from a Hough transformed image. If -criterion=max, the base line is detected at the maximum in the Hough space. Otherwise, we find a rotated rectangle which includes at least minBetweenBaseline white pixels and optimizes the target function (-criterion parameter) and meets the -maxValArea restriction. The base line is detected at the bottom of the rectangle. See (Stahlberg and Vogel, 2015) for a detailed discussion. Following parameters are available:

`blurMode` *String, Default: none* – Blur hough space. Available values are: 'none', 'median', 'mean'.

`blurRho` *Integer, Default: 3* – Kernel size of blur operation in rho direction

`blurTheta` *Integer, Default: 1* – Kernel size of blur operation in theta direction

`borderFactor` *Float, Default: 0.0* – Before Hough transform, the image is extended on top and bottom by borderFactor*height

`combination` *String, Default: lowest* – How the different baselines should be combined. Available: 'none': Do not combine baselines - Pass them separately 'lowest': Select the baseline with the lowest slope.

`configDumpFileName` *String, Default: <not set>* – This can be used to write a file containing all parameters of the used configuration.

`configFile` *String, Default: <not set>* – Configuration file (format: <key> <val>). Parameters in this file override values in the default configuration file. Command line arguments override all other settings. If this parameter is used within a config file, it has 'include once' semantics.

`deleteAboveAscenders` *Integer, Default: 0* – Set to 1 to remove pixels above the highest ascender. An ascender is a connected component which crosses the upper baseline. Ignored if the operation parameter is not set to 'align'

`deleteBelowDescenders` *Integer, Default: 0* – Set to 1 to remove pixels below the lowest descender. A descender is a connected component which crosses the lower baseline. Ignored if the operation parameter is not set to 'align'

`deltaLambda` *Float, Default: 0.05* – Lambda increment. Set to 0 to use only startLambda*.

`endLambdaBandMin` *Float, Default: 0.4* – Required fraction of white pixels between both base lines for the bandMin criterion.

`endLambdaBandWidth` *Float, Default: 0.4* – Required fraction of white pixels between both base lines for the bandWidth criterion.

`houghMax`	*Integer, Default: 0* – Set to 1 to include the hough space maximum.
`maxDegree`	*Float, Default: 20.0* – Maximum text rotation in degree. If this rotation is exceeded, the image is discarded. Note: We only check the range betweeen +-45 degree.
`maxValArea`	*Float, Default: 0.5* – The maximum within the band must be in the lower part of the band area. This restriction addresses the common assumption that the baseline is represented by a maximum in the Hough space. Set to negative value todisable this check. This is not used if criterion=max
`noTextLineOperation`	*String, Default: original* – This parameter decides what is passed through the pipeline if no text line within range has been detected: • 'original': Pass through original image. • 'bottom': Place baseline at bottom border of image
`operation`	*String, Default: align* – This parameter decides what is passed through the pipeline. Available values are: • 'none': Leave the images as they are. • 'draw': Draw upper and lower baselines. • 'align': Create a new image where the baseline is horizontal and at image height/2
`resolution`	*Integer, Default: 130* – Number of steps between -45 and +45 degree in Hough transform.
`startLambdaBandMin`	*Float, Default: 0.2* – Required fraction of white pixels between both base lines for the bandMin criterion.
`startLambdaBandWidth`	*Float, Default: 0.2* – Required fraction of white pixels between both base lines for the bandWidth criterion.
`truPath`	*String, Default: <not set>* – If .tru files are available (as for the IFN/ENIT database) the baseline error can be computed automatically. Use placeholders as in the global -outputPath parameter. The evaluation is written to stdout. Format: EvalBaseline <inputFileName> <refStartY> <refSlope> <hypoStartY> <hypoSlope> <Error> <StringError>
`useTruIfAvailable`	*Integer, Default: 1* – If this is set to 1, we take the reference baseline from the tru file if exists.

4.2.27 `invert` Operation

Invert the image. Following parameters are available:

`configDumpFileName`	*String, Default: <not set>* – This can be used to write a file containing all parameters of the used configuration.
`configFile`	*String, Default: <not set>* – Configuration file (format: <key> <val>). Parameters in this file override values in the default configuration file. Command line arguments override all other settings. If this parameter is used within a config file, it has 'include once' semantics.

4.2.28 `log` Operation

Element wise logarithm. Following parameters are available:

`configDumpFileName`	*String, Default: <not set>* – This can be used to write a file containing all parameters of the used configuration.

