TongGuOCR: A Layout-Aware and Token-Augmented OCR Framework for Chinese Historical Documents

Supplementary Information

Supplementary Note 1: Additional details of Layout-Aware Preprocessing

This Supplementary Note provides additional implementation details for the Layout-Aware Preprocessing module described in the main manuscript. The main manuscript describes the overall design, the penalty-based segmentation objective, and the role of Layout-Aware Preprocessing in TongGuOCR. This note gives further details on candidate block generation, penalty construction, dynamic programming, and Block Crop Refinement.

1.1 Overview

Layout-Aware Preprocessing converts an ordered text-line sequence into an ordered sequence of OCR-friendly crop images. It contains two main stages. The first stage, Recognition Block Construction, groups ordered text lines into recognition blocks by solving a layout-guided sequence segmentation problem. The second stage, Block Crop Refinement, converts each selected block into an OCR-friendly crop image by crop expansion, non-target text masking, local background filling, and square-canvas padding.

Given an ordered text-line sequence

$$L = [l_1, l_2, \ldots, l_n],$$

each text line l_i is associated with a bounding box b_i. The goal is to partition L into a sequence of recognition blocks

$$S = [B_1, B_2, \ldots, B_K],$$

where each B_t is a consecutive subsequence of L. This interval-based representation provides a structured search space for block construction and supports efficient global sequence segmentation.

1.2 Candidate block generation

Candidate recognition blocks are generated as consecutive intervals:

$$B_{i,j} = L[i:j], \quad 1 \le i < j \le n+1. \tag{1}$$

Here, $B_{i,j}$ contains text lines $l_i, l_{i+1}, \ldots, l_{j-1}$. The bounding box of a candidate block is defined as the minimum bounding rectangle (MBR) covering all text-line boxes in the interval:

$$\mathrm{BBox}(B_{i,j}) = \mathrm{MBR}(\{b_k \mid i \le k < j\}), \quad 1 \le i < j \le n+1. \tag{2}$$

Candidate enumeration is restricted by maximum interval length and extreme geometry pruning. The pruning step removes candidates that are clearly unsuitable for OCR, such as extremely large regions, very thin strip-like regions, and candidates with severe geometric inconsistency. This reduces the search space while keeping candidates that may form useful OCR inputs.

1.3 Penalty-based segmentation objective

Recognition Block Construction is formulated as a minimum-penalty sequence segmentation problem. For a valid segmentation $S = [B_1, \ldots, B_K]$, the total penalty is

$$\mathcal{P}(S) = \lambda_{\mathrm{blk}}\mathcal{P}_{\mathrm{blk}}(S) + \lambda_{\mathrm{adj}}\mathcal{P}_{\mathrm{adj}}(S) + \lambda_{\mathrm{page}}\mathcal{P}_{\mathrm{page}}(K). \tag{3}$$

The three penalty groups correspond to candidate-block quality, adjacent-block compatibility, and page-level block-number regularization. Larger penalty values indicate less desirable candidates or segmentations.

The block-level penalty group aggregates single-block penalties over all selected blocks:

$$\mathcal{P}_{\text{blk}}(S) = \sum_{t=1}^{K} p_{\text{blk}}(B_t). \tag{4}$$

The adjacent-block penalty group aggregates compatibility penalties over neighboring block pairs:

$$\mathcal{P}_{\text{adj}}(S) = \sum_{t=1}^{K-1} p_{\text{adj}}(B_t, B_{t+1}). \tag{5}$$

The page-level penalty depends on the number of selected blocks:

$$\mathcal{P}_{\text{page}}(K) = p_{\text{page}}(K). \tag{6}$$

1.4 Block-level penalty

The single-block penalty $p_{\text{blk}}(B)$ evaluates whether a candidate interval B forms an OCR-friendly recognition block. It combines geometric and layout cues related to character scale, local context, text compactness, crop shape, and line consistency:

$$\begin{aligned} p_{\text{blk}}(B) = {} & \omega_n P_n(B) + \omega_a P_a(B) + \omega_f P_f(B) \\ & + \omega_r P_r(B) + \omega_s P_s(B) + \omega_g P_g(B) \\ & + \omega_o P_o(B) + \omega_\Delta P_\Delta(B). \end{aligned} \tag{7}$$

Here, $P_n, \ldots, P_\Delta$ are normalized penalty terms, and all ω terms are non-negative weights. These terms denote semantic groups of geometric cues; in the implementation, a group may contain both soft and hard penalty components. The individual penalty terms are summarized in Supplementary Table 1.

Let $m(B)$ denote the number of text lines in B. Let

$$A(B) = \frac{\text{area}(\text{BBox}(B))}{\text{area}(I)}$$

denote the relative area of the candidate block in the page image I. The line-count term $P_n(B)$ penalizes candidates whose line count is outside the preferred range, while the area term $P_a(B)$ penalizes overly large candidates.

