AI in-the-loop: the AI-assisted digitisation workflow of the Fondazione Zeri's auction catalogues collection

Paolo Bonora[1,†], Tommaso Vitale[1,†]

[1]*Department of Classical Philology and Italian Studies - ALMA MATER STUDIORUM - Università di Bologna - Via Zamboni, 32, 40126 Bologna, Italia*

Abstract

This paper introduces an AI-assisted digitisation workflow tailored for auction catalogues. The application of Artificial Intelligence (AI) in this peculiar context serves a dual purpose: it functions both as a core component of the processing pipeline and as an assistant throughout its development. The workflow seamlessly integrates computer vision models for image processing, Multimodal Large Language Models (MLLMs) for feature analysis, and human-in-the-loop validation within a unified software solution. The development of the pipeline's core tools relied on an AI-assisted IDE to overcome resource limitations and respect temporal constraints. Experimental results underscore the transformative impact of AI as a key enabler, significantly advancing both the efficiency and the quality of digitisation processes.

Keywords

Digitisation workflow, AI coding, AI image processing, auction catalogues.

1. Introduction

Auction catalogues constitute a significant source for the study of art markets, provenance research, and cultural heritage. They represent a category of cultural heritage documentation, combining textual and visual information. Their digitisation requires not only technical transformations but also semantic validation to support discovery, interoperability, and preservation. Traditional digitisation pipelines focus on image capture and conversion, but for complex documents such as catalogues, additional layers such as page classification, structure, and indexing are essential to make the contents fully accessible.

Given the dimension of the collection (over 1,900 catalogues[1] with an estimated number of pages of up to 150k) and the project timeline (less than 10 months to complete the digitisation phase), a significant level of automation was identified as a prerequisite to accomplish the task. Balancing production speed and quality, intended both in terms of reproduction accuracy and metadata consistency, is challenging.

We modelled the workflow as a multi-step procedure, starting from the output of the acquisition of the catalogue to the delivery of the digital collection to the client.

The workflow design emphasises quality assurance, semantic structuring, and interoperability, ensuring that digitised catalogues meet archival standards and are suitable for long-term preservation and dissemination. A dual modelling approach was followed to track the dynamic states of digital objects and the procedural activities required to transition between them, supporting active monitoring by operators.

To ensure transparency and auditability, we adopted Business Process Model and Notation (BPMN) and Unified Modelling Language (UML) to draw the diagrams: the business process diagram, which

IRCDL 2026: 22nd Conference on Information and Research Science Connecting to Digital and Library Science, February 19-20, 2026, Modena, Italy

[†]Paolo Bonora authored Sections 1, 2, and 3.2. Tommaso Vitale authored Section 3.1. Both authors contributed equally to the remaining parts of the manuscript.

paolo.bonora@unibo.it (P. Bonora); tommaso.vitale@unibo.it (T. Vitale)

https://www.unibo.it/sitoweb/paolo.bonora/en (P. Bonora); https://lab.ficlit.unibo.it/tommaso.vitale (T. Vitale)

0000-0001-8337-3379 (P. Bonora); 0000-0003-3994-8985 (T. Vitale)

[1]Catalogues are in different languages: German (54%), French (29%), Italian (9%), and English (8%).

details the procedural flow, decision points, and termination conditions; the state diagram, which captures the discrete states of digital objects, metadata and their transitions. Figure 1 depicts the activity diagram, and Figure 2 represents the state diagram.

The diagrams provide a holistic view of the workflow that is designed to balance technical efficiency with quality assurance, ensuring that digitised catalogues are both authentic archival objects and trustworthy digital resources.

2. The Workflow

The workflow begins with the shooting stage, which produces RAW image files. These are converted first into TIF masters, serving as archival preservation copies, and then into JPG derivatives used for access and dissemination. An initial shooting validation step ensures that the images meet quality benchmarks for resolution, colour fidelity, and completeness. Failure results in a Rejected state, ensuring that only valid images will proceed. Then, a Computer Vision (CV) module perform image processing steps: Cropping, which removes borders and extraneous elements, and Splitting, which separates the two-page composite captures. Both steps include validation; failures result in Cropping rejected or Splitting rejected states. Once validated, pages are processed by a Multimodal LLM (MLLM) to assign a document-level category and identify functional page types (cover, table of contents, lot descriptions, etc.). The result is a tentative logical structure of the catalogue that must be validated by operators in the Structural Validation step. As the structure is validated, pages' filenames are then renamed according to archival conventions, ensuring traceability and interoperability. Final steps are designed to support quality assurance of the final product before delivery. Catalogue's pages metadata are indexed to support access through IIIF services and the final inspection. After QA inspection, the catalogue is ready for delivery. Figure 1 is a BPMN representation of the process [1].

