LineTR: Unified Text Line Segmentation for Challenging Palm Leaf Manuscripts

Vaibhav Agrawal, Niharika Vadlamudi, Muhammad Waseem, Amal Joseph, Sreenya Chitluri, and Ravi Kiran Sarvadevabhatla

Centre for Visual Information Technology
International Institute of Information Technology, Hyderabad 500032, India
ravi.kiran@iiit.ac.in

Abstract The dense and unstructured text in historical manuscripts presents significant challenges for precise line segmentation due to large diversity in sizes, scripts and appearances of the documents. Existing approaches tackle this complexity either by performing dataset-specific processing or training per-dataset models. This strategy hampers maintainability and scalability as newer manuscript collections get digitized and annotated. In this paper, we propose **LineTR**, a novel two-stage line segmentation approach which can process a diverse variety of challenging handwritten documents in a unified, dataset-agnostic manner. **LineTR**'s first stage processes context-adaptive image patches. It consists of a novel DETR-style network which generates parametric representations of text strike-through lines (scribbles) and a novel hybrid CNN-transformer network which generates a text energy map. A dataset-agnostic and robust post-processing procedure is applied on first-stage outputs to obtain document-level scribbles. In the second stage, these scribbles and the text energy map are used within a seam generation framework to obtain highly precise polygons enclosing the manuscript text lines. We also introduce three new diverse text line segmentation datasets comprising challenging Indic and South-East Asian manuscripts. Through experiments, ablations and evaluations, we show that **LineTR** generates significantly superior line segmentations - *all with a single model*. Our results also highlight the effectiveness of our unified model for good quality zero-shot inference on the newly introduced datasets. Project page: https://ihdia.iiit.ac.in/LineTR/.

Keywords: Text Line Segmentation · Historical Manuscripts · Deep Learning · Zero-Shot · Transformers

1 Introduction

Many approaches have been proposed in recent years to improve the quality of text line segmentation in challenging historical manuscripts [27,30,6,21,31]. Despite their successes, fundamental challenges remain. For instance, manuscript attributes such as size, aspect ratio, text line density, script, diacritics often change dramatically across datasets. For existing approaches, this diversity of

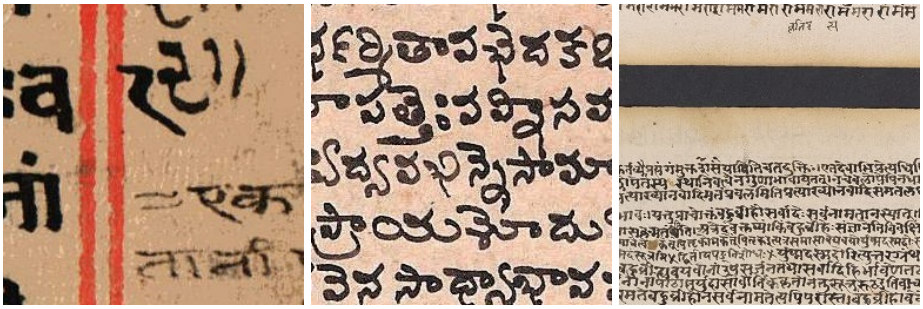
attributes is not easy to handle in a unified manner. As a compromise, dataset-specific hyperparameters are used in some cases. In other cases, dataset-specific models are trained. However, as new manuscript collections get digitized and annotated, the current strategies are not practical from maintenance and scalability point of view. Consequently, there is a practical need for an approach which can process a diverse variety of manuscripts in a unified, dataset-agnostic manner while delivering good line segmentation performance.

Motivated by this need, we propose **LineTR**, a unified, highly adaptive and precise text segmentation approach for challenging historical manuscripts. **LineTR** is designed as a two-stage pipeline [31]. To begin with, overlapping patches are sampled from input image in a context-adaptive manner. The first stage consists of [i] a novel DETR-style [8] deep network which generates parametric representations of text strike-through lines (scribbles) and [ii] a novel hybrid CNN-transformer network which generates a text energy map. A dataset-agnostic and robust post-processing procedure is applied on patch-level predictions from the first stage to obtain document-level scribbles. The scribbles and text energy map are used within an enhanced seam generation framework (the second stage) to obtain highly precise polygons enclosing the manuscript text lines.

Our first insight is that documents occur in various resolutions and aspect ratios, and resizing them to a fixed size causes problems. If the patch size is too large, then the number of learnable parameters as well as the memory and time complexity increase dramatically, and otherwise if the patch size is too small, then a lot of information in the image, such as the separation between the text lines is gone. This motivates a patch-based approach, similar to [31]. However, SeamFormer [31] samples patches of a fixed size (256×256), which brings in the issue of *bad context*. A patch of a fixed size can contain vastly varying amounts of information or *context*, depending on the resolution of the original document. (see Fig. 1), which is not desirable for a system which works generally across datasets. Therefore, an *adaptive* patching mechanism is required which can find the right patch size for a given document. We have shown in our experiments that this is a crucial design decision.

Our second insight is that a text line is a geometric structure (a curve), and this can serve as an excellent prior for the task of text-line detection [22,11,9,35,23,15]. Segmentation based approaches [31,25,5,27,7,19,17,20,16] do not utilize this prior, and instead predict a *blob* of pixels as the representation of a text-line, which leads to unwanted artifacts such as merging of adjacent blobs (see Fig 2). We approximate a text line within a patch using a straight line (see Fig. 3), and this approach has clear benefits over segmentation based approaches, as indicated by our results.

Finally, we also introduce three new diverse text line segmentation datasets consisting of challenging Indic and South-East Asian manuscripts. Through experiments, ablations and evaluations on new and existing palm leaf manuscript datasets, we show that **LineTR** generates significantly superior line segmentations compared to other competing approaches (Sec. 7) - all with a single unified



(a) too *little* context! (b) *good* context (c) too *much* context!

Figure 1: Fixed size patches (256×256, as used in [31]) sampled from different documents can result in *poor context*.

model. Additionally, we highlight the effectiveness of our unified model for good quality zero-shot inference on the newly introduced datasets (Sec. 7). Additional diagrams, zero-shot results and other details can be found on the project page.

