OCR4all – An Open Source Tool Providing a Full OCR Workflow

Christian Reul

Centre for Philology and Digitality (ZPD)
Chair for Artificial Intelligence and Applied Computer Science
University of Würzburg

24.09.2019

Agenda

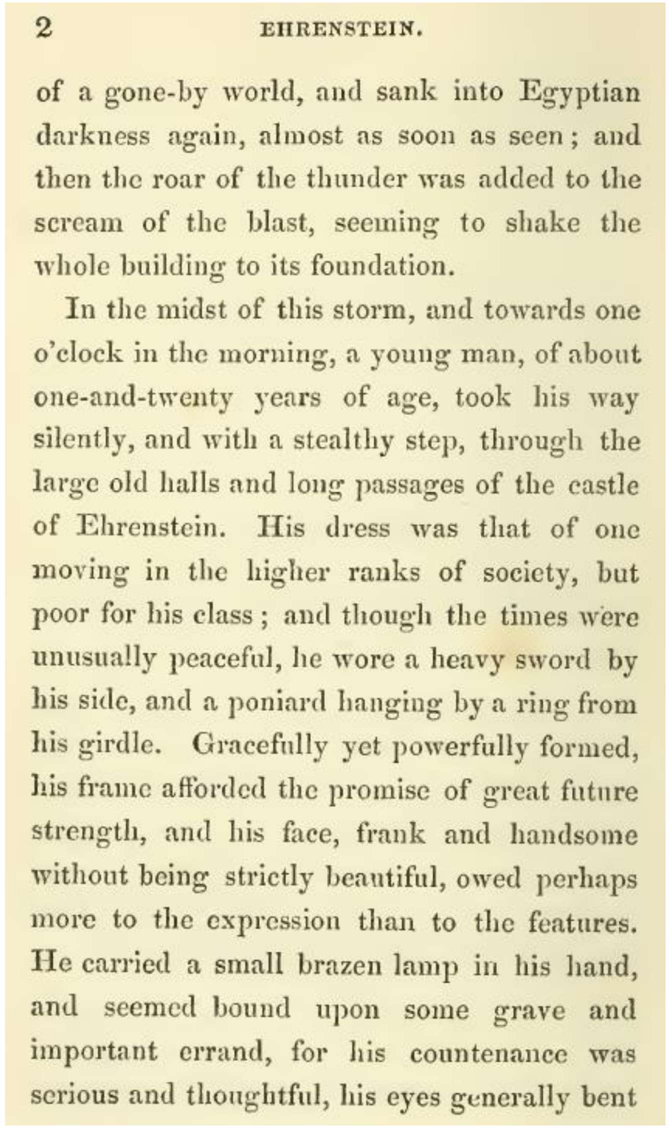
1. OCR Basics
2. OCR Engines ABBYY Finereader and OCR4all
3. OCR4all Hands-on Guided
4. OCR4all Hands-on Playground

Agenda

1. **OCR Basics**
2. OCR Engines ABBYY Finereader and OCR4all
3. OCR4all Hands-on Guided
4. OCR4all Hands-on Playground

What is OCR?

- **O**ptical **C**haracter **R**ecognition.
- One of the oldest Computer Vision tasks.
- Goal: Extract machine-actionable text from scanned images.
  - Searchable.
  - Annotatable.
  - …



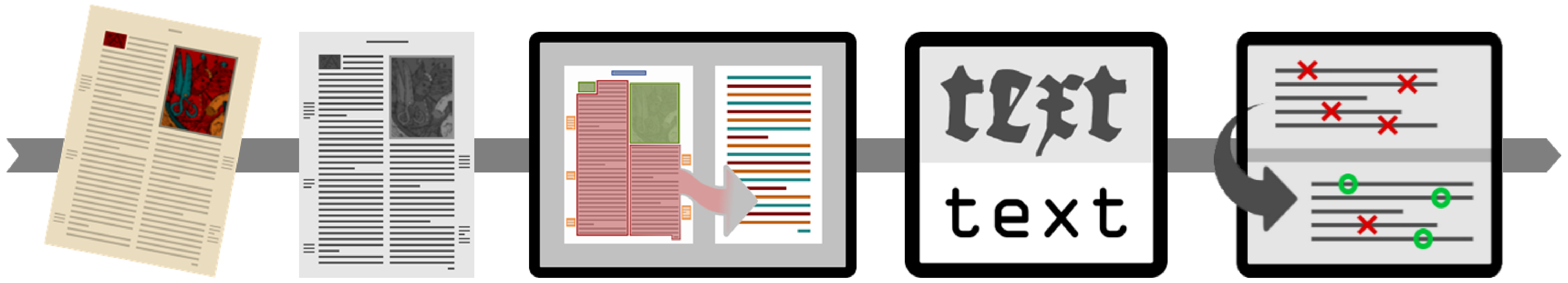
2 EHRENSTEIN.

of a gone-by world, and sank into Egyptian darkness again, almost as soon as seen; and then the roar of the thunder was added to the scream of the blast, seeming to shake the whole building to its foundation.

In the midst of this storm, and towards one o'clock in the morning, a young man, of about one-and-twenty years of age, took his way silently, and with a stealthy step, through the large old halls and long passages of the castle of Ehrenstein. His dress was that of one moving in the higher ranks of society, but poor for his class; and though the times were unusually peaceful, he wore a heavy sword by his side, and a poniard hanging by a ring from his girdle. Gracefully yet powerfully formed, his frame afforded the promise of great future strength, and his face, frank and handsome without being strictly beautiful, owed perhaps more to the expression than to the features. He carried a small brazen lamp in his hand, and seemed bound upon some grave and important errand, for his countenance was serious and thoughtful, his eyes generally bent

2 EIIRENSTEIN.
of a gone-by world, and sank into Egyptian
darkness again, almost as soon as seen; and
then the roar of the thunder was added to the
scream of the blast, seeming to shake the
whole building to its foundation.
In the midst of this storm, and towards one
o’clock in the morning, a young man, of about
one-and-twenty years of age, took his way
silently, and with a stealthy step, through the
large old halls and long passages of the castle
of Ehrenstein. His dress was that of one
moving in the higher ranks of society, but
poor for his class; and though the times were
unusually peaceful, he wore a heavy sword by
his side, and a poniard hanging by a ring from
his girdle. Gracefully yet powerfully formed,
his frame afforded the promise of great future
strength, and his face, frank and handsome
without being strictly beautiful, owed perhaps
more to the expression than to the features.
He carried a small brazen lamp in his hand,
and seemed bound upon some grave and
important errand, for his countenance was
serious and thoughtful, his eyes generally bent

OCR Basics – Workflow and Models



- Main steps: preprocessing, segmentation, OCR, postcorrection.
- Modern OCR approaches operate on line instead of character level.
- So-called *models* extract text from line images.
  - Can be trained ((Deep) Neural Networks).
  - Require training data consisting of image/text pairs.

Er wird eifrig geſammelt.

Er wird eifrig geſammelt.

OCR Basics – Mixed Models vs. Book-Specific Models

- Models perform best on data similar to what they were trained on.
- Mixed/Polyfont/Omnifont models:
  - Usually trained on a wide variety of sources and types.
  - Works best on newer prints (ca. 1800+) with somewhat regular typography.
  - Advantage: can be applied out of the box, without book specific training.
  - Disadvantage: not as accurate as book-specific models.
- Book-specific models:
  - Usually trained exclusively for the recognition of one specific book or type.
  - Often inevitable for older works (before 1800).
  - Advantage: in general considerably more accurate than mixed models.
  - Disadvantage: requires ground truth, i.e. transcriptions of text lines.

OCR Basics – Workflow Tools

- Many tools comprising a (more or less) full OCR workflow available.
  - Open-source/free and proprietary/fee-based.
  - Different strengths and weaknesses.
  - Tesseract, OCRopus, Omnipage, …
- In the following we focus on ABBYY Finereader and OCR4all.

ABBYY FineReader

Agenda

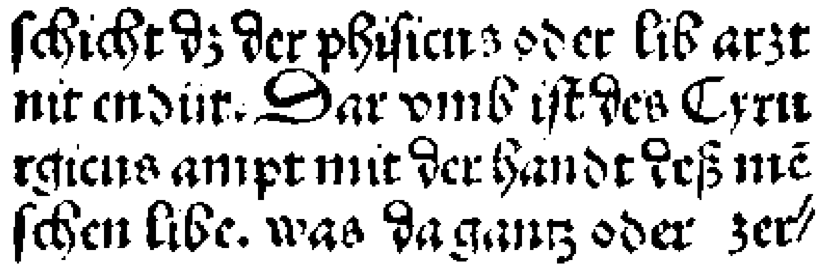
1. OCR Basics
2. **OCR Engines ABBYY Finereader and OCR4all**
3. OCR4all Hands-on Guided
4. OCR4all Hands-on Playground

ABBYY Finereader – Overview

- Undisputed commercial leader in document analysis and recognition.
- Clear focus on contemporary material and fully automatic mass processing.
  - Bills, contracts, business cards, … !
  - But also books, newspapers, …
- Very impressive from a purely technical point of view:
  - Rapid, robust, and precise.
  - Rich output (paragraphs, page numbers, italics, …) in many formats (.docx, .xml, .pdf, …).
- Costs money.
  - Different licensing models (**P**ages **P**er **Y**ear/**M**onth, **T**otal **P**age **C**ount, …).
  - Different technical solutions (Recognition Server, Linux CLI, Cloud SDK).
- Proprietary code (closed source).

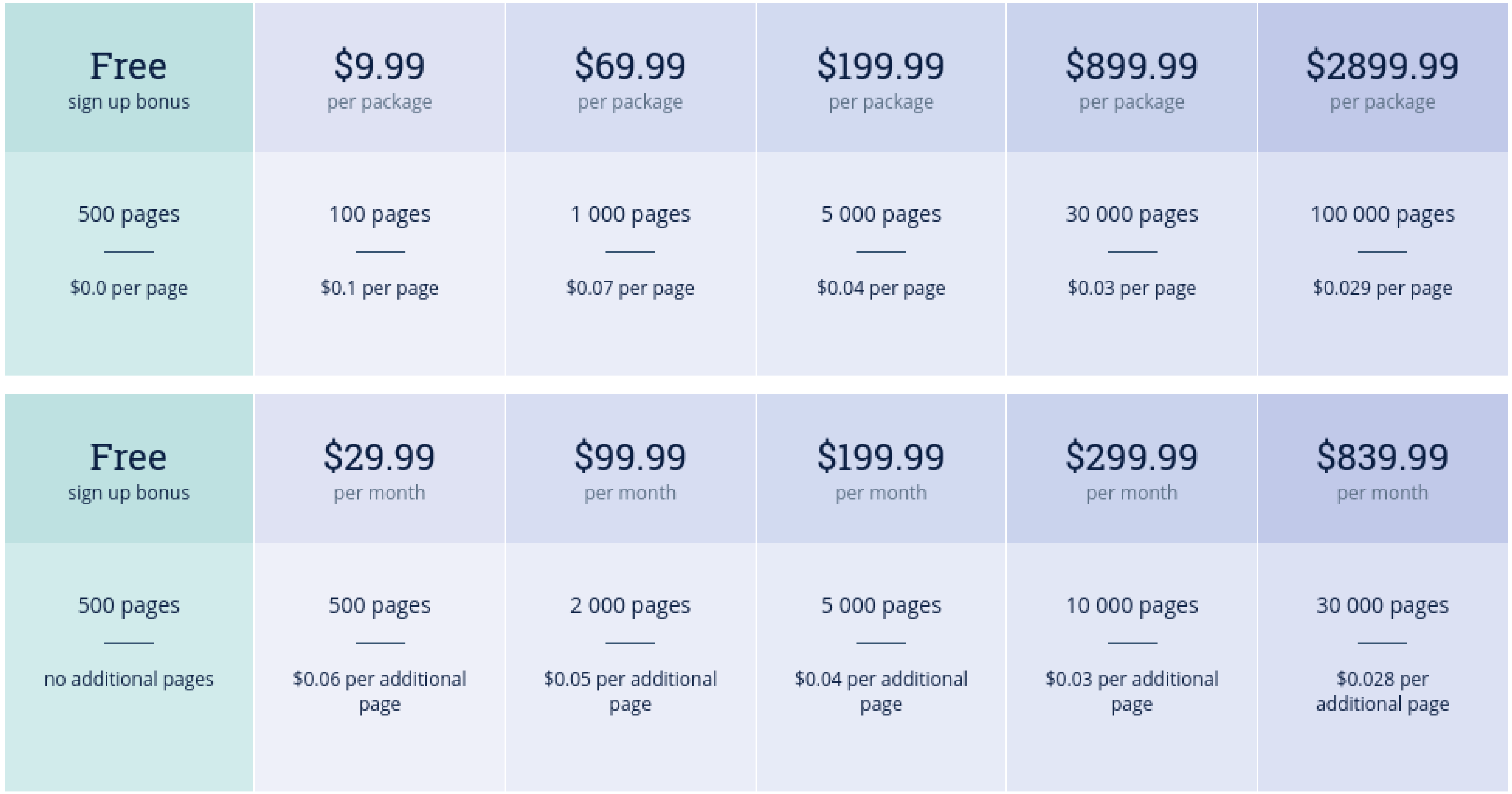
ABBYY Finereader – OCR on Historical Printings

- Comparatively poor support of historical printings.
  - Mainly 19th century and later.
  - Mostly useless for (very) early printed books.
  - Support of just six historical languages (in terms of dictionaries, language models, …).
  - Individual font training possible but laborious and ineffective.
- However, still very precise under certain circumstances:
  - Antiqua font (Fraktur/Gothic model does not seem as strong).
  - Decent print quality and state of preservation.
  - Language properties fit the „modern“ model, ...


ſchickt dz der phiſicus oder lib arzt
nit endiit. Dar vmb iſt des Cyru
rgicus ampt mit der hande deß mē
ſchen libe. was da gantz oder zer/

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

ABBYY Finereader – Prices (SDK Cloud)



Free sign up bonus	$9.99 per package	$69.99 per package	$199.99 per package	$899.99 per package	$2899.99 per package
500 pages $0.0 per page	100 pages $0.1 per page	1 000 pages $0.07 per page	5 000 pages $0.04 per page	30 000 pages $0.03 per page	100 000 pages $0.029 per page

Free sign up bonus	$29.99 per month	$99.99 per month	$199.99 per month	$299.99 per month	$839.99 per month
500 pages no additional pages	500 pages $0.06 per additional page	2 000 pages $0.05 per additional page	5 000 pages $0.04 per additional page	10 000 pages $0.03 per additional page	30 000 pages $0.028 per additional page

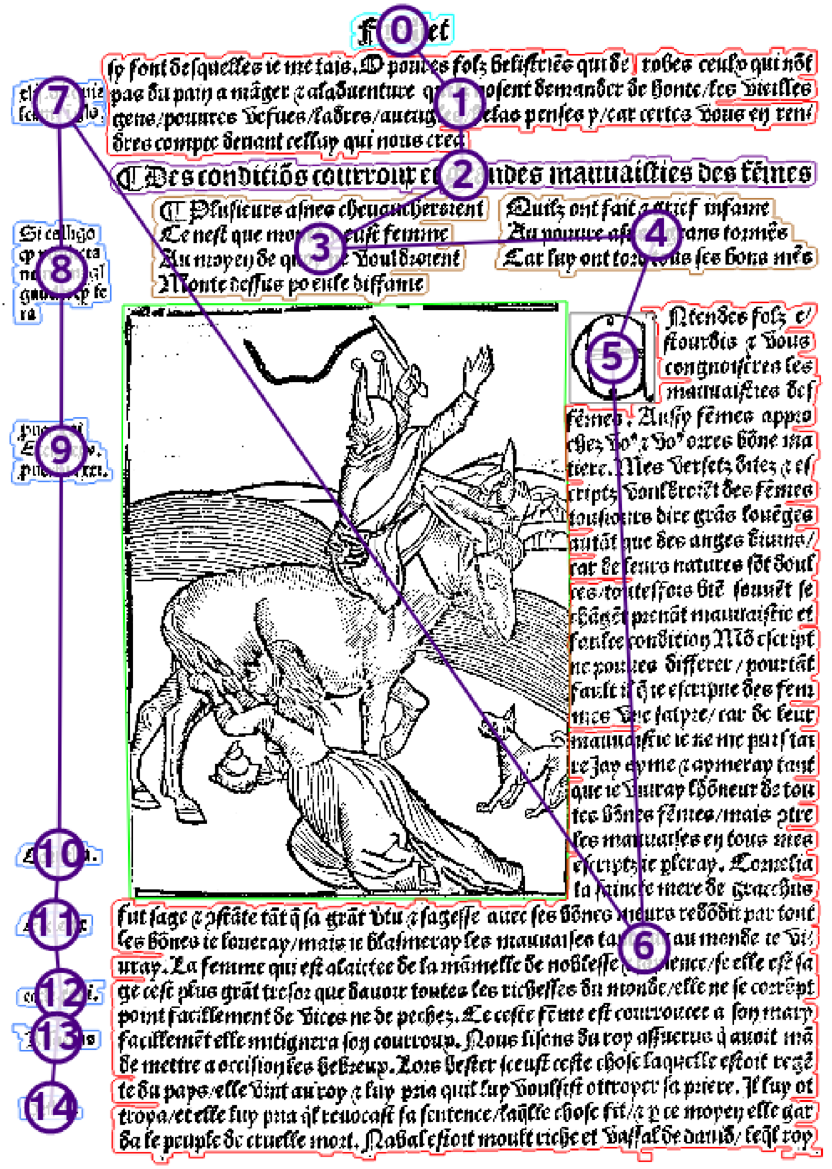
OCR4all – Overview

- 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.
  - Open source.

https://www.uni-wuerzburg.de/en/zpd/ocr4all/

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 imperative due to highly variant typography.
  - Usual goal: 100% accuracy regarding layout and OCR.
  - → 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!



