CURIO: Curvature-Aligned and Efficient OCR for Low-Resource Historical Manuscripts (Supplementary)

Sai Madhusudan Gunda, Tathagata Ghosh, Simran Singh Sandral, Ravi Kiran Sarvadevabhatla
International Institute of Information Technology Hyderabad
sai.gunda@research.iiit.ac.in, tathagata.ghosh@research.iiit.ac.in,
ssandral@gitam.in, ravi.kiran@iiit.ac.in

Contents

1. Sharada Manuscripts

Why Sharada is challenging. The *Śāradā* (Sharada) script—historically used to write Sanskrit and, later, Kashmiri—has very limited living usage today. Surviving materials are sparse, heterogeneous, and often degraded; expert readers are few, and public transcriptions are rare. Baselines are frequently *strongly curved* due to page curl, copy wear, and calligraphic ductus, with tight diacritics and ligature overhangs. This makes Sharada a *low-resource* setting in both data and expertise: few digitized corpora, minimal ground truth, and large variability in handwriting. Because training data are scarce and annotations are expensive, progress hinges on methods that generalize under data poverty: robust line extraction, curvature-aware rectification, and synthetic augmentation that respects Sharada's visual patterns. Fig. 1 The two collections used in our study—Ubay Utpal Vivritti [2] (UUV) and the Sharada Lipi Archive [3] (SLA)—exhibit a particularly broad spectrum of curvature profiles and line lengths, making them representative and challenging sets for OCR development (for stats see Tab. 1 , Fig. 2).

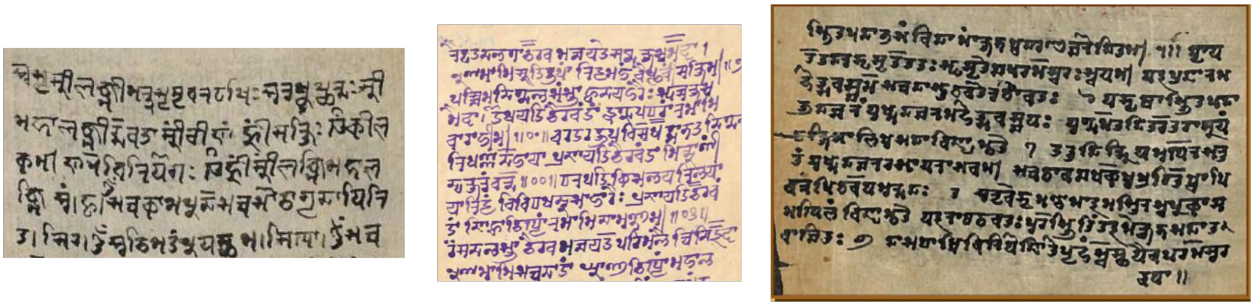


Figure 1. Example Sharada Leaves

2. Inpaint Line Segments

Extracted line segments using LineTR contain black regions which isolate them to remove it we inpainting on these regions. The inpainting process begins by first detecting where the paper ends and the unwanted regions (black padding and text) lie. Using the paper mask and the text mask, we define a combined inpainting mask that marks both the black borders and the ink strokes as areas to be filled. We then apply OpenCV's [7] Navier-Stokes-based inpainting algorithm [6], which propagates nearby paper texture into these masked zones, producing a continuous, text-free background that looks like natural manuscript paper. Since this also erases the original text, we restore it in a final compositing step: the text pixels from the original image are overlaid exactly onto the inpainted background. This ensures that the padding is replaced with plausible paper texture while the text itself is preserved at pixel-level fidelity (Fig. 3).

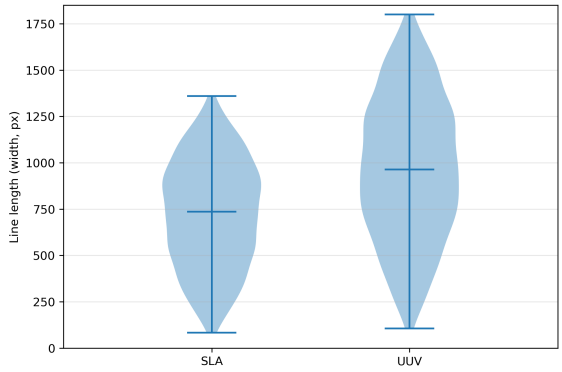
3. More on Line Rectification

Historical manuscripts often exhibit curved or warped text lines due to page curl, folds, and calligraphic ductus. To normalize these effects prior to recognition, we apply a dedicated line-level rectification stage.