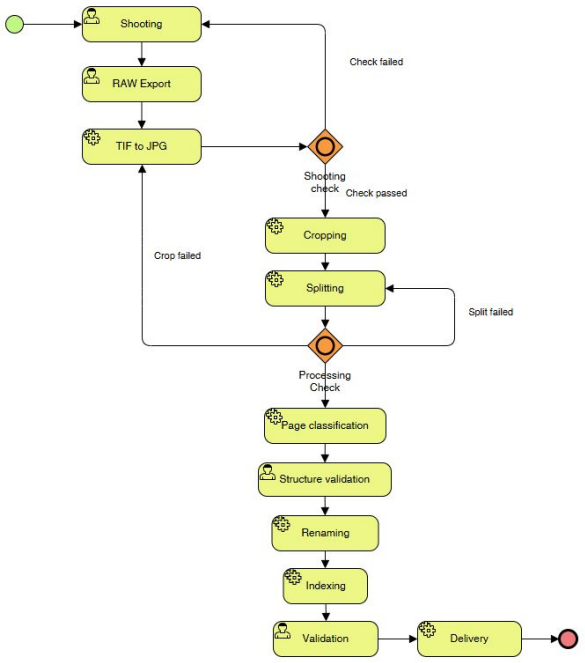

Figure 1: BPMN representation of the digitisation workflow.

The following table synthesises the digitisation workflow by aligning each step with its primary function and the associated validation checkpoint. This mapping highlights how technical transformations and semantic enrichments, made by AI and humans, are systematically controlled to ensure quality.

Table 1
Digital Workflow Steps and Validation

Workflow Step	Primary Function	Validation Checkpoint	Actor
Shooting (RAW capture)	Technical acquisition of source images[2]	Shooting validation (resolution, completeness, fidelity)	Human
RAW to TIF conversion	Creation of the archival master copy	File integrity and format compliance	Human
TIF to JPG conversion	Generation of access derivatives	Visual inspection and checksum verification	Procedural
Cropping	Removal of borders and extraneous content	Cropping validation (content completeness, correct framing)	AI
Splitting	Separation of multi-page or composite images	Splitting validation (page count, logical separation)	AI
Page Classification	Semantic identification of page types (cover, lot, index)	Page-type validation against controlled vocabulary	AI
Structural Validation	Verification of logical document structure	Schema validation (expected sequence, hierarchy)	Human
Renaming	Application of archival naming conventions	Filename compliance check (traceability, uniqueness)	Procedural
Indexing	Metadata assignment and identifier linking	Metadata validation (field completeness, controlled terms)	Procedural
Final Validation	Cross-check of all technical and semantic outputs	Delivery readiness check	Human
Ready for Delivery	Dissemination via repository or IIIF	Conformance to delivery specifications	Procedural

Figure 2 is a UML representation of the state transition according to the BPMN model. The state transitions are logged in a log file that traces the lifecycle of the catalogue from the shooting to the delivery. Iterations and loops due to QA failures are logged to trace critical issues.