ABBYY vs. OCR4all

ABBYY	Criteria	OCR4all
+++	Preprocessing	++
+++	Segmentation of moderate layouts, fully automatic	++
++	Segmentation of very complex layouts, semi automatic	+++
+++	OCR: mixed models (modern)	+
++	OCR: mixed models (Antiqua around 1900)	+
	OCR: mixed models (early printed books)	++
+	OCR: book-specific training	+++
+ (OK-ish)	Price	+++ (free)
+++	Speed	++
+(+)	Manual correction (segmentation, OCR, …)	+++
++	Rich output out of the box (paragraphs, semantic distinction, italics, …)	

ABBYY vs. OCR4all (with focus on the CADR)

ABBYY	Criteria	OCR4all
+++	Preprocessing	++
+++	**Segmentation of moderate layouts, fully automatic**	**++**
++	Segmentation of very complex layouts, semi automatic	+++
+++	OCR: mixed models (modern)	+
++	**OCR: mixed models (Antiqua around 1900)**	**+**
	OCR: mixed models (early printed books)	++
+	OCR: book-specific training	+++
+ (OK-ish)	**Price**	**+++ (free)**
+++	Speed	++
+(+)	Manual correction (segmentation, OCR, …)	+++
++	**Rich output out of the box (paragraphs, semantic distinction, italics, …)**	

ABBYY vs. OCR4all – Recommended Strategy

- ABBYY is probably better suited to your needs.
- If you have access to ABBYY, try a couple of pages and check the results.
  - Explicitly provide the recognition language (ABBYY usually detects it automatically but if it fails your results will suffer considerably).
  - If sufficient: process the rest of the book using ABBYY.
  - If not sufficient: use OCR4all.
    - Probably no fitting mixed models available.
    - Book-specific training required.
- If you can’t / don’t want to pay for ABBYY:
  - Register an account at https://cloud.ocrsdk.com/Account/Register (500 pages for free) and use the promo code (ask me) to get another 1,000 (Thanks to Deniz Güray @ ABBYY).
  - Cooperate with the ZPD.

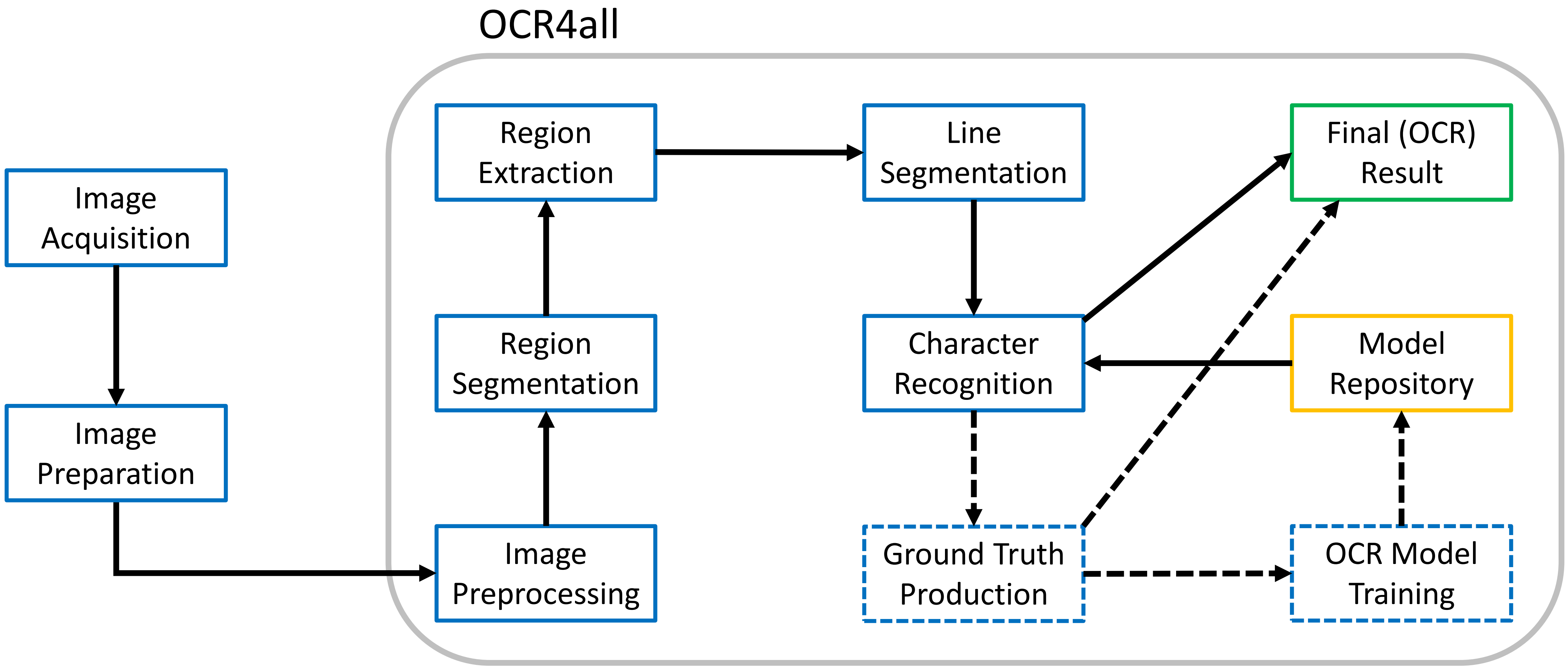
ABBYY vs. OCR4all – Cooperating with the ZPD

- What we offer:
  - ABBYY: We process your books using our page contingent.
  - OCR4all: We host a server version which you can use to process your books.
    - Technical support.
    - Access to our GPU infrastructure (rapid training!).
- Expected return service:
  - You produce GT (line images and the corresponding transcription) for your book.
    - One page GT for every 100 pages processed with ABBYY/OCR4all.
    - A minimum of five pages per book.
    - Can be done very efficiently (line synopsis, correcting the OCR result instead of transcribing from scratch, virtual keyboard for special characters, …).
  - GT will be used to
    - train mixed models which will be made publicly available.
    - perform evaluations, leading to insights and maybe a small publication.

Agenda

1. OCR Basics
2. OCR Engines ABBYY Finereader and OCR4all
3. **OCR4all Hands-on Guided**
4. OCR4all Hands-on Playground

OCR4all – Workflow (Main Steps Only)



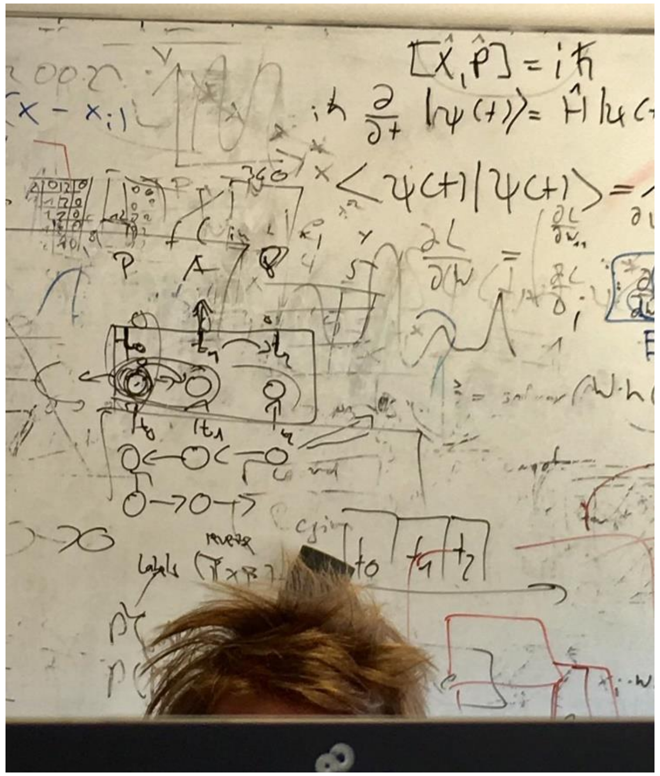
OCR4all – Submodule OCRopus

- 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 (in OCR4all now passed on to Calamari).
- Developed by Thomas M. Breuel (CS Prof. at University of Kaiserslautern, DFKI, Xerox, Google, currently NVIDIA).
- Open source.
- Breuel et al.: High-Performance OCR for Printed English and Fraktur using LSTM Networks.

OCR4all – Submodule Calamari

- Open source OCR engine based on OCRopus.
  - Developed by Christoph Wick at the University of Würzburg.
  - Wick, Reul, Puppe: Calamari - A High-Performance Tensorflow-based Deep Learning Package for Optical Character Recognition.
- Usage of deeper neural networks → significantly better recognition results.
- GPU support → much faster recognition and training.
- Native integration of advanced training and recognition concepts like pretraining, voting, and data augmentation.
- Many tweaks like enhanced modularity and cleaned up code.

Calamari OCR


Christoph at work.

Hands-on: Accessing OCR4all

- OCR4all originally designed to run locally and be accessed via a web GUI.
- For simplicity during this workshop → 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!

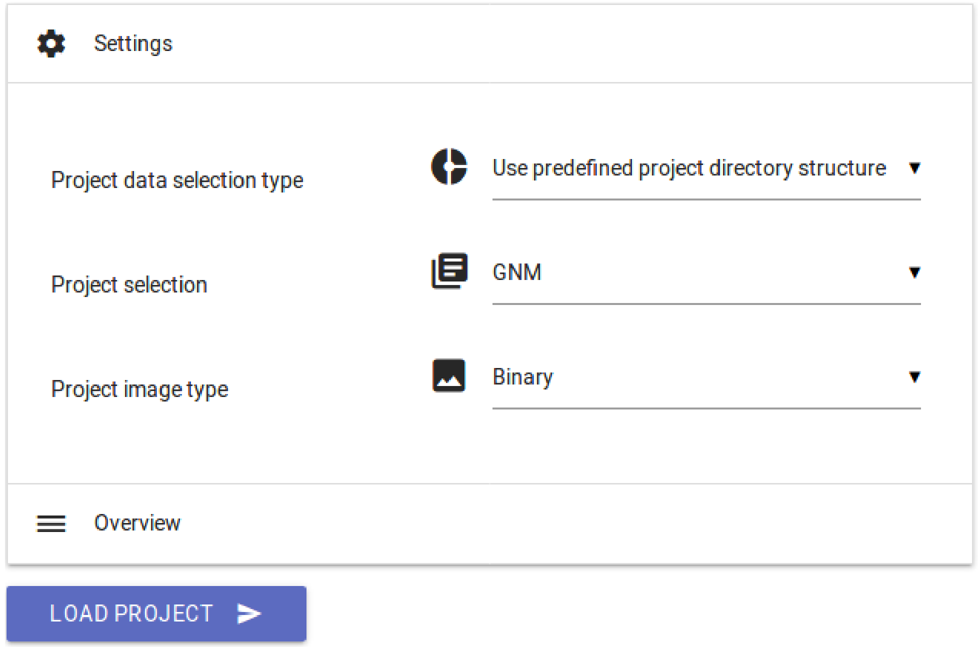
- Open your browser (preferably Chrome) and access

https://go.uniwue.de/ocr4all

(Backup: http://www.ocr4all-workshop.informatik.uni-wuerzburg.de)

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*/input/*images*.
- Go to "Settings" and select "ehrenstein" from the dropdown.
- Change "image type" to "Binary", **choose "Legacy" as your "Project processing mode",** and hit "Load Project".


Settings

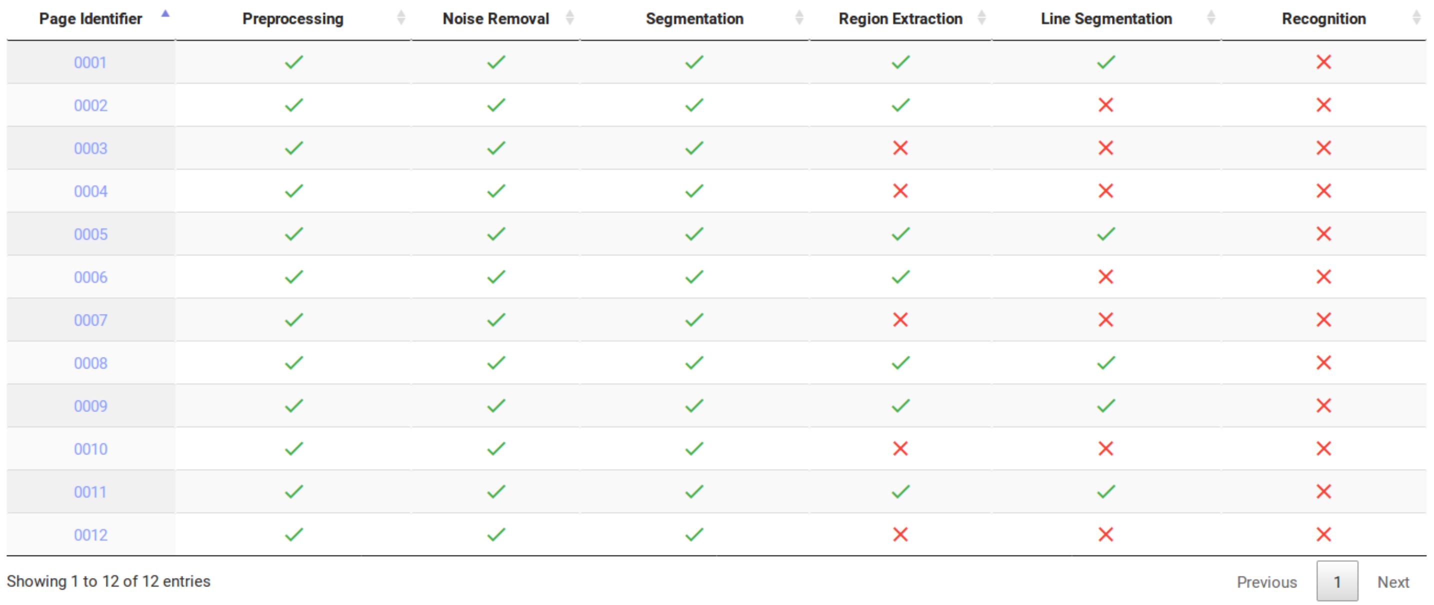
Project data selection type: Use predefined project directory structure