`configFile`	*String, Default: <not set>* – Configuration file (format: <key> <val>). Parameters in this file override values in the default configuration file. Command line arguments override all other settings. If this parameter is used within a config file, it has 'include once' semantics.

4.2.29 `morph` Operation

Performs morphological operations. Following parameters are available:

`configDumpFileName`	*String, Default: <not set>* – This can be used to write a file containing all parameters of the used configuration.
`configFile`	*String, Default: <not set>* – Configuration file (format: <key> <val>). Parameters in this file override values in the default configuration file. Command line arguments override all other settings. If this parameter is used within a config file, it has 'include once' semantics.
`kernelShape`	*String, Default: rect* – Kernel shape. 'rect' or 'ellipse'
`kernelSize`	*Integer, Default: 5* – Kernel size
`operation`	*String, Default: close* – One of the following morphology operations: close, open, erode, dilate

4.2.30 `multiChannelOtsu` Operation

This is Otsu thrsholding adapted for multichannel images. It uses greyscale standard otsu binarization for initial labeling, and then applies the k-means algorithm (k=2) for final binarization. Note: Channels > 3 (e.g. alpha channel) are not considered. Following parameters are available:

`blackDiscount`	*Float, Default: 0.5* – Increase this to make more pixels classified as white. Between 0 and 1
`configDumpFileName`	*String, Default: <not set>* – This can be used to write a file containing all parameters of the used configuration.
`configFile`	*String, Default: <not set>* – Configuration file (format: <key> <val>). Parameters in this file override values in the default configuration file. Command line arguments override all other settings. If this parameter is used within a config file, it has 'include once' semantics.
`extractRegions`	*Integer, Default: 0* – Extract regions.
`extractTextObjects`	*Integer, Default: 1* – Extract text lines.
`maxForegroundFraction`	*Float, Default: 0.2* – Maximum fractionof foreground pixels. If exceeded, increase blackDiscount parameter to produce more background
`maxIter`	*Integer, Default: 10* – Number of k-means iterations
`minLevel`	*Integer, Default: 0* – Minimum level in layout of region to be extracted.
`normalizeRegionChannels`	*Integer, Default: 0* – Normalize channels in top level regions before binarization.
`xmlPath`	*String, Default: <not set>* – Path to the xml files in PAGE format. The same placeholders as in the global outputPath can be used. If this-parameter is provided, binarization is done for each region separately. Otherwise, the algorithm is applied to the whole image. Note: The used PAGE library may break when using multiple threads

4.2.31 `normalize` Operation

The normalize operation rescales the values in the matrices to the given interval. The current value range is fetched from the first channel only. Following parameters are available:

`configDumpFileName`	*String, Default: <not set>* – This can be used to write a file containing all parameters of the used configuration.
`configFile`	*String, Default: <not set>* – Configuration file (format: <key> <val>). Parameters in this file override values in the default configuration file. Command line arguments override all other settings. If this parameter is used within a config file, it has 'include once' semantics.
`newMax`	*Float, Default: 255.0* – Maximum value of the new interval.
`newMin`	*Float, Default: 0.0* – Minimum value of the new interval.

4.2.32 `normalizeText` Operation

Assumes that images contain text lines with horizontal baseline at image center. Scales images suchthat the baseline is repositioned as defined by -below/aboveBaseline and scaled such that the first/last row is the nearest row to the baseline which sums up to less than -maxCut pixels. The resulting images will have the height belowBaseline+aboveBaseline. Following parameters are available:

`aboveBaseline`	*Integer, Default: 32* – Vertical distance in the output image from baseline to upper border.
`belowBaseline`	*Integer, Default: 16* – Vertical distance in the output image from baseline to lower border.
`configDumpFileName`	*String, Default: <not set>* – This can be used to write a file containing all parameters of the used configuration.
`configFile`	*String, Default: <not set>* – Configuration file (format: <key> <val>). Parameters in this file override values in the default configuration file. Command line arguments override all other settings. If this parameter is used within a config file, it has 'include once' semantics.
`maxBelowShrink`	*Float, Default: 3.0* – If scale below and above baseline differ by this factor (relative) use above bl scale for below bl.
`maxBelowStretch`	*Float, Default: 1.5* – If scale below and above baseline differ by this factor (relative) use above bl scale for below bl.
`maxCut`	*Float, Default: 1.0* – Maximum sum at cropped border
`minCroppedAboveAbsolute`	*Integer, Default: 20* – Distance of cropped border above baseline.
`minCroppedAboveRatio`	*Float, Default: 0.1* – Distance of cropped border above baseline (relative to image border.
`minCroppedBelowRatio`	*Float, Default: 0.02* – Distance of cropped border above baseline (relative to image border.