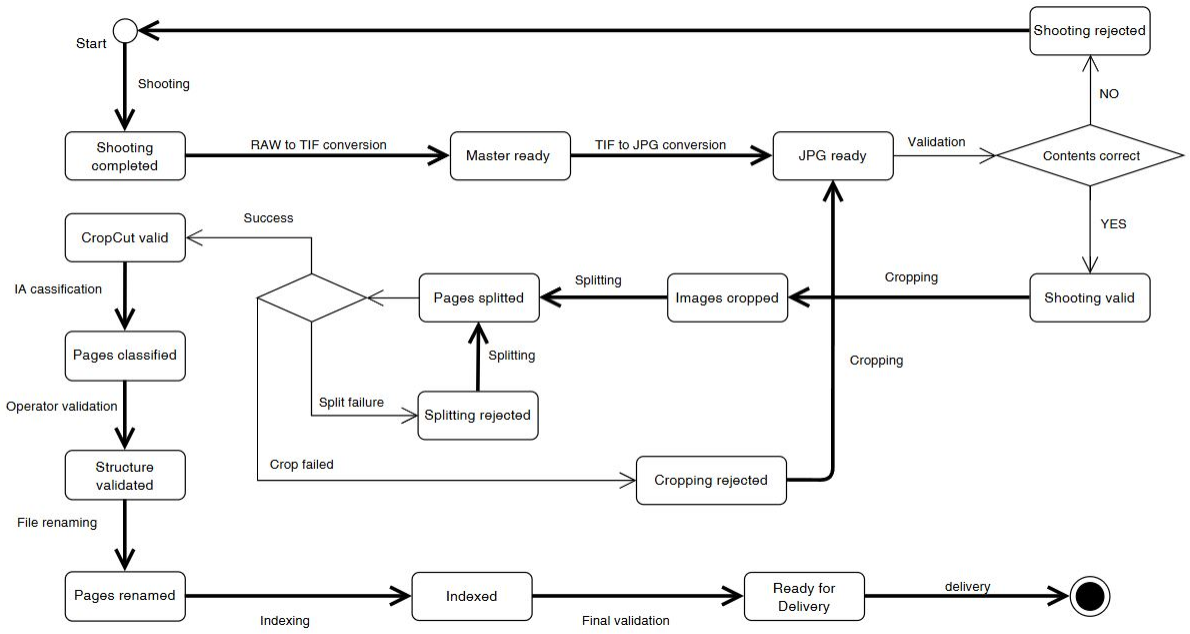

Figure 2: State transition diagram.

The design addresses some critical aspects of the workflow. Every technical transformation (e.g., cropping, splitting) is accompanied by a visual or structural validation step, ensuring that errors are

detected early and documented. Validation checkpoints create a transparent trail of decisions and outcomes, supporting reproducibility and governance. In this context, the AI plays a key role in terms of scalability and sustainability of the process (see Table 1). Each transition is logged within a log file, which represents the journey of each catalogue through the process route. The main purpose is to mark transitions and serve as the reference for routing catalogues. It also documents the output of each step and logs activities of human and automated actors. Both are used for quality assurance and performance monitoring.

3. The Tools

To implement the workflow, we developed a software solution that integrates the various phases into a single, coherent flow of activities. Individual tasks can be performed asynchronously, a relevant feature when coordinating computational activities performed in batch mode with supervision and quality control activities carried out by human operators. We designed a component-based system architecture, logically partitioned into three primary, high-level modules: Digit-Review, Digit-Check, and the MLLM Module. The system employs a modular, service-oriented design where encapsulated components communicate via specified interfaces. Operators access dedicated web UIs for monitoring and carrying out tasks. Figure 3 depicts the system's architecture.