Project selection: GNM

Project image type: Binary

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

Workflow – Image Preparation

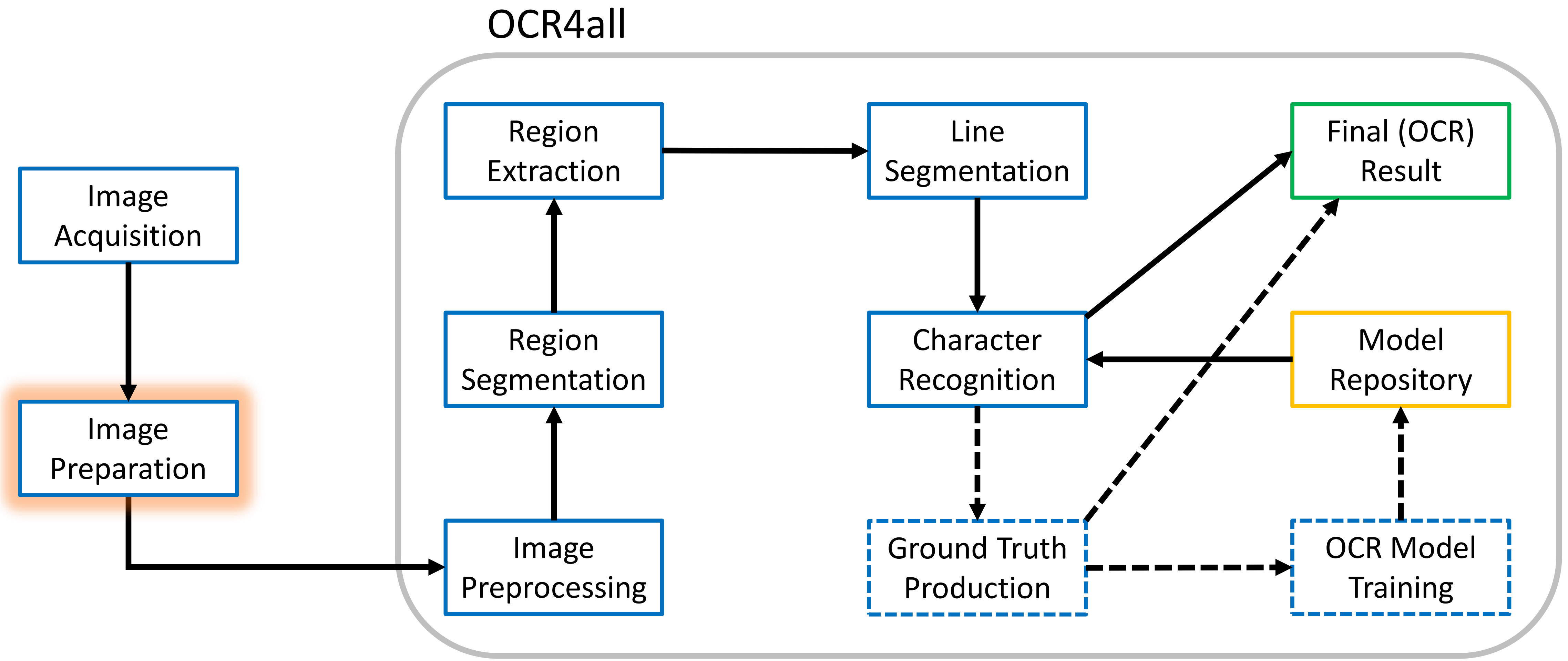
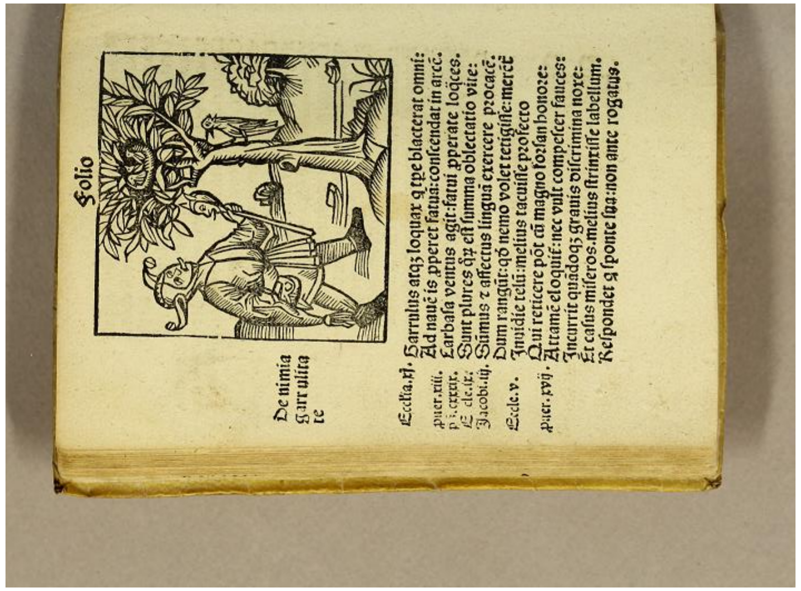
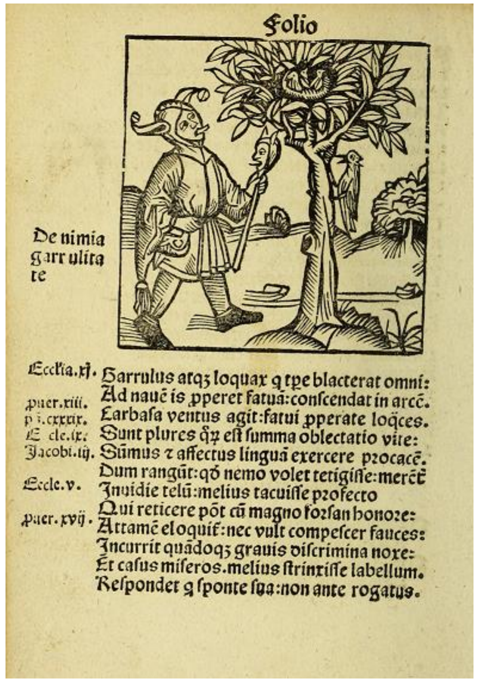


Image Preparation – Overview

- **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.





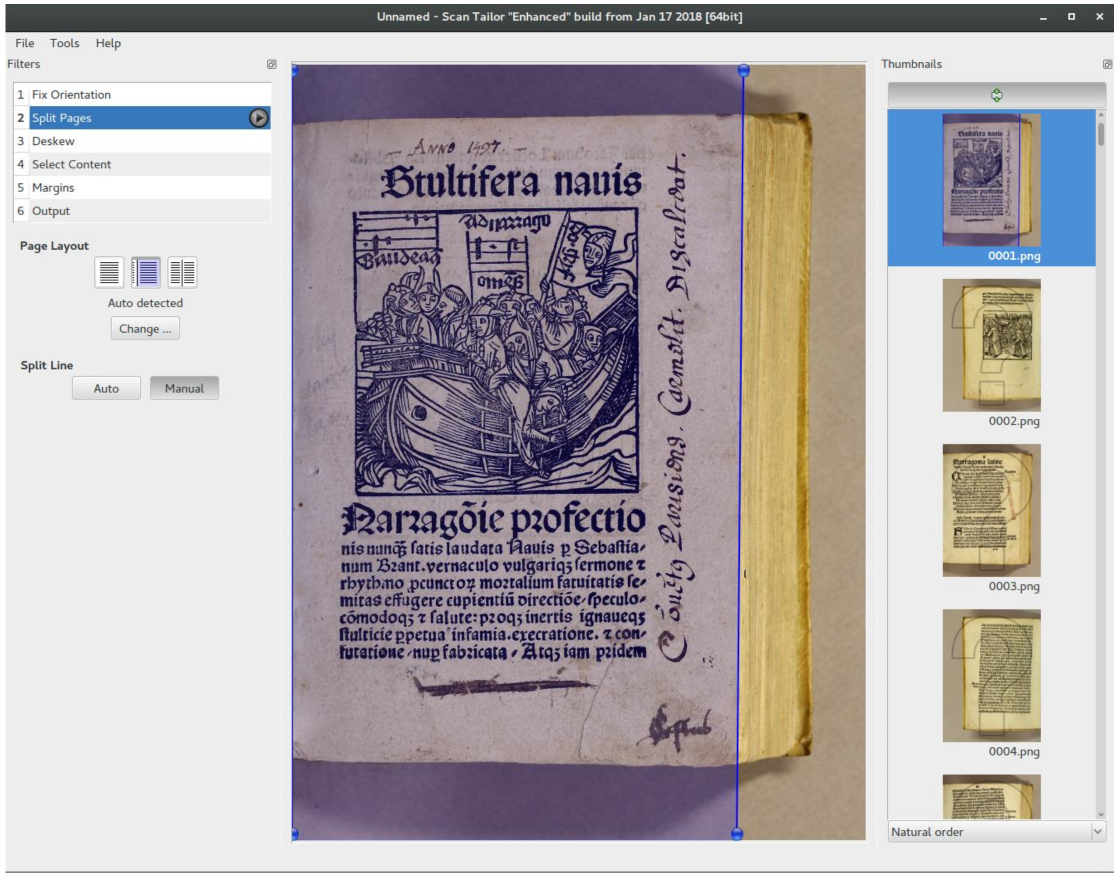
Folio

De nimia
garrulita
te

Eccl'ia.xj. Garrulus atq; loquax q tpe blacterat omni:
Ad nauē is pperet fatuā:conſcendat in arcē.
puer.xiiij. Carbaſa ventus agit:fatui pperate loq̄ces.
p i.cxxix. Sunt plures q̄z eſt ſumma oblectatio vite:
E cle.ix. Sūmus ⁊ affectus linguā exercere procacē.
Jacobi.iij. Dum rangūt:qđ nemo volet tetigiſſe:merc̄e
Eccle.v. Inuidie telū:melius tacuiſſe profecto
Qui reticere pōt cū magno forſan honore:
puer.xvij. Attamē eloquit:nec vult compeſcer fauces:
Incurrit quādoq; grauis diſcrimina noxe:
Et caſus miſeros.melius ſtrinxiſſe labellum.
Reſpondet q ſponte ſua:non ante rogatus.

Image Preparation – ScanTailor

- Many features:
  - Page splitting/periphery removal.
  - Rotation.
  - Content detection.
  - Deskewing.
  - Binarisation.
  - …
- Usually, the binarisation step can/should be skipped and be performed in OCR4all.
- A detailed video tutorial is available here.



Workflow – Image Preprocessing

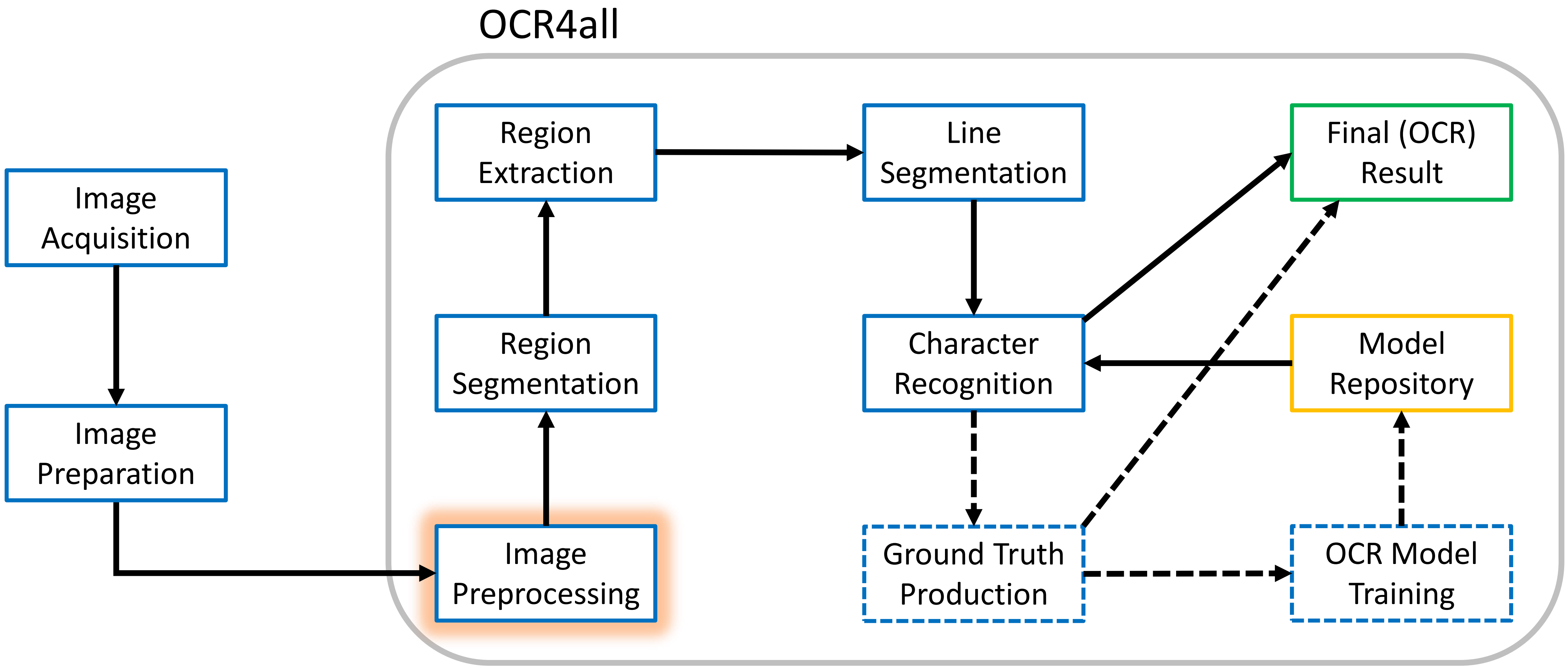
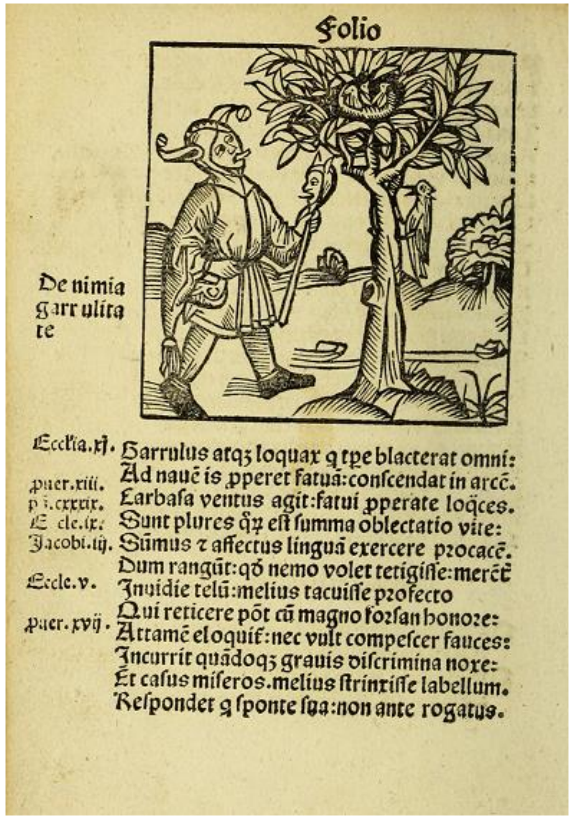


Image Preprocessing – Overview

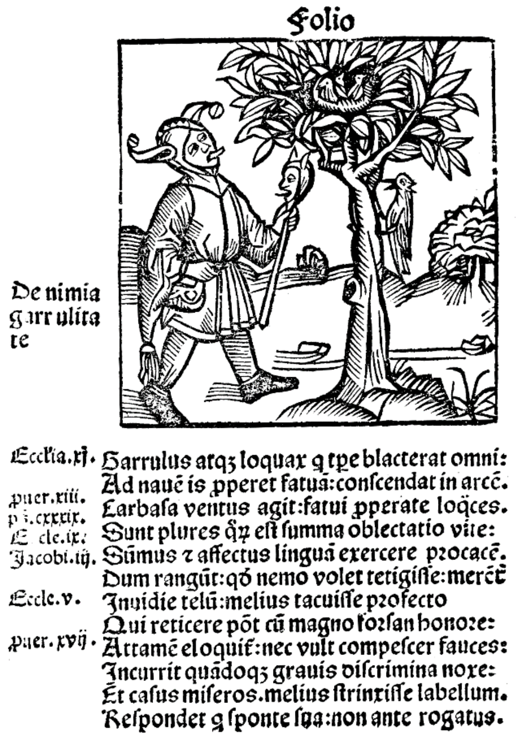
- **Input:** original image (color, grayscale, or binary).
  Output: deskewed binary (and grayscale) image(s).
- Two steps:
  - **Binarisation:** necessary for many image processing operations.
  - **Deskewing:** improves the upcoming line segmentation.
- Implemented in *ocropus-nlbin*.



Folio

De nimia garrulita te

Ecc̄lia.xj. Garrulus atq₃ loquax q tpe blacterat omni:
puer.xiiii. Ad nauē is ꝑperet fatuā: conſcendat in arcē.
p i.cxxix. Carbaſa ventus agit: fatui ꝑperate loq̄ces.
E cle.ix. Sunt plures q̄₃ eſt ſumma oblectatio vite:
Jacobi.iij. Sūmus ꝛ affectus linguā exercere procacē.
Dum rangūt: qđ nemo volet tetigiſſe: merēt
Eccle.v. Inuidie telū: melius tacuiſſe profecto
Qui reticere pōt cū magno forſan honore:
puer.xvij. Attamē eloquit: nec vult compeſcer fauces:
Incurrit quādoq₃ grauis diſcrimina noxe:
Et caſus miſeros. melius ſtrinxiſſe labellum.
Reſpondet q ſponte ſua: non ante rogatus.



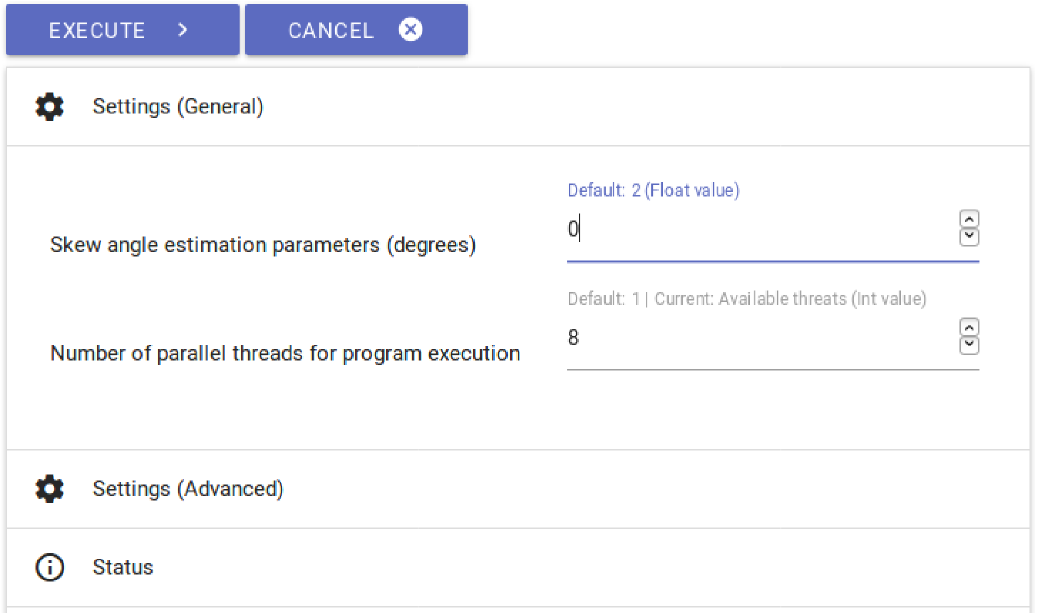
Folio

De nimia garrulita te

Ecc̄lia.xj. Garrulus atq₃ loquax q tpe blacterat omni:
puer.xiiii. Ad nauē is ꝑperet fatuā: conſcendat in arcē.
p i.cxxix. Carbaſa ventus agit: fatui ꝑperate loq̄ces.
E cle.ix. Sunt plures q̄₃ eſt ſumma oblectatio vite:
Jacobi.iij. Sūmus ꝛ affectus linguā exercere procacē.
Dum rangūt: qđ nemo volet tetigiſſe: merēt
Eccle.v. Inuidie telū: melius tacuiſſe profecto
Qui reticere pōt cū magno forſan honore:
puer.xvij. Attamē eloquit: nec vult compeſcer fauces:
Incurrit quādoq₃ grauis diſcrimina noxe:
Et caſus miſeros. melius ſtrinxiſſe labellum.
Reſpondet q ſponte ſua: non ante rogatus.