2 Related Work

For an overview of methods ranging from classical image processing to deep learning techniques, refer to Vadlamudi et al. [31]. In some approaches, the document is fed to a neural network which directly predicts the text line polygonal instances or regions [25,5,12,24,27,7,19,17,20,16]. These one-shot approaches typically involve downsampling the image to a fixed input dimension. Therefore, they do not work well for high-aspect ratio palm leaf manuscripts containing closely spaced lines. Despite promising results on other historical manuscripts, these approaches predict imprecise text-line polygons and exclude vital textual components (e.g. diacritics).

A popular alternative is to predict 1-D geometric structures related to the text line. These structures are often represented as underlines (baselines) [22,11,9] or strike-throughs [35,23]. Instead of a pixel-based representation, Kiessling et al. [15] propose an approach which predicts the Bezier coefficients of the baseline.

While some approaches treat the gap between adjacent baselines as text lines [2], others employ heuristics to obtain text-line polygons [26]. Since these approaches also involve input downsampling, they inherit the shortcomings of one-shot approaches mentioned earlier. In contrast, our approach for **LineTR** does not involve manuscript image downsampling.

Another popular line of research employs seam generation techniques for delineating text-line polygons or combining seams to form text line polygons [1]. These methods employ the conventional seam carving algorithm [4] on curated energy maps to detect baselines or to generate separators for text-line regions. The novelty in these approaches often originates from the proposed energy map. This is exemplified by Signed Distance Transform (SDT) for Arabic

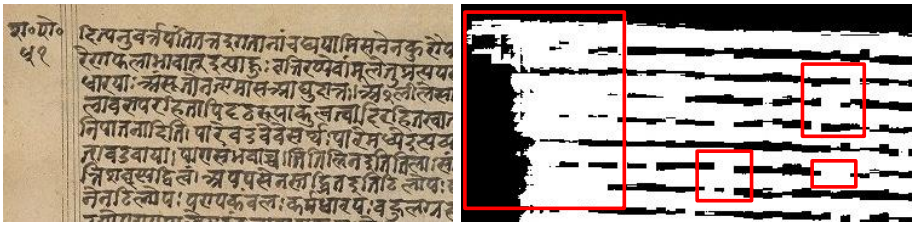


Figure 2: Issues with SeamFormer's [31] raw scribble map output on a dense Indic manuscript [27]. SeamFormer formulates scribble prediction as a per-pixel binary classification task, leading to extremely noisy predictions. Highlighted regions in the image cannot be post-processed to obtain distinct scribbles because of extreme merging.

manuscripts [26], geodesic distance transform energy map [3], a global energy map that incorporates diacritics [31]. These methodologies often rely on classical image processing techniques. Vadlamudi et al. [31] propose a hybrid two-stage approach in which strike-through regions predicted by a first stage network are used along with multiple energy maps to generate medial, upper and lower seams associated with the line polygon. While these approaches show promise, they necessitate intensive hyper-parameter tuning for the energy maps for every new dataset. While our approach uses seam generation, we introduce a hybrid CNN-transformer module which produces a single energy map and generalizes across diverse document styles and languages.

To avoid the drawbacks associated with image downsampling, some approaches partition the image into smaller patches and predict the text-line fragments [6]. The fragments are typically merged into document-level polygons using heuristic post-processing methods [31] or seam generation [6]. However, the post-processing is fragile and not viable when the lines have dense and uneven geometry as found in palm leaf manuscripts. Although SeamFormer [31] reduces the fragility by predicting text strike-through scribbles at patch level, the reliance on dataset-specific post-processing is not fully resolved due to noisy pixel-based representation of the scribbles. Often, the predictions merge into each other, as shown in Fig. 2. In contrast, we employ a parametric line representation for scribbles which enables dataset-agnostic post-processing. Unlike existing works which use fixed size patches, **LineTR** uses variable-sized patches which aids generalization.

As a rule, existing approaches train separate models and employ heuristics which are dataset-specific [6,31,35]. As emphasized earlier, ours is the first approach for text-line segmentation which works across multiple diverse datasets without requiring dataset-specific processing. As we have shown, this ability also enables **LineTR** to exhibit good zero-shot performance on unseen datasets.

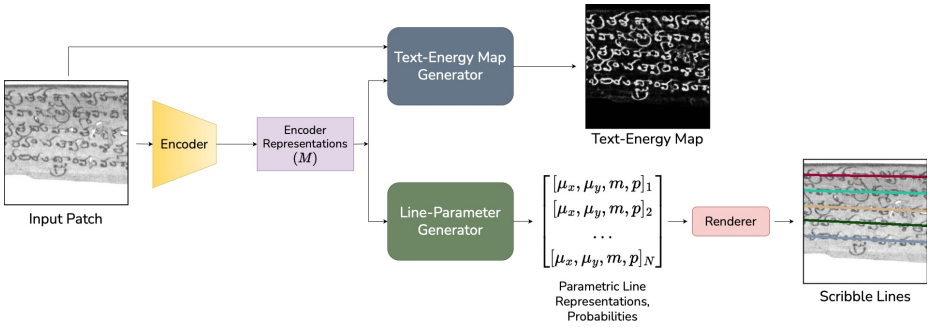


Figure 3: Stage-1 pipeline (Sec. 3.1). Scribbles lines are overlaid on the input patch for clarity.

3 The Proposed Approach: LineTR

We adopt a two stage approach to predict text line polygons. In *Stage-1* (see Fig. 3), the input image is first split into overlapping contextual patches of various sizes (Sec. 3.4.1). Each patch is processed by a deep network which predicts (a) parametric representations of text strike-through lines (scribbles) and (b) a continuous binary energy map (Sec. 3.1). The per-patch outputs are merged using an adaptive, data-agnostic post-processing module to obtain a global scribble map and a global binary energy map (Sec. 3.4.2). *Stage-2:* The global maps from *Stage-1* are processed using a seam generation algorithm to obtain tight-fitting polygons enclosing the text lines in the image (Sec. 3.5).

3.1 Stage-1

In this stage, the input patch is first processed by a shared encoder (see Fig. 3). The encoder's representations (M in Fig. 3,4) are fed to the Line-Parameter Generator which outputs parametric representations of scribble segments in the patch. The encoder representations are also fed to the Text-Energy Map Generator which outputs a continuous ($[0, 1]$) binary energy map. Next, we describe the individual components of *Stage-1* pipeline.