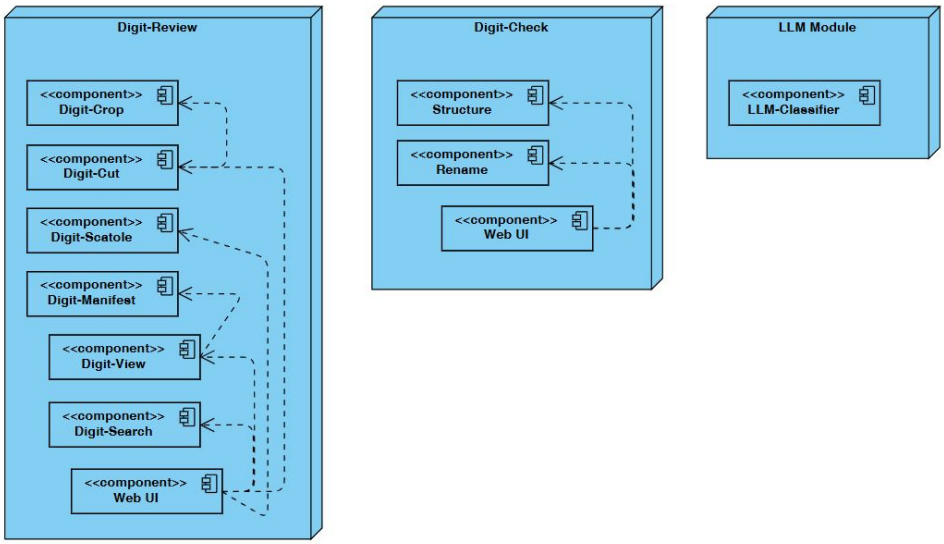

Figure 3: UML components and packages diagram.

The Digit-Review Module is dedicated to primary image-related activities: editing, viewing, and searching. It is internally composed of seven distinct components: Digit-Crop, Digit-Cut, Digit-Scatole, Digit-Manifest, Digit-View, Digit-Search. The internal architecture of this module is centralised around its Web UI component. This component acts as the primary consumer and orchestrator, holding dependencies on all six other components within the module. This structure suggests the Web UI drives the application's core functionality by invoking services for cropping, cutting, manifest management, viewing, and searching. The module implements the functionalities servicing the following workflow steps: Cropping, Splitting, Indexing, plus IIIF presentation and image access and search APIs. The Digit-Check module represents a distinct application dedicated to metadata management and quality assurance. It contains three components: Structure, Rename, and its own Web UI component. Implements the Page Classification, Structural Validation, and Renaming functionalities. The LLM module is designed as a specialised, self-contained service running on dedicated hardware. The LLM-Classifier component encapsulates the Pixtral multimodal LLM [2], which is responsible for classifying catalogue pages, the initial task of the catalogue's structure analysis. The solution relies on AI to achieve two

objectives: image layout analysis to extract the required information for editing the image into the format required for publication, and single-page features analysis to gather the basic set of metadata necessary to produce catalogue structural metadata (workflow steps: Page Classification and Structural Validation). The Split and Crop procedures, which require an accurate measurement of the page layout, would necessitate significant human effort. As consistency in results is also needed, the same amount of effort should be dedicated to quality checks. The development of a completely automated CV procedure frees operators from initial interventions and allows for the postponement of the QA. Extracting the structural features of the catalogue is challenging due to the very diverse nature of these documents. At the same time, accuracy is determinant in naming pages consistently and allows further metadata extraction procedures. In this case, the role played by the MLLM will enable operators to start from a draft structure that requires some minor corrections and final validation.

3.1. Computer Vision for layout metadata extraction

To automate the processing of digitised images, four complementary tools were developed and integrated into the workflow. Two Python-based components named Digit-Crop and Digit-Cut were designed to perform automatic image analysis. The first detects and removes black borders surrounding scanned images, while the second identifies the book spine area and splits the original image into two separate pages (left and right). In addition to these automated procedures, two graphical tools were implemented to enable human operators to visually inspect the results and manually correct or refine the output when necessary. This combined approach ensures both the efficiency of automated processing and the accuracy that can only be achieved through human supervision.

The script SurfaceCrop.py automatically detects and crops the main surface area within a set of images using OpenCV[3] [3]. Each image is converted to the LAB colour space and processed through Gaussian blurring and K-means clustering to segment foreground and background regions. The resulting clusters are thresholded using Otsu's method, followed by morphological operations to refine the segmentation mask. The largest contour is extracted to compute an adaptive bounding box with a configurable border. Finally, images are cropped accordingly, logged to a CSV file, and optionally checked for abnormal asymmetries in the cropping margins for the human operator review (Figure 4 depicts the corresponding UI).

Figure 4: UI for the validation of the cropping results.

The Digit-Cut component automatically detects and splits an image into left and right sections based on the estimated "coastline" or central dividing edge, using OpenCV. Each image is converted to grayscale and processed with the Canny edge detector to locate strong vertical edge responses. The algorithm identifies the dominant peak near the image centre by analysing the column-wise edge intensity profile, allowing for a tolerance in deviation and an optional offset. The detected division point is used to crop the image into two parts, which are saved separately. Results and geometric parameters are recorded in a CSV file, ensuring reproducibility and control of boundary detection accuracy. The operator visually evaluates the geometry detection through an overlay and edits it if it is not accurate

[3]opencv-python 4.11.0.86 https://github.com/opencv/opencv-python

(Figure 5 depicts the corresponding UI).