4.2.33 `normalizeUpperBaseline` Operation

This operation normalizes the position of the upper baseline. The input image should be an aligned image with lower baseline in image center (see houghTextLine operation) The upper baseline is estimated at the maximum in the derivative of the horizontal projection profile above the lower baseline. The image is modified s.t. the upper baseline is at a predefined height. Note: If you apply the normalizeText operation after this, the maxCut, minCroppedAboveRatio, and minCroppedAboveAbsoluteparameters should be equal. Following parameters are available:

`configDumpFileName`	*String, Default: <not set>* – This can be used to write a file containing all parameters of the used configuration.
`configFile`	*String, Default: <not set>* – Configuration file (format: <key> <val>). Parameters in this file override values in the default configuration file. Command line arguments override all other settings. If this parameter is used within a config file, it has 'include once' semantics.
`keepCoreZoneAspectRatio`	*Integer, Default: 0* – Set to 1 to keep the aspect ratio in the core zone between upper and lower baseline. Otherwise, the core zone is stretched/shrinked in order to reposition the upper baseline. Only applicable if operation=align
`maxCut`	*Float, Default: 1.0* – Maximum sum at cropped border
`maxStretchFactor`	*Float, Default: 4.0* – Works with keepCoreZoneAspectRatio=1. Maximum horizontal stretching factor
`minCroppedAboveAbsolute`	*Integer, Default: 20* – Distance of cropped border above baseline.
`minCroppedAboveRatio`	*Float, Default: 0.5* – Distance of cropped border above baseline (relative to image border.
`minStretchFactor`	*Float, Default: 0.25* – Works with keepCoreZoneAspectRatio=1. Minimum horizontal stretching factor
`newUpperBaseline`	*Float, Default: 0.4* – New ratio between distance between baselines and highest ascender - lower baseline distance. This isonly applicable in combination with -operation=align
`operation`	*String, Default: align* – What should be done after the upper baseline is found 'align': Reposition the baseline to a predefined height 'draw': Draw a line indicating the upper baseline
`upperBaselineHighest`	*Float, Default: 0.8* – Highest possible ratio between distance between baselines and highest ascender - lower baseline distance
`upperBaselineLowest`	*Float, Default: 0.2* – Lowest possible ratio between distance between baselines and highest ascender - lower baseline distance

4.2.34 `orientationCorrection` Operation

Brings rotated text documents in an upright position. This is done by finding the maximum squared variance angle in the Hough transformed image. Following parameters are available:

`configDumpFileName`	*String, Default: <not set>* – This can be used to write a file containing all parameters of the used configuration.
`configFile`	*String, Default: <not set>* – Configuration file (format: <key> <val>). Parameters in this file override values in the default configuration file. Command line arguments override all other settings. If this parameter is used within a config file, it has 'include once' semantics.
`maxAngle`	*Float, Default: 45.0* – Maximum skew angle.
`noCorrection`	*Integer, Default: 0* – Pass thru image without modification.
`reloadOriginal`	*Integer, Default: 0* – Reload the original image and rotate it. Otherwise

use image in the pipeline.

`resolution`	*Integer, Default: 90* – Resolution of the Hough transform.
`sobelKSize`	*Integer, Default: 3* – Size of the sobel kernel.

4.2.35 `outlierRemove` Operation

Removes outlier. Outlier are identified by differing by -tolerance times standard derivation from the mean. Input needs to be 1 channel float. Following parameters are available:

`configDumpFileName`	*String, Default: <not set>* – This can be used to write a file containing all parameters of the used configuration.
`configFile`	*String, Default: <not set>* – Configuration file (format: <key> <val>). Parameters in this file override values in the default configuration file. Command line arguments override all other settings. If this parameter is used within a config file, it has 'include once' semantics.
`tolerance`	*Float, Default: 3.0* – Tolerance parameter for outlier detection.

