Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

OCR4all –
An Open Source Tool Providing a Full OCR Workflow

Christian Reul and Frank Puppe

Chair for Artificial Intelligence and Applied Computer Science
University of Würzburg

16.04.2018

Contents and Goals of the Workshop

Things to be looked at:

- The OCR workflow implemented in OCR4all.
  - Hands-on experience using several historical prints.
  - Basic concepts behind the tools/steps in the workflow.
- Additional OCR related tools.
- More advanced OCR concepts.

Things to (hopefully) be achieved:

- General understanding of the workflow and the concepts behind it.
- Interest in further usage of OCR4all.
- Basic knowledge about advanced OCR concepts.
- Testing of and feedback for OCR4all.

Motivation – Why not (only) Abbyy?

- Costs money.
- Proprietary code (closed source).
  - No individual adaptations.
  - Future unclear.
- Relatively poor support of historical prints.
  - Mainly 19th century and later.
  - Supports only six languages (in terms of dictionaries, language models, …).
  - Individual font training possible but laborious and ineffective.
- Combination with open source tools possible and effective.

OCR4all – Motivation and General Idea

- Open source OCR tools powerful but can be overwhelming for inexperienced users:
  - No Windows compatibility.
  - Complicated setup (missing dependencies, …).
  - No comfortable GUI but unfamiliar command line usage.
  - …
- Idea behind OCR4all:
  - Comprehensible for and applicable by any given user.
  - Entire workflow encapsulated into a single Docker image.
    - Platform independence.
    - Easy installation.
  - Based on several open source tools (mainly OCRopus).
  - (Soon to be) open source.

OCR4all – Current Status and Goals

- Initially designed to support the OCR of (very) early printed books.
  - Considered impossible only a few years ago.
  - Complicated layout analysis.
  - Book specific model training.
  - …

    → Users accept a certain amount of manual effort.
- Already applicable to a wide variety of prints.
- Main goal: Further increase of the degree of automation and robustness.
- Work in progress!

OCR4all – Main Submodule OCRopus

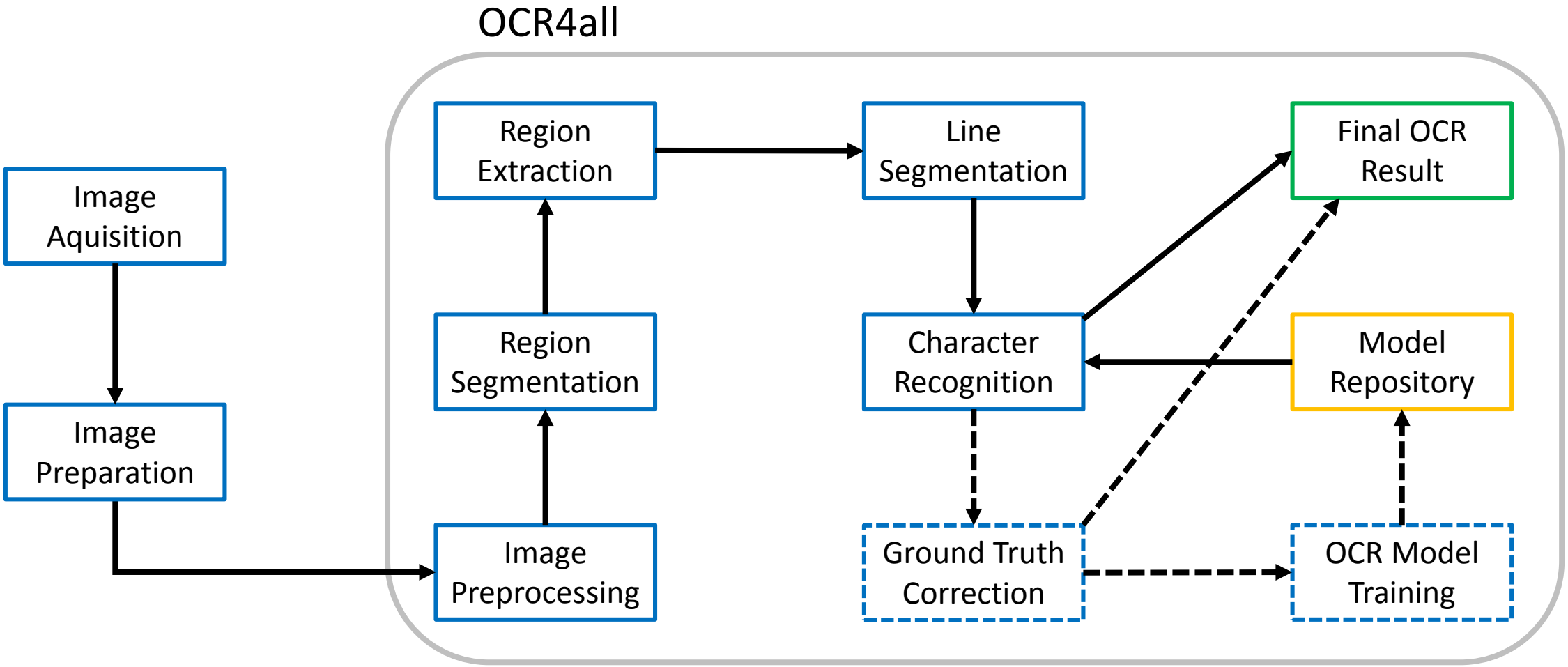
- Centrepiece of the OCR4all workflow.
- Python-based command line tools for document analysis and OCR.
  - Preprocessing.
  - Layout analysis and line segmentation.
  - Character recognition and model training.
- Developed by Thomas M. Breuel (University of Kaiserslautern/DFKI, Xerox, Google, currently Nvidia).
- Open source.
- Breuel et al.: High-Performance OCR for Printed English and Fraktur using LSTM Networks.

Further Reading

The practical part of this workshop will focus on OCR4all. For tutorials regarding the usage of the OCRopus command line scripts please see:

- Springmann and Fink: OCR and postcorrection of early printings for digital humanities.
  - Workshop that also covers varies additional topics like other OCR engines or post correction.
  - Highly recommended!
- Springmann: Ocrocis tutorial.
- OCRopus Github Wiki.
- …

OCR4all – Workflow (Main Steps Only)



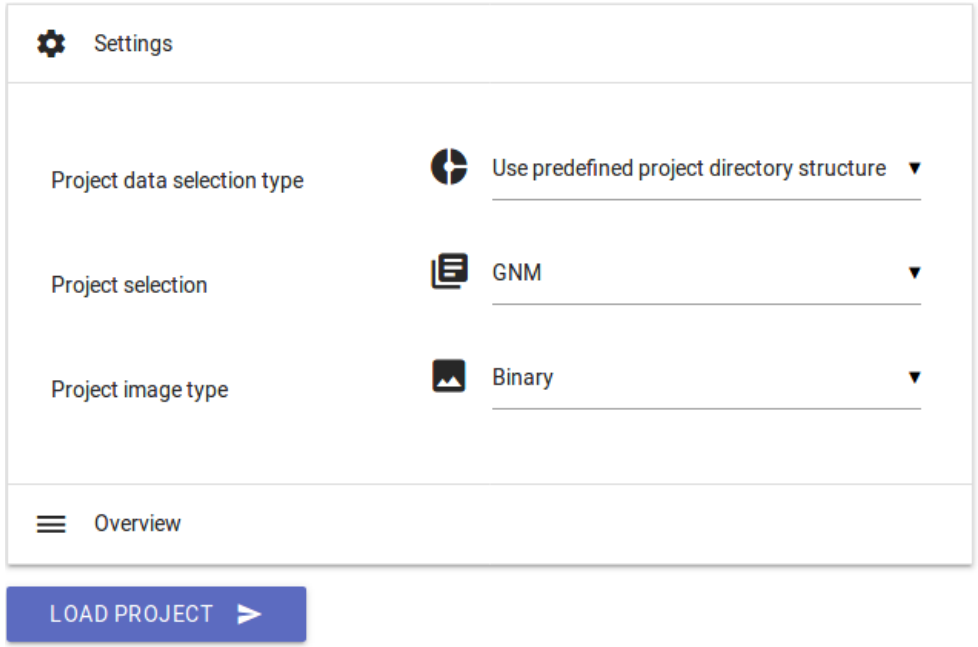
Hands-on: Accessing OCR4all

- OCR4all originally designed to run locally and be accessed via a web GUI.
- Local technical restrictions → deployed on our server as a true web app.
- Resources have to be shared among the participants.
  - 32 CPUs → We should be fine.
  - Every user has access to all the data → Please work only on your assigned projects!

- Open Firefox and access supergirl.informatik.uni-wuerzburg.de:8082/OCR4all_Web/
  - “supergirl” is the name of the server.
  - Yes… supergirl…

Hands-on: Selecting a Book to Process

- “data” folder on your host machine gets mirrored into docker.
- Expected input format for a book: …/data/*book_name*/Original/*images*.
- Go to “Settings” and select “GNM” from the dropdown.
- If necessary, change “image type” to “binary” and hit “Load Project”.



Hands-on: Project Overview

- The “Project Overview” shows the progress of your project on a page basis.
- By selecting a page’s “Page Identifier” you can get a closer look at the already produced output like…
  - the original image.
  - different preprocessing steps.
  - extracted segments.
  - segmented lines.

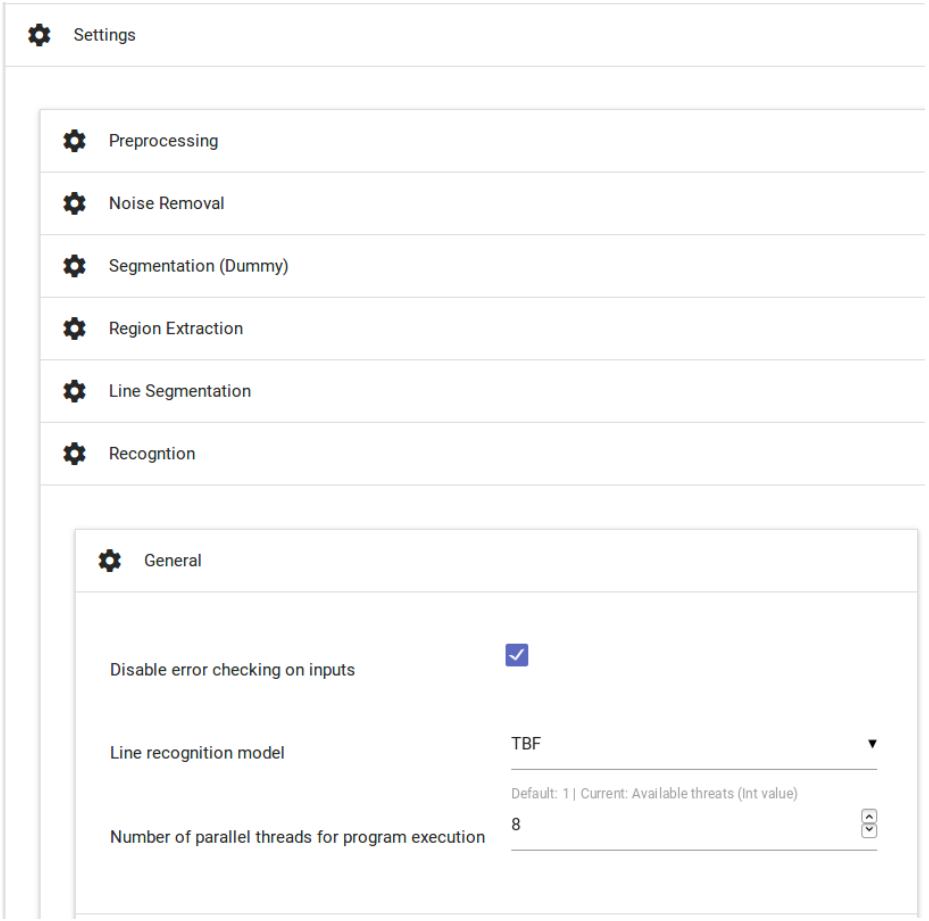
Page Identifier	Preprocessing	Noise Removal	Segmentation	Region Extraction	Line Segmentation	Recognition
0001	✓	✓	✓	✓	✓	✗
0002	✓	✓	✓	✓	✗	✗
0003	✓	✓	✓	✗	✗	✗
0004	✓	✓	✓	✗	✗	✗
0005	✓	✓	✓	✓	✓	✗
0006	✓	✓	✓	✓	✗	✗
0007	✓	✓	✓	✗	✗	✗
0008	✓	✓	✓	✓	✓	✗
0009	✓	✓	✓	✓	✓	✗
0010	✓	✓	✓	✗	✗	✗
0011	✓	✓	✓	✓	✓	✗
0012	✓	✓	✓	✗	✗	✗

Showing 1 to 12 of 12 entries

Previous 1 Next

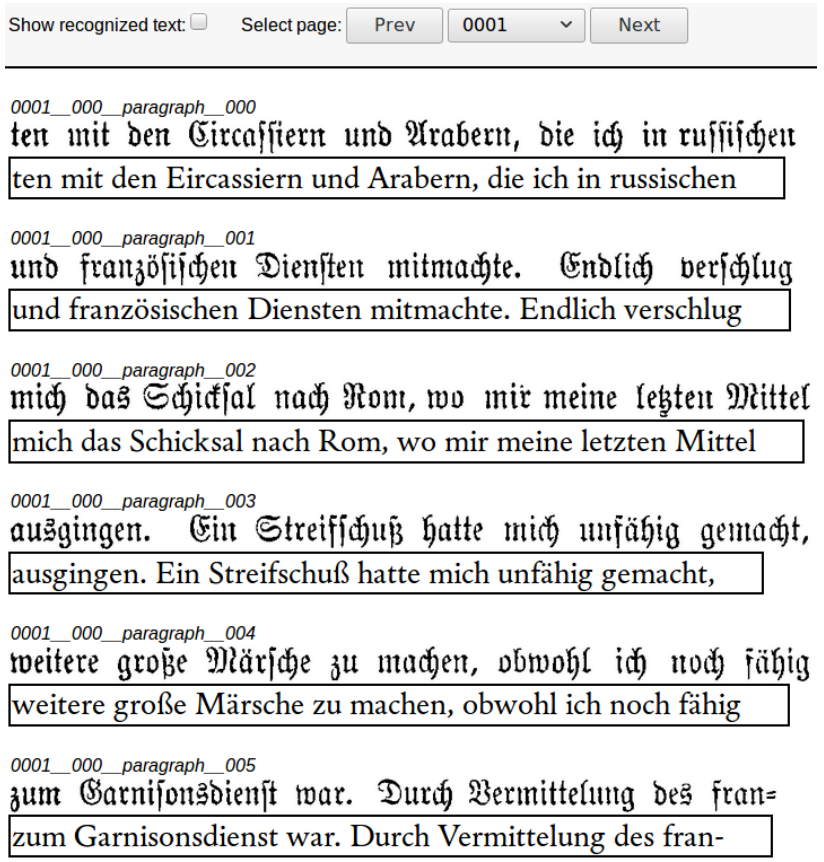
Hands-on: A First Fully Automated Run

- Select “Centralized Process Flow” from the menu (☰) on the top left.
- Choose “TBF” as your OCR model (“Settings” → “Recognition”).
- Keep the “Number of parallel threads for program execution” to the default.
- Hit execute and wait for the success notification.



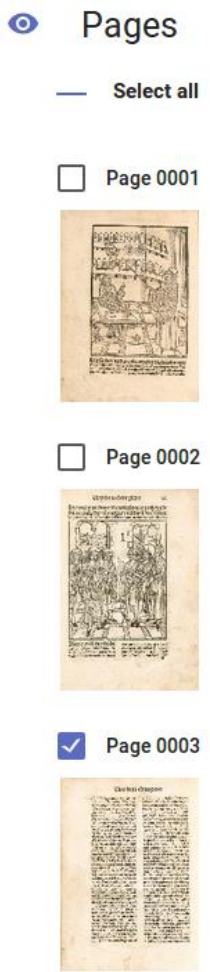
Hands-on: Checking the Results

- Go to “Ground Truth Correction”.
- The line images are aligned with the corresponding OCR result.
- Navigate through the pages and get a first impression of the results.