3.1.1 The Encoder This is comprised of a Vision Transformer (ViT) [10] which outputs feature map M – the encoder representation.

3.2 Line-Parameter Generator

3.2.1 Scribble Representation: We represent each scribble line using three parameters (Fig. 4)– μ_x, μ_y and m where (μ_x, μ_y) represents the mid-point of the scribble segment and m represents the slope (see Fig. 5). μ_x and μ_y are normalized wrt patch dimensions by dividing with image width and height respectively so that $\mu_x, \mu_y \in [0, 1]$.

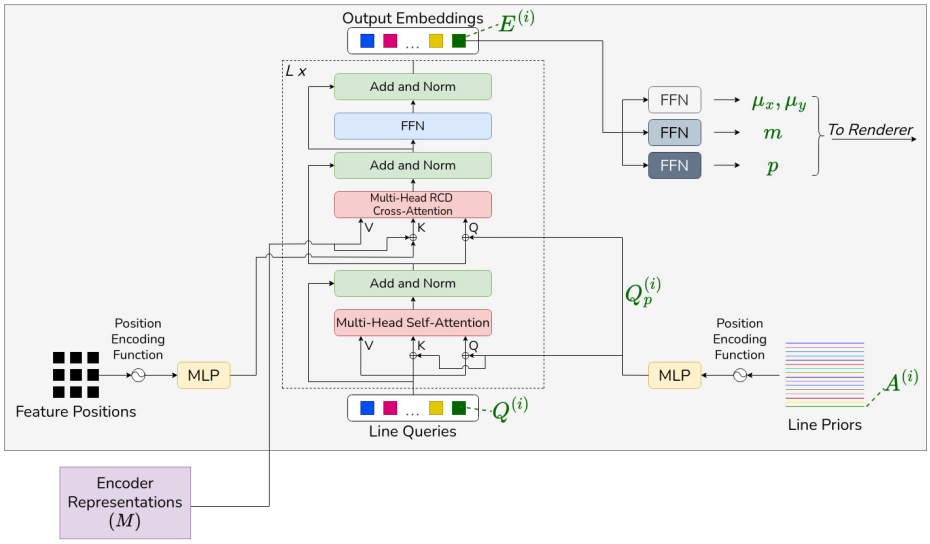


Figure 4: Line-Parameter Generator

3.2.2 Architectural Details We introduce a novel DETR-style [8] framework to predict parametric representations for each scribble line. A set of N learnable embeddings, which we call 'Line Queries' are fed to a transformer decoder (see Fig. 4). Within the decoder, these line queries are processed along with encoder representations M to obtain latent representations for the scribble lines ('Output Embeddings' in Fig. 4). These latent representations are transformed via lightweight feed-forward networks (FFN) to obtain scribble line parameters μ_x, μ_y, m and associated probabilities p (see top-right in Fig. 4). Next, we describe some key components of the transformer decoder.

Line Queries: We first define 'Line Priors'. These are N horizontal lines distributed uniformly throughout the image (see bottom right corner of Fig. 4). Formally, the i-th line prior $A^{(i)}$ is parameterized as $\mu_x = 0.5, \mu_y = i/N, m = 0$. We define positional query $Q_p^{(i)}$ as the positionally encoded and transformed version of $A^{(i)}$. Each line query $Q^{(i)}$ is first initialized to zero. We add the positional query $Q_p^{(i)}$ at the inputs of the attention layers (see Fig. 4).

Self Attention: There are two types of attention in the transformer decoder – self attention and cross attention [33]. The self attention is applied within the *Line Queries* Q (defined previously). Note that the positional queries are added to the line queries before computing the attention scores.

Cross Attention: The outputs of *Self Attention* are fed to the cross attention module. Due to the large size of the encoder feature representation M, the number of attention weights is quite large, which leads to slow convergence in DETR-style frameworks [8,34,37]. To counter this, we adopt a decoupled

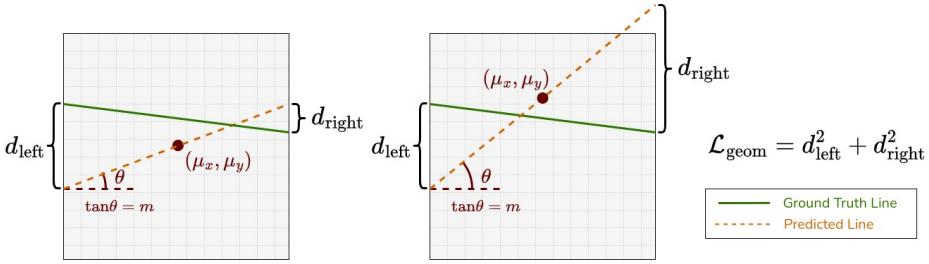


Figure 5: Our proposed loss function for penalizing the misalignment between two lines. If the scribble line touches the top or the bottom patch boundary, then we use interpolation to calculate the loss as shown in the figure on the right.

row-column-based attention proposed by Wang et al. [34] which leads to faster convergence.

Final Predictions: The line queries undergo successive layers of self-attention and cross-attention with the encoder representations. Ultimately, each query $Q^{(i)}$ is transformed into an output embedding $E^{(i)}$ - see Fig. 4. Each $E^{(i)}$ either represents a scribble or ϕ (the empty class). We obtain the scribble-line parameters μ_x, μ_y, m along with line probability score p for each embedding using feed-forward networks (see Fig. 4).

3.2.3 Optimization: Having obtained the predicted parameters and line probability scores for each query, we match each ground truth line to a query such that the assignment is one-to-one and optimal using the Hungarian Algorithm.

Let $\{l_i\}_{i=1}^{N_0}$ be the set of ground truth lines sorted by their μ_y values. We define the median vertical gap δ between the sorted lines as $\delta = \texttt{median}\{\mu_{y_2} - \mu_{y_1}, \mu_{y_3} - \mu_{y_2}, ..., \mu_{y_{N_0}} - \mu_{y_{N_0-1}}\}$. Let $\hat{l}(\hat{\mu_x}, \hat{\mu_y}, \hat{m})$ be a predicted line with associated probability p and $l(\mu_x, \mu_y, m)$ be a ground truth line. We propose a geometry-based loss function for penalizing the misalignment between the predicted line $\hat{l}$ and the ground truth line l. Let d_{left} and d_{right} be the vertical distances between the lines at the left and the right patch boundary respectively (see Fig. 5). We define the geometric loss $\mathcal{L}_{\text{geom}}$ as: $\mathcal{L}_{\text{geom}}(\hat{l}, l) = d^2_{\text{left}} + d^2_{\text{right}}$.