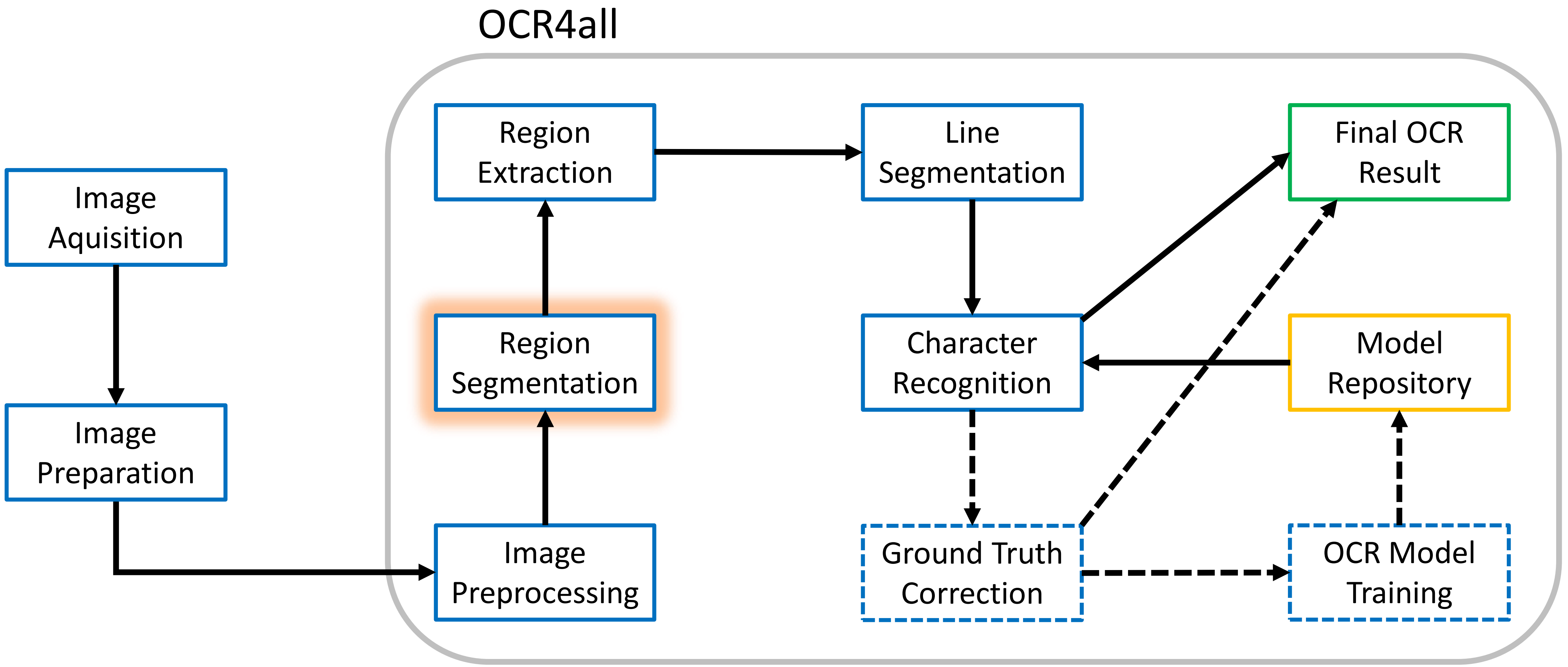
Hands-on: Preprocessing

- Select “Preprocessing” from the menu (☰) on the top left..
- **Change the “Number of parallel threads for program execution” to “2”.**
  - **Do this for every single one of the upcoming steps.**
  - When working locally it’s usually best to use all available CPUs (auto detect).
- Check the results by selecting preprocessed pages in the “Project Overview” view.



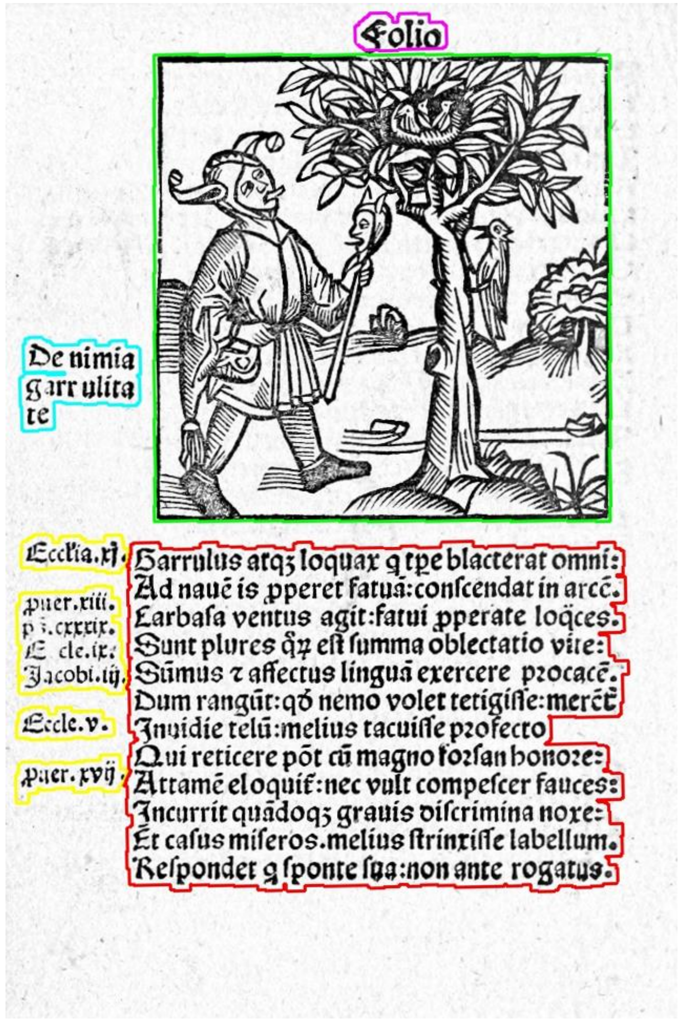
Setting	Default	Value
Skew angle estimation parameters (degrees)	Default: 2 (Float value)	0
Number of parallel threads for program execution	Default: 1 | Current: Available threads (Int value)	8

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.
  - LAREX (semi-automatic, deferred).



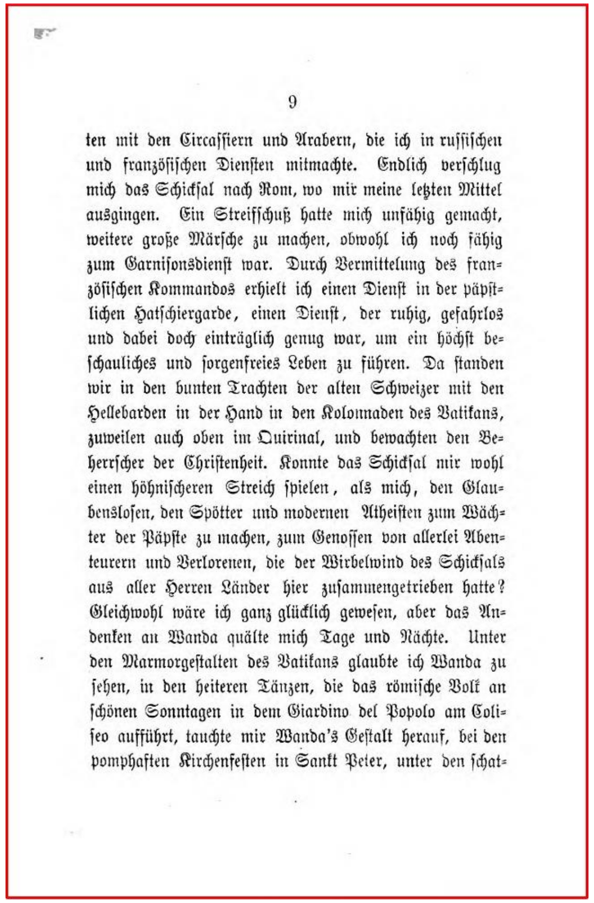
Folio

De nimia garrulita te

Eccl'ia.xi. Garrulus atqꝫ loquax q tpe blacterat omni:
Ad nauē is ꝑperet fatuā:conſcendat in arcē.
ꝓuer.xiii. Carbaſa ventus agit:fatui ꝑperate loq̃ces.
pſ.cxxxix. Sunt plures q̄ꝝ eſt ſumma oblectatio vite:
Eccle.ix. Sūmus ⁊ affectus linguā exercere procacē.
Jacobi.iij. Dum rangūt:qđ nemo volet tetigiſſe:mercē
Eccle.v. Inuidie telū:melius tacuiſſe profecto
Qui reticere pōt cū magno forſan honore:
ꝓuer.xvij. Attamē eloquit:nec vult compeſcer fauces:
Incurrit quādoqꝫ grauis diſcrimina noxe:
Et caſus miſeros.melius ſtrinxiſſe labellum.
Reſpondet q ſponte ſua:non ante rogatus.

Region Segmentation – Dummy Segmentation

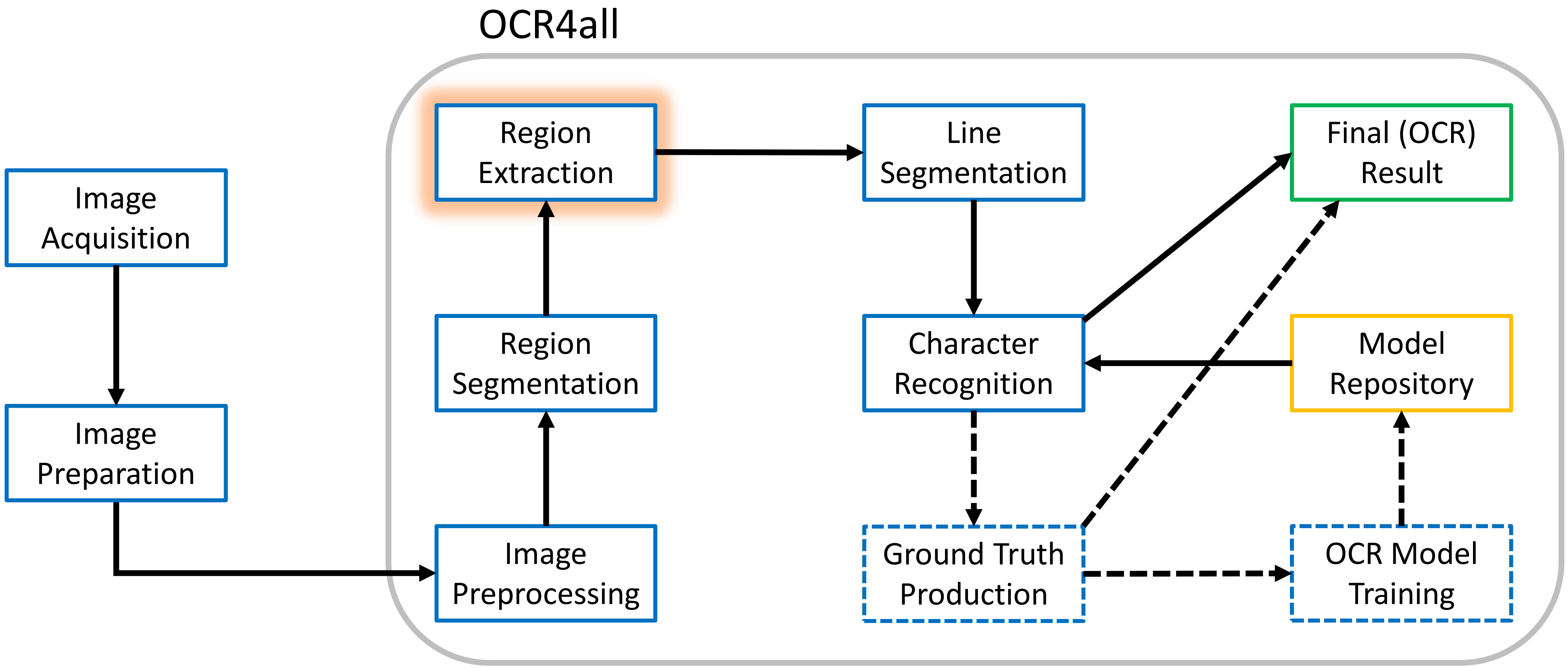
- Considers the entire page as a single segment.
- Leaves text/image segmentation and column separation to the upcoming line segmentation.
- Fully automatic and very fast.
- Often completely satisfactory when dealing with simple layouts (e.g. standard 19th century novels).
- No image markup.
- No semantic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



9

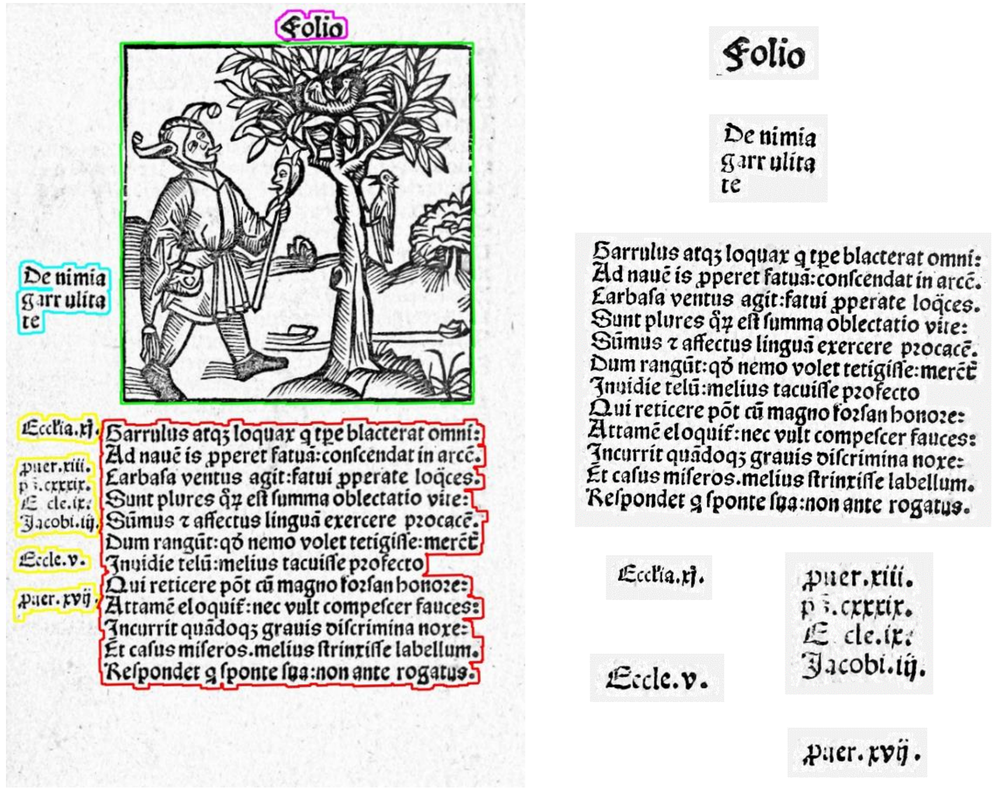
ten mit den Circaſſiern und Arabern, die ich in ruſſiſchen
und franzöſiſchen Dienſten mitmachte. Endlich verſchlug
mich das Schickſal nach Rom, wo mir meine letzten Mittel
ausgingen. Ein Streifſchuß hatte mich unfähig gemacht,
weitere große Märſche zu machen, obwohl ich noch fähig
zum Garniſonsdienſt war. Durch Vermittelung des fran=
zöſiſchen Kommandos erhielt ich einen Dienſt in der päpſt=
lichen Hatſchiergarde, einen Dienſt, der ruhig, gefahrlos
und dabei doch einträglich genug war, um ein höchſt be=
ſchauliches 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 Be=
herrſcher der Chriſtenheit. Konnte das Schickſal mir wohl
einen höhniſcheren Streich ſpielen, als mich, den Glau=
benslosen, den Spötter und modernen Atheiſten zum Wäch=
ter der Päpſte zu machen, zum Genoſſen von allerlei Aben=
teurern und Verlorenen, die der Wirbelwind des Schickſals
aus aller Herren Länder hier zuſammengetrieben hatte?
Gleichwohl wäre ich ganz glücklich geweſen, aber das An=
denken 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 Coli=
ſeo aufführt, tauchte mir Wanda's Geſtalt herauf, bei den
pomphaften Kirchenfeſten in Sankt Peter, unter den ſchat=

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*.