Hands-on: Experimenting with Books and Models

- Perform another run through using book “Cirurgia”, image type “gray” and model “TBF”.
- Select only a single page without images for processing.
- Check the results.
- Change the model to “Cirurgia” and go again.
- Check the results.



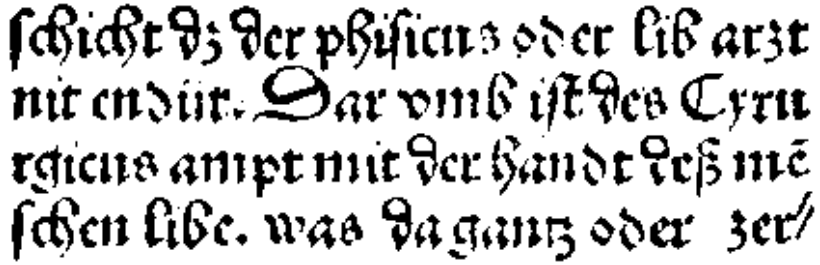
Comparing the Results

- The book “Cirurgia” was printed in 1499 using broken script, i.e. Fraktur in a wider sense.
- Abbyy (Old German) and the TBF model produce completely useless recognition results.
- Standard models don’t fit the printing types.
- The book specific OCRopus model yields an excellent character error rate of well below 2%.

schichr d; der phisicn» oder kbarzr
nir c„di«r.^ar vmb iffdc» Lzn»
rgicir» aniprnncdcc handrdcß n»k
schcn kbe. wa» daganiz »derzcr-

scJucErpBexpHgcnoder llarzrmr
cndut. Saromb usktes Efru
rgtans ampt mu Vcr handtkeßmtä
sckeni lbe. was dngumn oder zer?

ſchicht dz der piſicus oder lib arzt
nit en dut. Dar vmb iſt des yru
rgieus am pt mit der handt drß mẽ
ſchen libe. was da gantz oder zer=



A First Assessment

Expectable OCR quality highly depends on several factors:

- Condition of the book.
- Layout: can vary from trivial to extremely complex.
- Printing type and language:
  - Some strong polyfont models available (Abby, OCRopus, …) for books printed ca. 1800 or later.
  - Early prints usually require book/type specific model training.

Different books require different approaches:

- Trade-off: degree of automation ↔ versatility of an approach.
- Rule of thumb: methods that work for older/more difficult prints will also work for newer/less difficult ones but not necessarily vice versa!

Workflow – Image Preparation

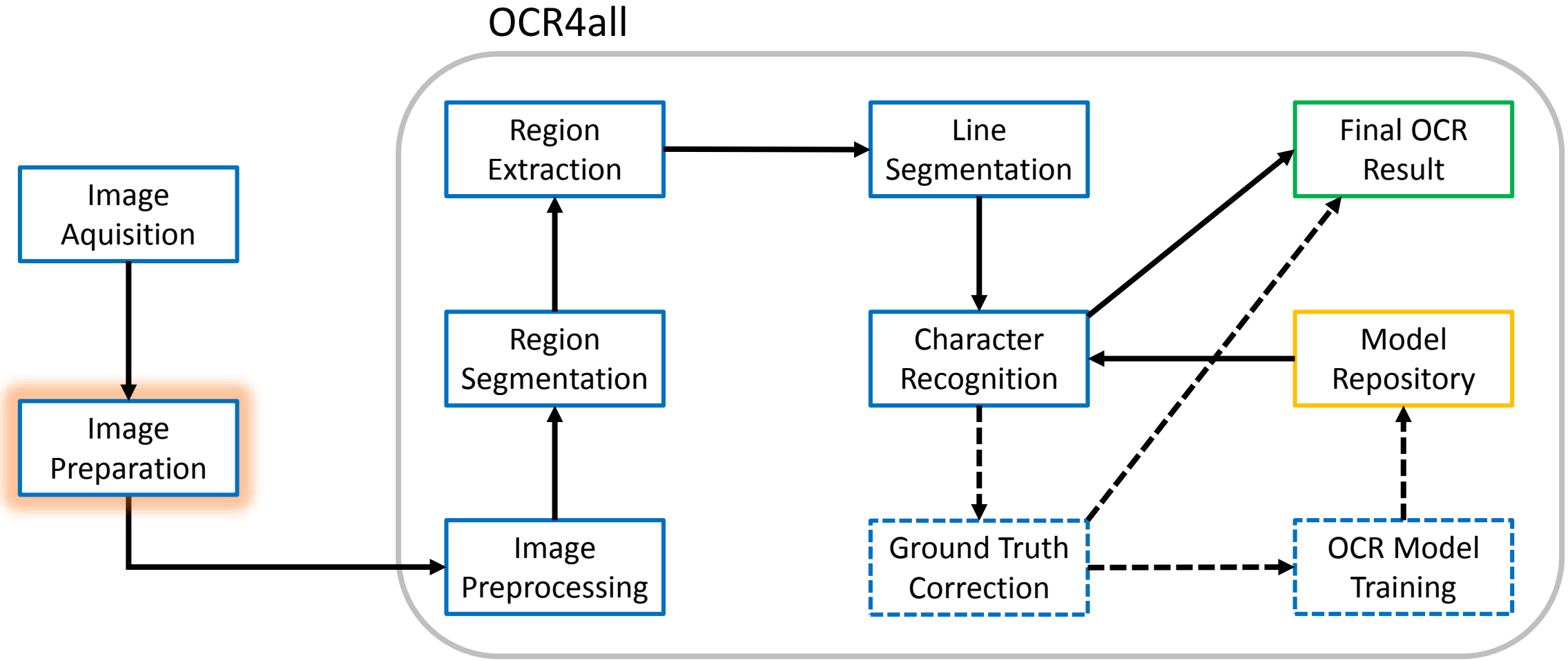


Image Preparation – Overview

- **Input:** unprepared image.
  Output: prepared image.
- Necessary actions vary from book to book.
  - Page splitting.
  - Rotation.
  - Periphery removal.
  - …
- Not (yet) integrated into OCR4all
  → open source tool ScanTailor.

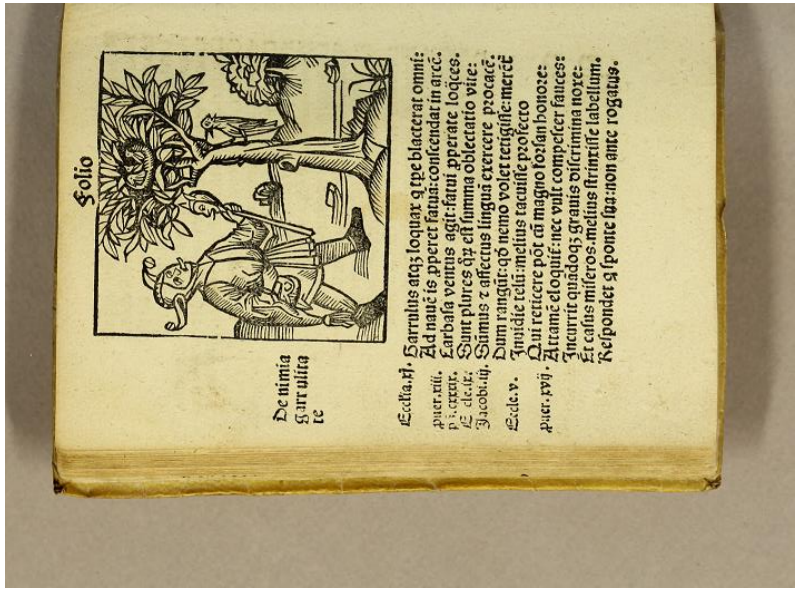
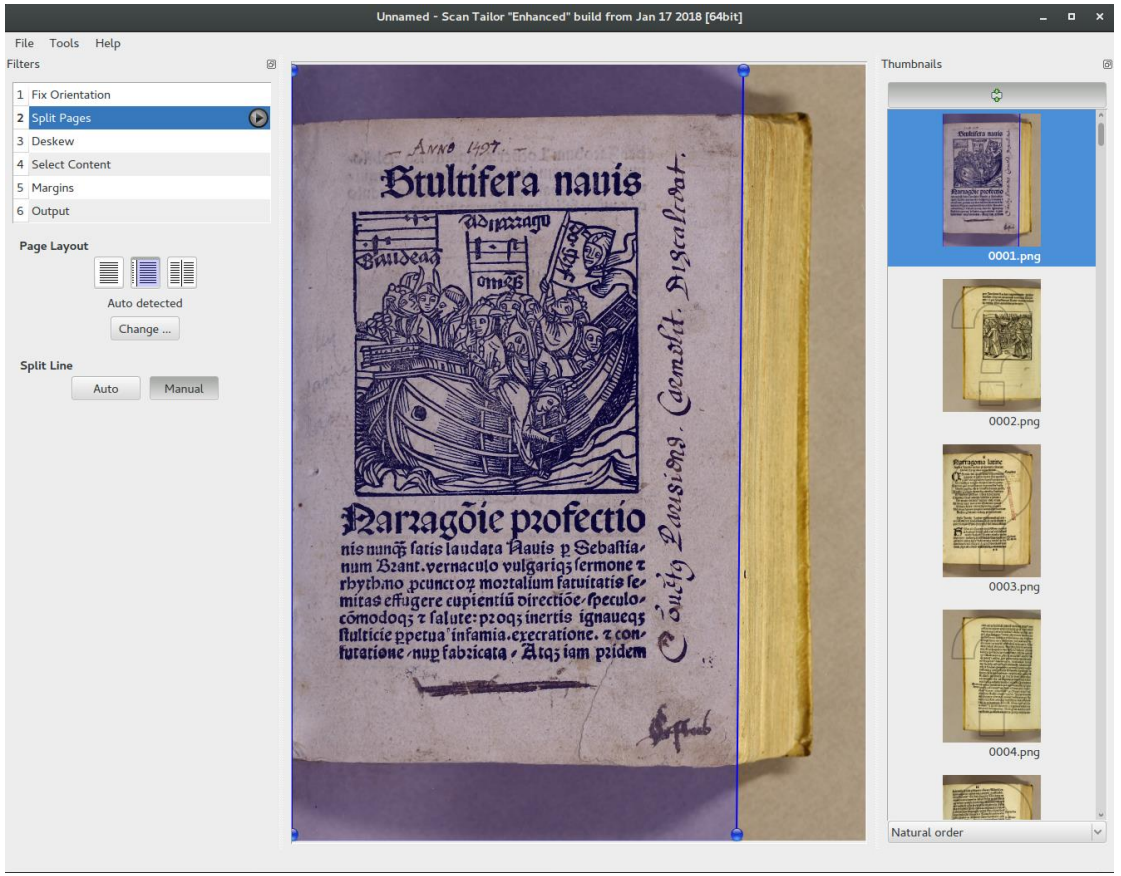


Image Preparation – ScanTailor

- Many features:
  - Page splitting/periphery removal.
  - Rotation.
  - Content detection.
  - Deskewing.
  - Binarisation.
  - …
- Usually, the binarisation step should be skipped, since OCRopus produces better results.
- A detailed video tutorial is available here.



Workflow – Image Preprocessing

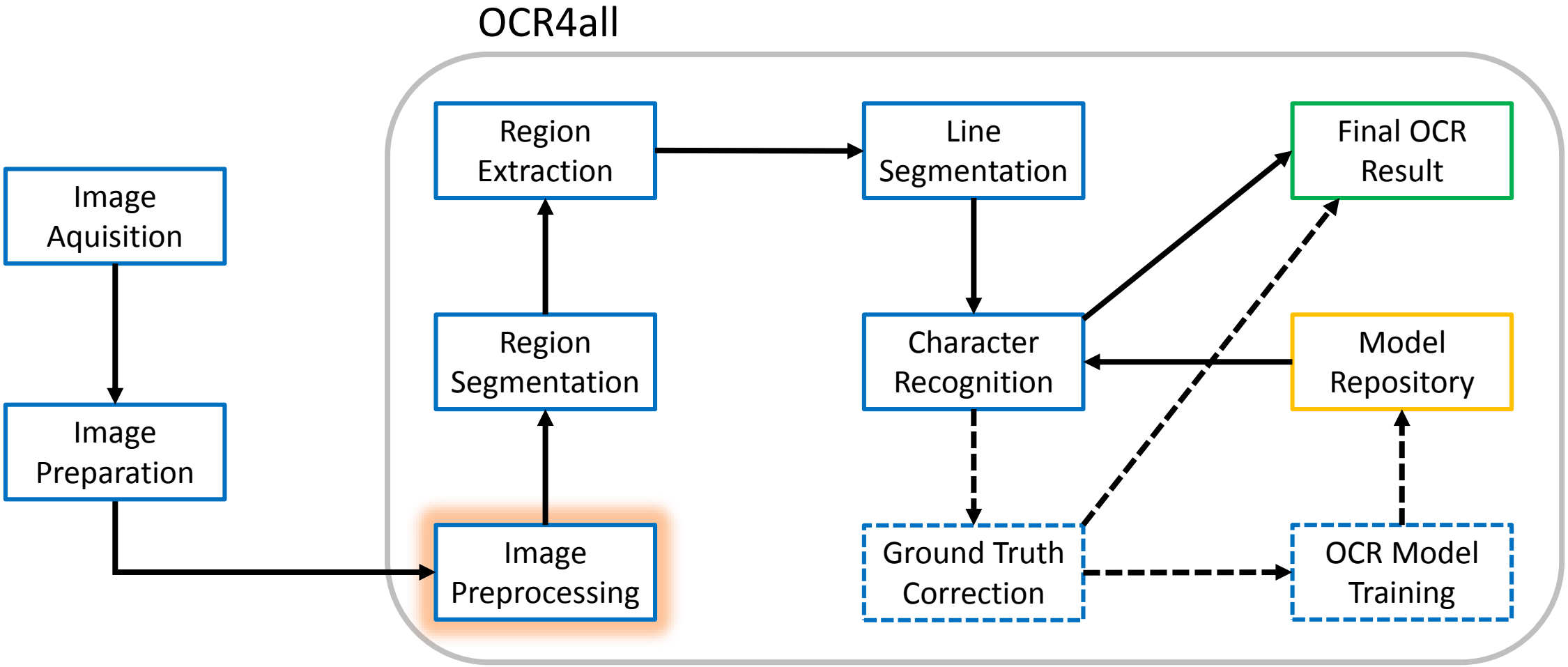
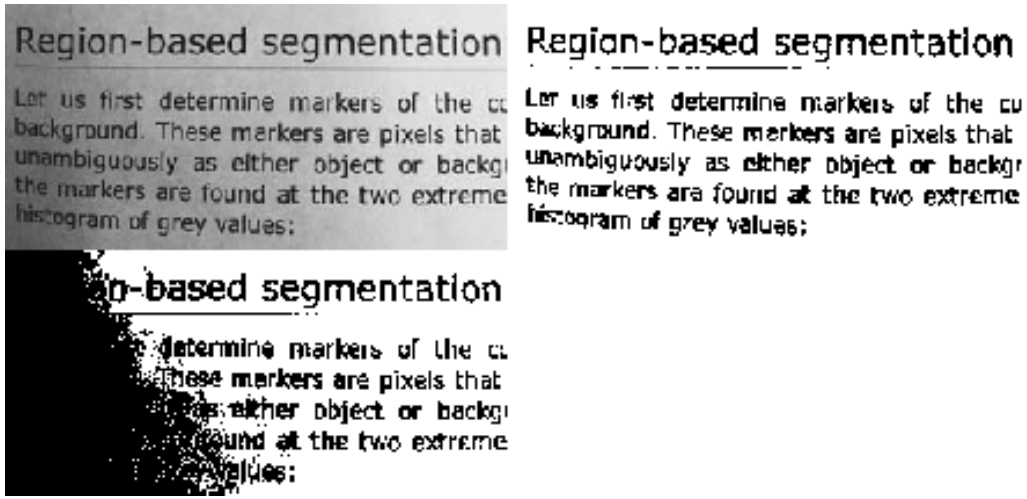


Image Preprocessing – Overview

- **Input:** original image (color, grayscale, or binary).
  Output: deskewed binary (and grayscale) image(s).
- Two steps:
  - **Binarisation:** necessary for many image processing operations.
  - **Deskewing:** improves the upcoming line segmentation.
- Implemented in ocropus-nlbin.