Finally, we define the matching cost function $\mathcal{C}_{\text{match}}$ as follows:

$$\mathcal{C}_{\text{match}}(\hat{l}, l) = \lambda_{\text{geom}} \cdot \frac{\mathcal{L}_{\text{geom}}(\hat{l}, l)}{\delta} - \lambda_p \cdot p$$

where $\lambda_{\text{geom}}, \lambda_p \in \mathbb{R}$ are hyperparameters. After the matching, some queries would be assigned to a line. The rest of the queries would be assigned ϕ (the empty class). Let $g : \mathbb{Z}^+ \to \{\mathbb{Z}^+, \phi\}$ be the obtained matching, such that $g(i) = j$ if the i^{th} prediction is matched to the j^{th} ground truth line, and $g(i) = \phi$ if it is

assigned ϕ. We define the optimization loss $\mathcal{L}_{\text{opt}}$ as follows:

$$\mathcal{L}_{\text{opt}} = \mathcal{L}_1 + \mathcal{L}_2$$
$$\mathcal{L}_1 = \sum_{i=1;g(i)=\phi}^{N} \left[\lambda_1 \mathcal{L}_{\text{focal}}(p_i, -) \right]$$
$$\mathcal{L}_2 = \sum_{i=1;g(i)\neq\phi}^{N} \left[\lambda_2 \mathcal{L}_{\text{focal}}(p_i, +) + \lambda_{\text{geom}} \frac{\mathcal{L}_{\text{geom}}(\hat{l}, l)}{\delta^2} \right]$$

where $\lambda_{\text{geom}}, \lambda_1, \lambda_2 \in \mathbb{R}$ are hyperparameters, and $\mathcal{L}_{\text{focal}}$ stands for the focal loss [18]:

$$\mathcal{L}_{\text{focal}}(p_i, +) = -(1 - p_i)^{\gamma} \log(p_i)$$
$$\mathcal{L}_{\text{focal}}(p_i, -) = -(p_i)^{\gamma} \log(1 - p_i)$$

where γ is a hyperparameter. This completes the description of 'Line-Parameter Generator' module in *Stage-1*. Next, we describe the 'Text-Energy Map Generator' (see Fig. 3).

3.3 Text-Energy Map Generator

The text-energy map generator is used to generate a continuous binary ($[0, 1]$) map to be used as an input to the seam generation algorithm in *Stage-2*. It uses a hybrid CNN-transformer architecture (see Fig. 6). The input patch is fed to a CNN encoder which outputs a feature map P. This feature map and representation M from the backbone encoder are fed to a transformer decoder. Within the decoder, self attention is applied to M and the result is processed along with P via a standard cross attention mechanism [33]. The resulting output is decoded to the target binary map via a CNN decoder containing skip connections with intermediate feature maps of the CNN encoder.

For optimization, we use the focal loss [18]:

$$\mathcal{L} = -\alpha \cdot y \cdot (1-\hat{y})^{\gamma} \log \hat{y} - (1 - \alpha) \cdot (1-y) \cdot \hat{y}^{\gamma} \log(1-\hat{y})$$

where $\hat{y}$ represents the sigmoid activation layer's output (shaded blue in Fig. 6), $y \in \{0, 1\}$ represents the ground truth, and α, γ are hyperparameters.

3.4 Stage-1 Inference and Post-Processing

In this section, we describe the mechanism by which global document-level strike-through scribbles and Text-Energy map are obtained during test time (inference). The exact algorithms and visual explanations of the mechanisms can be found on the project page.

3.4.1 Context-Adaptive Patching: Documents occur in various sizes. To ensure that the patches contain enough contextual information for effective scribble line prediction, we introduce an adaptive patching mechanism. Given the input image, we sample patches of various sizes, and perform inference to obtain

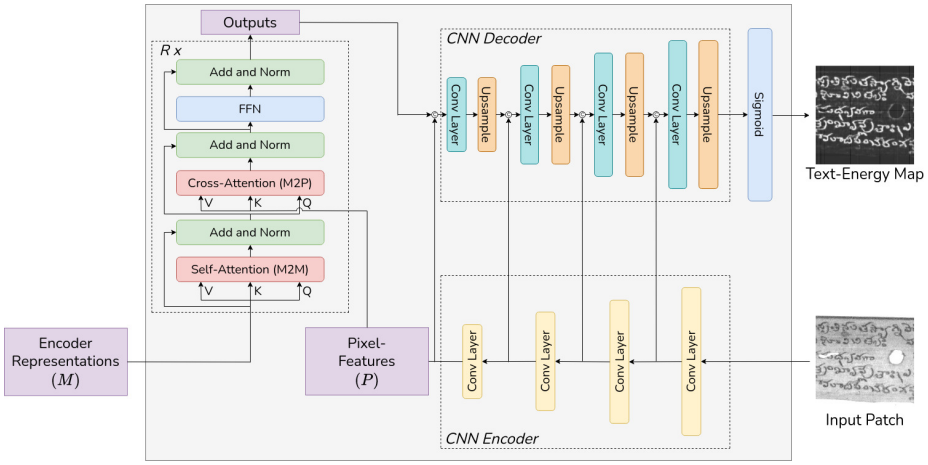


Figure 6: Text-Energy Map Generator

initial scribble line outputs. These are used to estimate the average spacing between text lines. A patch size t is calculated, such that the patches capture *suitable* context in the document. We then sample patches of size t and perform inference to obtain scribble-line predictions. These predictions tend to be more accurate because the patches are context-adapted. See Fig. 7 for a visual description. Refer to the project page for more details.

3.4.2 Combining patch-level outputs: After obtaining context adapted patches (Sec. 3.4.1) and patch-level scribble-line predictions (Sec. 3.2), we construct the global scribble map $\mathcal{S}$ using an iterative *Projection-Merging Algorithm*. Roughly, the algorithm involves rendering each patch's line predictions on the original document, which are then clustered based on distance between them. Each cluster so formed represents a text line in the document. The exact algorithm along with a visual explanation can be found on the project page.