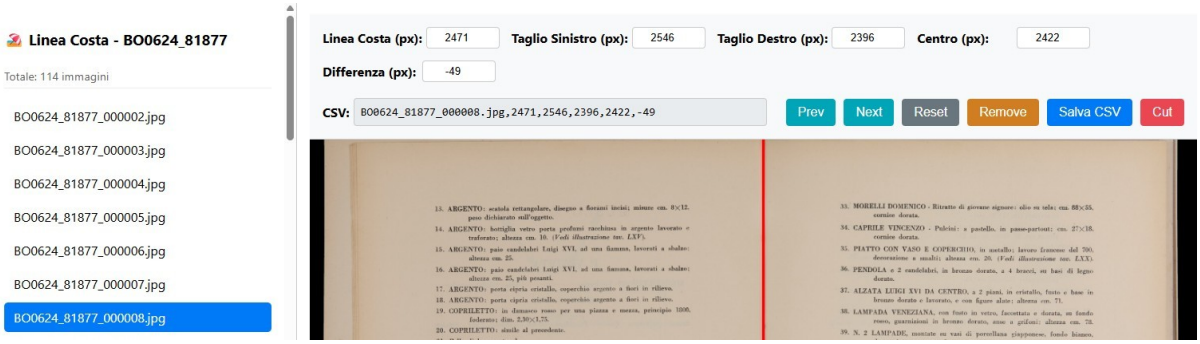

Figure 5: UI for the validation of the splitting results.

3.2. MLLMs for page features analysis

One of the most challenging tasks is to determine the structure of the catalogue. Determining the different parts of the catalogue represented in the scans, such as the covers (front and back), the table of contents, item description pages, etc., is mandatory to order the file sequence and names. Starting from a basic taxonomy, aligned with the given file naming convention, we designed a prompt strategy to make Pixtral 124B MLLMs[4] identify for each page the corresponding class, the page number (if present), and whether the page presents some graphical elements.

The prompt is the following:

```
prompt = f"""Here is the image with its corresponding filename:
- {filename}

[IMG]

The image is a single page of an auction catalogue. Your task is threefold:

1. Classify the part of the catalogue it represents. You MUST choose one of the
    following categories EXACTLY as written, without any deviation, misspellings, or
    variations, providing a precise and distinct classification:
    - "External Front Cover": The very first page/cover of the entire catalogue,
    typically sturdy and often decorative, showing the main title, auction house, and
    year.
    - "Title Page": The page immediately following the External Front Cover (or the
    first page if there's no sturdy external cover), often containing more detailed
    information like the full title, auction dates, location, and details about the
    collection or auctioneers.
    - "Table of Contents": A page listing sections or contents of the catalogue, usually
     with page numbers.
    - "Item Description": Pages primarily showing detailed descriptions of items for
    sale, often numbered.
    - "External Back Cover": The very last page/cover of the entire catalogue, usually
    sturdy and often containing contact information, logos, or a summary.
    - "Blank Page": A page with no significant content.
    - "Guard Page": An additional blank or printed leaf (a single sheet forming two
    pages) that is inserted into a book, often before the title page, illustrations, or
    delicate plates.
    - "Tissue": A thin, translucent, protective sheet, often made of paper or a similar
    material, that is placed between illustrations or facing pages.
    - "Reference": A page containing a color checker.

    You MUST use the EXACT ORIGINAL FILENAME provided above for the 'filename' field.

2. Determine if the page contains any graphical elements (e.g., images, photos,
    illustrations, diagrams). If it does, set 'hasGraphics' to true; otherwise, set it
    to false. You MUST include this field.
```