Binarisation – Basics

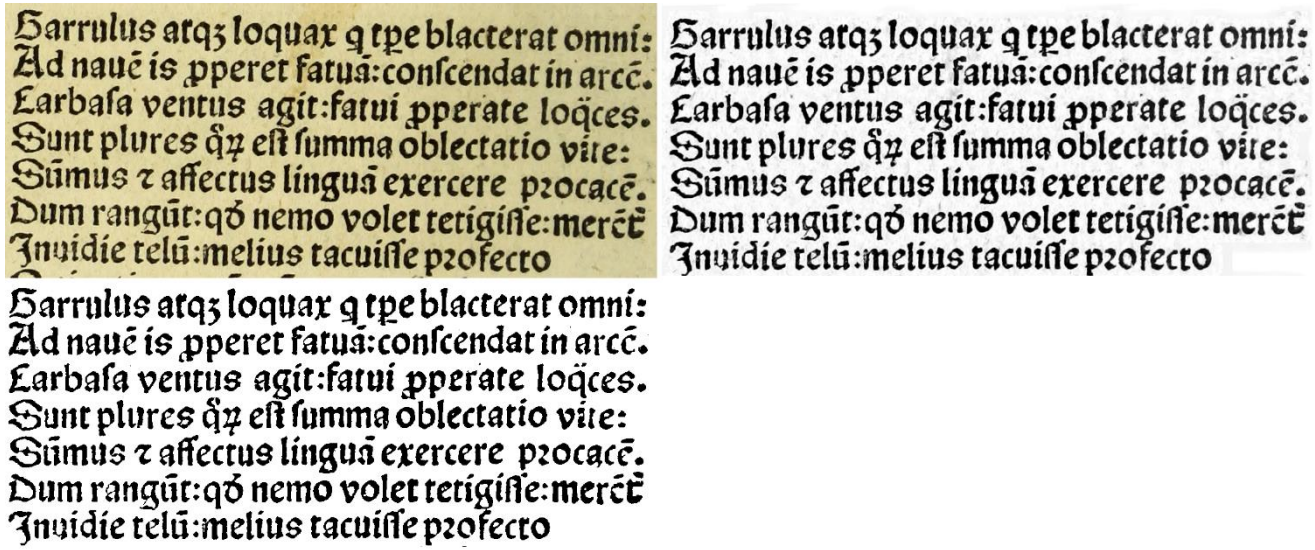
- Goal: Reduce a color or grayscale image to a binary one.
- Image types:
  - **Color:** Three values (RGB) per pixel with a certain intensity range, e.g. 0-255.
  - **Grayscale:** One value per pixel (e.g. 0-255), calculated from RGB.
  - **Binary:** Only two valid pixel values (0 = black, 1 or 255 = white), calculated from grayscale.
- Two general approaches:
  - **Global threshold** T: if gray(x) < T: 0; else: 1.
  - **Local threshold**: Adaptive threshold based on local grayscale intensities.



Example binarisation result of a scan segment: grayscale image (tl), global threshold (Otsu, bl), local threshold (Sauvola, tr).

Binarisation – OCRopus (and OCR4all) Approach

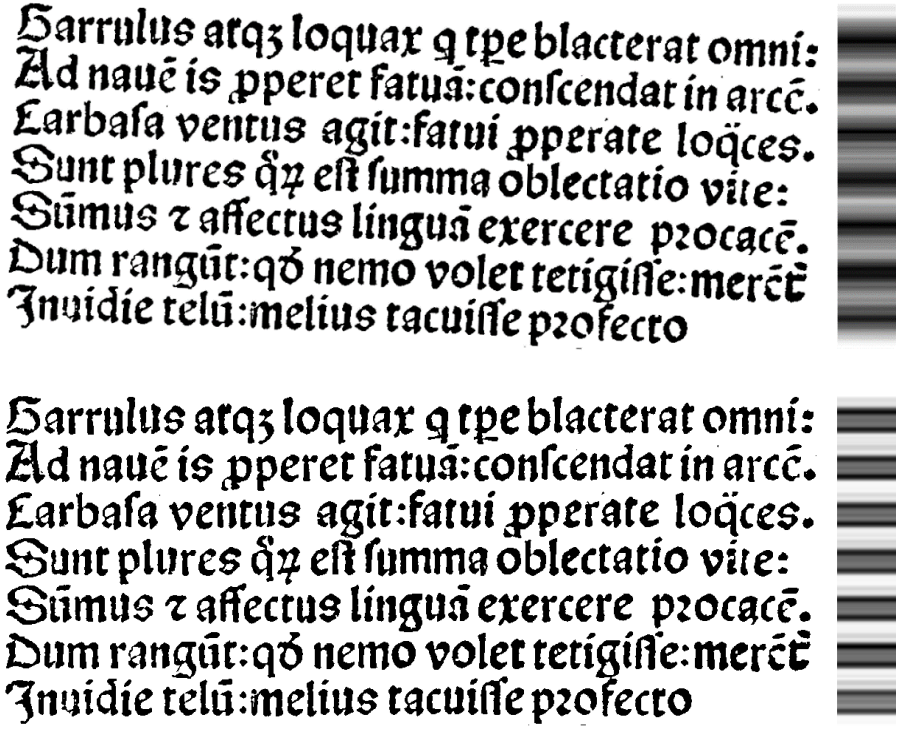
- Local adaptive background estimation using percentile filters.
- Outputs a binary image and a flattened (filtered background) grayscale image.
- Afzal et al.: Robust Binarization of Stereo and Monocular Document Images Using Percentile Filter.



Example nlbin result of a scan segment:
original image (tl), flattened grayscale (tr), binary (bl).

Deskewing

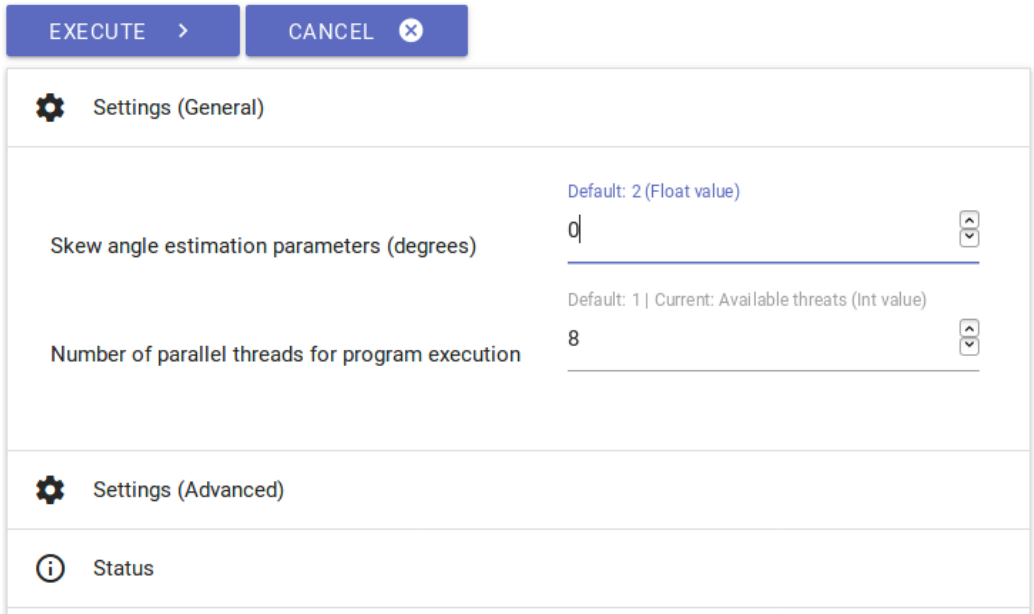
- Basic idea: Straight text lines will maximize the standard deviation of the black pixel counts for each row in the image.
- Brute force approach:
  - For different angles
    (default: -2.00°, -1.75°, -1.50°, …, 2.00°):
    - Rotate binary image by the given angle.
    - Count black pixels in each row.
    - Calculate the standard deviation of the counts.
  - Choose angle that yielded the maximal standard deviation.



Example deskewing input (top) and output (bottom).
On the right: black pixel row counts (more = darker).

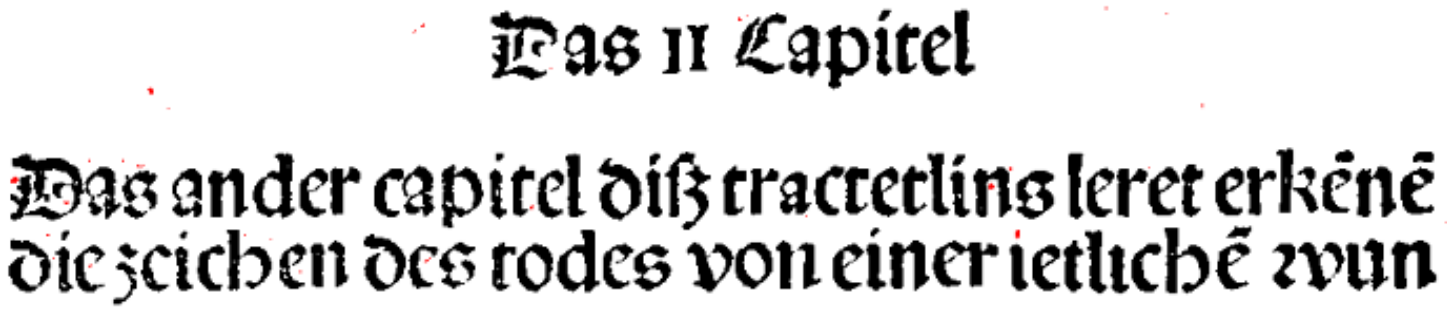
Hands-on: Preprocessing

- Select “Cirurgia” as book and set “image type” to “gray”.
- Select “Preprocessing” from the menu and perform a run with the default settings.
- Check the results by selecting preprocessed pages in the “Project Overview” view.
- Perform the preprocessing again with “Skew angle estimation parameters” set to “0”.
- Check the results.



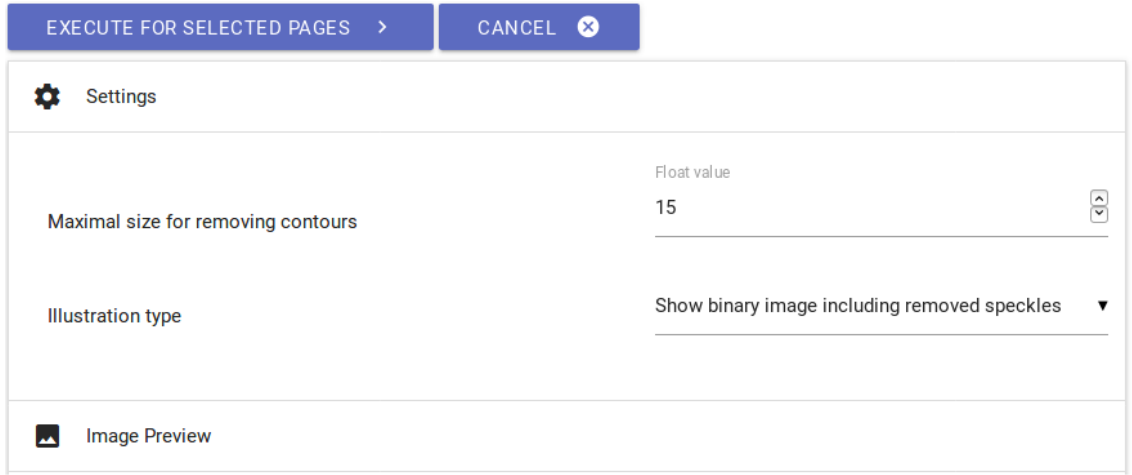
Noise Removal

- **Input:** noisy binary image.
  Output: binary image with no/less noise.
- Detects contours and removes the ones smaller than a given area threshold.
- Unnecessary for most books and not part of the default fully automatic workflow.
- Can be very useful when using LAREX (coming up) for segmentation.

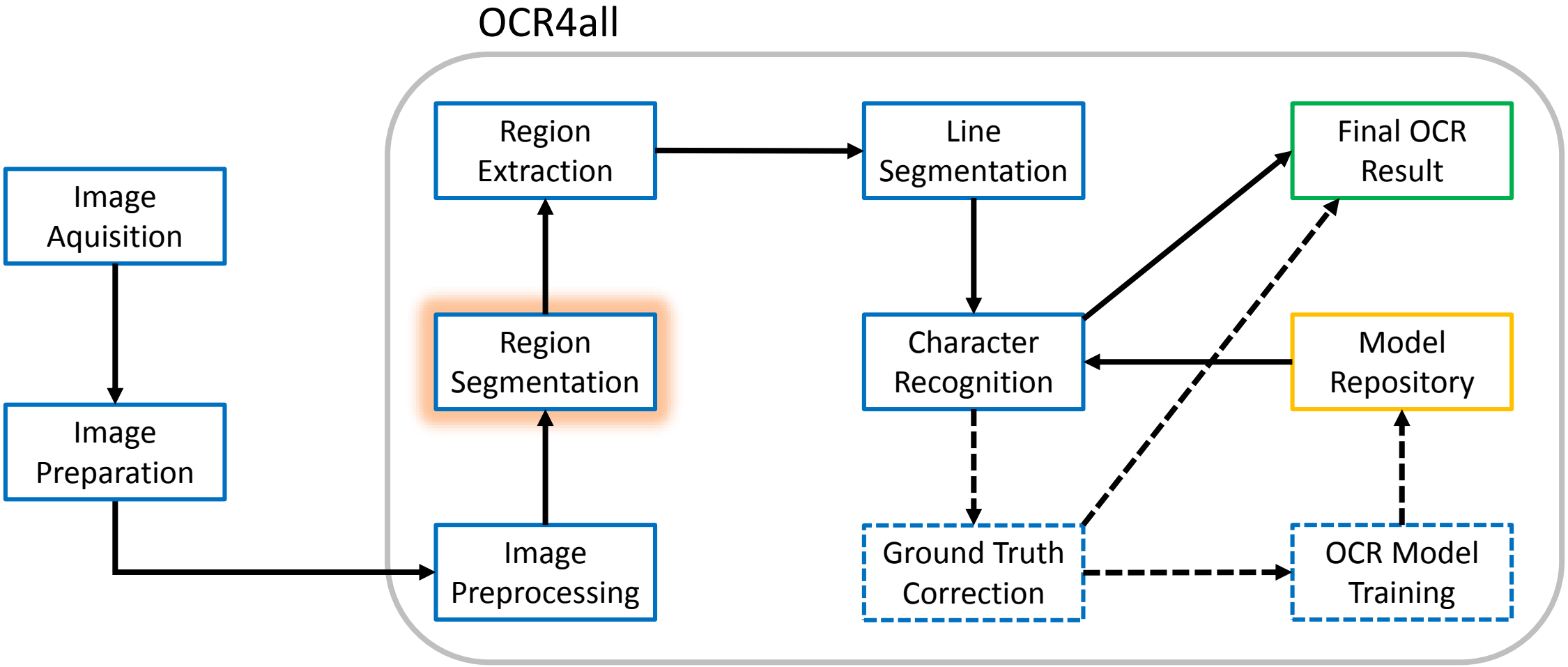


Hands-On: Noise Removal

- Book: “Cirurgia”, image type : “gray”.
- Go to “Noise Removal”.
- Select a page and open “Image Preview”.
- Play around with different pages and the “Maximal size for removing contours” value.
- Select a fitting value (15?) and execute.