4.2.36 `polynomialTextLine` Operation

Fits a polynom to the data points in order to guess the text line. Following parameters are available:

`configDumpFileName`	*String, Default: <not set>* – This can be used to write a file containing all parameters of the used configuration.
`configFile`	*String, Default: <not set>* – Configuration file (format: <key> <val>). Parameters in this file override values in the default configuration file. Command line arguments override all other settings. If this parameter is used within a config file, it has 'include once' semantics.
`dataPoints`	*String, Default: minima* – Strategy for data point retrival. 'bbq': Use all bottom foregound points (see bbq op) 'minima': Use only minima of bottom foreground points
`flatBorders`	*Integer, Default: 1* – If this is set to 1, the baseline left and right from all data points is set to a straight horizontal line at the height of the curve at the last data point. This usually reduces artefacts due to large slopes in the curves outside the data point range.
`minDataPoints`	*Integer, Default: 2* – Minimum number of data points.
`minHeight`	*Integer, Default: 1* – Minimum distance to top border for a polynom in order to be used for estimating the polynomial
`operation`	*String, Default: align* – This parameter decides what is passed through the pipeline. Available values are: • 'none': Leave the images as they are. • 'draw': Draw upper and lower baselines. • 'align': Create a new image where the baseline is horizontal and at image height/2
`order`	*Integer, Default: 4* – Order of the polynom
`outlierFactor`	*Float, Default: -1.0* – Remove data points this factor times stdDev from median. Set to negative value to disable outlier detection
`threshold`	*Float, Default: 0.5* – Threshold for detecting the lowest point

4.2.37 `printMax` Operation

Print information about the maximum. Following parameters are available:

`configDumpFileName`	*String, Default: <not set>* – This can be used to write a file containing all parameters of the used configuration.
`configFile`	*String, Default: <not set>* – Configuration file (format: <key> <val>). Parameters in this file override values in the default configuration file. Command line arguments override all other settings. If this parameter is used within a config file, it has 'include once' semantics.

4.2.38 `projectionLineSegmentation` Operation

Line segmentation using vertical projection. This operation first fits a sinus function to the profile. The frequency of the best fit is then used to determine the kernel size for a blur operation on the projection profile.Lines are extracted from valley to valley in the smoothed profile. Following parameters are available:

`analysisMode`	*Integer, Default: 0* – Set to 1 to output an image explaining the line segmentation by showing the smoothed vertical projection over the image plus the found boundaries
`configDumpFileName`	*String, Default: <not set>* – This can be used to write a file containing all parameters of the used configuration.
`configFile`	*String, Default: <not set>* – Configuration file (format: <key> <val>). Parameters in this file override values in the default configuration file. Command line arguments override all other settings. If this parameter is used within a config file, it has 'include once' semantics.
`maxLineHeight`	*Integer, Default: 130* – Maximum line height in pixel.
`maxLineHeightVariance`	*Float, Default: 2.0* – If a line is taller than this parameter times the estimated average line height -> outlier.
`maxProjectionRatio`	*Float, Default: 0.8* – If the minimum next to a maximum is larger than this parameter times the maximum in the projection, ignore this maximum.
`minLineCount`	*Integer, Default: 10000* – Minimal line count in one segment. This can be used for outlier detection. Set to a high value to disable this feature.
`minLineHeight`	*Integer, Default: 10* – Minimal line height in pixel.
`sinusExp`	*Integer, Default: 1* – Sinus exponent used for line height estimation. Should be uneven.

4.2.39 `rectSum` Operation

This operation calculates the sum of elements within rectangles in the images. The rectangles have a common edge point (fix point) but their width and height vary. Produces a maxWidth times maxHeight matrix storing the sum within corresponding rectangles. Following parameters are available:

`configDumpFileName`	*String, Default: <not set>* – This can be used to write a file containing all parameters of the used configuration.
`configFile`	*String, Default: <not set>* – Configuration file (format: <key> <val>). Parameters in this file override values in the default configuration file.

Command line arguments override all other settings. If this parameter is used within a config file, it has 'include once' semantics.

`fixX`	*Integer, Default: 0* – x coordinate of fix point
`fixY`	*Integer, Default: 0* – y coordinate of fix point
`maxHeight`	*Integer, Default: 0* – Largest rectangle height. Can also be negative. If it is set to 0, use imageHeight-fixY
`maxWidth`	*Integer, Default: 0* – Largest rectangle width. Can also be negative. If it is set to 0, use imageWidth-fixX

4.2.40 `reduce` Operation

Calculates the projections of the images in the pipeline. The images are reduced to a single column or row (see dim parameter). Following parameters are available:

`configDumpFileName`	*String, Default: <not set>* – This can be used to write a file containing all parameters of the used configuration.
`configFile`	*String, Default: <not set>* – Configuration file (format: <key> <val>). Parameters in this file override values in the default configuration file. Command line arguments override all other settings. If this parameter is used within a config file, it has 'include once' semantics.
`dim`	*Integer, Default: 1* – Dimension along which the reduction is done. E.g. 1 reduces the image to a single column.
`mode`	*String, Default: sum* – Accumulation mode. Available modes: sum, avg, max, min, sqrSum

4.2.41 `reducedAlcmTransform` Operation

Applies a steerable ellipsoid filter to create an adaptive local connectivity map. The ALCM of each direction is reduced horizontally to a single col. The i-th col of the resulting image corresponds to the direction i. i encodes the angle 180*i/resolution. If resolution=2, i=0 is the horizontal, i=1 the vertical ALCM. The resulting image has the width -resolution Following parameters are available:

`configDumpFileName`	*String, Default: <not set>* – This can be used to write a file containing all parameters of the used configuration.
`configFile`	*String, Default: <not set>* – Configuration file (format: <key> <val>). Parameters in this file override values in the default configuration file. Command line arguments override all other settings. If this parameter is used within a config file, it has 'include once' semantics.
`kHeight`	*Integer, Default: 6* – Height of the ellipse kernel.
`kWidth`	*Integer, Default: 30* – Width of the ellipse kernel.
`resolution`	*Integer, Default: 2* – See operation description.

4.2.42 `renderPageXmlTranscriptions` Operation

This operation scans a page xml file for text regions. The text regions are written to the document images in the pipeline by inserting solid rectangles with the text of the xml files in it. For example, this can be used to generate a translated version of the document image after the text has been ocred and translated. Following parameters are available:

`align`	*String, Default: center* – Text alignment, 'right', 'left', or 'center'
`bgColorEstimateBorder`	*Integer, Default: 2* – Controls the way the background color for text areas is estimated. The color is the average of the pixel colors at the border of the text area. This is the thickness of that border.
`configDumpFileName`	*String, Default: <not set>* – This can be used to write a file containing all parameters of the used configuration.
`configFile`	*String, Default: <not set>* – Configuration file (format: <key> <val>). Parameters in this file override values in the default configuration file. Command line arguments override all other settings. If this parameter is used within a config file, it has 'include once' semantics.
`fontFamily`	*String, Default: Arial* – Font Family. Only used if renderWithOpenCV is not set
`maxFontSize`	*Integer, Default: 100* – Maximum font size. Only used if renderWithOpenCV is not set
`minFontSize`	*Integer, Default: 12* – Minimum font size. Only used if renderWithOpenCV is not set
`minLevel`	*Integer, Default: 0* – Minimum level in layout of region to be extracted.
`renderWithOpenCV`	*Integer, Default: 0* – Set to 1 to use OpenCV's putText function for rendering the text. This is faster but does only support ASCII characters.
`xmlPath`	*String, Default: %base%idx.xml* – Path to the xml files in PAGE format. The same placeholders as in the global outputPath can be used.

4.2.43 `removeDiacritics` Operation

Remove diacritics (small and quadractic connected components) in the image Following parameters are available:

`configDumpFileName`	*String, Default: <not set>* – This can be used to write a file containing all parameters of the used configuration.
`configFile`	*String, Default: <not set>* – Configuration file (format: <key> <val>). Parameters in this file override values in the default configuration file. Command line arguments override all other settings. If this parameter is used within a config file, it has 'include once' semantics.
`maxHeight`	*Float, Default: 0.2* – Maximum height of a diacritic relative to image height
`maxWidth`	*Float, Default: 0.3* – Maximum width of a diacritic relative to image height

4.2.44 `removeLargeComponents` Operation

Remove large connected components in the image – i.e. components which exceed either the maximum width or the maximum height. Following parameters are available:

`configDumpFileName`	*String, Default: <not set>* – This can be used to write a file containing all parameters of the used configuration.
`configFile`	*String, Default: <not set>* – Configuration file (format: <key> <val>). Parameters in this file override values in the default configuration file. Command line arguments override all other settings. If this parameter is used within a config file, it has 'include once' semantics.

`maxHeight`	*Float, Default: 0.3* – Maximum height of a connected component relative to the image height.
`maxWidth`	*Float, Default: 0.2* – Maximum width of a connected component relative to the image width.