Folio

De nimia
garr ulita
te

Garrulus atqz loquax q tpe blacterat omni:
Ad nauē is pperet fatuā:conſcendat in arcē.
Carbaſa ventus agit:fatui pperate loq̈ces.
Sunt plures q̄z eſt ſumma oblectatio vite:
Sūmus ꝛ affectus linguā exercere procacē.
Dum rangūt:qđ nemo volet tetigiſſe:merc̄ē
Inuidie telū:melius tacuiſſe profecto
Qui reticere pōt cū magno forſan honore:
Attamē eloquiſ:nec vult compeſcer fauces:
Incurrit quādoqz grauis diſcrimina noxe:
Et caſus miſeros.melius ſtrinxiſſe labellum.
Reſpondet q ſponte ſua:non ante rogatus.

Eccl'ia.xj.

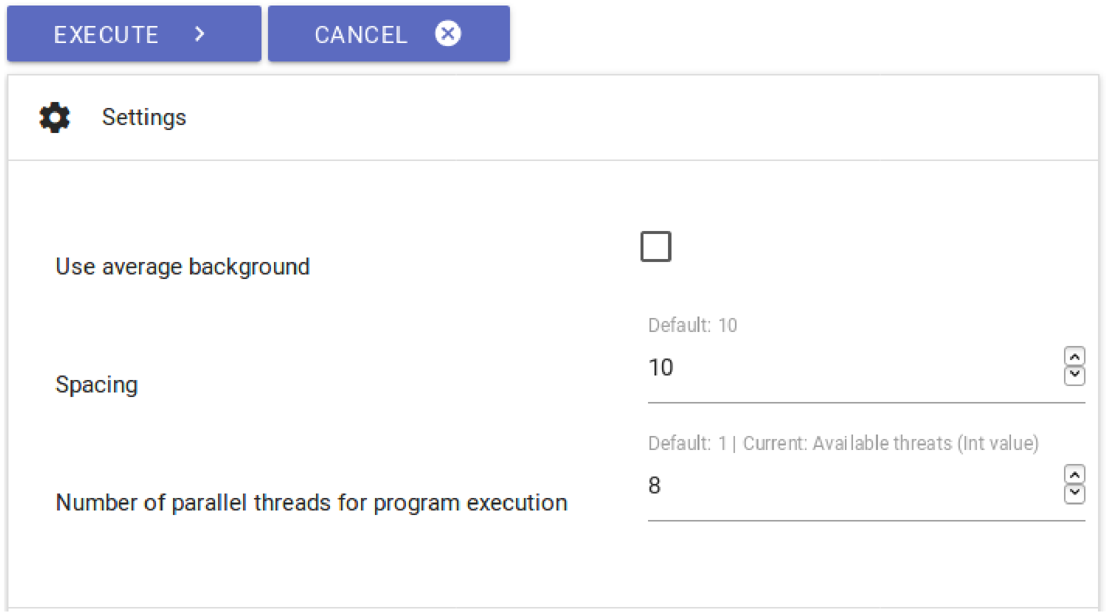
puer.xiii.
p ī.cxxxix.
E cle.ix:
Jacobi.iij.

Eccle.v.

puer.xvij.

Hands-on: Region Extraction

- 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”.

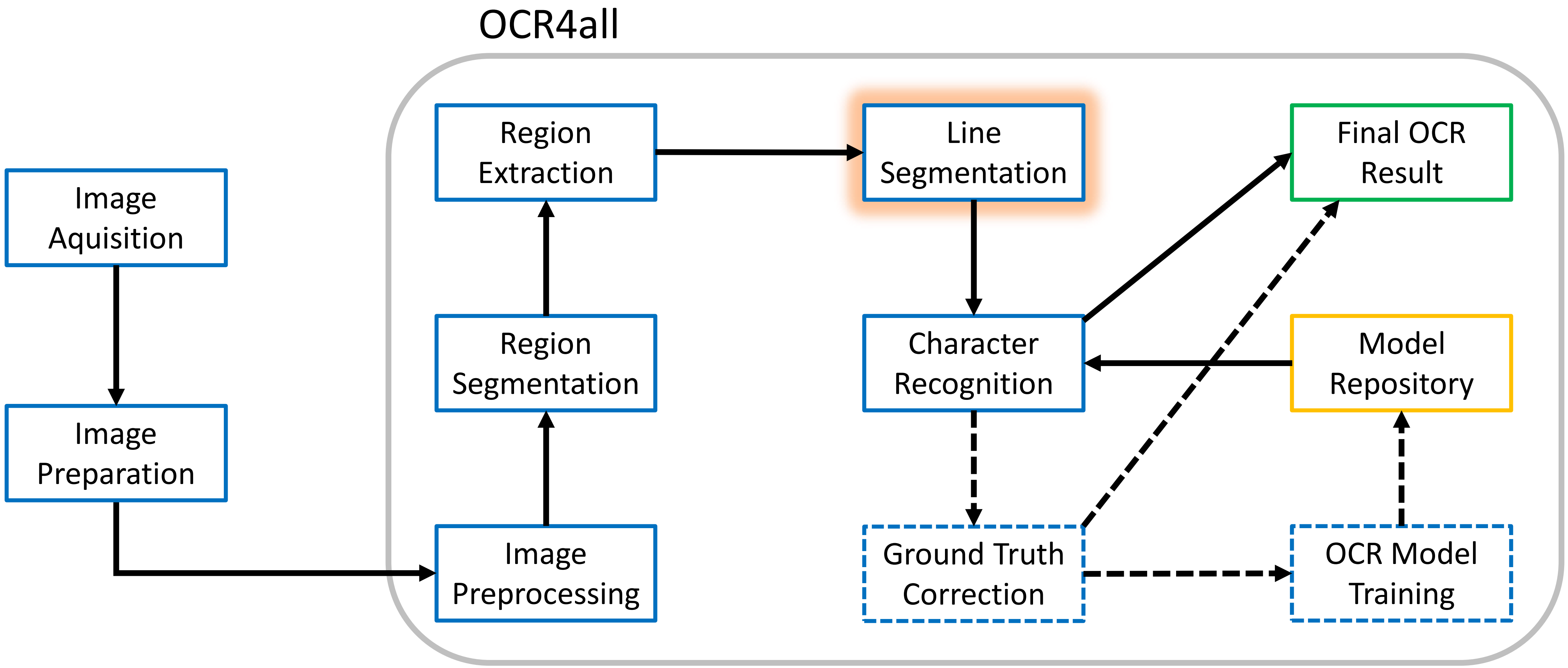


Use average background ☐

Spacing — Default: 10 — 10

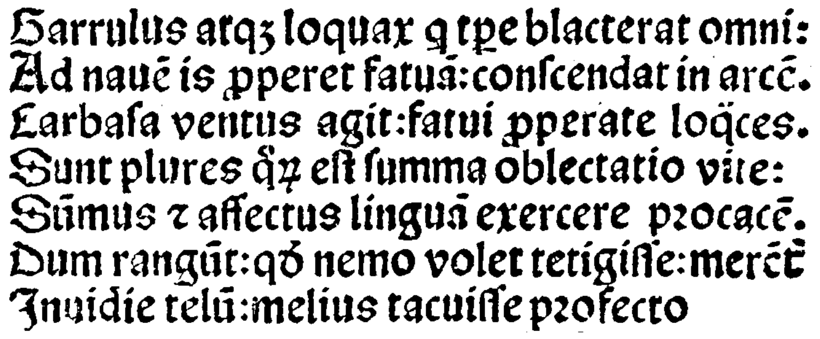
Number of parallel threads for program execution — Default: 1 | Current: Available threats (Int value) — 8

Workflow – Line Segmentation

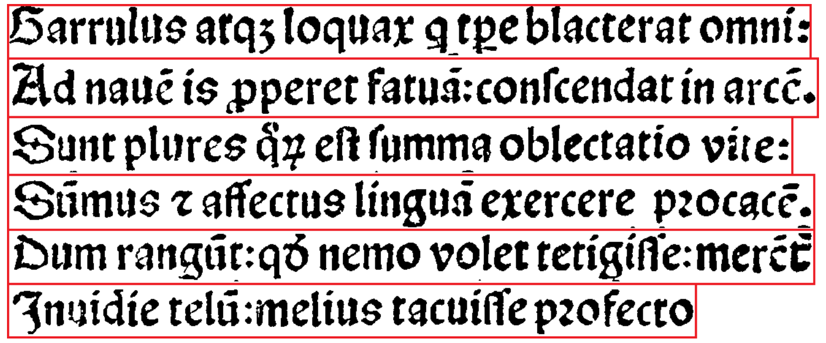


Line Segmentation – Overview

- **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.
- Adapted from *ocropus-gpageseg*.



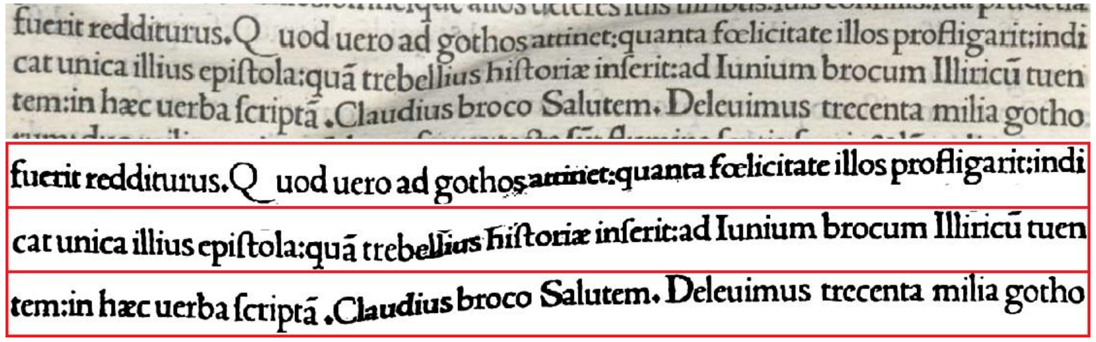
Garrulus atqꝫ loquax q̄ tpe blacterat omni:
Ad nauē is ꝑperet fatuā: conſcendat in arcē.
Carbaſa ventus agit: fatui ꝑperate loq̄ces.
Sunt plures q̄ꝫ eſt ſumma oblectatio vite:
Sūmus ꝛ affectus linguā exercere pꝛocacē.
Dum rangūt: qꝺ nemo volet tetigiſſe: merē̄
Inuidie telū: melius tacuiſſe pꝛofecto



Garrulus atqꝫ loquax q̄ tpe blacterat omni:
Ad nauē is ꝑperet fatuā: conſcendat in arcē.
Sunt plures q̄ꝫ eſt ſumma oblectatio vite:
Sūmus ꝛ affectus linguā exercere pꝛocacē.
Dum rangūt: qꝺ nemo volet tetigiſſe: merē̄
Inuidie telū: melius tacuiſſe pꝛofecto

Line Segmentation – Conclusion

- Very robust on Latin scripts.
- In general, *ocropus-gpageseg* can also deal with more complex layouts:
  - Simple text/non-text segmentation.
  - Decent column detection.
- However, complex layouts require a precise region segmentation.

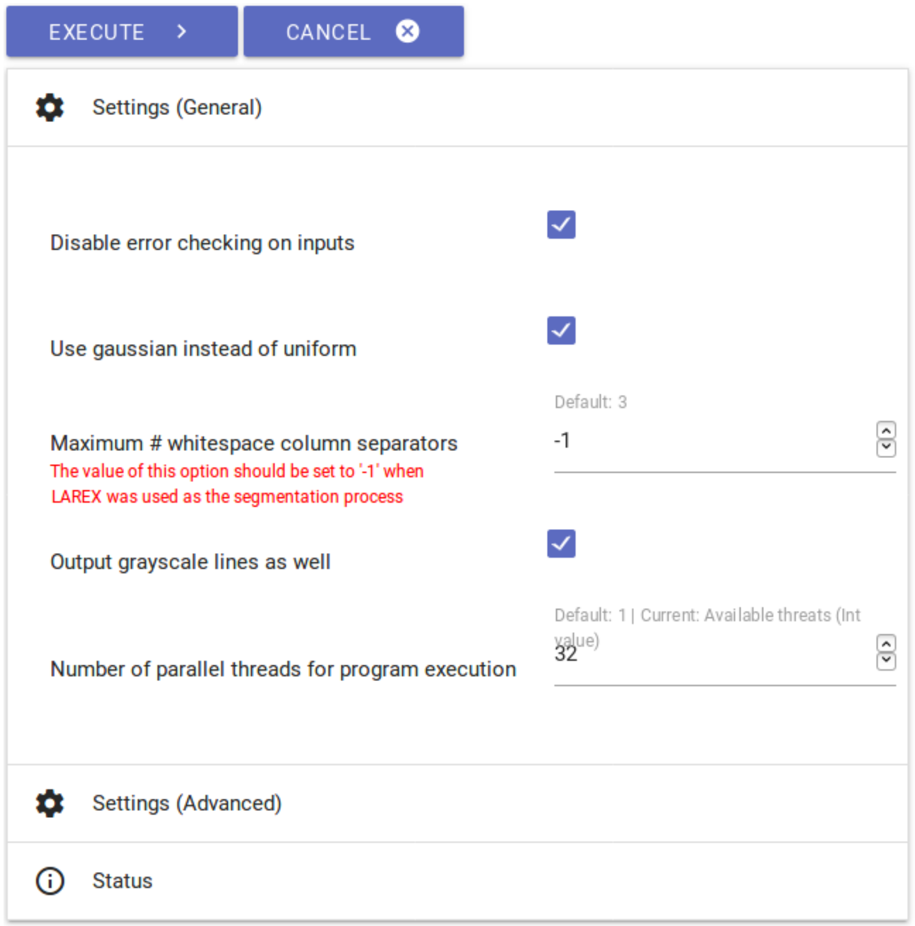


fuerit redditurus.Q uod uero ad gothos attinet:quanta fœlicitate illos profligarit:indi
cat unica illius epiſtola:quã trebellius hiſtoriæ inferit:ad Iunium brocum Illiricũ tuen
tem:in hæc uerba ſcriptã .Claudius broco Salutem. Deleuimus trecenta milia gotho

fuerit redditurus.Q uod uero ad gothos attinet:quanta fœlicitate illos profligarit:indi
cat unica illius epiſtola:quã trebellius hiſtoriæ inferit:ad Iunium brocum Illiricũ tuen
tem:in hæc uerba ſcriptã .Claudius broco Salutem. Deleuimus trecenta milia gotho

Hands-on: Line Segmentation

- Go to “Line Segmentation”.
- Keep all settings to the default.
- Hit “Execute” and check the results …
  - in the “Project Overview”.
  - by using “Ground Truth Production”.
- Important parameter: “Maximum # of whitespace column separators”.
  - Controls the column detection.
  - A value of “-1” skips the search for columns.



Disable error checking on inputs ✓

Use gaussian instead of uniform ✓

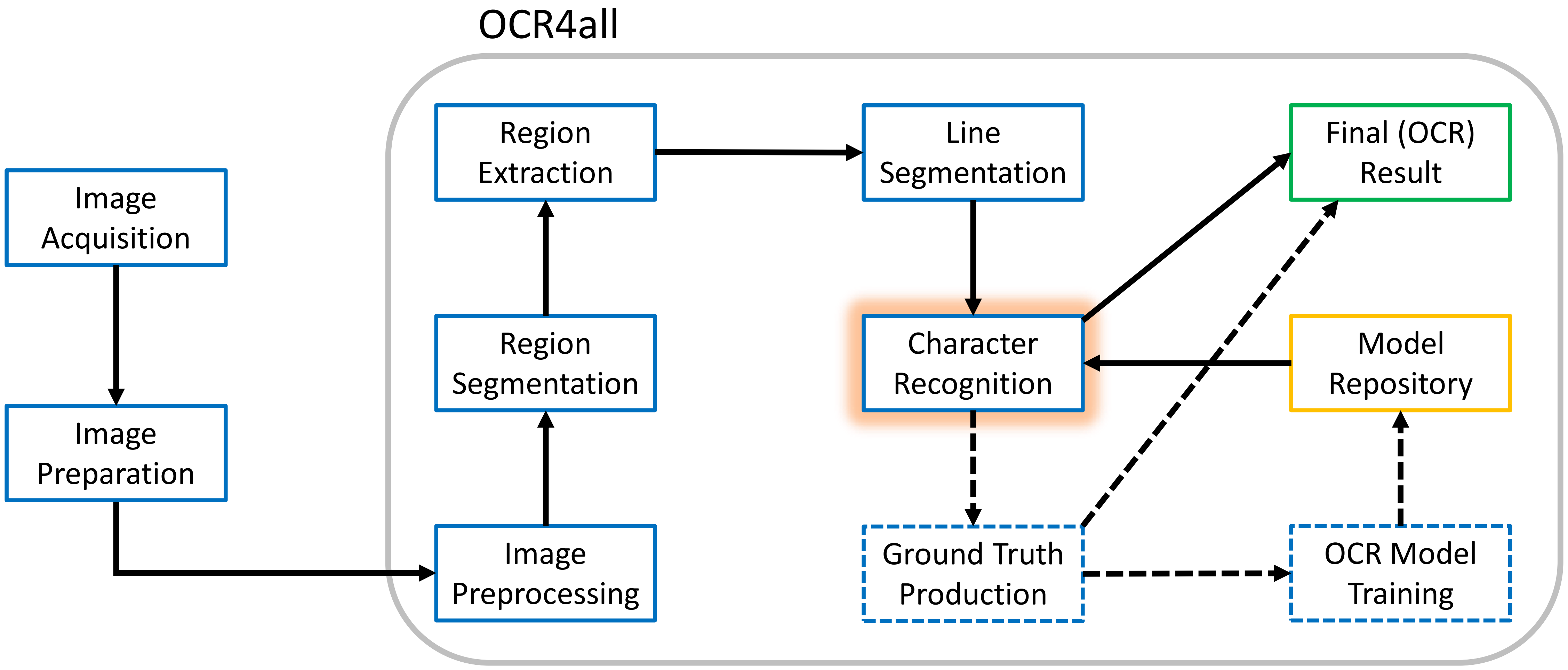
Maximum # whitespace column separators — Default: 3 — -1