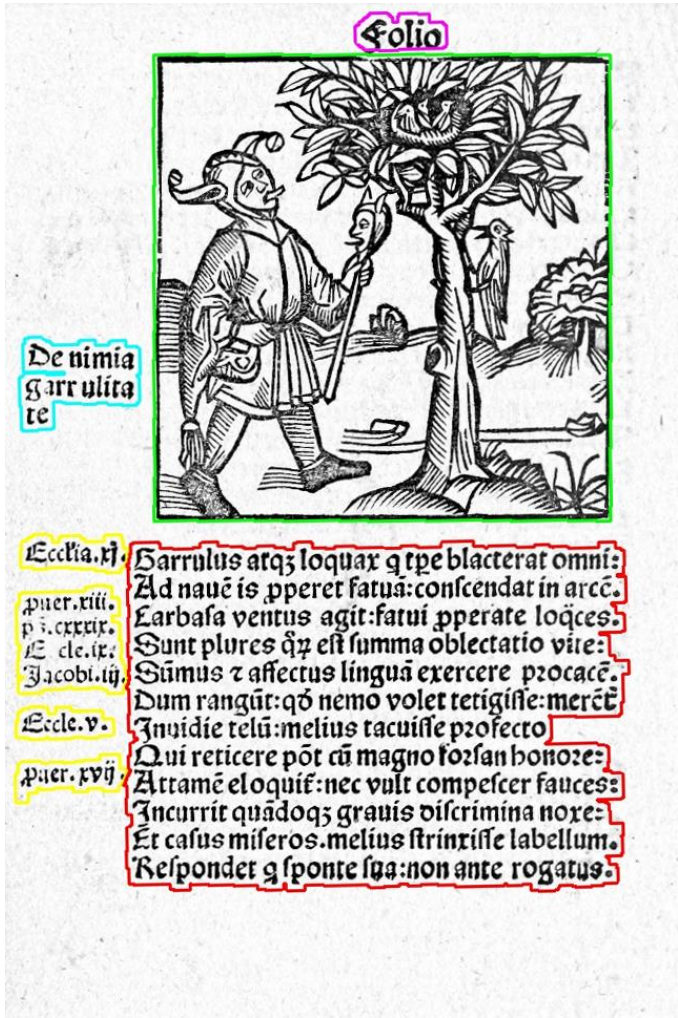


Workflow – Region Segmentation



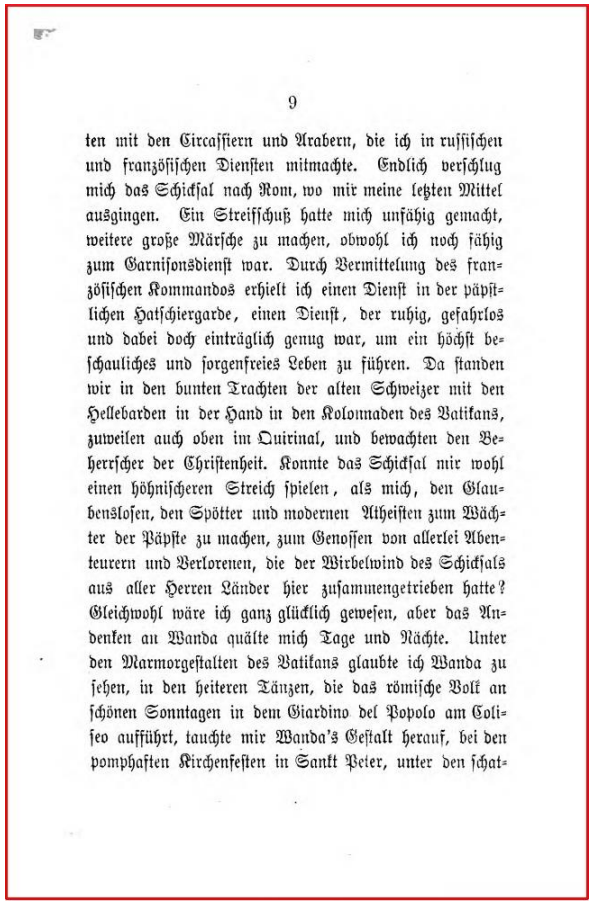
Region Segmentation

- **Input:** preprocessed image.
  Output: structural information about regions (position and type) and their reading order.
- Different manifestations:
  - Text/Image segmentation.
  - Fine-grained semantic distinction between text types (running text, heading, marginalia, …).
  - …
- Tools/Methods:
  - Dummy segmentation.
  - Pixel classifier (coming soon).
  - LAREX.



Region Segmentation – Dummy Segmentation

- Considers the entire page as a single segment.
- Leaves text/image segmentation and column separation to the upcoming line segmentation.
- Fully automatic and very fast.
- Often completely satisfactory when dealing with simple layouts (e.g. standard 19th century novels).
- No image markup.
- No semantic distinction of text parts.



9

ten mit den Circaſſiern und Arabern, die ich in ruſſiſchen und franzöſiſchen Dienſten mitmachte. Endlich verſchlug mich das Schickſal nach Rom, wo mir meine letzten Mittel ausgingen. Ein Streifſchuß hatte mich unfähig gemacht, weitere große Märſche zu machen, obwohl ich noch fähig zum Garniſonsdienſt war. Durch Vermittelung des franzöſiſchen Kommandos erhielt ich einen Dienſt in der päpſtlichen Hatſchiergarde, einen Dienſt, der ruhig, gefahrlos und dabei doch einträglich genug war, um ein höchſt beſchauliches und ſorgenfreies Leben zu führen. Da ſtanden wir in den bunten Trachten der alten Schweizer mit den Hellebarden in der Hand in den Kolonnaden des Vatikans, zuweilen auch oben im Quirinal, und bewachten den Beherrſcher der Chriſtenheit. Konnte das Schickſal mir wohl einen höhniſcheren Streich ſpielen, als mich, den Glaubenloſen, den Spötter und modernen Atheiſten zum Wächter der Päpſte zu machen, zum Genoſſen von allerlei Abenteurern und Verlorenen, die der Wirbelwind des Schickſals aus aller Herren Länder hier zuſammengetrieben hatte? Gleichwohl wäre ich ganz glücklich geweſen, aber das Andenken an Wanda quälte mich Tage und Nächte. Unter den Marmorgeſtalten des Vatikans glaubte ich Wanda zu ſehen, in den heiteren Tänzen, die das römiſche Volk an ſchönen Sonntagen in dem Giardino del Popolo am Coliſeo aufführt, tauchte mir Wanda's Geſtalt herauf, bei den pomphaften Kirchenfeſten in Sankt Peter, unter den ſchat-

Region Segmentation – LAREX

- Text/non-text separation not always sufficient.
- Desired additional semantic distinction.

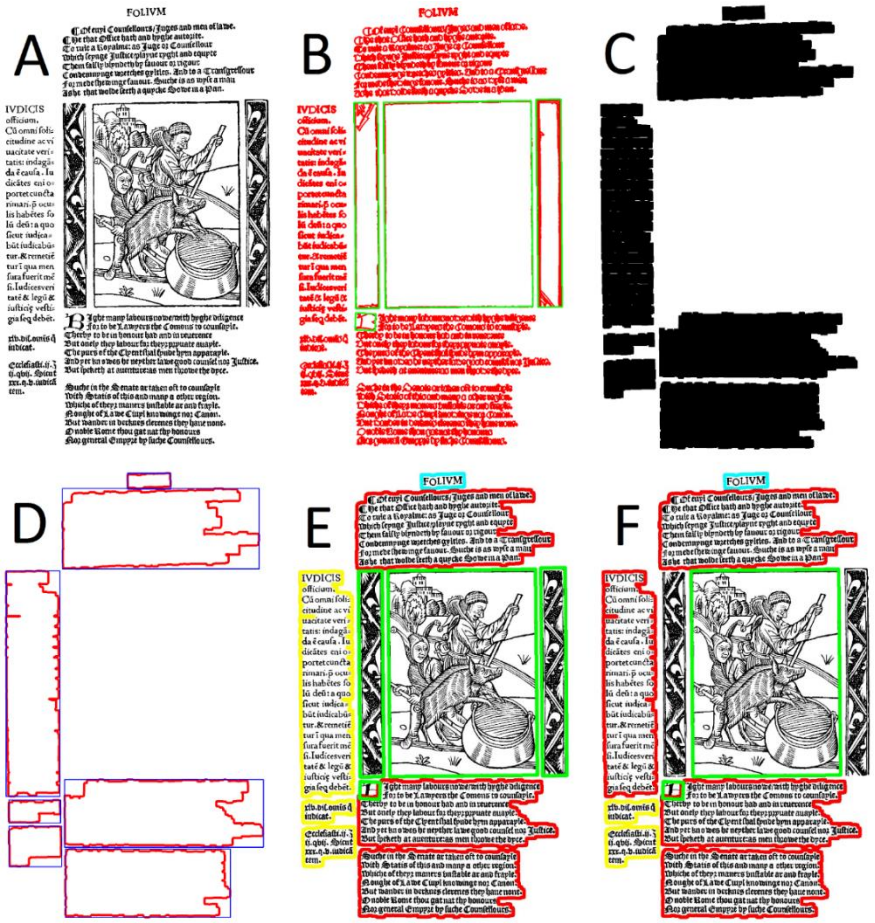
- Semi-automatic.
- Intuitive, adaptable and comprehensible.
- Open source.
- Layout Analysis and Region EXtraction.
- Reul, Springmann, Puppe: LAREX - A semi-automatic open-source Tool for Layout Analysis and Region Extraction on Early Printed Books.

- Two basic assumptions: within a book
  - characters, words, and lines that semantically belong together are closer to each other than the ones that don't.
  - the general layout follows certain rules with regard to the positioning of regions.

LAREX – Workflow

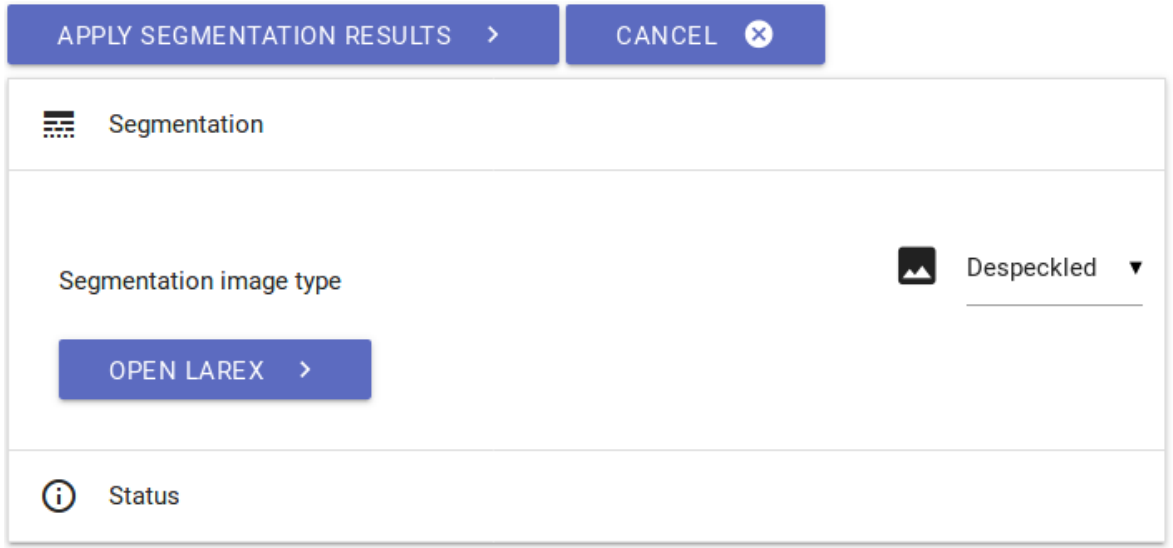
Input: binary, grayscale, or color image.
Output: classified segments as PageXML.

1. Pre-processing (A).
2. Image detection (A → B).
    1. Image/non-image classification.
    2. Image removal.
3. Text classification (C → D, E).
    1. Foreground dilation.
    2. Classification using rules (position and size).
4. Manual corrections (E → F).
5. Conversion to output format.



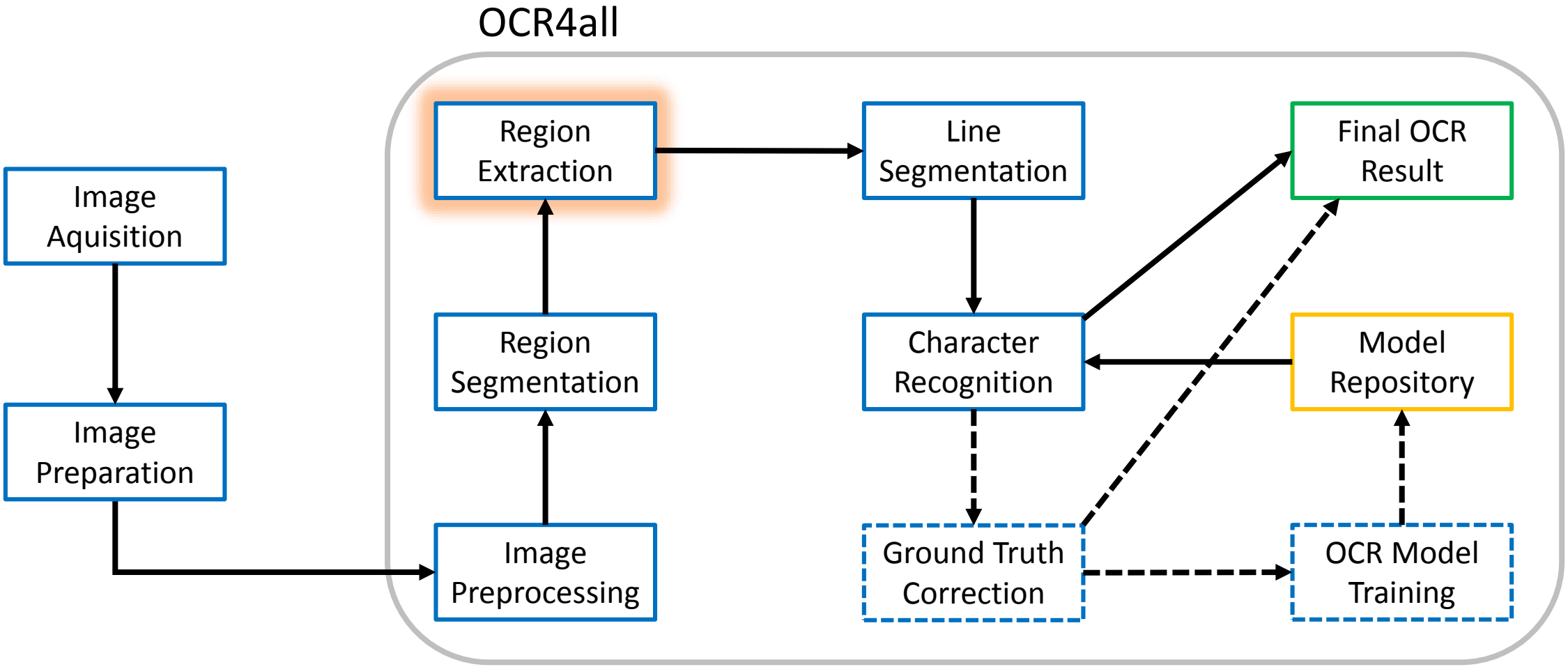
Hands-On: Segmenting with LAREX

- Book: “Cirurgia”, image type: “gray”.
- Go to “Segmentation” → “LAREX”.
- Select “Despeckled” as input images and hit “Open LAREX”.
- Follow the instructions.
- When finished:
  “Apply segmentation results”!



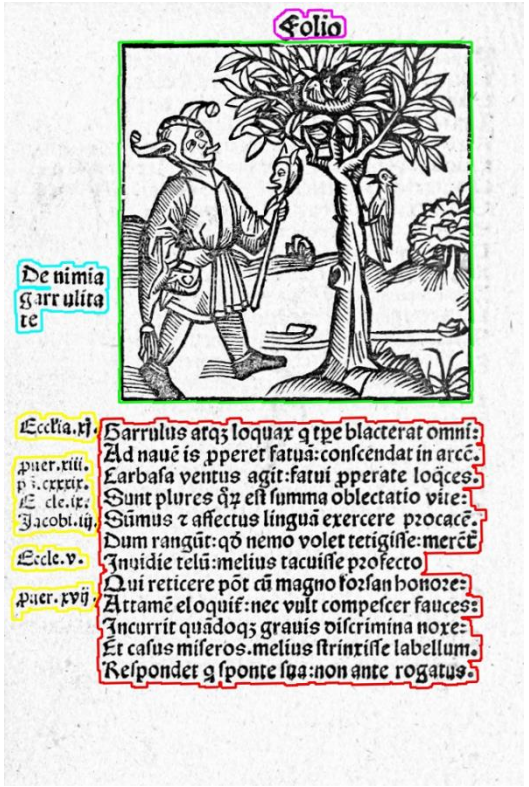
- A short user manual is available here.