4.2.45 `removeSmallComponents` Operation

Remove small connected components in the image – i.e. components which are smaller than both the minimum width and height. Following parameters are available:

`configDumpFileName`	*String, Default: <not set>* – This can be used to write a file containing all parameters of the used configuration.
`configFile`	*String, Default: <not set>* – Configuration file (format: <key> <val>). Parameters in this file override values in the default configuration file. Command line arguments override all other settings. If this parameter is used within a config file, it has 'include once' semantics.
`minHeight`	*Float, Default: 0.3* – Minimum height of a connected component relative to the image height.
`minWidth`	*Float, Default: 0.2* – Minimum width of a connected component relative to the image width.

4.2.46 `removeUnderline` Operation

Remove underlines in text line images based on bottom point analysis: Record lowest foreground point ,look for straight lines, estimate line thickness with median height from segments just above a straight line, override with black. Following parameters are available:

`configDumpFileName`	*String, Default: <not set>* – This can be used to write a file containing all parameters of the used configuration.
`configFile`	*String, Default: <not set>* – Configuration file (format: <key> <val>). Parameters in this file override values in the default configuration file. Command line arguments override all other settings. If this parameter is used within a config file, it has 'include once' semantics.
`foregroundThreshold`	*Float, Default: 0.5* – Threshold for foreground
`maxHeight`	*Integer, Default: 10000* – Minimum distance from underline to image bottom in pixel
`maxRelHeight`	*Float, Default: 0.6* – Minimum height of the underline relative to the image height
`maxThickness`	*Integer, Default: 6* – Maximum thickness of underline in pixels.
`maxVariation`	*Integer, Default: 6* – Minimum width of connected underline in pixels.
`minRelWidth`	*Float, Default: 0.0* – Minimum width of connected underline relative to image width.
`minWidth`	*Integer, Default: 20* – Minimum width of connected underline in pixels.
`thicknessFactor`	*Float, Default: 1.5* – Underlines are removed up to a thickness of thicknessFactor*median thickness of this segment.

4.2.47 `removeVertTextMargin` Operation

Removes black space on top and bottom of the image, assuming that it contains only one single text line. Following parameters are available:

`configDumpFileName`	*String, Default: <not set>* – This can be used to write a file containing all parameters of the used configuration.
`configFile`	*String, Default: <not set>* – Configuration file (format: <key> <val>). Parameters in this file override values in the default configuration file. Command line arguments override all other settings. If this parameter is used within a config file, it has 'include once' semantics.
`minRowSum`	*Float, Default: 3.0* – Rows with less than minPixelCount white pixels are marked as black.
`minTextHeight`	*Integer, Default: 15* – Minimal text height in pixel.

4.2.48 `scale` Operation

Scales the images in the pipeline. Following parameters are available:

`configDumpFileName`	*String, Default: <not set>* – This can be used to write a file containing all parameters of the used configuration.
`configFile`	*String, Default: <not set>* – Configuration file (format: <key> <val>). Parameters in this file override values in the default configuration file. Command line arguments override all other settings. If this parameter is used within a config file, it has 'include once' semantics.
`xScale`	*Float, Default: 1.0* – Scale in x direction.
`yScale`	*Float, Default: 1.0* – Scale in y direction.

4.2.49 `sobel` Operation

Applies a sobel filter to the image and calculates derivatives in x or y direction. Following parameters are available:

`borderType`	*String, Default: constant* – Border type. Available values: constant, replicate
`configDumpFileName`	*String, Default: <not set>* – This can be used to write a file containing all parameters of the used configuration.
`configFile`	*String, Default: <not set>* – Configuration file (format: <key> <val>). Parameters in this file override values in the default configuration file. Command line arguments override all other settings. If this parameter is used within a config file, it has 'include once' semantics.
`kSize`	*Integer, Default: 3* – Size of the sobel kernel.
`xOrder`	*Integer, Default: 0* – Order of derivative in x direction.
`yOrder`	*Integer, Default: 1* – Order of derivative in y direction.

4.2.50 `splitTextLines` Operation

Expects input images to have horizontal baselines in the center of the image. Following parameters are available:

`configDumpFileName`	*String, Default: <not set>* – This can be used to write a file containing all parameters of the used configuration.
`configFile`	*String, Default: <not set>* – Configuration file (format: <key> <val>).

Parameters in this file override values in the default configuration file. Command line arguments override all other settings. If this parameter is used within a config file, it has 'include once' semantics.

`maxCut`	*Float, Default: 2.0* – Maximal sum at split borders
`minWidth`	*Float, Default: 4.0* – Minimal width of a child relative to image height.