The value of this option should be set to '-1' when LAREX was used as the segmentation process

Output grayscale lines as well ✓

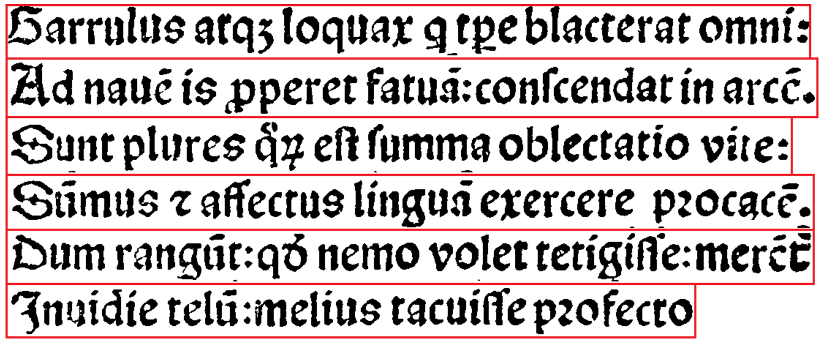
Number of parallel threads for program execution — Default: 1 | Current: Available threats (Int value) — 32

Workflow – Character Recognition



Character Recognition – Overview

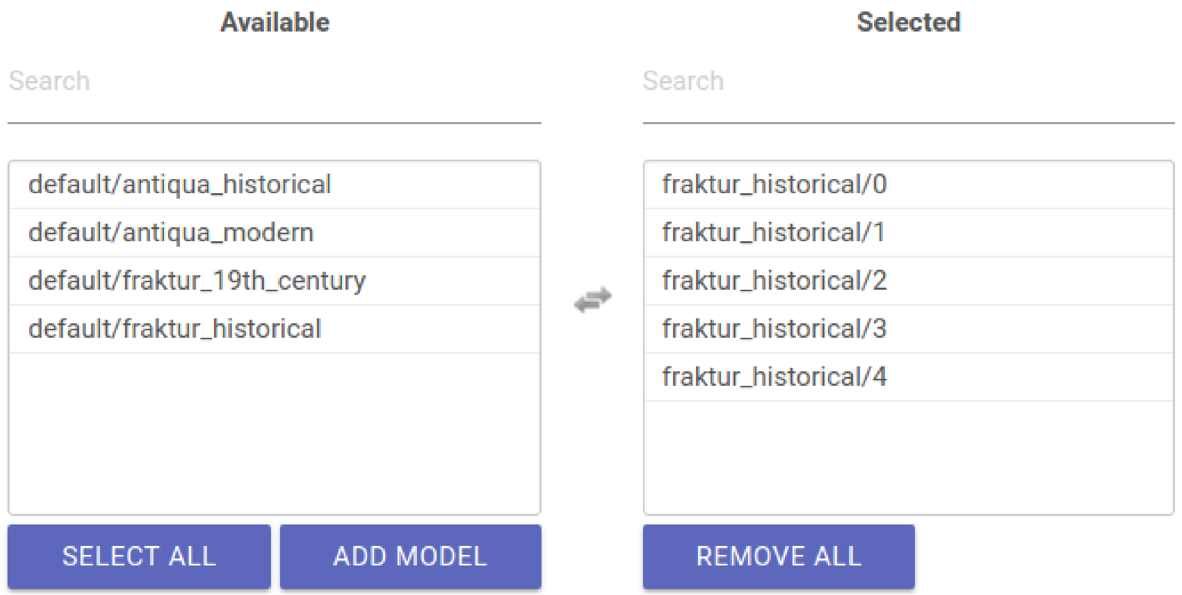
- **Input:** line images and an OCR model.
  Output: recognized text lines.
- Works on binary and grayscale images.
- Grayscale images can offer additional information.
- Rules of thumb/recommendations:
  - Models trained on grayscale/binary lines should be used to recognize grayscale/binary lines.
  - Utilize grayscale lines for book-specific training.
  - Mixed models should be trained on and applied to binary lines.


Garrulus atqꝫ loquax q̄ tpe blacterat omni:
Ad nauē is pperet fatuā:conſcendat in arcē.
Sunt plures q̄ꝫ eſt ſumma oblectatio vite:
Sūmus ꝛ affectus linguā exercere pꝛocacē.
Dum rangūt:qꝺ nemo volet tetigiſſe:mercē
Jnuidie telū:melius tacuiſſe pꝛofecto

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

Hands-on: Character Recognition

- Go to “Recognition”.
- Select “antiqua_modern” as “Line recognition models”.
- Hit “Execute” and check the results by using “Ground Truth Production”.

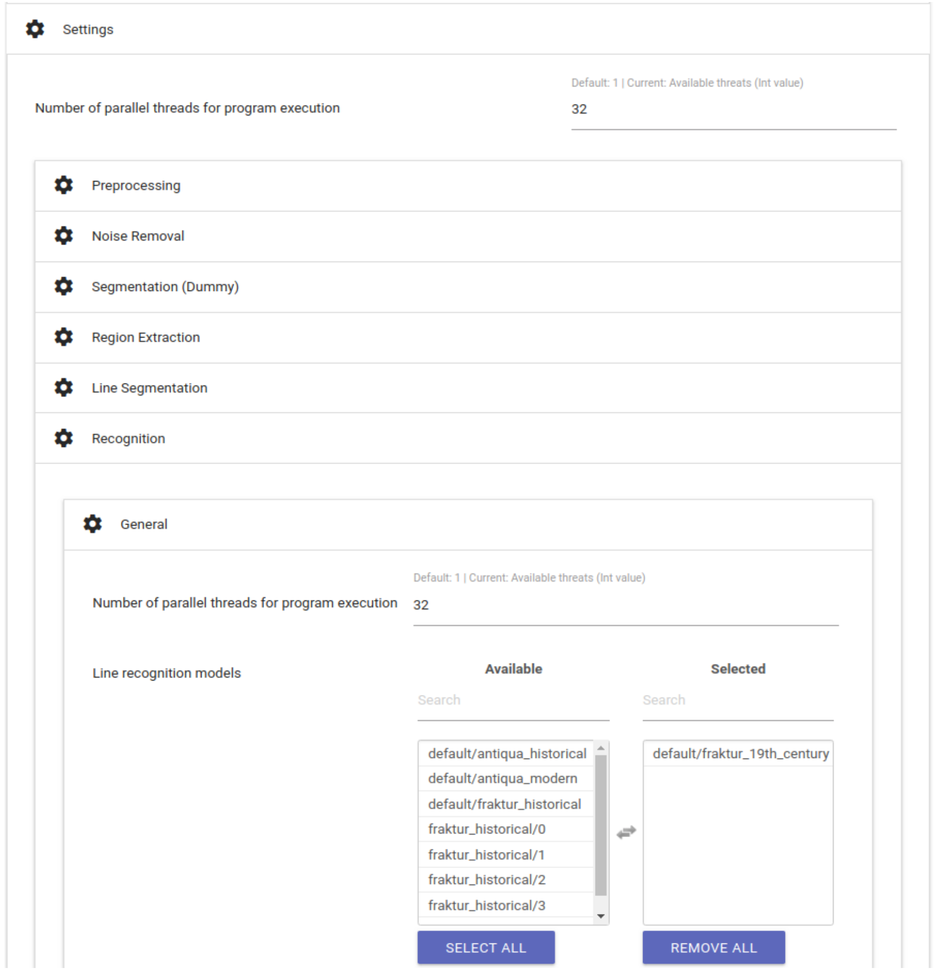


default/antiqua_historical
default/antiqua_modern
default/fraktur_19th_century
default/fraktur_historical

fraktur_historical/0
fraktur_historical/1
fraktur_historical/2
fraktur_historical/3
fraktur_historical/4

Hands-on: A Fully Automated Run

- Steps that don‘t require manual intervention can/should be run at once.
- Go to “Process Flow”.
- Again choose “antiqua_modern” as your OCR model (“Settings” → “Recognition”).
- Hit execute and wait for the success notification.
- Check the results.



Number of parallel threads for program execution: 32 (Default: 1 | Current: Available threads (Int value))

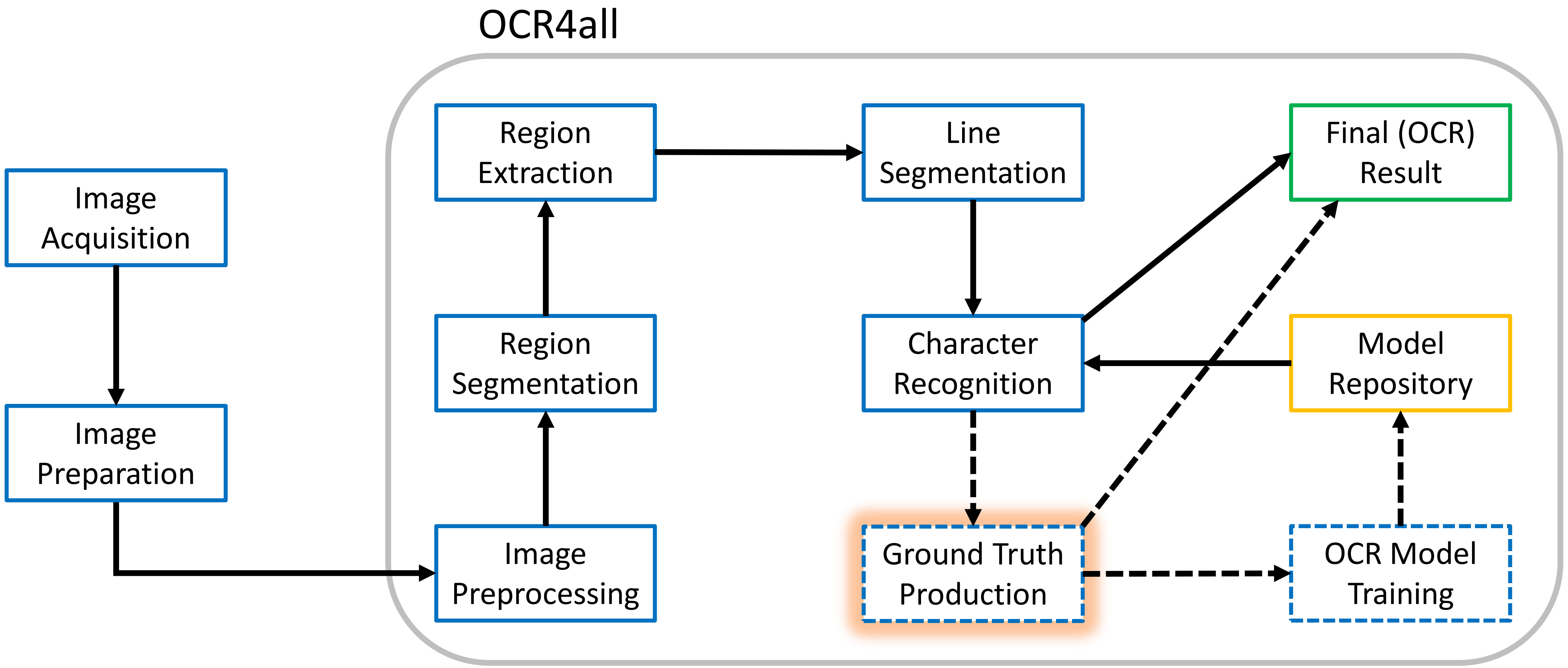
Number of parallel threads for program execution: 32 (Default: 1 | Current: Available threads (Int value))

Line recognition models

Available: default/antiqua_historical, default/antiqua_modern, default/fraktur_historical, fraktur_historical/0, fraktur_historical/1, fraktur_historical/2, fraktur_historical/3

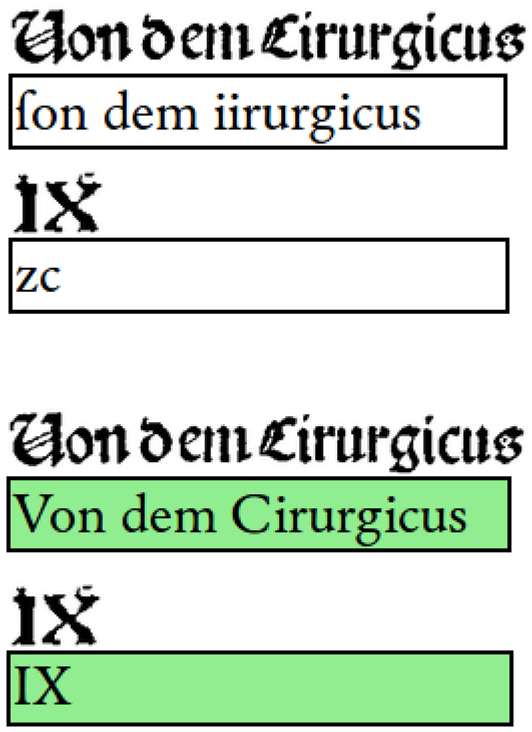
Selected: default/fraktur_19th_century

Workflow – Ground Truth Production



Ground Truth Production – Overview

- **Input:** line image and OCR result (optional).
  Output: corrected text (= ground truth GT).
- GT is required for the upcoming OCR model training.
- Configurable virtual keyboard allows to insert non-standard characters.



ſon dem iirurgicus

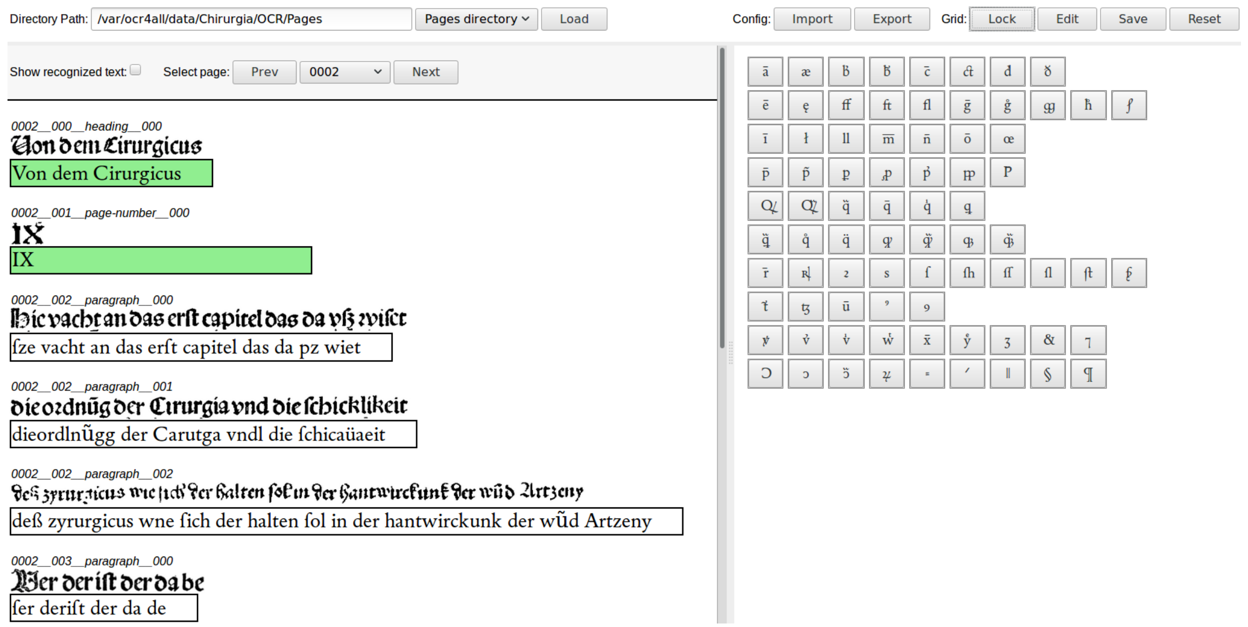
zc

Von dem Cirurgicus

IX

Hands-on: Ground Truth Production

- Go to “Ground Truth Production”.
- Select a line by clicking on 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 upcoming evaluation step: please make sure to at least save some (pseudo) corrections!