Workflow – Region Extraction



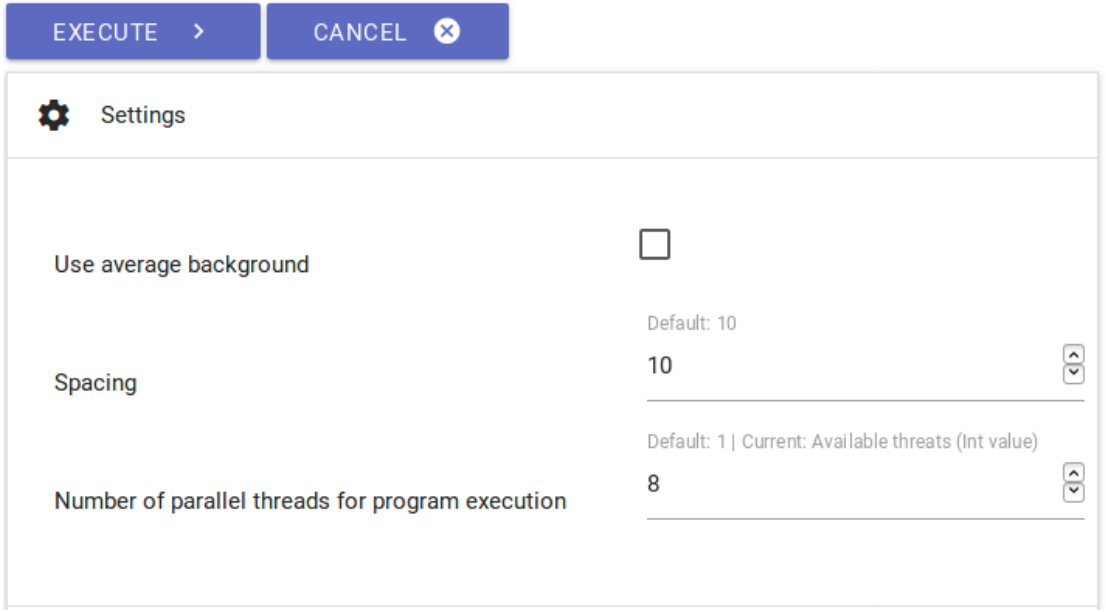
Region Extraction – Overview

- **Input:** binary/grayscale image and segmentation information (PageXML).
  Output: cut out text segments.
- Region polygons are extracted and copied into new image → no noise from adjacent regions.
- After the extraction the images are individually deskewed using ocropus-nlbin.
- Globally unique segment identifier:
  *page-id*__*reading order index*__*segment type*.png
  e.g. 0003__001__heading.png

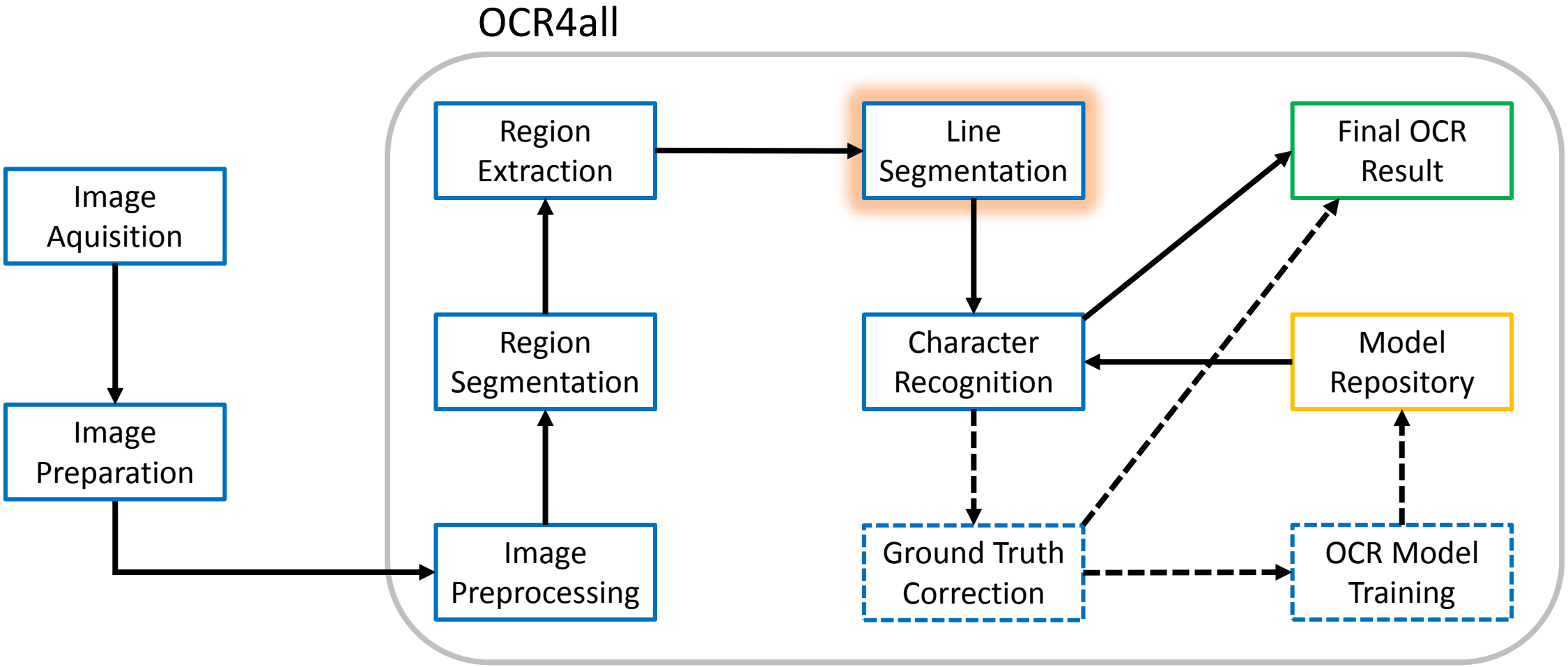


Hands-on: Region Extraction

- Book: “Cirurgia”, image type: “gray”.
- Go to “Region Extraction”.
- Keep all settings to the default:
  - 10 pixel spacing around the regions.
  - White background.
- Hit “Execute” and check the results in the “Project Overview”.

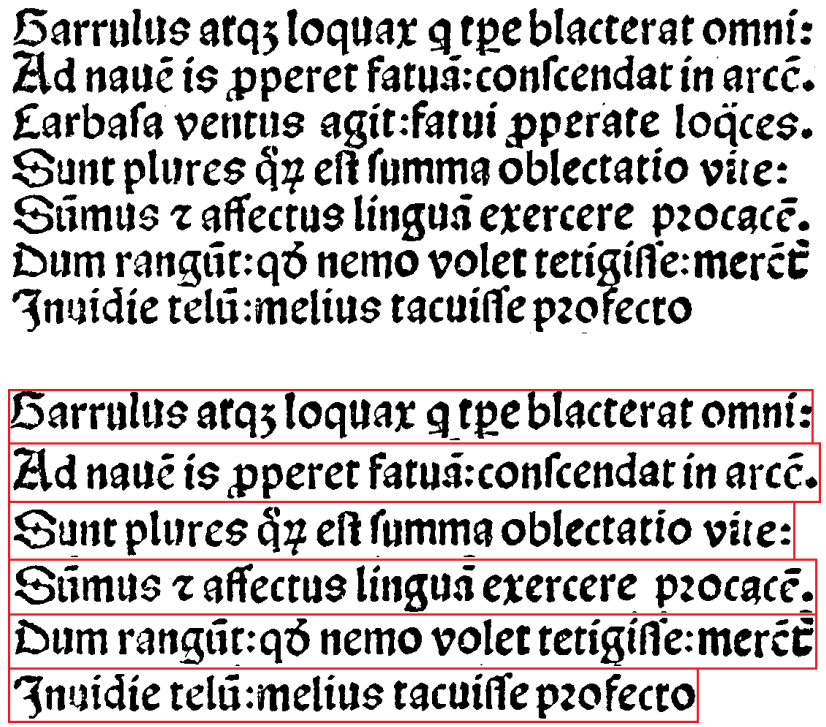


Workflow – Line Segmentation



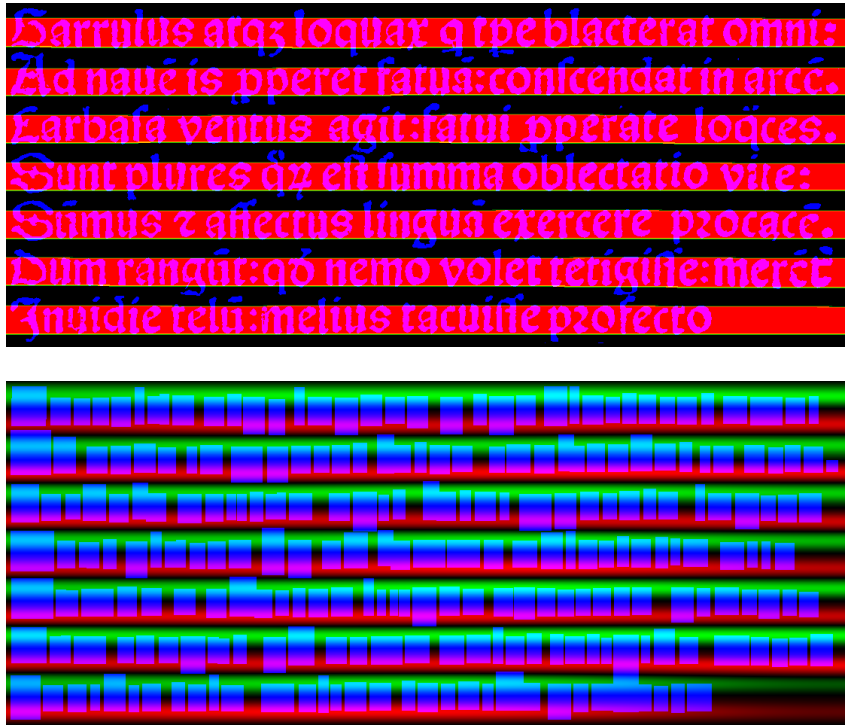
Line Segmentation – Overview

- **Input:** Preprocessed segment image.
  Output: Lines in reading order.
- Operates on binary images but can also output grayscale.
- Inevitable since the OCR operates on line level.
- Implemented in ocropus-gpageseg.
- Globally unique line identifier:
  *segment-id*__*line reading order index*.png
  e.g. 0003__001__paragraph__004.png



Line Segmentation – Algorithm

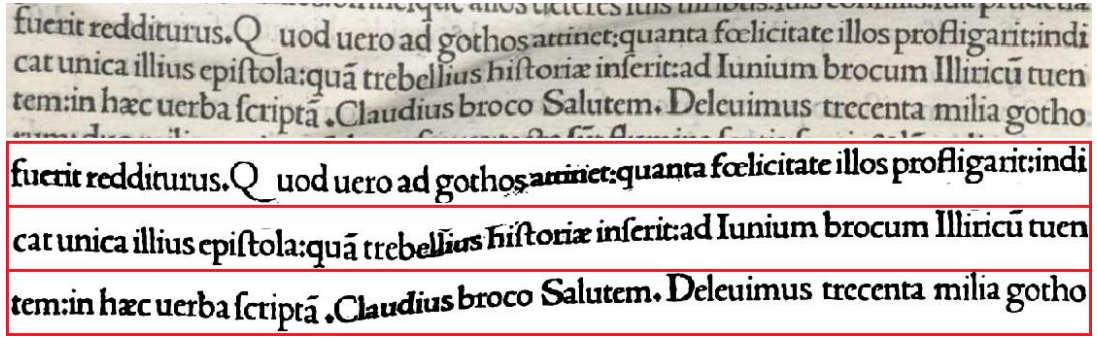
- Scale/x-height estimation.
- Exclusion of components which are unlikely to be letters (too small/big according to scale).
- Seed generation by horizontal blurring.
- Seed expansion and component extraction.
- Generation of final line by copying the components into a rectangle
  → no noise from adjacent lines.



Example of running gpageseg on a scan segment: results of the seed generation (top) and the horizontal blurring (bottom).

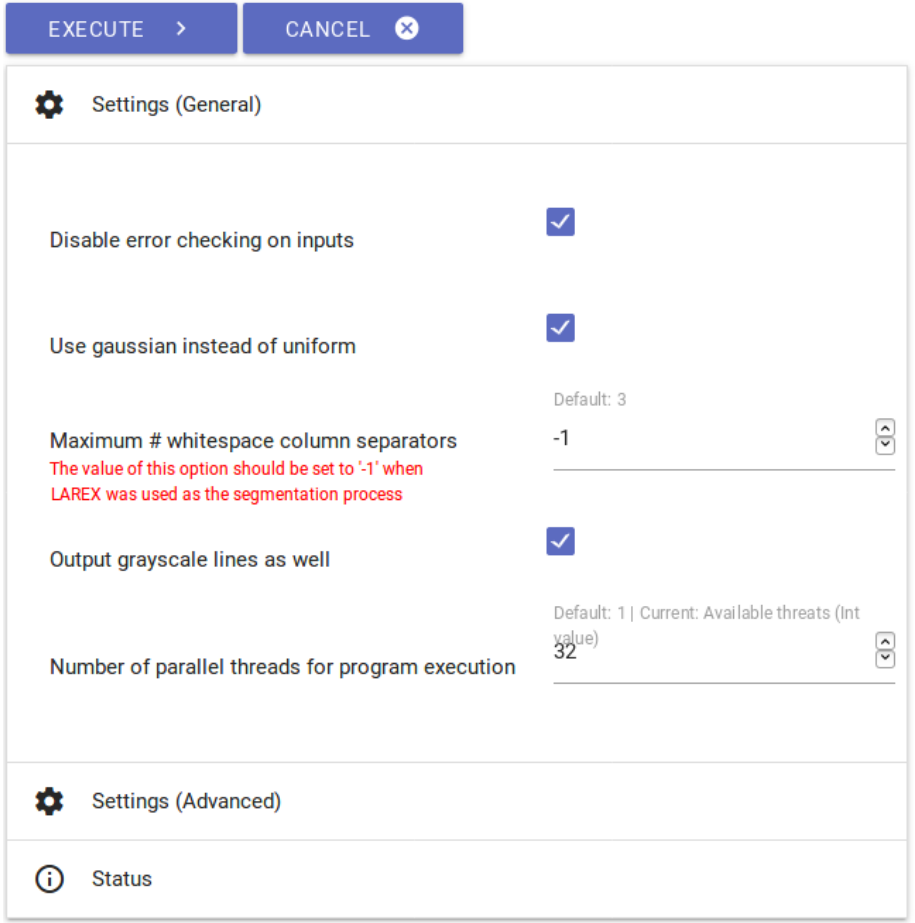
Line Segmentation – Conclusion

- Very robust on Latin scripts.
- Large variations of the x-height within a segment (title page!) can cause problems.
- In general, ocropus-gpageseg can also deal with entire pages:
  - Simple text/non-text segmentation.
  - Decent column detection.
  - Used for dummy segmentation.
- However,
  - complex layouts require preceding segmentation steps like region segmentation etc..
  - Works best when column detection is turned off.