4.2.51 `subtractMean` Operation

Mean normalization (Subtract mean) Following parameters are available:

`configDumpFileName`	*String, Default: <not set>* – This can be used to write a file containing all parameters of the used configuration.
`configFile`	*String, Default: <not set>* – Configuration file (format: <key> <val>). Parameters in this file override values in the default configuration file. Command line arguments override all other settings. If this parameter is used within a config file, it has 'include once' semantics.

4.2.52 `tee` Operation

tee writes the current set of images to the file system similarly to the linux command tee. The images in the pipeline remain unchanged. Following parameters are available:

`configDumpFileName`	*String, Default: <not set>* – This can be used to write a file containing all parameters of the used configuration.
`configFile`	*String, Default: <not set>* – Configuration file (format: <key> <val>). Parameters in this file override values in the default configuration file. Command line arguments override all other settings. If this parameter is used within a config file, it has 'include once' semantics.
`normalize`	*Integer, Default: 1* – Normalize between 0 and 255 before storing.
`outputPath`	*String, Default: tee-%base%idx%ext* – Path to the output files. See the -outputPath parameter of prepocressor for more information.

4.2.53 `textSkewCorrection` Operation

Corrects the text skew resulting from italic writing styles. Assumes that the base line is centered, i.e. pixels on the middle horizontal lines are not affected by this transformation. Following parameters are available:

`configDumpFileName`	*String, Default: <not set>* – This can be used to write a file containing all parameters of the used configuration.
`configFile`	*String, Default: <not set>* – Configuration file (format: <key> <val>). Parameters in this file override values in the default configuration file. Command line arguments override all other settings. If this parameter is used within a config file, it has 'include once' semantics.
`fromDegree`	*Float, Default: -50.0* – Minimal degree considered by Hough transform.
`maxDegree`	*Float, Default: 15.0* – If the detected text skew exceeds this parameter (in degrees), the detection is assumed to be incorrect and no correction is applied.
`resolution`	*Integer, Default: 90* – Resolution of Hough transform.
`sobelKSize`	*Integer, Default: 3* – Size of the sobel kernel.
`toDegree`	*Float, Default: 50.0* – Maximal degree considered by Hough transform.

4.2.54 `thinning` Operation

Line thinning as proposed by T.Y. Zhang and C.Y. Suen. Following parameters are available:

`configDumpFileName` *String, Default: <not set>* – This can be used to write a file containing all parameters of the used configuration.

`configFile` *String, Default: <not set>* – Configuration file (format: <key> <val>). Parameters in this file override values in the default configuration file. Command line arguments override all other settings. If this parameter is used within a config file, it has 'include once' semantics.

4.2.55 `threshold` Operation

The threshold command creates binary images. It is based on the OpenCV function threshold() and thus supports simple, adaptive, and otsu's thresholding. Following parameters are available:

`configDumpFileName` *String, Default: <not set>* – This can be used to write a file containing all parameters of the used configuration.

`configFile` *String, Default: <not set>* – Configuration file (format: <key> <val>). Parameters in this file override values in the default configuration file. Command line arguments override all other settings. If this parameter is used within a config file, it has 'include once' semantics.

`maxVal` *Integer, Default: 255* – Positive value after binarization (passed through to OpenCV).

`threshold` *Float, Default: 127.0* – Binarization threshold (passed through to OpenCV).

`type` *String, Default: BINARY,OTSU* – Thresholding type. See OpenCVs documentation for the threshold function. Connect options with ','. Available options are: BINARY, BINARY_INV, TRUNC, TOZERO, TOZERO_INV, OTSU

4.2.56 `transpose` Operation

Transpose the image. Following parameters are available:

`configDumpFileName` *String, Default: <not set>* – This can be used to write a file containing all parameters of the used configuration.

`configFile` *String, Default: <not set>* – Configuration file (format: <key> <val>). Parameters in this file override values in the default configuration file. Command line arguments override all other settings. If this parameter is used within a config file, it has 'include once' semantics.

4.2.57 `vertTextSegmentation` Operation

Extracts vertical cuts of text areas. The areas are identified by a constant horizontal gradient of the vertical projection. Following parameters are available:

`concatChildren` *Integer, Default: 0* – Set to 0 to pass each found text segment separately through the pipeline. Set to 1 to concat all children to a single image removing the non-text areas.

`configDumpFileName` *String, Default: <not set>* – This can be used to write a file containing all parameters of the used configuration.