To obtain the global Text-Energy Outputs, we sample the original image into non-overlapping patches and pass them through the Text-Energy Map generator (Sec. 3.3) to obtain patch-level outputs. These are then combined to form the global Text-Energy map $\mathcal{B}$. See the project page for more details.

3.5 Stage-2

The outputs of *Stage-1* are a list of global scribbles $\mathcal{S}$ and a *continuous* Text-Energy binary map $\mathcal{B}$ of the complete input image. These are processed by the seam generation pipeline from SeamFormer [31] to obtain tight fitting polygons enclosing the text lines.

We introduce two crucial modifications to the default approach in SeamFormer [31]. Instead of thresholding the binary map, we use output from *Stage-1* as it is, i.e. $\mathcal{B}$ contains floating point values in the range $[0, 1]$. This avoids loss

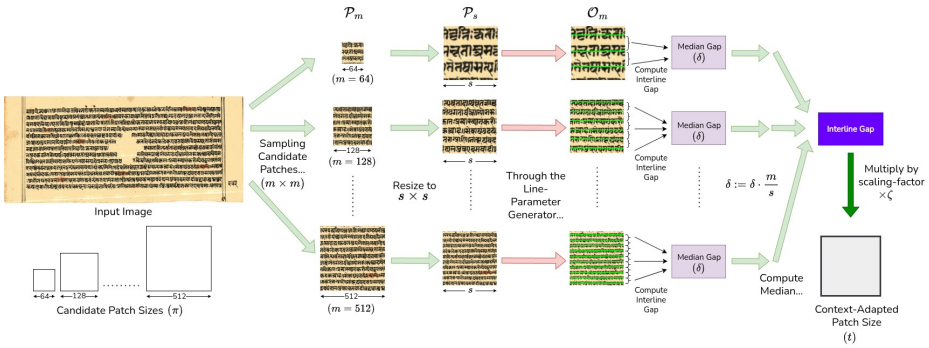


Figure 7: First, *candidate* patches of various sizes $m = \{64, 128, \ldots 512\}$ are sampled from the document. Line-parameter predictions are made for each patch by resizing them to the input shape expected by the model ($s = 256$) and then passing them through the Line-Parameter Generator (Sec. 3.2). An interline-gap value δ for each patch is calculated using the line predictions for the patch. These patch-level interline gap predictions are then scaled to the size of the original document ($\delta := \delta \cdot \frac{m}{s}$) and averaged to obtain an estimate of the average interline gap value for the whole document. This averaged value is multipled by a scaling factor ζ, which roughly represents the expected number of text lines seen in a patch to obtain the final context-adapted patch size (t). Patches of size t are then sampled from the original document, and are fed to the Line-Parameter Generator to get the final predictions.

of crucial text presence information caused by thresholding. The second modification is to discard other energy maps used in SeamFormer [31] (smoothing map, diacritic map and sobel map). This eliminates the need for determining energy map weight coefficients. As shown via experiments (Table 3), the quality of our Text-Energy map makes other maps redundant in practice.

4 Implementation Details

Architectural Details: The reference input shape for Stage-1 model is $H_0 \times W_0 \times 3$, where $H_0 = 256$, $W_0 = 256$. The ViT backbone (see Fig. 3) uses a patch size of 16×16, and the output feature map M is of dimension $H \times W \times C$, where $H = 16$, $W = 16$ and $C = 256$, where C is the model's embedding dimension. The encoder consists of 8 encoder layers. Learnable position embeddings are used. The Line-Parameter Generator (Fig. 4) uses $N = 200$ line queries. The transformer decoder consists of $L = 8$ decoder layers. The probability threshold p_0 for line-prediction is set to 0.9. The CNN encoder for Text-Energy Map Generator (Fig. 6) has 4 convolutional layers, each of which reduces the spatial dimensions of the input image by half. The output of this encoder (P) has the same shape as the output of the backbone (M). The transformer decoder consists

Table 1: Dataset Statistics (Sec. 5). Languages : `hin` - Hindi, `tel` - Telugu, `tam` - Telugu, `sun` - Sundanese, `ban` - Balinese, `khm` - Khmer, `mal` - Malayalam, `jav` - Javanese. Newly introduced datasets are shaded pink.

	I2	SD	BL	KH	KG	WM	UB	SM
Name	Indiscapes2	Sundanese	Balinese	Khmer	KgathaM	Wikimedia	Upamiti Bori	SVMP
Train Images	907	31	47	50	313	0	0	0
Test Images	229	30	49	200	79	60	30	30
Avg Lines	11	4	4	5	9	6	5	11
Min Size	442x207	2530x333	2505x637	2244x322	2636x410	2042x1440	2592x1728	4567x1331
Max Size	9184x1064	3159x352	5759x561	8224x696	3404x501	2324x1814	2592x1728	10277x3281
Aspect Ratio	3	9	10	11	7	3.5	1.5	2
Language	`hin,tel,tam`	`sun`	`ban`	`khm`	`mal`	`ban,jav`	`hin`	`tel`
Source	[27]	[28]	[13]	[32]	[31]	Wikimedia	Private	Private

of $R = 3$ decoder layers. The CNN decoder consists of 4 layers. Each layer consists of a convolutional layer followed by upsampling. In the loss function, we use $\alpha = 0.25$.

Training Details: We perform *Stage-1* training in two phases. In the first phase, we train only the ViT backbone and the Line-Parameter Generator. In the second phase, we freeze both of them, and train only the Text-Energy Map Generator. In the first phase, we use a learning rate of 5×10^{-5}, and train for 60 epochs. We then reduce the learning rate to 10^{-5} and train for another 20 epochs. In the second phase, we train only the energy map generator with a learning rate of 10^{-4} for 50 epochs. We use the AdamW optimizer, with the coefficients set to PyTorch's defaults. Our implementation is based on the distributed PyTorch Lightning framework. We trained our model on 4 NVIDIA RTX 2080Ti GPUs, with 24 images on each GPU. The first phase of training took around 32 hours, and the second phase took around 7 hours.