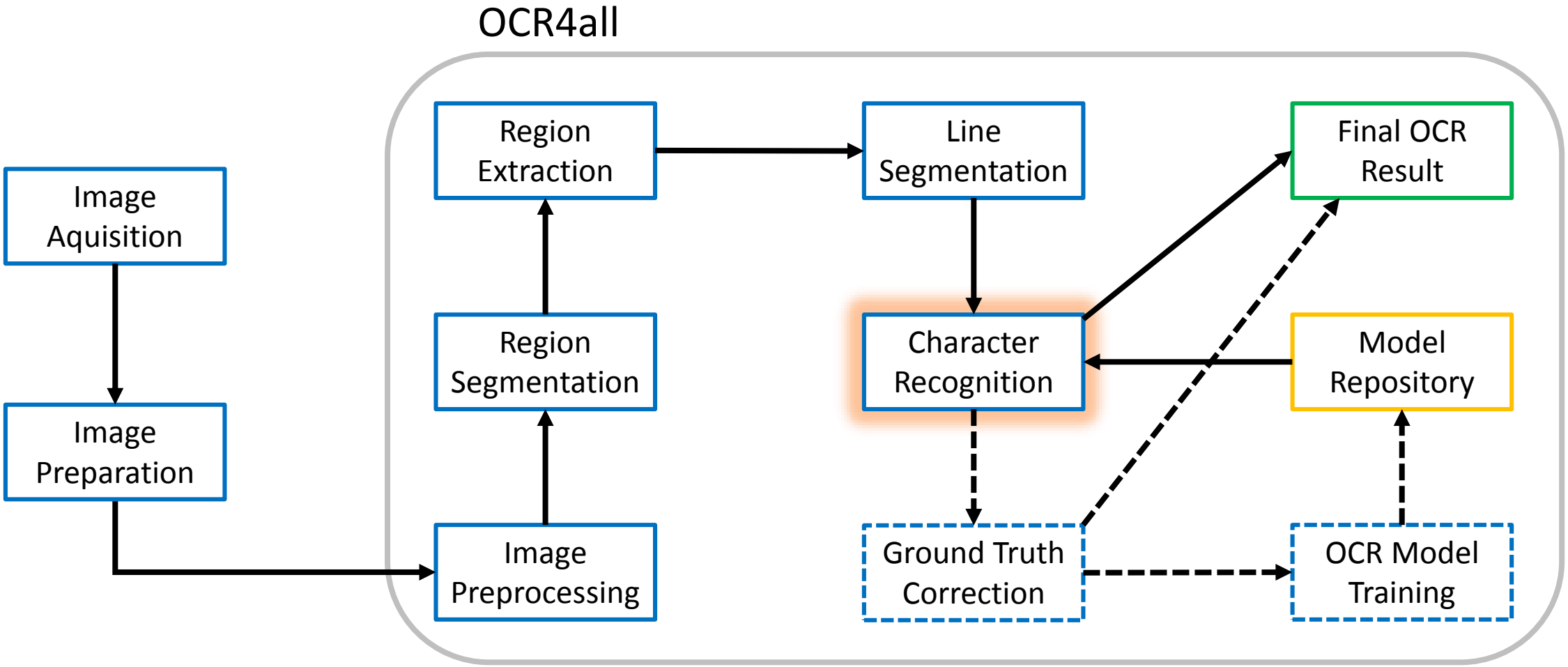


Hands-on: Line Segmentation

- Book: “Cirurgia”, image type: “gray”.
- Go to “Line Segmentation”.
- Keep all settings to the default except the “Maximum # of whitespace column separators” (3 → -1).
- Hit “Execute” and check the results
  - in the “Project Overview”.
  - by using “Ground Truth Correction”.

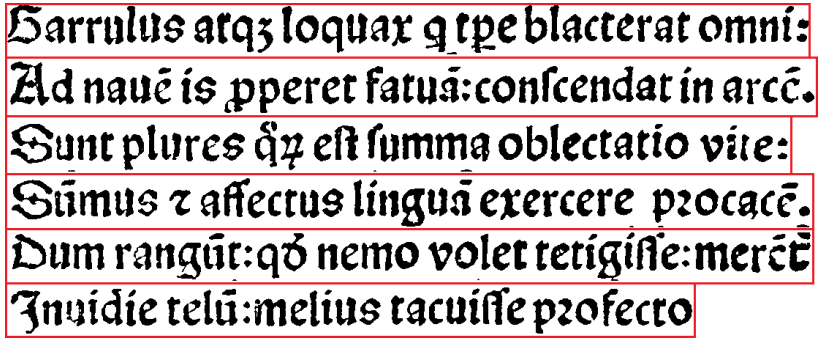


Workflow – Character Recognition



Character Recognition – Overview

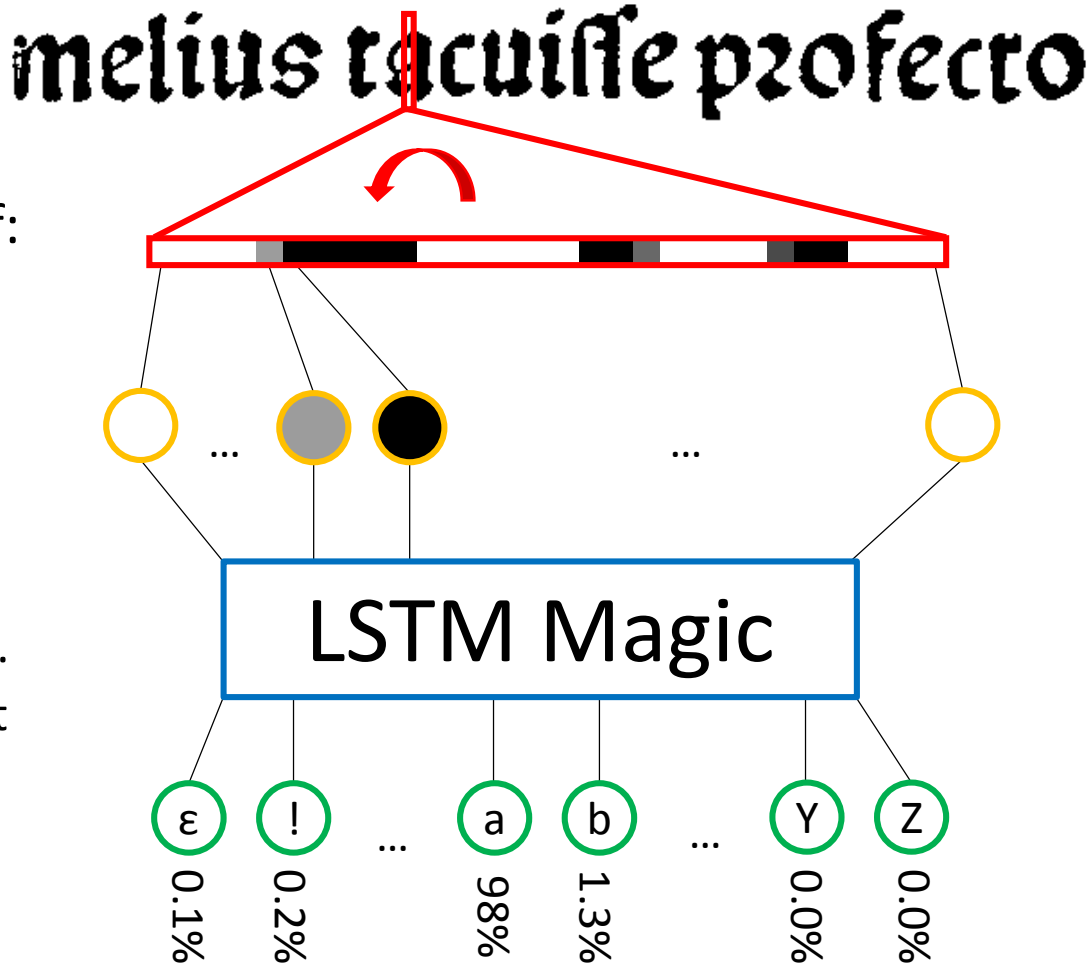
- **Input:** line images and an OCR model.
  Output: recognized text lines.
- Works on binary and grayscale images.
- Grayscale images can offer additional information.



Earrulus atqȝ loquax q Epe blacterat omni:
Ad nauē is pperet fatuā:conſcenda in arcē
Sunt plures qu eſt ſurmma oblectatio vite:
Sūmus ꝛ affectus linguſta exercere pꝛocacē.
Dum rangūt:qꝺ̆ nemo volet tetigiſſe:merct
Jnuidie telū:melius tacuiſſe pꝛofecto

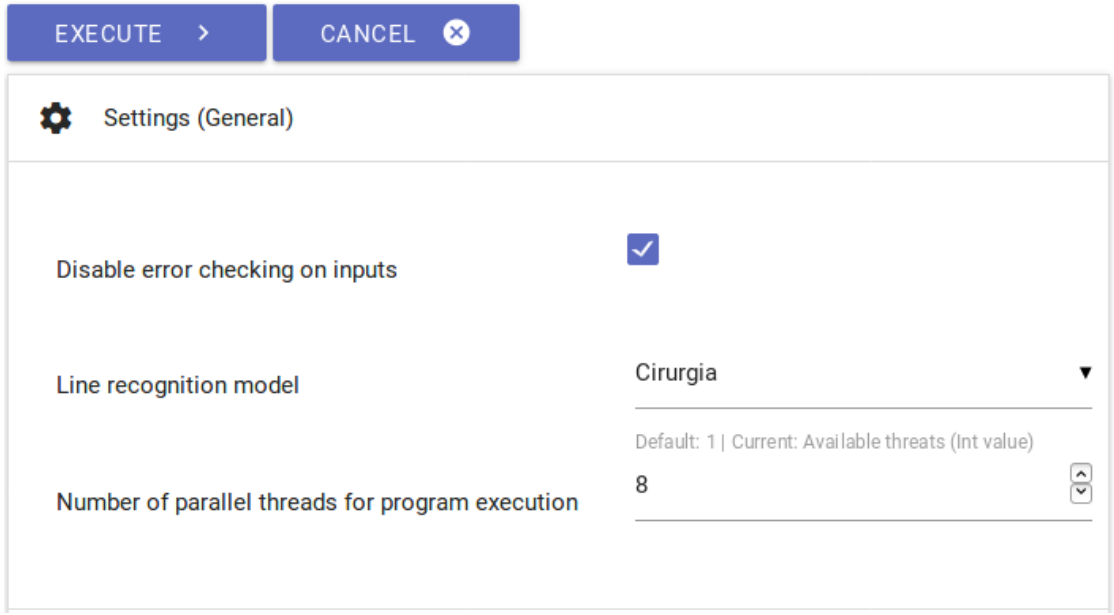
Character Recognition – Methodology

- Line image is normalized to a certain height (default: 48px) and cut into slices of 1px width.
- Model stores a LSTM Neural Network consisting of:
  - input nodes (# line height): pixel values of the slice.
  - A LSTM layer with hidden states.
  - output nodes (# codec size): characters in the codec.
- Recognition:
  - Input: one slice at a time.
    Output: probability distribution over the entire codec.
  - A line gets recognized from left to right and from right to left.
  - CTC algorithm combines the single probability distributions to generate the most likely output.

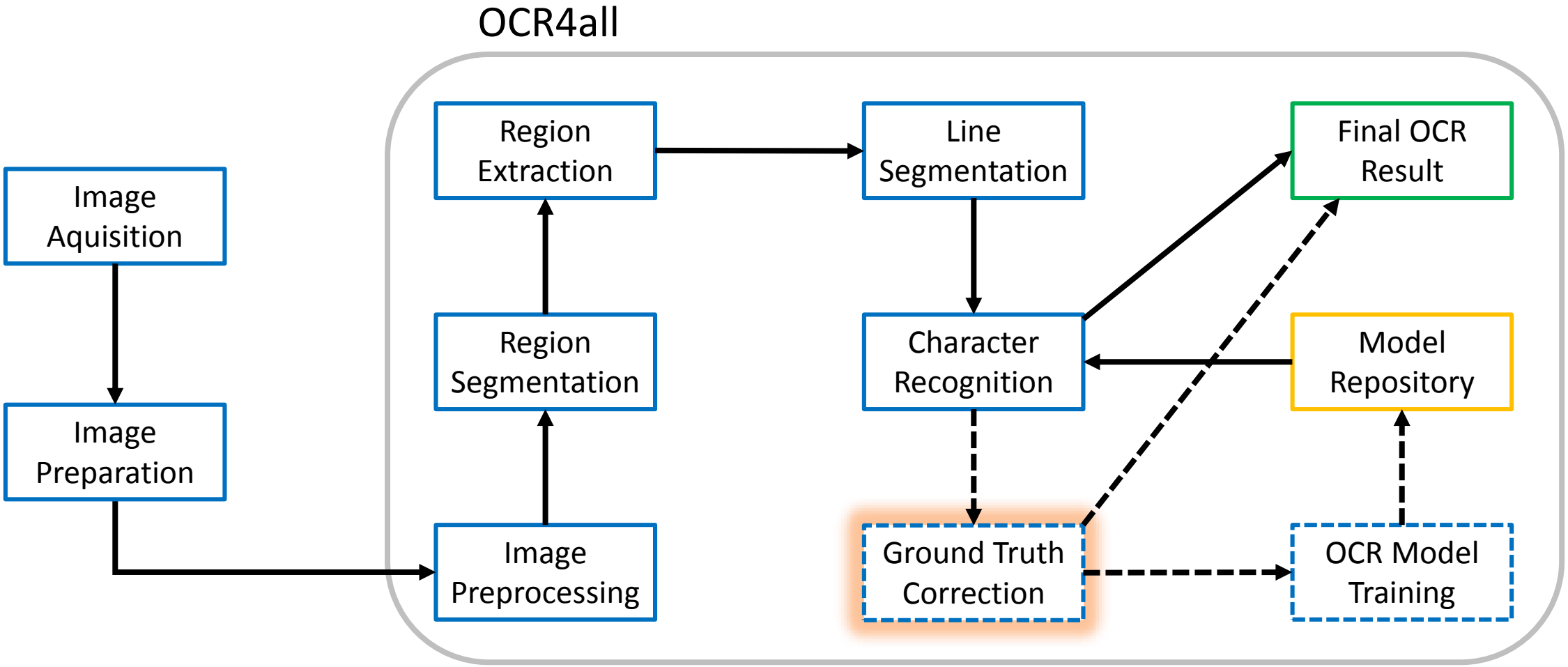


Hands-on: Character Recognition

- Book: “Cirurgia”, image type: “gray”.
- Go to “Recognition”.
- Keep all settings to the default except the “Line recognition model” (“Cirurgia”).
- Hit “Execute” and check the results by using “Ground Truth Correction”.

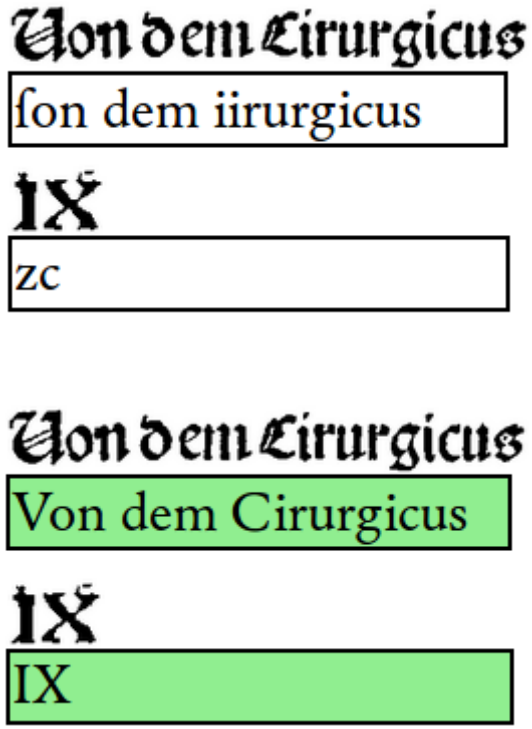


Workflow – Ground Truth Correction



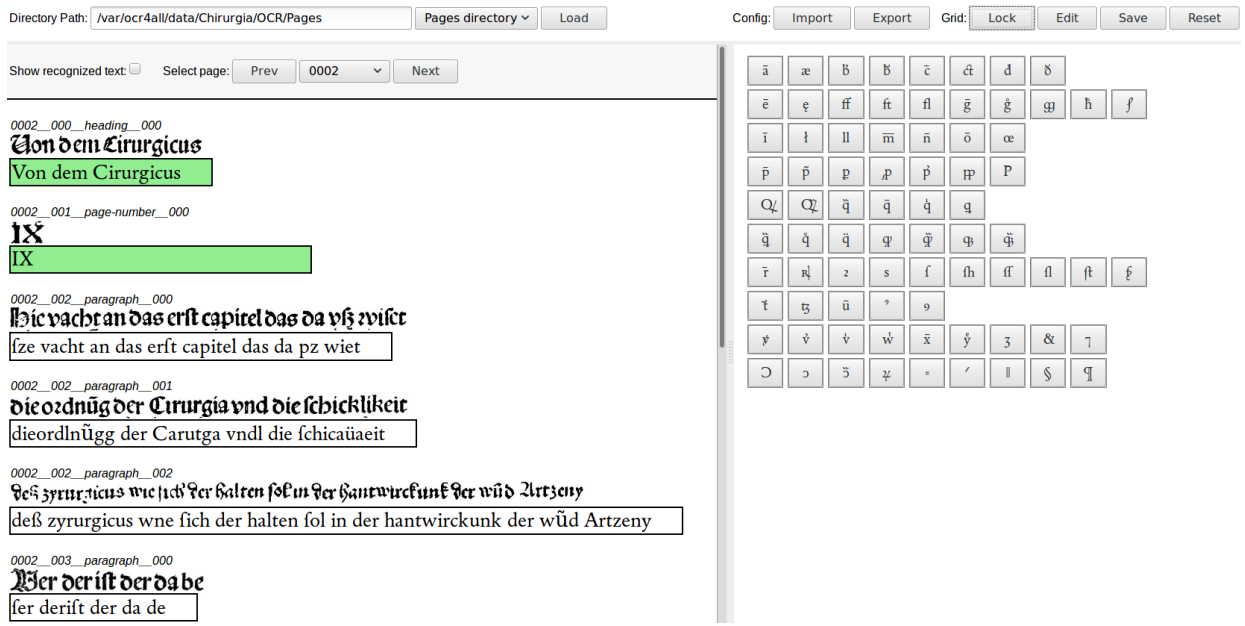
Ground Truth Correction – Overview

- **Input:** line image and OCR result (optional).
  Output: corrected text (= ground truth GT).
- Configurable virtual keyboard allows to insert special characters.
- Different line files connected via their IDs:
  - Binary: *line-id*.bin.png
  - Gray: *line-id*.nrm.png
  - OCR: *line-id*.txt
  - GT: *line-id*.gt.txt



Hands-on: Ground Truth Correction

- Book: “Cirurgia”, image type: “gray”.
- Go to “Ground Truth Correction”.
- Select a line by clicking the OCR text.
- Make corrections (if necessary) by
  - typing regularly.
  - selecting characters from the virtual keyboard on the right.
- GT is saved (green background) when the line gets deselected.
- In view of the following step: Please make sure to at least save some (pseudo) corrections!