0002__000__heading__000
Von dem Cirurgicus

0002__001__page-number__000
IX

0002__002__paragraph__000
ſze vacht an das erſt capitel das da pz wiet

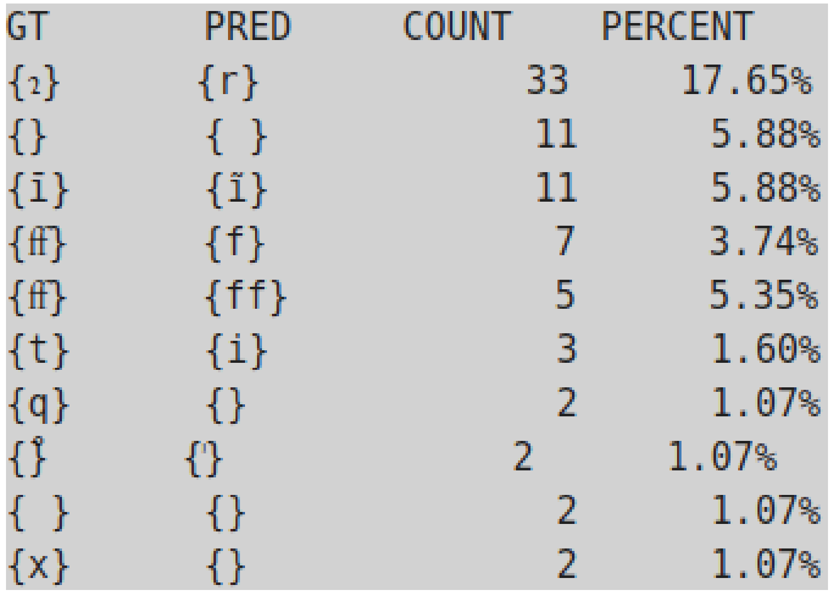
0002__002__paragraph__001
dieordlnũgg der Carutga vndl die ſchicaüaeit

0002__002__paragraph__002
deß zyrurgicus wne ſich der halten ſol in der hantwirckunk der wũd Artzeny

0002__003__paragraph__000
ſer deriſt der da de

Evaluation – Overview

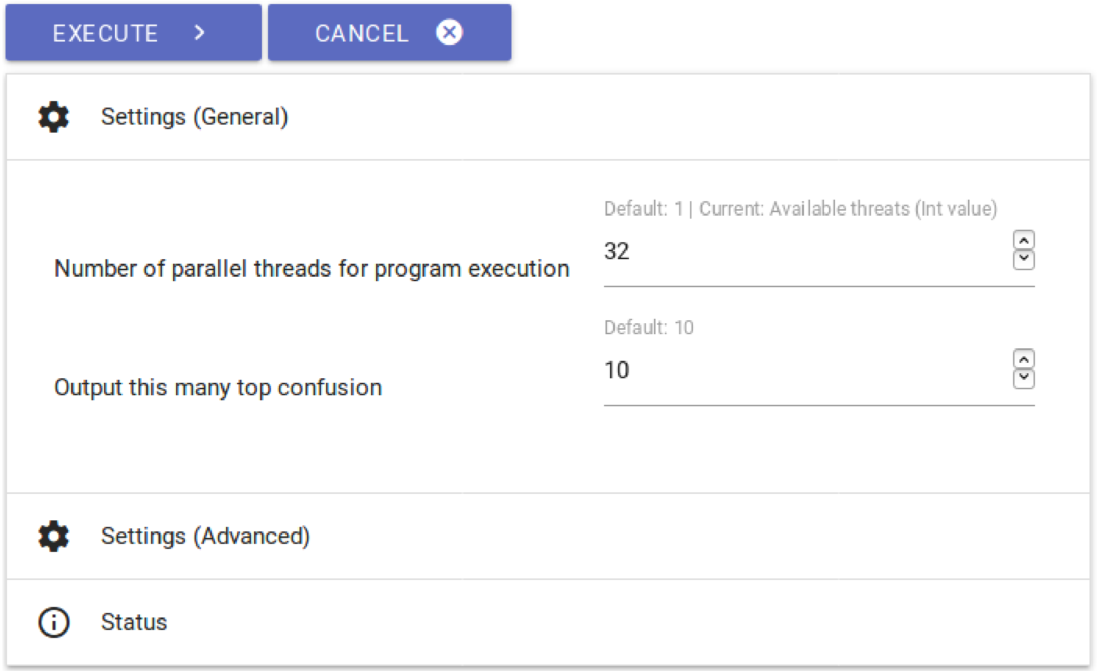
- **Input:** line-based OCR and GT pairs.
  Output: error statistics.
- Implemented in Calamari’s eval script.
- Character Error Rate (CER) = # errors / len(GT)
- Shows type and impact of the most frequent errors.
- Three different error types:
  - **Insertions**: character(s) present in OCR but not in GT.
  - **Deletions**: character(s) present in GT but not in OCR.
  - **Substitutions**: character(s) recognized as (an)other one(s).



```
GT         PRED       COUNT      PERCENT
{ꝛ}        {r}            33      17.65%
{}         { }            11       5.88%
{ī}        {ĩ}            11       5.88%
{ﬀ}        {f}             7       3.74%
{ﬀ}        {ff}            5       5.35%
{t}        {i}             3       1.60%
{q}        {}              2       1.07%
{ͦ}       {'}            2      1.07%
{ }        {}              2       1.07%
{x}        {}              2       1.07%
```

Hands-on: Evaluation

- Go to “Evaluation”.
- Keep all settings to the default and hit “Execute”.



Number of parallel threads for program execution — Default: 1 | Current: Available threads (Int value) — 32

Output this many top confusion — Default: 10 — 10

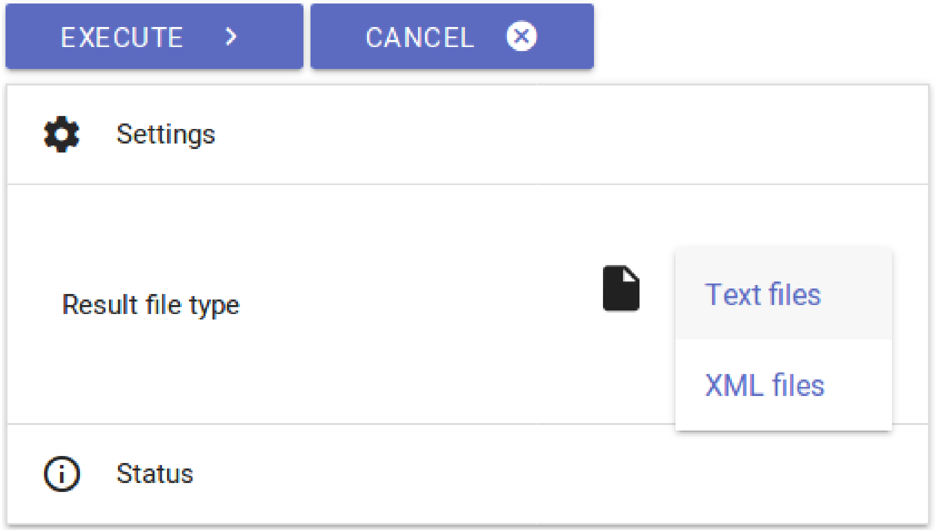
Result Generation – Overview

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 simultaneously.

Hands-on: Result Generation

- Go to “Result Generation”.
- Run the extraction using both available “Result file types”.



Agenda

1. OCR Basics
2. OCR Engines ABBYY Finereader and OCR4all
3. OCR4all Hands-on Guided
4. **OCR4all Hands-on Playground**

Hands-on: Try Another Book on Your Own

- Work with the books you submitted to us.
  - There is one copy of each book available on the server.
  - If more copies are required → ask me.
- Process the books analogously to “ehrenstein” (legacy mode!).
  - Use the “Process Flow” (limit the CPUs to 2).
  - Choose the entire “antiqua_modern” voting ensemble (models 0-4) as your “recognition model”.
  - Check the results.
- Experiment with book-specific training.
  - Produce some real GT (two pages?).
  - Notify me when you are done so we can run the training on the GPU servers.
  - Apply the resulting voting ensemble to previously unseen pages.
  - Correct the output for one of these new pages.
  - Perform an “Evaluation” on this new page and check the CER and the confusion table.

What Next?

- Consider a cooperation with the ZPD.
- If interested, feel free to try out OCR4all using your own hardware and material.
- Don’t hesitate to ask for help.
  - Installation.
  - Project specific consulting.
  - Bug fixes.
  - Server support if your own hardware is not sufficient.
  - …

- We depend on your (brutally honest) feedback!
- … but please keep in mind that OCR4all is work in progress.

Thank you for your Attention!

Acknowledgements:

- **OCR4all web app**: Dennis Christ, Alexander Hartelt, Nico Balbach, Yannik Herbst, Kevin Chadbourne, Andreas Büttner
- **LAREX web app**: Nico Balbach
- **Calamari**: Christoph Wick
- **Distribution via Docker**: Björn Eyeselein
- **Testing, Guides, and User Support**: Maximilian Wehner, Maximilian Nöth, …
- **Ideas and Feedback**: Dr. Uwe Springmann, Prof. Dr. Frank Puppe, Christine Grundig, …
- **Funding**: Chair for Artificial Intelligence (Prof. Dr. Frank Puppe) and Centre for Philology and Digitality at the University of Würzburg, BMBF Project "Kallimachos"

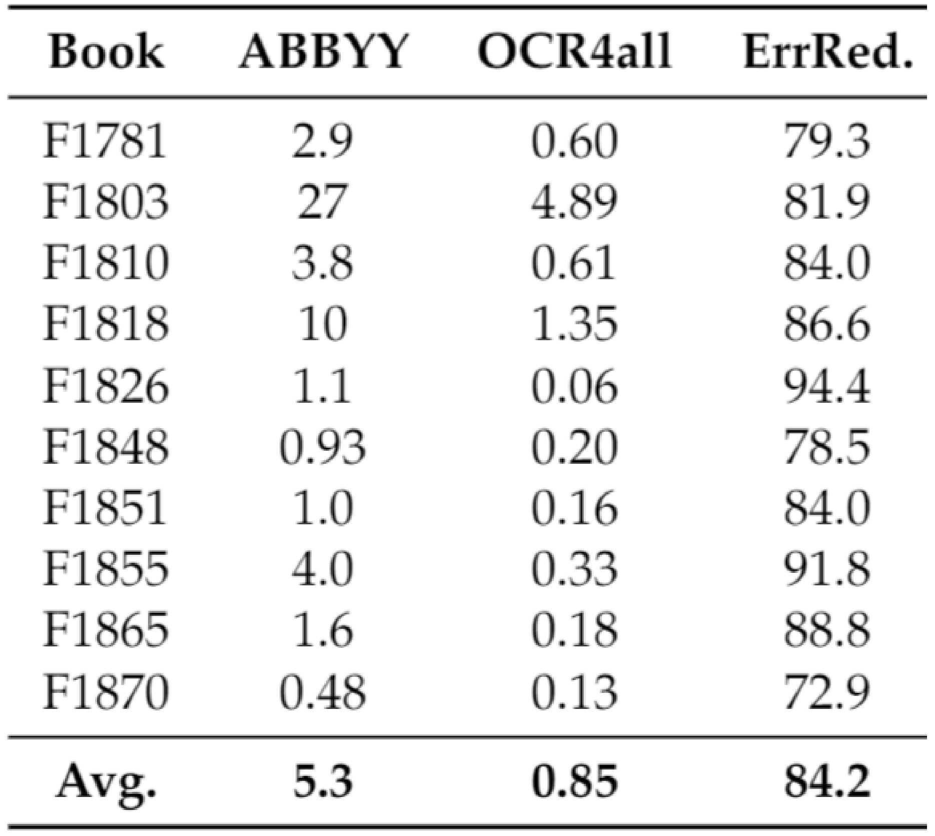
Working with OCR4all – What to expect?

- Early prints (15-16th cent.) with complex layout:
  - Assisted segmentation including fine-grained semantic distinction (up to several hours).
  - GT production for book-specific iterative model training (up to several hours).
  - Good chances to reach 99%+ character accuracy, depending on the state of the training.
- Standard 19th cent. novel:
  - Fully automated process using a fitting default model (if available).
  - Very good chances to reach 99%+ character accuracy.
- Fluent passage for works from the 17-18th cent.

- Modern prints:
  - Recommendation: ABBYY Finereader.
  - Usage of OCR4all basically possible.

Comparison with ABBYY on 19th Century Fraktur Novels

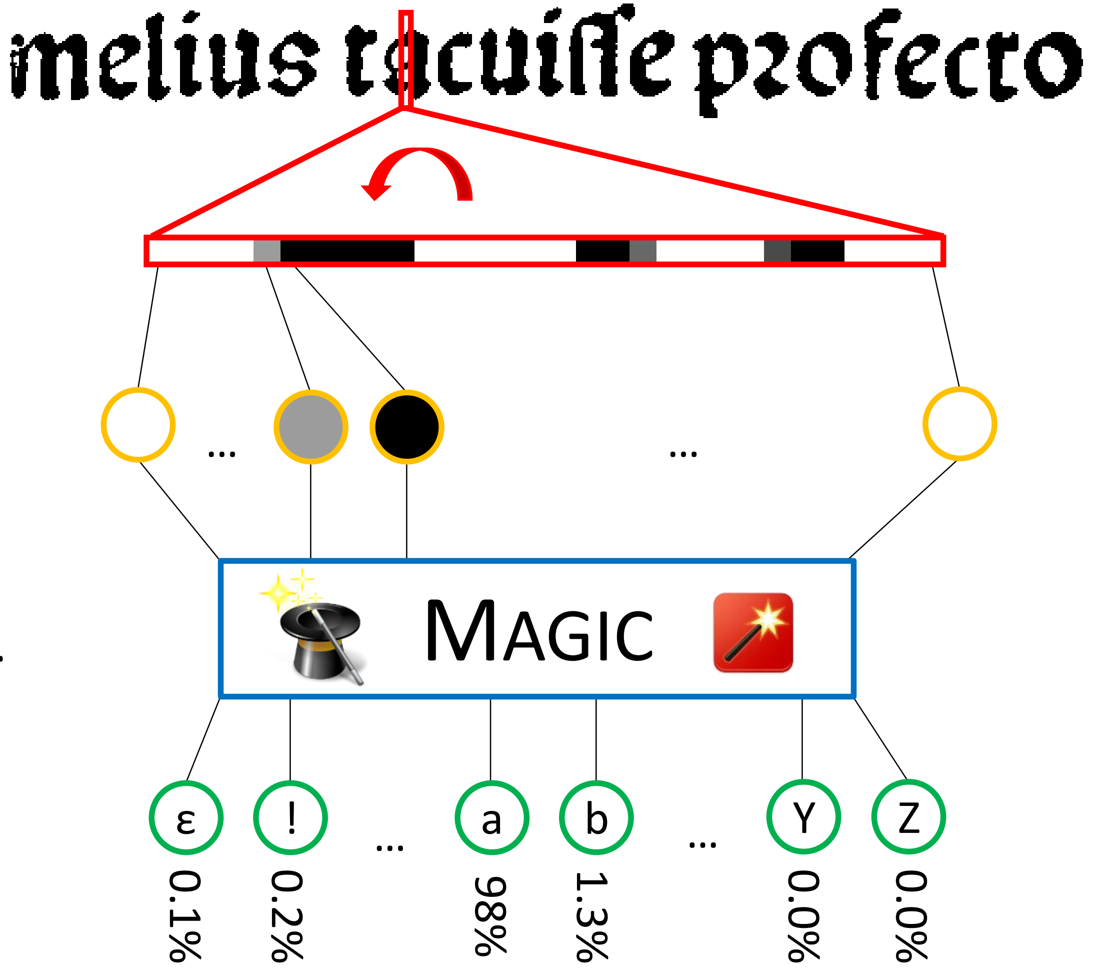
- Case study on ten (mostly) 19th century Fraktur models.
- OCR4all considerably outperformed ABBYY (over 80% less errors).
- To keep in mind:
  - Moderate layout.
  - Very strong mixed model available:
    Reul, Springmann, Wick, Puppe: State of the Art Optical Character Recognition of 19th Century Fraktur Scripts using Open Source Engines.
  - ABBYY Fraktur/Gothic model supposedly weaker than its Antiqua counterpart.
- Still very promising results .
- Strong Antiqua mixed model needed (GT!).



Book	ABBYY	OCR4all	ErrRed.
F1781	2.9	0.60	79.3
F1803	27	4.89	81.9
F1810	3.8	0.61	84.0
F1818	10	1.35	86.6
F1826	1.1	0.06	94.4
F1848	0.93	0.20	78.5
F1851	1.0	0.16	84.0
F1855	4.0	0.33	91.8
F1865	1.6	0.18	88.8
F1870	0.48	0.13	72.9
Avg.	**5.3**	**0.85**	**84.2**

Character Recognition – Methodology

- Line image is normalized to a certain height (default: 48px) and cut into slices of 1px width.
- Model stores a Neural Network consisting of:
  - input nodes (# line height): pixel values of the slice.
  - Several convolutional and one LSTM layer(s).
  - 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.