[4]We adopted the quantised version by RedHatAI: Pixtral-Large-Instruct-2411-hf-quantized.w4a16 which run on a local infrastructure based on a dual NVIDIA L40s GPUs.

```
Again, use the EXACT ORIGINAL FILENAME for the 'filename' field.

3. Finally, identify the main page number of the catalogue, if clearly visible on the
    page. It must be at the top or bottom end of the page. Provide it as a string in the
     'page_number' field. If no clear page number is visible or if the page type does
    not typically have one (e.g., covers), set this field to "N/A".

Your entire response MUST be a single JSON array containing a single object. No other
    text, explanations, or code blocks are allowed outside this JSON array.
All string values within the JSON (especially item descriptions) must be enclosed in
    standard double quotes (") and any internal double quotes, backslashes, or newlines
    MUST be properly escaped (e.g., \\\" for double quotes, \\\\ for backslashes, \\n
    for newlines, \\r for carriage returns).
Omit the 'item_description' field if not an item description page.

**The value for "classification" MUST be one and only one of the following exact strings
    : "External Front Cover", "Title Page", "Table of Contents", "Item Description", "
    External Back Cover", "Blank Page", "Guard Page", "Tissue".**

The structure should be as follows:
[
  {{
    "filename": "{filename}",
    "classification": "<classification>",
    "hasGraphics": <true/false>,
    "page_number": "<extracted_page_number_string_if_applicable_or_N/A>"
  }}
]
BEGIN_JSON_OUTPUT"""
```

The results are saved in a single JSON file for each catalogue (an excerpt is presented in Figure 6[5]).


```
{
    "filename": "BO0624_81875_000001.jpg",
    "classification": "Front Cover",
    "hasGraphics": true,
    "page_number": "N/A",
    "description": "Nessuna descrizione disponibile per questa immagine.",
    "is_table": false,
    "page_number_is_valid": null,
    "page_number_new": "_copertina_anteriore"
}
```


Figure 6: Excerpt of the JSON file representing the catalogue's structure.

This preliminary draft catalogue structure is then presented to the operator in the Digit-Check web application for validation (see Figure 7)[6].

The human review stage serves two primary functions: identifying inconsistencies and integrating structural metadata[7]. This manual step is a prerequisite for executing the page numbering algorithm. The algorithm assigns page numbers based on page typology and content, such as differentiating between pages containing images and those comprising solely text. An operator then evaluates the algorithmic output and modifies or supplements the results where needed. Final validation of the structural and page numbering metadata is a two-stage process. It begins with a preliminary, rule-based automatic assessment, followed by a final validation by the operator. The resulting metadata, generated

[5]The page is identified by the corresponding image filename, which serves as a unique key derived from the library identifier, the catalogue inventory number, the incremental shoot number, and the appendix generated during the page-splitting procedure. The classification, page number, and table indicator are obtained from the MLLM output, whereas the revised page number is produced by the numbering algorithm executed by the operator following validation.

[6]No significant hallucination was encountered, just minor output oscillation in class names (lower cases when capital expected or word drops, i.e.: "External Cover" instead of "External Front Cover" as stated in the prompt).

[7]Typical errors affect "Table of Contents" and "Title Page", often confused with "Item Description". Pages that can be discriminated by their position, such as "Blank Pages" and "Guard Pages", are often misclassified (the prompt processes pages one by one without reference to the absolute position within the catalogue).