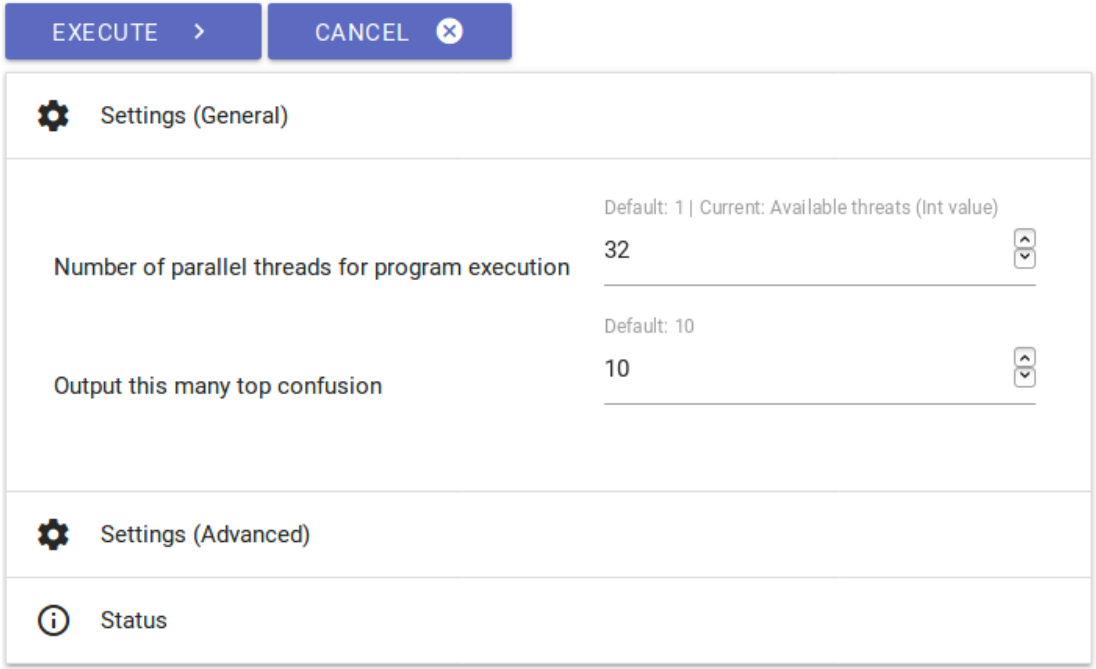
Evaluation – Overview

- **Input:** line-based OCR and GT pairs.
  Output: error statistics.
- Character Error Rate (CER) = number of errors / length of GT
- Three different error types:
  - **Insertions**: character(s) present in OCR but not in GT.
  - **Deletions**: character(s) present in GT but not in OCR.
  - **Substitutions**: character(s) recognized as (an)other one(s).
- Implemented in ocropus-econf:
  - Three columns: frequency | OCR result | GT
  - "_" marks an insertion.

```
errors          545
missing           0
total         43933
err           1.241 %
errnomiss     1.241 %
24
        _
15      n       u
14
        _       ū
13      ū       i
12      _
11      n       _
10      ū       ā
9       u       n
8       c       e
8       e       c
```

Hands-on: Evaluation

- Book: “Cirurgia”, image type: “gray”.
- Go to “Evaluation”.
- Keep all settings to the default and hit “Execute”.
- Change the “Context for confusion matrix” to 10 and go again.



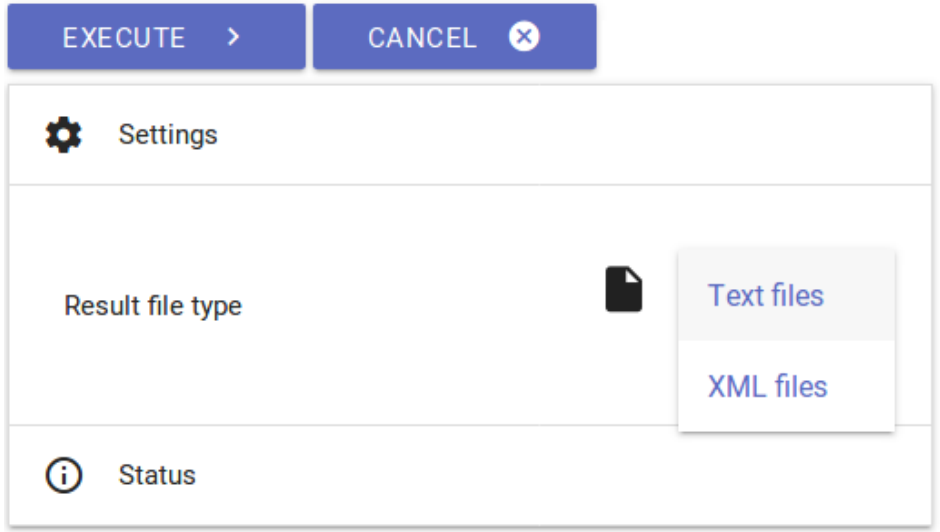
Result Generation – Overview

As of now, support of two output formats:

- Plain text:
  - **Input:** (corrected) OCR results on a line basis.
    Output: (corrected) OCR results combined to pages and the entire book.
  - Uses GT (if present) or OCR.
- PageXML:
  - **Input:** deskewing angles, segmentation data (region locations and types), line coordinates, OCR results, GT.
    Output: PageXML files on a page basis.
  - Stores region and line information like position and type.
  - Can store OCR and GT.
  - Is used as input for the OCR viewer (to be discussed).

Hands-on: Result Generation

- Book: “Cirurgia”, image type: “gray”.
- Go to “Result Generation”.
- Run the extraction using both available “Result file types”.



Short and Long Term Goals

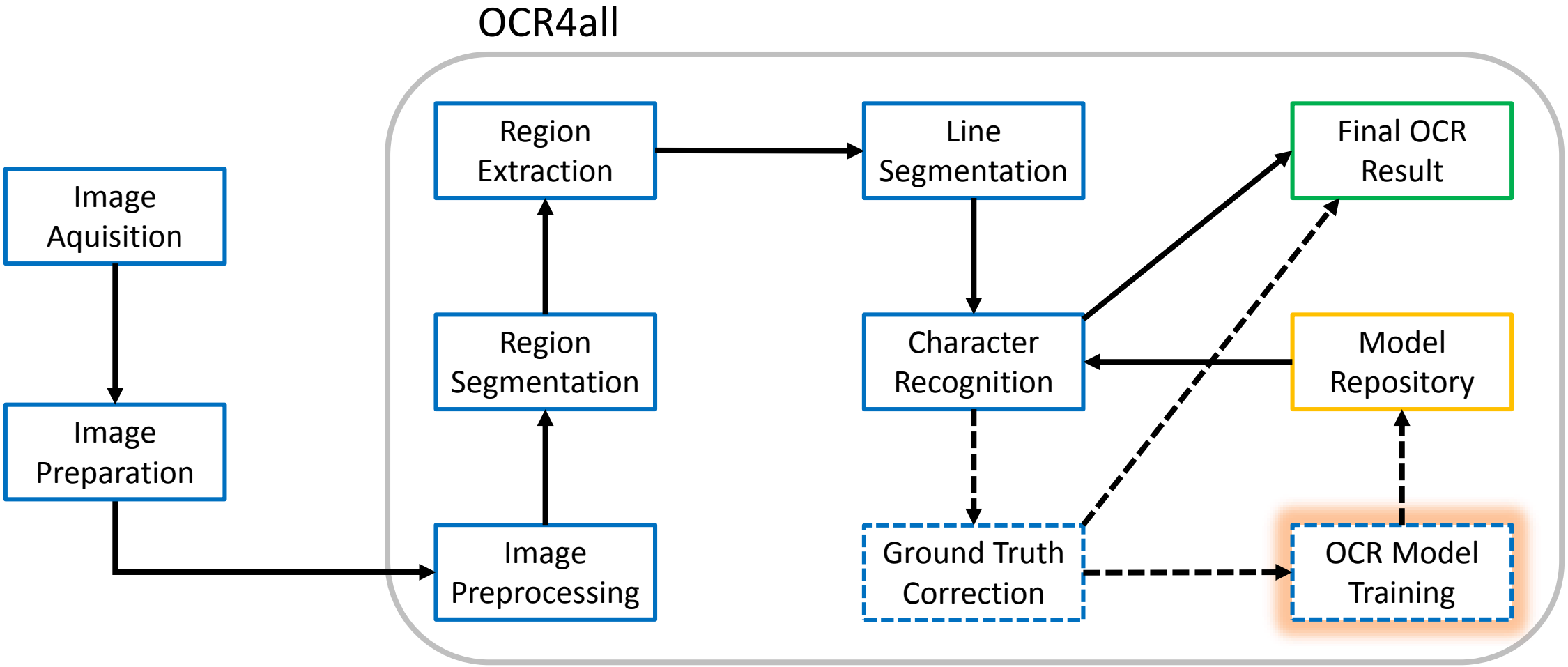
(Very) soon to be integrated:

- Book specific model training.
- Pretraining using existing models.
- Combination of several OCR results by voting.
- OCRopus clone Calamari.
- Pixel Classifier for region segmentation.
- OCR Viewer for result presentation.

(Proven) concepts and ideas:

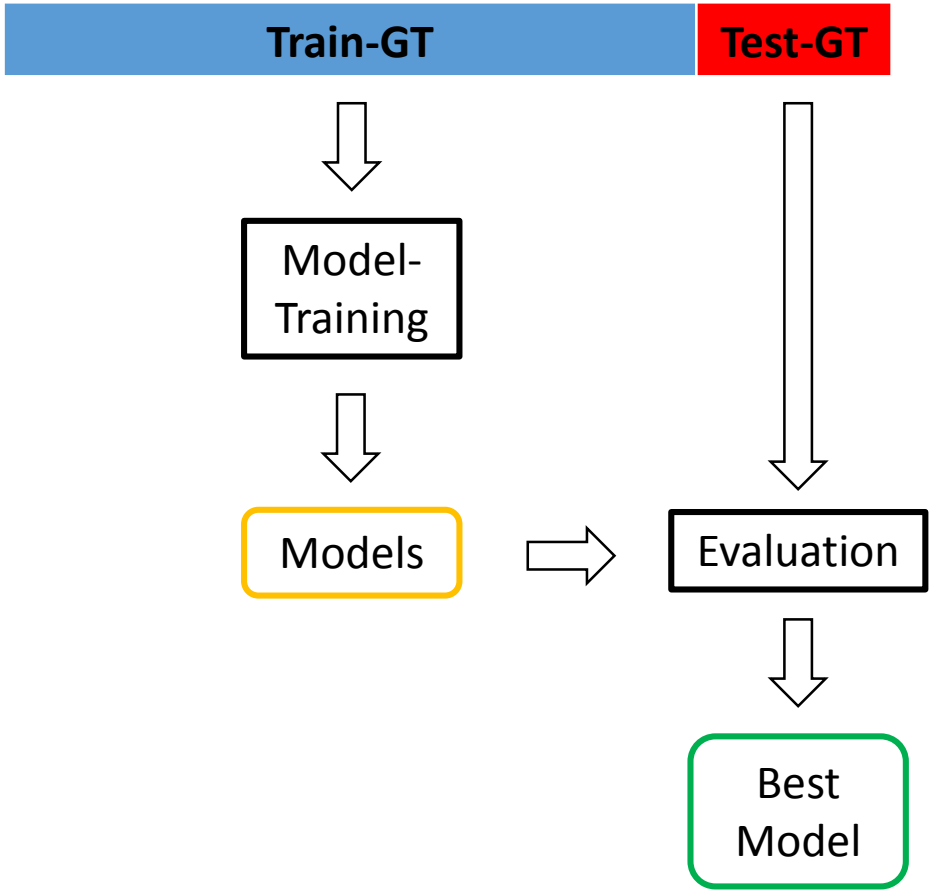
- Integrating Abbyy results.
- Incorporating postcorrection (dictionaries, language models, …).

Workflow – OCR Model Training



OCRopus Model Training

- **Input:** line-based GT (image/transcription pairs).
  Output: OCRopus model.
- General methodology:
  - Random initialization of the network weights.
  - Network predicts a line.
  - Compare result to GT and update weights accordingly.
  - Repeat prediction and update step.
- Training procedure:
  - Allocation of the available GT in training and test set.
  - Training of multiple models using the training set.
  - Determining the best model:
    - Apply each model to the test set.
    - Choose the model with the lowest error rate.

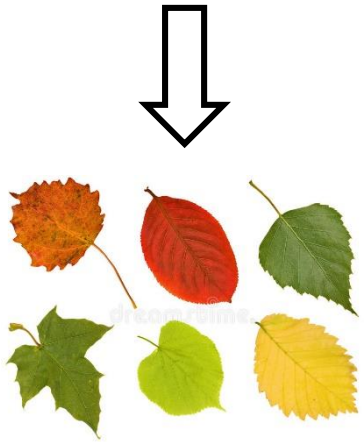


Pretraining

- Basic Idea: train a Neural Network on a similar domain with different data and perform the „real“ training afterwards.
- Hope: profit from learning general features.

Example: leaf species classification

Pretraining



Real training

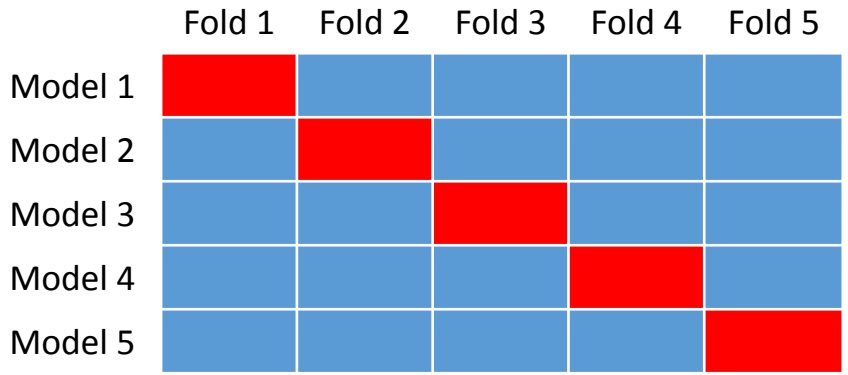
Application to OCR (on early printed books):

- Start training from standard models like LH, FRK, or ENG (part of OCR4all).
- Reul, Wick, Springmann, Puppe: Transfer Learning for OCRopus Model Training on Early Printed Books.