5 Datasets

To evaluate the proposed model and baselines, we use palm leaf manuscript datasets introduced in earlier works [13,28,31,32] - see Table 1 (shaded blue). In addition, we annotate three new challenging manuscript datasets (WM,UB,SM) for zero-shot evaluation (shaded pink). The diversity in terms of scripts, image aspect ratios, image dimension ranges, number of lines seen in Table 1 underscores the challenge involved in palm-leaf manuscript text line segmentation. The new datasets were annotated using the HInDoLA document image annotation tool [29]. Instead of annotating polygons from scratch, a semi-automatic approach was implemented and integrated into the tool. The annotators added strike-through scribbles for text lines. These scribbles and a binarized version of input image were processed by seam generation module from SeamFormer [31] to obtain text line polygon predictions. The predicted polygon boundaries were adjusted to accommodate missing diacritics and ensure correct enclosure of text lines. Empirically, this semi-automatic approach provided a 75% reduction in annotation time compared to the purely manual variant.

6 Experiments

We compare **LineTR** against various state-of-the-art approaches developed for handwritten historical manuscripts. For fair and consistent comparison, we train all the models (ours, existing approaches) on a single large-scale dataset obtained by combining the training sets of existing palm leaf manuscript datasets - Indiscapes2 [27] [I2], KGatham [31] [KG], and Challenge B dataset of ICFHR 2018 Competition On Document Image Analysis Tasks for Southeast Asian Palm Leaf Manuscripts [14] containing manuscripts from Balinese [13] [BL], Khmer [32] [KH], and Sundanese [28] [SD] languages. We report the performance metrics on the respective test sets of these datasets. For existing approaches, we follow the training instructions mentioned in their corresponding papers.

Trivedi et al. [30] demonstrate that Average Hausdorff Distance (AvgHD) is a better measure of prediction performance for line polygon boundaries. Therefore, we report AvgHD in addition to the standard Intersection over Union (IoU) metric. To assess the zero-shot generalizability performance, we report these metrics on three newly introduced datasets unseen during training - WM, UB, and SM - see Table 1 for an overview of the datasets.

7 Results

As Table 2 shows, **LineTR** clearly outperforms other models by a significant margin across all datasets. The consistently poor scores among other baselines is an outcome of loose fit predicted text regions, often missing crucial textual elements such as diacritics. Also, baseline methods [20,27,7] typically approach text line segmentation as a per-pixel classification task, resizing images with large aspect ratios to a fixed lower size. This resizing tends to merge adjacent predicted text, particularly in dense text documents. In other baselines [31,1,22], the suboptimal results are due to excessive dataset-specific decisions. **LineTR**'s numbers on unseen datasets are on par with ones encountered during training. This shows its zero-shot generalization ability.

7.1 Ablations

For ablation experiments, we report metrics on the combined test sets of the seen benchmark datasets - see Table 3. The results suggest that a fine balance is required so that number of *Line Queries* (N) in 'Line-Parameter Generator' (Sec. 3.2) is neither too many nor too few. Compared to using an arbitrarily threshold binary map similar to SeamFormer [31]'s approach, our unthresholded text energy map (Sec. 3.3) is a noticeably better choice. This is due to the loss of text-information caused due to a fixed threshold value. To demonstrate the importance of context-adaptive patching (Sec. 3.4.1) for generating training data, we considered patches of fixed-size, similar to SeamFormer [31]. However, we found that the combined dataset training is extremely unstable. In fact, the optimization did not even converge. Finally, we replace $\mathcal{L}_{\text{geom}}$ (Sec. 3.2.3) by the EA-loss [36]. This setting also caused the network to not converge.

Table 2: Comparative evaluation of **LineTR** using benchmark datasets. All the baseline models are trained on **the pooled dataset** (Sec. 6) using default settings mentioned in respective works to assess their adaptability. The results are reported on test set of each dataset and three additional unseen datasets (zero-shot).

	I2[27]	SD[28]	BL[13]	KH[32]	KG[31]	WM*	UB*	SM*
				AvgHD ↓				
Doc-UFCN [7]	68.60	71.17	74.52	95.90	38.79	41.39	71.10	174.14
SeamFormer [31]	11.82	8.92	104.75	49.03	7.83	11.54	9.46	130.98
Palmira [27]	15.81	6.50	301.21	203.59	7.50	24.11	18.88	15.78
LCG [1]	16.82	39.65	95.18	44.50	29.72	317.98	481.14	1034.88
dhSegment [22]	60.33	66.77	415.24	43.60	319.57	150.43	213.32	414.63
docExtractor [20]	77.25	33.86	43.16	48.36	40.24	87.75	95.51	260.70
LineTR	**1.86**	**1.30**	**22.62**	**14.97**	**2.09**	**0.94**	**1.01**	**3.04**
				IoU ↑				
Doc-UFCN [7]	0.23	0.10	0.08	0.11	0.12	0.15	0.10	0.16
SeamFormer [31]	0.51	0.53	0.32	0.37	0.49	0.49	0.76	0.43
Palmira [27]	0.72	0.66	0.39	0.41	0.62	0.53	0.54	0.69
LCG [1]	0.37	0.12	0.12	0.18	0.20	0.02	0.01	0.07
dhSegment [22]	0.34	0.12	0.03	0.08	0.12	0.10	0.13	0.09
docExtractor [20]	0.03	0.01	0.00	0.02	0.12	0.03	0.03	0.02
LineTR	**0.80**	**0.73**	**0.62**	**0.69**	**0.81**	**0.66**	**0.82**	**0.82**

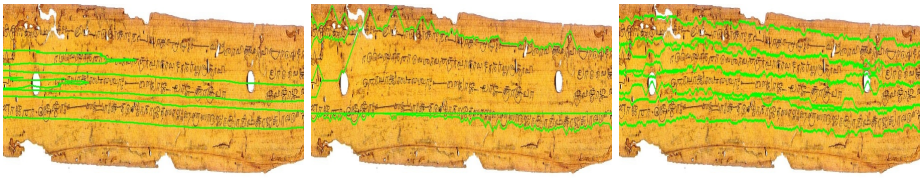
* Newly introduced datasets with zero-shot baseline testing.