The text fill ratio used by $P_f(B)$ is defined as

$$\text{fill}(B) = \frac{\sum_{l_i \in B} \text{area}(b_i)}{\text{area}(\text{BBox}(B))}. \tag{8}$$

A low fill ratio indicates that the candidate contains too much blank area or combines lines from spatially inconsistent regions. The corresponding penalty $P_f(B)$ increases as the fill ratio decreases.

The aspect-ratio term $P_r(B)$ penalizes candidate boxes with highly imbalanced width and height. If the width and height of $\text{BBox}(B)$ are w_B and h_B, the aspect-ratio imbalance is measured as

$$r(B) = \max\left(\frac{w_B}{h_B}, \frac{h_B}{w_B}\right). \tag{9}$$

The strip-like term $P_s(B)$ further penalizes narrow strip-like crops, which are not suitable as OCR inputs after square padding.

The line-consistency terms $P_g(B)$ and $P_o(B)$ are computed from neighboring text-line boxes inside the candidate interval. Horizontal gaps are normalized by the page width. Vertical overlap is measured between neighboring line boxes and normalized by the smaller line height. For vertical historical document pages, lines in the same local region usually share a stable vertical band. Therefore, a sharp drop in vertical overlap suggests that the candidate is crossing into another column, table-like region, or marginal area.

The block-growth term $P_\Delta(B)$ is computed during interval expansion. It penalizes candidates whose bounding box changes abruptly when a new line is appended, such as a large increase in area or a large shift of the stable vertical band. This helps prevent unrelated lines from being merged into the same recognition block.

1.5 Adjacent-block penalty

The single adjacent-block penalty $p_{\text{adj}}(B, B')$ evaluates the geometric compatibility between two neighboring recognition blocks B and B'. Let

$$R = \text{BBox}(B), \qquad R' = \text{BBox}(B').$$

Table 1 Block-level penalty terms. Each term measures one property of a candidate recognition block.

Term	Cue	Penalized case
$P_n(B)$	Number of text lines	The candidate contains too few lines, which produces fragmented inputs, or too many lines, which makes the crop text-rich and difficult to recognize.
$P_a(B)$	Relative block area	The candidate covers an overly large region of the page and may reduce character scale after resizing.
$P_f(B)$	Text fill ratio	The candidate contains large blank regions or combines spatially inconsistent lines.
$P_r(B)$	Aspect ratio	The candidate has an overly elongated shape.
$P_s(B)$	Strip-like geometry	The candidate forms a narrow strip-like crop that is unstable after square padding.
$P_g(B)$	Horizontal gap	Neighboring lines inside the candidate have large horizontal gaps.
$P_o(B)$	Vertical overlap	Neighboring lines inside the candidate have low vertical overlap, often indicating a transition to another local region.
$P_\Delta(B)$	Block growth stability	Adding a line causes abnormal expansion of the candidate region or a large change in its stable vertical band.

The penalty is computed from geometric relations between R and R':

$$\begin{aligned} p_{\text{adj}}(B, B') = \gamma_{\text{ov}} P_{\text{ov}}(B, B') + \gamma_{\text{con}} P_{\text{con}}(B, B') \\ + \gamma_{\text{tiny}} P_{\text{tiny}}(B, B') \\ + \gamma_{\text{split}} P_{\text{split}}(B, B'). \end{aligned} \tag{10}$$

Here, $P_{\text{ov}}, P_{\text{con}}, P_{\text{tiny}}$, and P_{split} are normalized geometric penalty terms, and all γ terms are non-negative weights. The adjacent-block penalty terms are summarized in Supplementary Table 2.

The overlap term P_{ov} is based on the intersection-over-union or intersection ratio between neighboring block boxes. The containment term P_{con} penalizes cases where one block is largely inside another block. The relative-size term P_{tiny} penalizes tiny blocks attached to or embedded in much larger neighboring regions. The split-boundary term P_{split} discourages unstable boundaries between closely related line groups.

These terms reduce cross-block interference and discourage unstable partitions. They are especially useful in dense vertical pages, table-like layouts, and pages with marginal annotations, where spatial proximity alone may lead to unsuitable grouping.