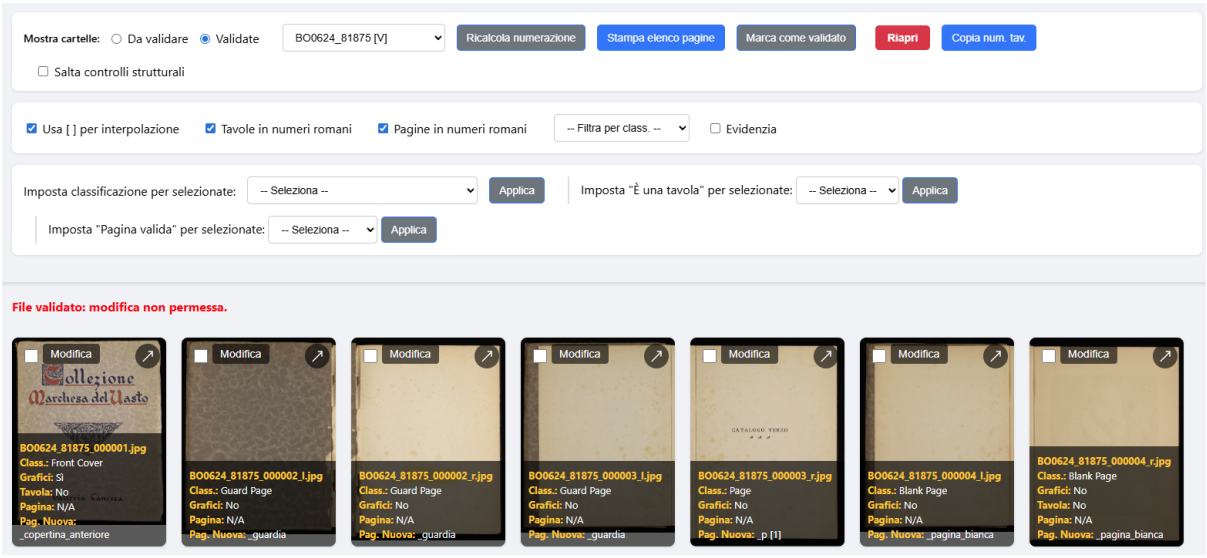

Figure 7: UI for the validation of the catalogue's structure.

through this synergy between the AI tool and the human operator, constitutes the foundation upon which all subsequent workflow steps depend.

3.3. LLMs for coding

Given the scale of the expected output and the inherent complexity of the initial requirements, an agile development methodology was adopted. This approach is based on the rapid prototyping of applications designed to support discrete workflow steps. To mitigate resource constraints and adhere to a stringent project timeline, AI-powered coding assistants were integrated from the outset of the design phase, playing a crucial role in the solution architecture. A multi-faceted AI strategy was employed. The primary tool utilised was the Microsoft Copilot integration within the VS Code IDE. In instances where the integrated models proved insufficient or a comparative analysis was necessary, supplemental models such as Gemini Pro 2.5 and Claude Sonnet 3.x were leveraged. While a formal assessment of AI assistant effectiveness is outside the scope of this discussion, their role in accelerating the prototyping cycle and subsequent refinement was critical. This integration enabled the testing of new features and advancements in a production-like environment, with feedback often being integrated within the same working day. This methodology significantly expedited the development process, facilitating the timely delivery of efficient software solutions to support production activities from the project's inception. This AI-assisted approach was extended to post-deployment maintenance and tuning phases. Notably, the coding assistants were also employed to design and implement the validation algorithms required to integrate human-in-the-loop quality inspections at various workflow stages. A rough estimation of the total effort needed for the development of the software tools indicates that at least 82 person-months have been invested[8]. Given that the team consists of the two authors, the development effort achieved in 6 months is approximately six times greater than what would have been achievable without the support of AI.

4. Discussion

The role that AI played in the digitisation process is twofold: it automates two critical steps of the post-production process while accelerating the development of the software solution that manages it. The role of AI in the digitisation process is twofold: it automates two critical post-production

[8]This estimation is based on the number of lines of code (LOC) retrieved from the project Git repository using `git2effort` [4], assuming an average productivity of 500 LOC per month per team member [5].

steps and accelerates the development of the software solution that manages them. The workflow we presented has been running since April 2025. As of today, 1585 catalogues have been processed [9], amounting to a total of 83263 images. To assess the accuracy of the computer-vision–based Cropping and Splitting phases, we can note that within the second and third batch, 285 catalogues were flagged as "Cropping rejected" and 194 as "Splitting rejected." 27 catalogues required a second run of the M-LLM