Voting – General Idea and Cross Fold Training

- Combining the results of several strong but different models improves accuracy.
- How to obtain different voters using only a single engine (OCRopus)?
- Cross fold training:
  - Divide the GT in N distinct folds.
  - Train N models:
    - Define one fold as test set.
    - Perform a training using the N-1 remaining folds.
    - Select the best model by evaluating on the test set.

```
GT: inde marien namen
---------------------
M1: inide maricn namen
M2: inde maricn namen
M3: inde marien namen
M4: iade marien namen
M5: inde maricn namen
```



Voting – Combining the Results

- Each model/voter recognizes each line.
- Aligning the results on line level:

$$\text{i}\begin{Bmatrix}\text{ni}\\ \text{n}\\ \text{n}\\ \text{a}\\ \text{n}\end{Bmatrix}\text{de marie}\begin{Bmatrix}\text{c}\\ \text{c}\\ \text{e}\\ \text{e}\\ \text{c}\end{Bmatrix}\text{n namen}$$

- Resolving the disagreements by confidence voting using the intrinsic OCRopus confidence values.
- Reul, Springmann, Wick, Puppe: Improving OCR accuracy on Early Printed Books by Utilizing Cross Fold Training and Voting.

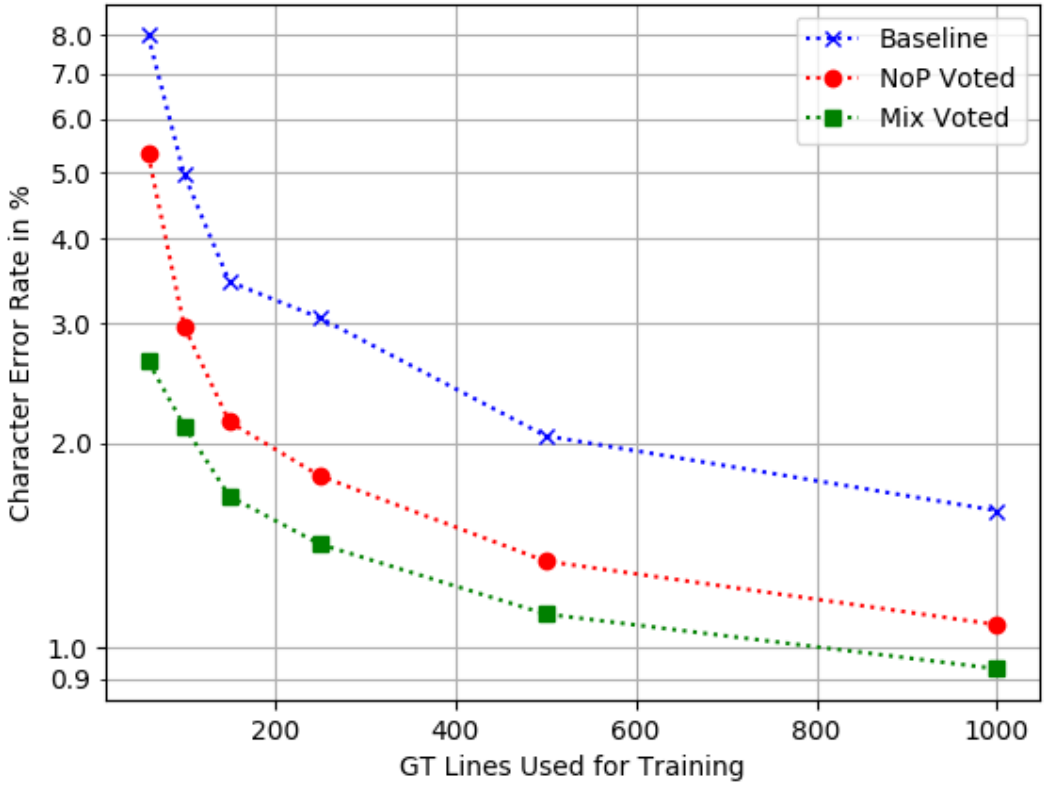
Char	x_S	x_E	Conf	Alternatives
i	120	123	87.54%	b=8.66%, f=2.94%
a	126	136	96.65%	n=45.78%, r= 23.65%, m=9.24%, k=8.32%, [...]
d	142	149	99.93%	ã=4.83%, V=4.17%, O=1.13%
e	155	160	99.15%	all alternatives $< 1\%$

$$[\ldots]\ \text{marie}\begin{Bmatrix}\text{c} = 67\%, \text{e} = 38\%\\ \text{c} = 93\%, \text{e} = 20\%\\ \text{c} = 0\%, \text{e} = 99\%\\ \text{c} = 8\%, \text{e} = 98\%\\ \text{c} = 90\%, \text{e} = 50\%\end{Bmatrix}\text{n namen}$$

actual output, alternative

Combination of Pretraining and Voting

- Evaluated on six early printed books.
- Pretrained Voting (●) on average produces 53% less errors than the standard OCRopus approach (Baseline ●).
- Also applicable to mixed/polyfont models → no (book specific) training required.
- Reul, Springmann, Wick, Puppe: Improving OCR Accuracy on Early Printed Books by Combining Pretraining, Voting, and Active Learning.



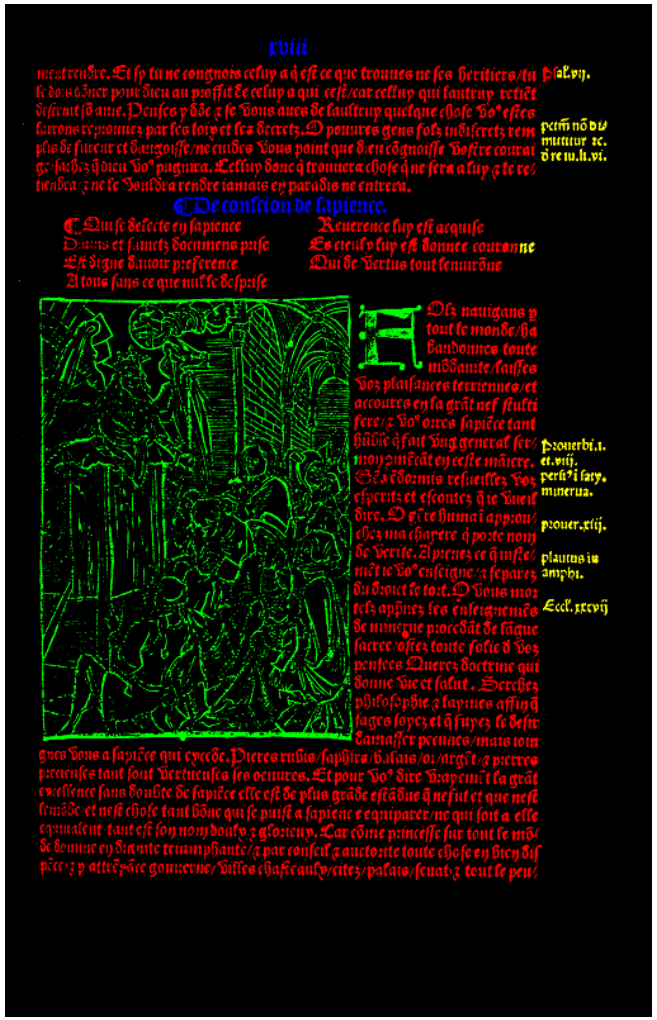
Calamari

- Open source OCRopus clone by Christoph Wick.
- Alternative to/replacement for OCRopus in OCR4all.

- Native integration of confidence voting and pretraining.
- Usage of deeper neural networks → better recognition results.
- GPU support → much faster recognition and training.
- Many minor tweaks like enhanced modularity and cleaned up code.

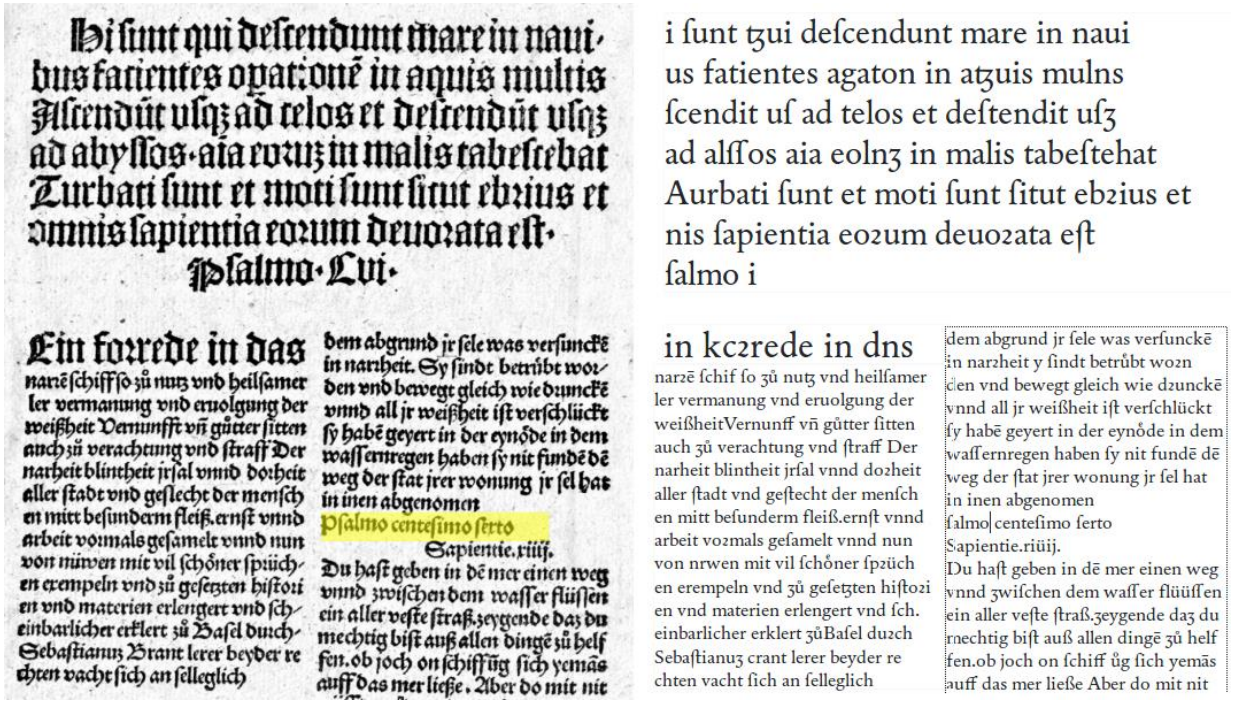
Segmentation – Pixel Classifier

- Fully Convolutional Network (FCN):
  - Predicts a label for every pixel.
  - Trained on images and segmentation masks as GT.
- Different manifestations:
  - Fine grained semantic distinction:
    - Can deal with complex layouts.
    - Requires book specific GT, e.g. 5-20 pages segmented using LAREX.
  - Text/Image separation only:
    - Generic models applicable to a variety of prints.
    - Trained on a huge amount of example pages.
- Post-Correction possible by using LAREX.
- Wick and Puppe: [Fully Convolutional Neural Networks for Page Segmentation of Historical Document Images](#).



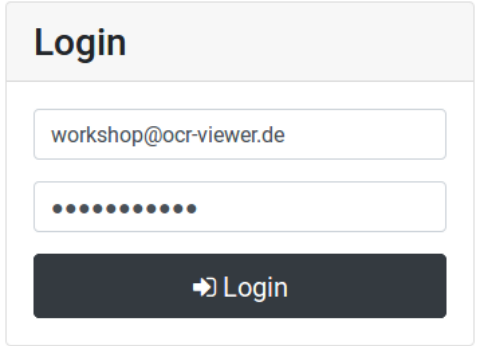
OCR Viewer

- Synopsis of scan and OCR based on the PageXML data.
- Already many features:
  - Region alignment.
  - Highlighting of corresponding lines.
  - Simple post correction.
  - User administration.
  - …
- Work in progress:
  - Integrated annotation tools.
  - Native integration into OCR4all.
  - …



Hands-on: OCR-Viewer

- Open Firefox and go to http://ocr-viewer.informatik.uni-wuerzburg.de.
- Login:
  - User: workshop@ocr-viewer.de
  - Password: workshop123
- Choose the book “Das Narrenschiff: GW5049” and hit “view”.
- Clear the cache (ctrl+shift+r) if a page isn’t displayed correctly.



Integrating Abbyy Results – Abbyy XML

- Optionally, Abbyy XML stores a lot of additional information.
- Some of it is in fact very useful:
  - Image and text region coordinates.
  - Line and character coordinates.
  - Recognition attributes like
    - Character confidence.
    - Word from dictionary?
    - Suspicious character?
    - …

```
−<line baseline="323" l="118" t="290" r="1084" b="332">
  −<formatting lang="OldGerman" ff="Arial" fs="8.">
      <charParams l="118" t="300" r="141" b="325" wordFirst="1" wordLeftMost="1"
      wordFromDictionary="1" wordNormal="1" wordNumeric="0" wordIdentifier="0"
      charConfidence="100" serifProbability="255" wordPenalty="0"
      meanStrokeWidth="36">w</charParams>
      <charParams l="144" t="303" r="155" b="324" wordFromDictionary="1"
      wordNormal="1" wordNumeric="0" wordIdentifier="0" charConfidence="29"
      serifProbability="255" wordPenalty="0" meanStrokeWidth="36">e</charParams>
      <charParams l="159" t="293" r="168" b="324" wordFromDictionary="1"
      wordNormal="1" wordNumeric="0" wordIdentifier="0" charConfidence="83"
      serifProbability="255" wordPenalty="0" meanStrokeWidth="36">l</charParams>
      <charParams l="171" t="292" r="195" b="332" wordFromDictionary="1"
      wordNormal="1" wordNumeric="0" wordIdentifier="0" charConfidence="70"
      serifProbability="255" wordPenalty="0" meanStrokeWidth="36">c</charParams>
      <charParams l="171" t="292" r="195" b="332" wordFromDictionary="1"
      wordNormal="1" wordNumeric="0" wordIdentifier="0" charConfidence="70"
      serifProbability="255" wordPenalty="0" meanStrokeWidth="36">h</charParams>
      <charParams l="198" t="301" r="210" b="324" wordFromDictionary="1"
      wordNormal="1" wordNumeric="0" wordIdentifier="0" charConfidence="100"
      serifProbability="255" wordPenalty="0" meanStrokeWidth="36">e</charParams>
```

Integrating Abbyy Results – Opportunities I

- Considering OCR results for voting:
  - Abbyy's recognition strengths and weaknesses (probably) differ considerably from OCRopus'
    → different types of errors → voting is promising.
  - Problem: Abbyy XML confidence values often inexplicable.
  - Attributes like "word from dictionary" and "suspicious" helpful.
- Making use of the very robust segmentation:
  - Even if the OCR failed the line segmentation might still be very usable.
  - Extract the lines and use OCRopus from there!

Integrating Abbyy Results – Opportunities II

- Using the Internet Archive as a free resource.
  - Provides a vast amount of scanned books.
  - Many books have been processed using Abbyy.
    - "Default" recognition settings → (depending on the book) OCR might be useless.
    - But high quality segmentation still available, extractable, and usable!
    - See Archiscribe by Johannes Baiter (available at GitHub).
- Fully automatic book specific OCRopus training using Abbyy pseudo GT.
  - Extract words Abbyy is quite sure about and construct pseudo GT from it.
  - Run a book specific training.
  - Example pseudo GT:

Erinnerung! einsehen, Zufriedenheit Herzog? erkämpfen?"

Thank you for your Attention!

Acknowledgements:

Dennis Christ (OCR4all web app)

Alexander Hartelt (OCR4all web app)

Nico Balbach (LAREX web app)

Uwe Springmann (ideas and feedback)

Christoph Wick (Calamari, pixel classifier, constant nagging)

Andreas Büttner (PageXML export)

Stefan Olbrecht (OCR-Viewer)

…