Table 3: Performance scores for **LineTR** ablative variants. (Sec. 7.1). DNC=Did not converge

Ablation Type	Pipeline Component	Ablation Details	IoU ↑	Avg HD ↓
Architectural	Line-Parameter Generator	$N = 100$	0.45	145.39
		$N = 300$	0.61	85.33
		$N = 400$	0.62	28.72
	Text-Energy Map Generator	Threshold the energy map[31]	0.53	8.14
Optimization	Line-Parameter Generator	Replace $\mathcal{L}_{\text{geom}}$ by EA-loss[36]	*DNC*	*DNC*
Dataset	Data-Preparation	Sample patches of fixed size	*DNC*	*DNC*
LineTR			**0.74**	**7.13**

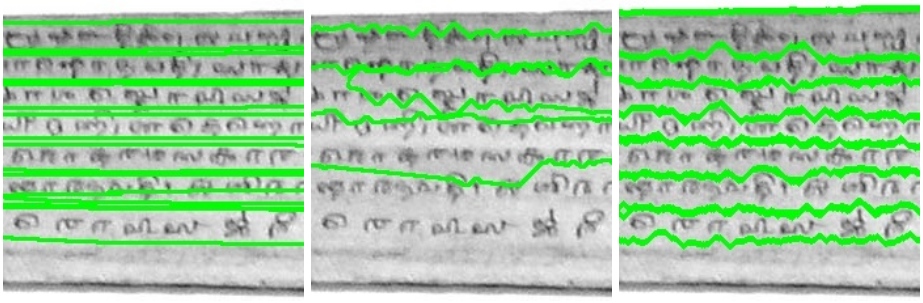
7.2 Qualitative Results

As Fig. 8 shows, both of **LineTR**'s closest competitors – Palmira [27] and SeamFormer [31] – fail when the text-lines have a curvature spread across the document width. But **LineTR** is able to detect all the text-lines accurately. Similarly, **LineTR** outperforms Palmira and SeamFormer on images where the density of text is very high (see Fig. 9). Figure 10 shows the zero-shot outputs of **LineTR** on the newly introduced datasets.



(a) Palmira[27] (b) SeamFormer[31] (c) **LineTR**

Figure 8: Performance comparison on a challenging image with curved text-lines



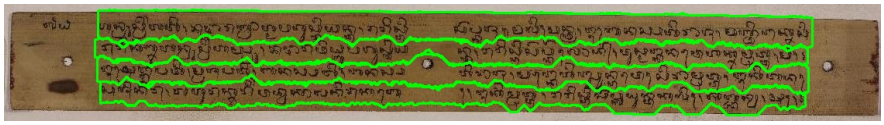
(a) Palmira[27] (b) SeamFormer[31] (c) **LineTR**

Figure 9: Performance comparison on a challenging image with very dense text

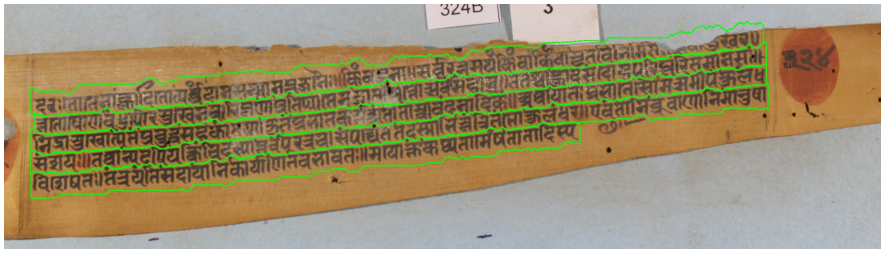
8 Conclusion

LineTR is a novel dataset-agnostic approach for robust text line segmentation in diverse and challenging historical manuscripts. Similar to recent successful approaches, we use a two stage approach - scribble generation and scribble-conditioned polygon generation. However, our unique and novel design choices make a significant difference. The choice of predicting per-patch scribble line parameters in the first stage helps avoid the difficulties of pixel-based scribble representation. Our adaptive patch extraction ensures sufficient context capture for predicting scribble line parameters. A sensible design for Text-Energy Map Generator not only simplifies second stage processing, it also improves overall results.

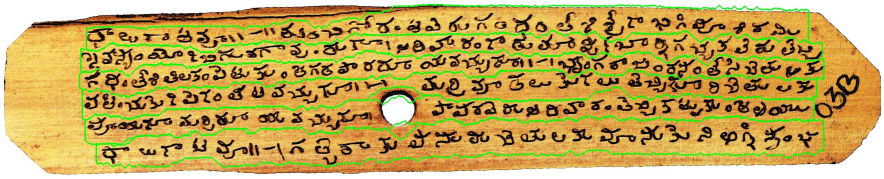
Unlike existing approaches, **LineTR**'s methodology does not require dataset-specific fine-tuning. Another distinction is that the training process results in a single model and does not require dataset-specific models. These features make **LineTR** advantageous from a maintainability and scalability point of view. **LineTR** not only outperforms strong baselines but also exhibits good zero-shot performance on unseen datasets. This showcases its generalizability and utility for the community.



(a) WM



(b) UB



(c) SM

Figure 10: Zero-shot outputs of **LineTR** on the newly introduced datasets

Acknowledgements

This work is supported by Ministry of Electronics and Information Technology (MeiTY), Government of India.

References

1. Alberti, M., Vögtlin, L., Pondenkandath, V., Seuret, M., Ingold, R., Liwicki, M.: Labeling, cutting, grouping: an efficient text line segmentation method for medieval manuscripts. In: 2019 IPAR (ICDAR). pp. 1200–1206. IEEE (2019) 3, 12, 13
2. Arvanitopoulos, N., Süsstrunk, S.: Seam carving for text line extraction on color and grayscale historical manuscripts. In: 2014 14th. pp. 726–731. IEEE (2014) 3
3. Asi, A., Saabni, R., El-Sana, J.: Text line segmentation for gray scale historical document images. In: Proceedings of the 2011 workshop on historical document imaging and processing. pp. 120–126 (2011) 4
4. Avidan, S., Shamir, A.: Seam carving for content-aware image resizing. In: ACM SIGGRAPH 2007 papers, pp. 10–es (2007) 3
5. Barakat, B., Droby, A., Kassis, M., El-Sana, J.: Text line segmentation for challenging handwritten document images using fully convolutional network. In: 2018 16th (ICFHR). pp. 374–379. IEEE (2018) 2, 3