`configFile`	*String, Default: <not set>* – Configuration file (format: <key> <val>). Parameters in this file override values in the default configuration file. Command line arguments override all other settings. If this parameter is used within a config file, it has 'include once' semantics.
`dilate`	*Float, Default: 0.05* – Safety margin which is added to the found text segments, relative to the page width. Set to negative to disable dilation.
`minMargin`	*Float, Default: 0.05* – Minimal vertical distance between two text areas relative to the page width.
`minSlope`	*Float, Default: 0.25* – Minimal gradient in the vertical projection of a text area. The values of the vertical projection range from 0 to 1, and the projection is resized to -resolution
`minWidth`	*Float, Default: 0.1* – Minimal width of a text area relative to the page width.
`morph`	*String, Default: openFirst* – Morphology operations on the segmentation. • openFirst: opening, then closing • closeFirst: closing, then opening • none: No morphology operation.
`outputSegmentation`	*Integer, Default: 0* – Output an 1xresolution array indicating the classification of the columns in text and non-text columns. If this is set to 0, the input image is cutted according the text segmentations.
`resolution`	*Integer, Default: 100* – Resolution for the vertical projection. 100 is a good value even for documents with largely differing sizes.
`symmetric`	*Integer, Default: 0* – Set to 1 to ensure that the middle of the input image corresponds to the middle of the output image. Useful for preserving the baseline position. Only applies if the number of children equals one (e.g. enforced by -concatChildren)
`transpose`	*Integer, Default: 0* – Set to 1 to transpose the image before applying segmentation algorithm. This results producing horizontal instead of vertical cuts.

4.2.58 `visualizePageXml` Operation

Visualizes line segmentations and confidences in page XML files.Note: The used PAGE library may break with multiple threads! Following parameters are available:

`configDumpFileName`	*String, Default: <not set>* – This can be used to write a file containing all parameters of the used configuration.
`configFile`	*String, Default: <not set>* – Configuration file (format: <key> <val>). Parameters in this file override values in the default configuration file. Command line arguments override all other settings. If this parameter is used within a config file, it has 'include once' semantics.
`extractRegions`	*Integer, Default: 1* – Extract regions.
`extractTextObjects`	*Integer, Default: 0* – Extract text lines.
`fontThickness`	*Integer, Default: 4* – Thickness of label font. Only used if -lineLabels is set to 'page-ids' or 'consecutive'.
`lineLabels`	*String, Default: none* – This defines the way how line labels are displayed:

- 'none': Do not show any line labels.
- 'page-ids': Use line ids defined in the page xml file.
- 'consecutive': Number lines starting from 1.

`minLevel` *Integer, Default: 0* – Minimum level in layout of region to be extracted.

`showConfidences` *Integer, Default: 0* – If this is set to 1, the border color around lines corresponds to the OCR confidence as defined in the page xml file. If this is set to 0 the border is always red. NOTE: This function is NOT implemented yet!

`xmlPath` *String, Default: %base%idx.xml* – Path to the xml files in PAGE format. The same placeholders as in the global outputPath can be used.

4.2.59 `writeRects` Operation

Writes the rectangles for each child into a file. Following parameters are available:

`configDumpFileName` *String, Default: <not set>* – This can be used to write a file containing all parameters of the used configuration.

`configFile` *String, Default: <not set>* – Configuration file (format: <key> <val>). Parameters in this file override values in the default configuration file. Command line arguments override all other settings. If this parameter is used within a config file, it has 'include once' semantics.

`outputPath` *String, Default: %base%idx-rect.txt* – Path to the rectangle files to create. The same placeholders as in the global outputPath can be used.

References

G. Bradski and A. Kaehler. *Learning OpenCV: Computer vision with the OpenCV library.* O'Reilly Media, Inc., 2008.

X. Huang, A. Acero, H.-W. Hon, and Raj R. *Spoken language processing: A guide to theory, algorithm, and system development.* Prentice Hall PTR, 2001.

D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlíček, Y. Qian, P. Schwarz, et al. The Kaldi speech recognition toolkit. 2011.

J. A. Sanchez, A. H. Toselli, V. Romero, and E. Vidal. ICDAR2015 Competition HTRtS: Handwritten Text Recognition on the tranScriptorium Dataset. In *ICDAR*. IEEE, 2015.

F. Stahlberg and S. Vogel. Document Skew Detection Based on Hough Space Derivatives. In *ICDAR*. IEEE, 2015a.

F. Stahlberg and S. Vogel. Detecting dense foreground stripes in Arabic handwriting for accurate baseline positioning. In *ICDAR*. IEEE, 2015b.

Index