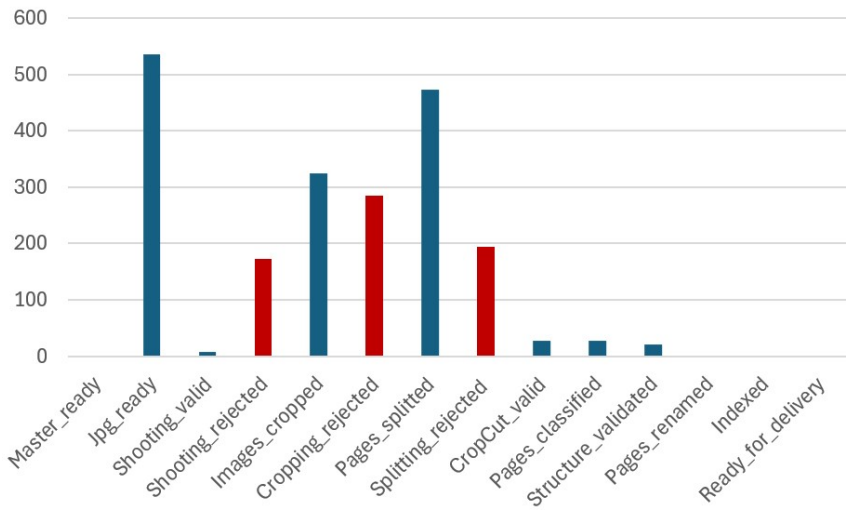

Figure 8: Deviation from optimal route of the second and third batch.

page classification after the structure was modified mainly due to new pages addiction. 21 catalogues required more than one round of human validation before being approved. Finally, 263 catalogues out of 1048 followed the optimal processing route, which consists of 11 steps. The longest route required 29 steps, while the average route involved 13,56 steps.

5. Conclusion

Integrating AI tools within a structured process is crucial for achieving significant improvements in overall efficiency and producing results that would be unattainable through a purely human-in-the-loop workflow. While the human element remains unavoidable for the present, the process itself must be properly designed to allow the AI to play a truly effective role. Looking ahead, the trajectory points toward a fully agentic framework in which AI becomes the central player of digitisation's post-production workflow, while humans shift into a supervisory role, serving as referees who ensure fairness, compliance, and strategic oversight.

Acknowledgments

The experimentation described in this paper was carried out at ADLab, the analogue-to-digital conversion laboratory of the Department of Classical Philology and Italian Studies at the University of Bologna, with the support of the Digital Humanities Advanced Research Centre at the University of Bologna. The authors would like to thank the team members who contributed to the project: Silvia Samorì, Aurora Perinotti, Catalina Salguero Palacino and Marco Serra.

[9]The collection is divided into four batches, of which only the first, second and third have been completed so far. The data presented here are relative to batch 2 and 3, consisting of 1048 catalogues.

Declaration on Generative AI

During the preparation of this work, the authors used DeepL to do a grammar and spelling check. After using the tool, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] M. Geiger, S. Harrer, J. Lenhard, G. Wirtz, Bpmn 2.0: The state of support and implementation, Future Generation Computer Systems 80 (2018) 250–262. URL: https://www.sciencedirect.com/science/article/pii/S0167739X17300250. doi:https://doi.org/10.1016/j.future.2017.01.006.

[2] P. Agrawal, S. Antoniak, E. B. Hanna, B. Bout, D. Chaplot, J. Chudnovsky, D. Costa, B. D. Monicault, S. Garg, T. Gervet, S. Ghosh, A. Héliou, P. Jacob, A. Q. Jiang, K. Khandelwal, T. Lacroix, G. Lample, D. L. Casas, T. Lavril, T. L. Scao, A. Lo, W. Marshall, L. Martin, A. Mensch, P. Muddireddy, V. Nemychnikova, M. Pellat, P. V. Platen, N. Raghuraman, B. Rozière, A. Sablayrolles, L. Saulnier, R. Sauvestre, W. Shang, R. Soletskyi, L. Stewart, P. Stock, J. Studnia, S. Subramanian, S. Vaze, T. Wang, S. Yang, Pixtral 12b, 2024. URL: https://arxiv.org/abs/2410.07073. arXiv:2410.07073.

[3] A. Gangal, P. Kumar, S. Kumari, Complete scanning application using opencv, 2021. URL: https://arxiv.org/abs/2107.03700. arXiv:2107.03700.

[4] G. Robles, J. M. González-Barahona, C. Cervigón, A. Capiluppi, D. Izquierdo-Cortázar, Estimating development effort in free/open source software projects by mining software repositories: a case study of openstack, in: Proceedings of the 11th Working Conference on Mining Software Repositories, 2014, pp. 222–231.

[5] B. W. Boehm, Software engineering economics, in: Software pioneers: Contributions to software engineering, Springer, 2011, pp. 641–686.