Ground Truth Production – Guidelines I

- The network can learn pretty much anything …
  - Resolving ligatures: “ﬀl” → “ffl” (“f” + “f” + “l”)
  - Replacing symbols by their meaning: “&” → “et”.
  - Labeling typographical properties instead of characters: “Hello **World**” → “nnnnn bbbbb”.
  - …

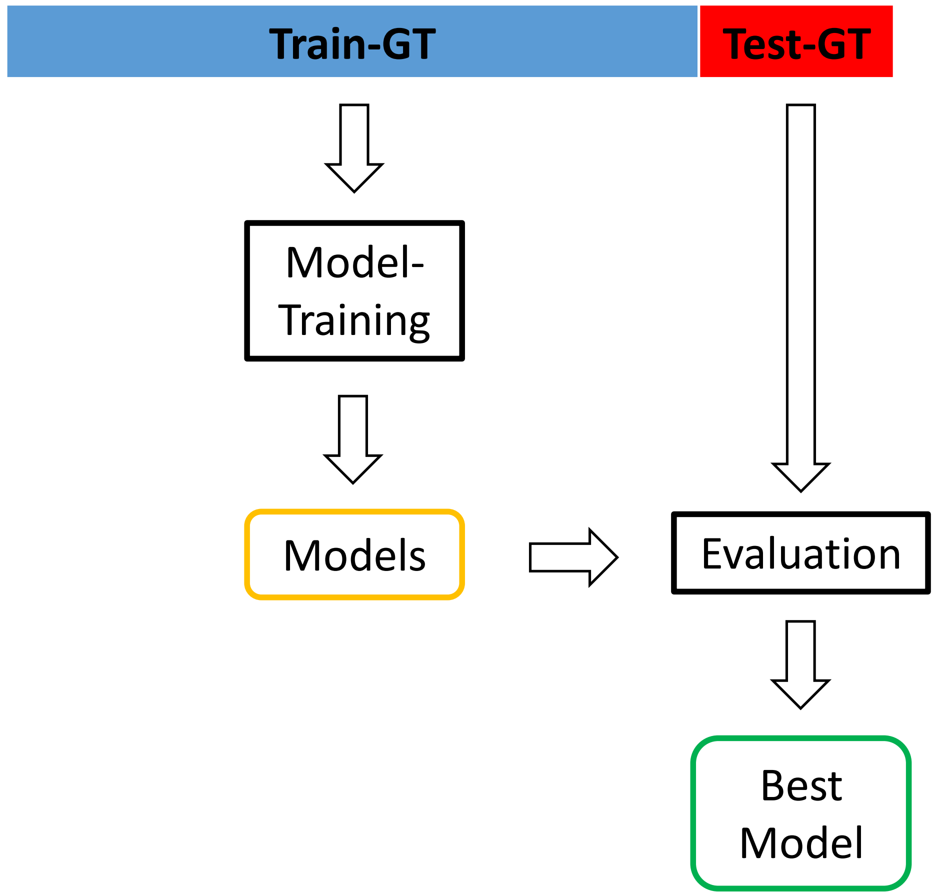
  … if it is provided **consistent training examples**.
- N : 1 is fine, e.g. mapping both glyphs “ſ” and “s” to “s”.
- … but 1 : N confuses the network.
- See the Medieval Unicode Font Initiative (MUFI) for recommendations regarding the representation of characters.

Ground Truth Production – Guidelines II

- Only use what you really need!
- Keep the codec …
  - as small as possible.
  - as big as needed.
- Regularisation scripts can help to automatically adjust the GT/codec.
  - Resolving ligatures, regularizing hyphens lengths, …
  - Not yet natively integrated into OCR4all.
  - Often a one way street!
    - Be careful when applying.
    - Consider backups.
    - Get help if needed.

OCR Model Training (OMT) – Basics

- **Input:** line-based GT (image/transcription pairs).
  Output: Calamari model(s).
- General methodology:
  - Random initialization of the network weights.
  - Current network predicts a line.
  - Compare result to GT and update weights accordingly.
  - Repeat prediction and update step.
- Training procedure (start training!):
  - 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.



OMT – Early Stopping: Theory

- Deep neural networks possess a great number of trainable parameters.
  → Exhaustive search for best settings impossible or at least highly impractical.
  → Training is never really finished.
- Assumption: the closer our models get to the global optimum, the smaller the chances to find a (significantly) better model within a reasonable time frame.
- Solution: early stopping.
  - Always after a fixed number of training steps (*early stopping frequency*):
    - Evaluate the actual model on the validation set.
    - If achieved accuracy > current top value: store new best accuracy and model.
  - Continue training until X (*early stopping nbest*) new models performed worse than the current best model.

OMT – Early Stopping: Example

- Training using 1,000 lines of GT.
- Default Calamari settings:
  - Training set: 800 lines.
  - Validation set: 200 lines.
  - Early stopping frequency: 400 (half an epoch).
  - Early stopping nbest: 5.

Model ID	Training Step	Accuracy on Validation Set	Best Model	Remaining Nbest
0	400	94.22%	0	5
1	800	96.11%	1	5
2	1,200	95.97%	1	4
3	1,600	98.71%	3	5
4	2,000	97.23%	3	4
5	2,400	98.45%	3	3
6	2,800	98.71%	3	2
7	3,200	98.00%	3	1
8	3,600	98.38%	3	0

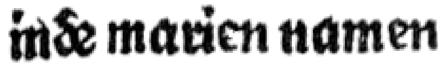
Early stopping → best model: 3 with validation acc. of 98.71%.

OMT – Pretraining

- The default approach is to start the training from scratch and initialize the weights randomly, leading to …
  - (considerably) longer training times.
  - exclusive usage of the information provided by the new GT.
- Existing (mixed) models can provide a valid starting point.
  - E.g. the available OCR4all standard models.
  - The codecs of the models and the GT should somewhat match.
  - Even models, that don’t match the printing type can be very helpful.
- Reul, Wick, Springmann, Puppe: Transfer Learning for OCRopus Model Training on Early Printed Books.

OMT – Voting: General Idea and Cross Fold Training

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


inde marien namen

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

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Model 1	🟥	🟦	🟦	🟦	🟦
Model 2	🟦	🟥	🟦	🟦	🟦
Model 3	🟦	🟦	🟥	🟦	🟦
Model 4	🟦	🟦	🟦	🟥	🟦
Model 5	🟦	🟦	🟦	🟦	🟥

OMT – Voting: Combining the Results

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

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

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

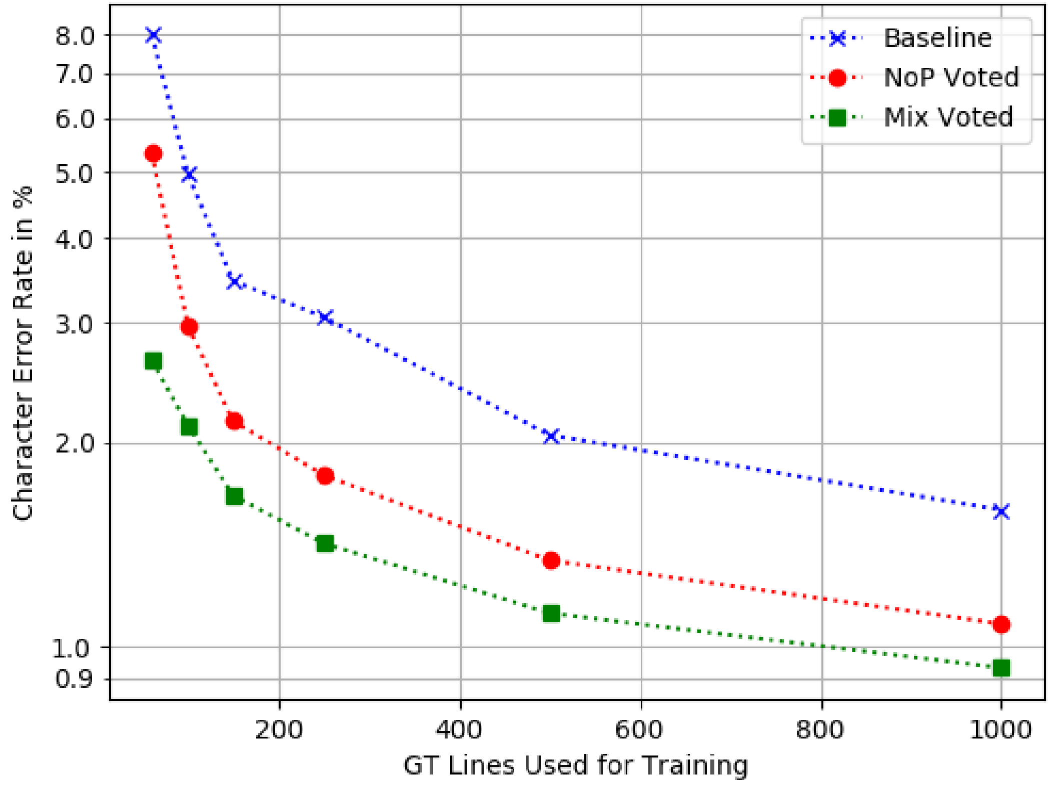
Char	x_S	x_E	Conf	Alternatives
i	120	123	87.54%	b=8.66%, f=2.94%
a	126	136	96.65%	n=45.78%, r= 23.65%, m=9.24%, k=8.32%, [...]
d	142	149	99.93%	ã=4.83%, V=4.17%, O=1.13%
e	155	160	99.15%	all alternatives $<$ 1%

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

actual output, alternative

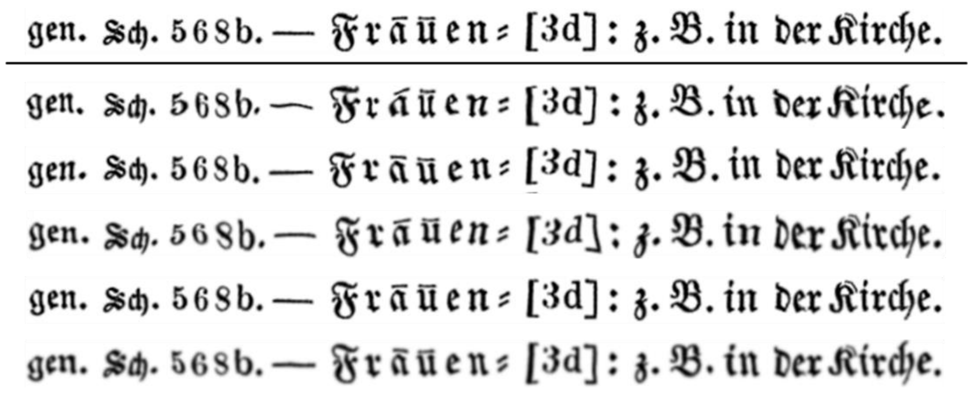
OMT – Combining Pretraining and Voting

- 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.



OMT – Data Augmentation

- In general: more GT (training examples) → better models.
- Idea: take existing GT images and slightly alter them to generate more training examples.
- Originally implemented in OCRopus3 submodule OCRodeg.
- Recommended two step approach:
  - Train on all available data (real + aug).
  - Refine the models by training only on real data, using the models from step 1 as a starting point.
  - Natively integrated into Calamari.


gen. Sch. 568b. — Frāūen⸗ [3d]: z. B. in der Kirche.

gen. Sch. 568b. — Frāūen⸗ [3d]: z. B. in der Kirche.

gen. Sch. 568b. — Frāūen⸗ [3d]: z. B. in der Kirche.

gen. Sch. 568b. — Frāūen⸗ [3d]: z. B. in der Kirche.

gen. Sch. 568b. — Frāūen⸗ [3d]: z. B. in der Kirche.

gen. Sch. 568b. — Frāūen⸗ [3d]: z. B. in der Kirche.

OMT – Iterative Training Approach

- Problem setting:
  - Generating GT is a cumbersome task, especially when transcribing from scratch.
  - Correcting an existing OCR output can be (considerably) more efficient, depending on the quality.
- Idea: minimizing the required human effort by increasing the computational effort.
- Solution: iterative training:
  1. Transcribe a small number of lines from scratch or correct the output of a fitting mixed model, if available.
  2. Train a book specific model/voting ensemble using all available GT.
  3. Apply the model/voting ensemble to further lines.
  4. Correct the output.
  5. Repeat steps 2-4.

OMT – Guidelines I

No “optimal” solution. Best strategy depends on several factors:

- The material at hand.
- Temporal flexibility of the user.
- Available hardware for training.
- Quality and error distribution of the available OCR output.
- Overall goal: correcting the entire book ↔ stopping as soon as a certain quality is reached.
- …

→ Experiment and figure out what works best for your material and in your setting.

→ Share and discuss your experiences with other users!

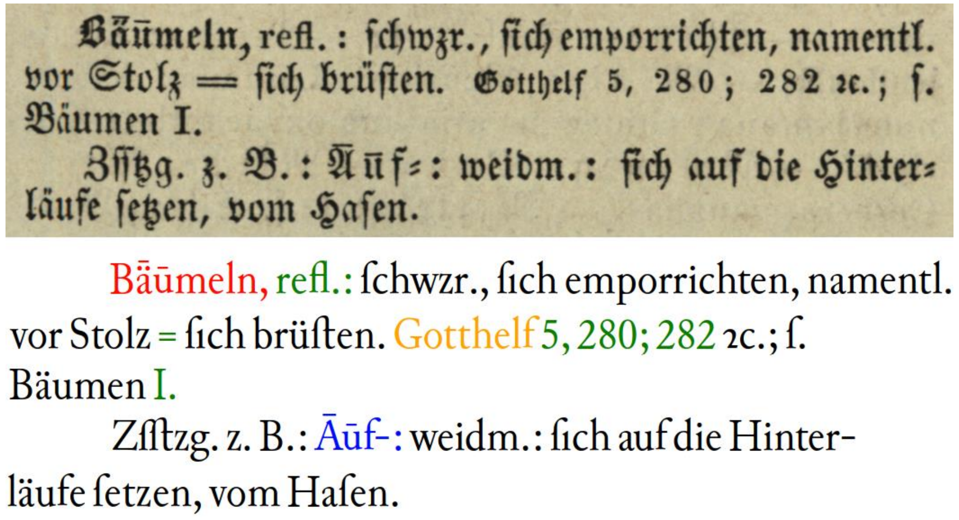
OMT – Guidelines II

Solid and proven default approach for book specific training:

- Start out by transcribing/correcting the OCR output for one or two pages (30-60 lines) and approximately double the amount of pages during each iteration.
- Try to distribute your GT over the entire book.
- Voting: use whenever feasible as it gifts you accuracy.
- Pretraining: highly recommended if somewhat fitting (codec!) models are available.
- Data augmentation:
  - Recommended during the first few iterations.
  - Gets less effective and efficient with a growing amount of GT.
  - Won't hurt, at least when using the two step approach.

Typography Recognition I

- Goal: Gathering the content of a historical lexicon by performing a precise automatic recognition of typographical attributes.
- The typography within a dictionary represents semantic meaning.
- Material: Daniel Sanders' *Wörterbuch der deutschen Sprache*.
- Idea: treat the task as two separate sequence classification problems: OCR and typography recognition.



Bāūmeln, refl.: ſchwzr., ſich emporrichten, namentl. vor Stolz = ſich brüſten. Gotthelf 5, 280; 282 ꝛc.; ſ. Bäumen I.

Zſſtzg. z. B.: Āūf-: weidm.: ſich auf die Hinter-läufe ſetzen, vom Haſen.

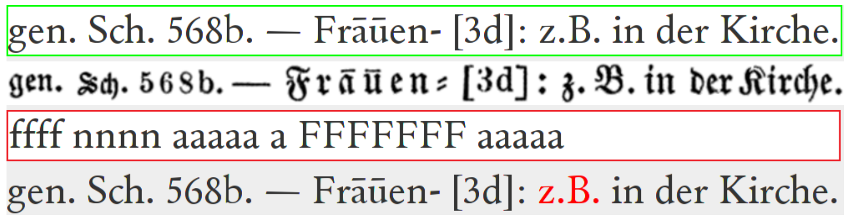
Bāūmeln, refl.: ſchwzr., ſich emporrichten, namentl. vor Stolz = ſich brüſten. Gotthelf 5, 280; 282 ꝛc.; ſ. Bäumen I.

Zſſtzg. z. B.: Āūf-: weidm.: ſich auf die Hinter-läufe ſetzen, vom Haſen.

Recognition input (top) and output (bottom) of an example article, showing different typography labels.

Typography Recognition II

- Very efficient typography GT production by building from the OCR GT.
- Train separate OCR and typography models, apply them individually, and combine the outputs.
- Results using 765 training lines:
  - OCR CER: 0.35%
  - Typography tagging: <1.5% coWER
    - **co**llapsed **W**ord **E**rror **R**ate.
    - aaaa fffffffff fff NNNNN ffff → affNf

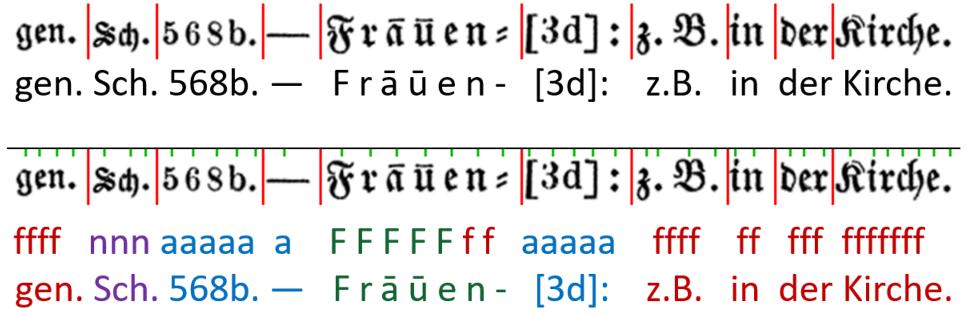


gen. Sch. 568b. — Frāūen- [3d]: z.B. in der Kirche.

ffff nnnn aaaaa a FFFFFFF aaaaa

gen. Sch. 568b. — Frāūen- [3d]: z.B. in der Kirche.

Assisted typography GT production. From top to bottom: OCR GT, line image, typography GT (in progress), helper.



gen. Sch. 568b. — F r ā ū e n - [3d]: z.B. in der Kirche.

ffff nnn aaaaa a F F F F F f f aaaaa ffff ff fff fffffff

gen. Sch. 568b. — F r ā ū e n - [3d]: z.B. in der Kirche.

Typography alignment for an example line. From top to bottom: Line image with OCR whitespace positions (|). Textual OCR output. Typography output with character positions ('). (Slightly flawed) textual typography output. Final combined output with typography classes assigned on word level.