1.6 Page-level penalty

The page-level penalty $p_{\text{page}}(K)$, where K is the number of selected recognition blocks, regularizes the number of recognition blocks on a page. It discourages both excessive fragmentation and the degenerate solution of using the entire page as a single block. We first define the normalized quadratic deviation from the preferred range $[K_{\min}, K_{\max}]$ as

$$q_{\text{range}}(K) = \begin{cases} \left(\dfrac{K_{\min} - K}{K_{\min}}\right)^2, & K < K_{\min}, \\ 0, & K_{\min} \le K \le K_{\max}, \\ \left(\dfrac{K - K_{\max}}{K_{\max}}\right)^2, & K > K_{\max}. \end{cases} \tag{11}$$

An additional quadratic term is applied only when the block count exceeds a soft upper cap $K_{\text{soft}} > K_{\max}$:

$$q_{\text{cap}}(K) = \left(\frac{[K - K_{\text{soft}}]_+}{K_{\text{soft}}}\right)^2. \tag{12}$$

The page-level penalty is then

$$\begin{aligned} p_{\text{page}}(K) = \rho_{\text{range}} q_{\text{range}}(K) + \rho_{\text{cap}} q_{\text{cap}}(K) \\ + \rho_{\text{single}} \mathbb{I}(K = 1). \end{aligned} \tag{13}$$

Here, $[x]_+ = \max(x, 0)$, $\mathbb{I}(\cdot)$ is the indicator function, and all ρ terms are non-negative weights. The preferred-range term penalizes block counts below $K_{\min}$ or above $K_{\max}$, the soft-cap term strengthens the penalty for severe fragmentation, and the single-block term prevents the optimization from choosing one whole-page block, which

Table 2 Adjacent-block penalty terms. Each term measures one geometric relation between neighboring block boxes.

Term	Cue	Penalized case
P_{ov}	Box overlap	Neighboring block boxes overlap excessively.
P_{con}	Containment relation	One neighboring block box is largely contained by the other.
P_{tiny}	Relative size imbalance	A tiny block is embedded in or attached to a much larger neighboring region.
P_{split}	Split-boundary stability	The split creates unstable neighboring crops or separates highly related line groups.

would weaken the purpose of Layout-Aware Preprocessing.

1.7 Dynamic programming solution

After candidate intervals are enumerated, dynamic programming is used to find the minimum-penalty block sequence. Let $D(B_{i,j}, k)$ denote the minimum accumulated penalty for a partial segmentation that ends with candidate block $B_{i,j}$ and contains k blocks. The recurrence is

$$\begin{aligned} D(B_{i,j}, k) = \lambda_{\text{blk}} p_{\text{blk}}(B_{i,j}) \\ + \min_{h<i} \left[D(B_{h,i}, k-1) \right. \\ \left. + \lambda_{\text{adj}} p_{\text{adj}}(B_{h,i}, B_{i,j}) \right]. \end{aligned} \tag{14}$$

Here, the minimization is over valid predecessor intervals $B_{h,i}$ that end immediately before $B_{i,j}$. The terminal state is selected by adding the page-level penalty:

$$\begin{aligned} (k^*, B^*_{\text{end}}) = \arg \min_{k, B_{i,n+1}} \Big[D(B_{i,n+1}, k) \\ + \lambda_{\text{page}} p_{\text{page}}(k) \Big]. \end{aligned} \tag{15}$$

The minimum-penalty recognition block sequence within the retained candidate space is recovered by backtracking from the selected terminal block B^*_{end}. We then apply a lightweight local merge step to suppress isolated tiny blocks. A tiny block is considered for merging with either adjacent block, and the merged interval is retained only when its geometry remains suitable for recognition according to fill-ratio, aspect-ratio, and strip-like constraints. The local decision is evaluated using the same block-level and adjacent-block costs as the sequence-segmentation objective. This post-processing step reduces unstable fragmentation while preserving the order and consecutive coverage of the text-line sequence.

1.8 Block crop refinement

After Recognition Block Construction, each selected block is converted into an OCR-friendly crop image. For a block B_t, we first compute its union bounding box $\text{BBox}(B_t)$ and expand it by a fixed margin p:

$$\Omega_t = \text{Expand}(\text{BBox}(B_t), p). \tag{16}$$

The expanded crop is clipped to the page boundary.

Because the expanded crop may include lines from neighboring blocks, non-target text regions are masked before recognition. We rasterize line boxes belonging to the current block into M_t^{own}, and line boxes from other blocks overlapping the crop into M_t^{nt}. The masked region is

$$M_t = M_t^{\text{nt}} \setminus M_t^{\text{own}}. \tag{17}$$

The masked pixels are filled with a local background color estimated from non-text pixels in the crop. If no reliable non-text pixels are available, the local crop appearance is used as the fallback background estimate.

Finally, the processed crop is placed at the center of a square canvas:

$$s_t = \alpha \cdot \max(w_t, h_t), \tag{18}$$

where w_t and h_t are the width and height of the processed crop, s_t is the side length of the square canvas, and α is a fixed padding scale. The remaining canvas area is filled with the same local background color. This padding strategy regularizes the input shape while preserving the original crop aspect ratio.