6. Barakat, B.K., Droby, A., Alaasam, R., Madi, B., Rabaev, I., Shammes, R., El-Sana, J.: Unsupervised deep learning for text line segmentation. In: 2020 25th (ICPR). pp. 2304–2311. IEEE (2021) 1, 4
7. Boillet, M., Kermorvant, C., Paquet, T.: Multiple document datasets pre-training improves text line detection with deep neural networks. In: 2020 25th (ICPR). pp. 2134–2141. IEEE (2021) 2, 3, 12, 13
8. Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S.: End-to-end object detection with transformers. In: ECCV (2020) 2, 6
9. Chamchong, R., Fung, C.C.: Text line extraction using adaptive partial projection for palm leaf manuscripts from thailand. In: ICFHR (2012) 2, 3
10. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N.: Image is worth 16x16 words: Transformers for image recognition. ICLR (2021) 5
11. Grüning, T., Leifert, G., Strauß, T., Michael, J., Labahn, R.: A two-stage method for text line detection in historical documents. (IJDAR) **22**(3), 285–302 (2019) 2, 3
12. Jindal, A., Ghosh, R.: Text line segmentation in indian ancient handwritten documents using faster r-cnn. MTA pp. 1–20 (2022) 3
13. Kesiman, M.W.A., Burie, J.C., Wibawantara, G.N.M.A., Sunarya, I.M.G., Ogier, J.M.: Amadi_lontarset: the first handwritten balinese palm leaf manuscripts dataset. In: 2016 15th (ICFHR). pp. 168–173. IEEE (2016) 11, 12, 13
14. Kesiman, M.W.A., Valy, D., Burie, J.C., Paulus, E., Suryani, M., Hadi, S., Verleysen, M., Chhun, S., Ogier, J.M.: Icfhr 2018 competition on document image analysis tasks for southeast asian palm leaf manuscripts (2018) 12
15. Kiessling, B.: Curt: End-to-end text line detection in historical documents with transformers. pp. 34–48. Springer (2022) 2, 3
16. Kurar Barakat, B., Cohen, R., Droby, A., Rabaev, I., El-Sana, J.: Learning-free text line segmentation for historical handwritten documents. Applied Sciences (2020) 2, 3
17. Li, D., Wu, Y., Zhou, Y.: Linecounter: Learning handwritten text line segmentation by counting. In: ICIP (2021) 2, 3
18. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection. In: ICCV. pp. 2980–2988 (2017) 8
19. Mechi, O., Mehri, M., Ingold, R., Amara, N.E.B.: Text line segmentation in historical document images using an adaptive u-net architecture. In: 2019 IPAR (ICDAR). pp. 369–374. IEEE (2019) 2, 3
20. Monnier, T., Aubry, M.: docExtractor: An off-the-shelf historical document element extraction. In: ICFHR (2020) 2, 3, 12, 13
21. Nguyen, T.N., Burie, J.C., Le, T.L., Schweyer, A.V.: An effective method for text line segmentation in historical document images. In: ICPR. IEEE (2022) 1
22. Oliveira, S.A., Seguin, B., Kaplan, F.: dhsegment: A generic deep-learning approach for document segmentation. In: 2018 16th (ICFHR) (2018) 2, 3, 12, 13
23. Paulus, E., Burie, J.C., Verbeek, F.J.: Text line extraction strategy for palm leaf manuscripts. Pattern Recognition Letters **174**, 10–16 (2023) 2, 3
24. Prusty, A., Aitha, S., Trivedi, A., Sarvadevabhatla, R.K.: Indiscapes: Instance segmentation networks for layout parsing of historical indic manuscripts. In: ICDAR. pp. 999–1006 (2019) 3
25. Renton, G., Soullard, Y., Chatelain, C., Adam, S., Kermorvant, C., Paquet, T.: Fully convolutional network with dilated convolutions for handwritten text line segmentation. (IJDAR) **21**, 177–186 (2018) 2, 3

26. Saabni, R., El-Sana, J.: Language-independent text lines extraction using seam carving. In: 2011 IPAR. pp. 563–568. IEEE (2011) 3, 4
27. Sharan, S., Aitha, S., Kumar, A., Trivedi, A., Augustine, A., Sarvadevabhatla, R.K.: Palmira: a deep deformable network for instance segmentation of dense and uneven layouts in handwritten manuscripts. In: ICDAR (2021) 1, 2, 3, 4, 11, 12, 13, 14
28. Suryani, M., Paulus, E., Hadi, S., Darsa, U.A., Burie, J.C.: The handwritten sundanese palm leaf manuscript dataset from 15th century. In: 2017 14th IAPR IPAR (ICDAR). vol. 1, pp. 796–800. IEEE (2017) 11, 12, 13
29. Trivedi, A., Sarvadevabhatla, R.K.: Hindola: A unified cloud-based platform for annotation, visualization and machine learning-based layout analysis of historical manuscripts. In: ICDARW. vol. 2, pp. 31–35. IEEE (2019) 11
30. Trivedi, A., Sarvadevabhatla, R.K.: Boundarynet: An attentive deep network for semi-automatic layout annotation. In: ICDAR (2021) 1, 12
31. Vadlamudi, N., Krishna, R., Sarvadevabhatla, R.K.: Seamformer: High precision text line segmentation for handwritten documents. In: IPAR. pp. 313–331. Springer (2023) 1, 2, 3, 4, 9, 10, 11, 12, 13, 14
32. Valy, D., Verleysen, M., Chhun, S., Burie, J.C.: A new khmer palm leaf manuscript dataset for document analysis and recognition: Sleukrith set. In: International Workshop on Historical Document Imaging and Processing (2017) 11, 12, 13
33. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30** (2017) 6, 8
34. Wang, Y., Zhang, X., Yang, T., Sun, J.: Anchor detr: Query design for transformer-based detector. In: AAAI. vol. 36, pp. 2567–2575 (2022) 6, 7
35. Wigington, C., Tensmeyer, C., Davis, B., Barrett, W., Price, B., Cohen, S.: Start, follow, read: End-to-end full-page handwriting recognition. In: ECCV (2018) 2, 3, 4
36. Zhao, K., Han, Q., Zhang, C.B., Xu, J., Cheng, M.M.: Deep hough transform for semantic line detection. IEEE TPAMI **44**(9), 4793–4806 (2021) 12, 13
37. Zhu, X., Su, W., Lu, L., Li, B., Wang, X., Dai, J.: Deformable transformers for end-to-end object detection. arXiv preprint arXiv:2010.04159 (2020) 6