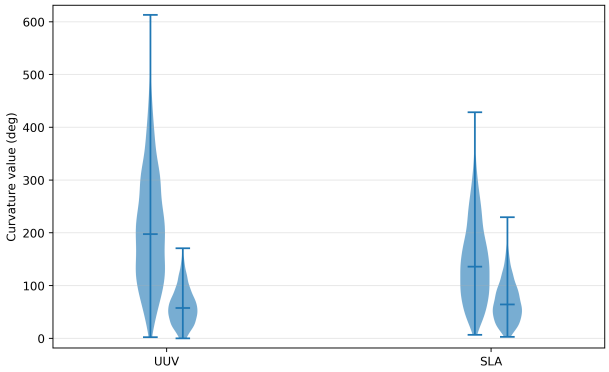
LineTR [5] is used to detect line instances and also tight polygonal mask , a baseline scribble. The raw polygons can clip ascenders/descenders and diacritics, we expand each mask by a small uniform dilation to create a safety margin. This preserves fine marks that are semantically important in low-resource scripts and stabilizes the subsequent warp by avoiding extreme boundary constraints.

Source	# Pages	# Lines	# Chars/Line	Width (px)		Curvature (°)		Train/Val/Test
				Min	Max	Pre-R	Post-R	60/20/20 %
UUV	89	1340	12–200	106	1800	2.3–612.8	0.0–170.6	804 / 268 / 268
SLA	123	1364	5–86	83	1350	6.8–428.4	1.1–229.7	818 / 273 / 273

Table 1. **Real data statistics** for Sharada scripts. Curvature is reported pre- (*Before*) and post- (*After*) rectification.



(a) Distribution of line lengths (in pixels) across SLA and UUV collections.



(b) Curvature statistics before and after rectification, showing reduction of extreme curvature values.

Figure 2. Quantitative statistics of Sharada manuscripts. (a) highlights the wide range of line lengths, while (b) demonstrates how rectification normalizes curvature distributions.

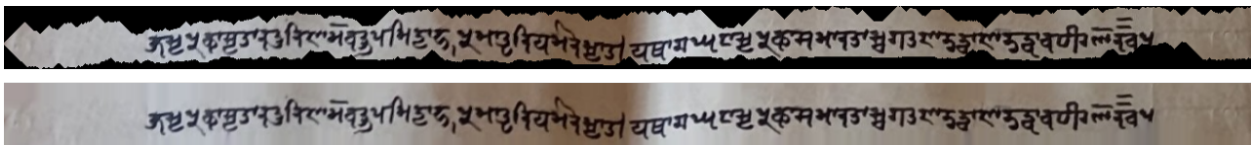


Figure 3. Above : Line segment using LineTR . Below : Inpainted Line segment

Each cropped line is then rectified with a single smooth warp, implemented as a calibration-style spatial transformer using paired control points and inspired by Curve Rectification [10]. The warp models the baseline as a smooth global deformation and yields dense remapping fields that project the text onto a flat, canonical strip. We use Lanczos resampling [7, 9] during warping to maintain edge sharpness and preserve intricate diacritics while correcting page curl, mild folds, and stylistic tilt in one pass.

For getting the new polygons, the rectifier provides a dense map from rectified $\rightarrow$ source (an "inverse" image map). To transfer these the other way (source $\rightarrow$ rectified), we numerically invert this map and forward-map polygons via scattered interpolation with a nearest-neighbor fallback to keep all vertices. Baseline scribbles from LineTR are mapped with the same procedure, producing rectified baselines used later for curvature-aligned synthesis.

To quantify residual curvature, we use a scribble-centric turning metric. Let the scribble be a polyline $\{\mathbf{p}_0, \ldots, \mathbf{p}_N\}$ (resampled to near-uniform arc-length spacing). With segments $\Delta\mathbf{v}_i = \mathbf{p}_{i+1} - \mathbf{p}_i$ and unit directions $\hat{\mathbf{v}}_i = \Delta\mathbf{v}_i / \|\Delta\mathbf{v}_i\|$, the local turn at vertex i is

$$\Delta\theta_i = \arccos\big(\mathrm{clip}(\hat{\mathbf{v}}_{i-1} \cdot \hat{\mathbf{v}}_i, -1, 1)\big), \quad \ell_i = \tfrac{1}{2}\big(\|\Delta\mathbf{v}_{i-1}\| + \|\Delta\mathbf{v}_i\|\big),$$

(or, equivalently, a numerically stable $\mathrm{atan2}$-based angle difference). The score

$$\textsc{MeanAbsTurnDegPer100px} = \left(\frac{1}{N-1}\sum_{i=1}^{N-1}\frac{|\Delta\theta_i|}{\ell_i}\right) \cdot \frac{180}{\pi} \cdot 100,$$

reports mean directional change (degrees) per 100 pixels: 0 indicates a straight line; larger values mean tighter bends. We skip degenerate joints with $\ell_i < \epsilon$ and optionally apply light pre-smoothing to suppress pixel-scale jitter. This discrete average approximates mean absolute curvature density in familiar units, and rectification consistently lowers it (see Fig. 2b , Table 1) while preserving glyph detail (Fig. 4).

4. Text Corpus

We compile our synthetic text material from large, publicly available Sanskrit corpora, including GRETIL [1], DCS [11], DSBC [12], and Sanskrit.org [15]. These repositories cover a wide range of prose and verse genres and serve as a convenient base for generating realistic input. All of these sources provide text in IAST (International Alphabet of

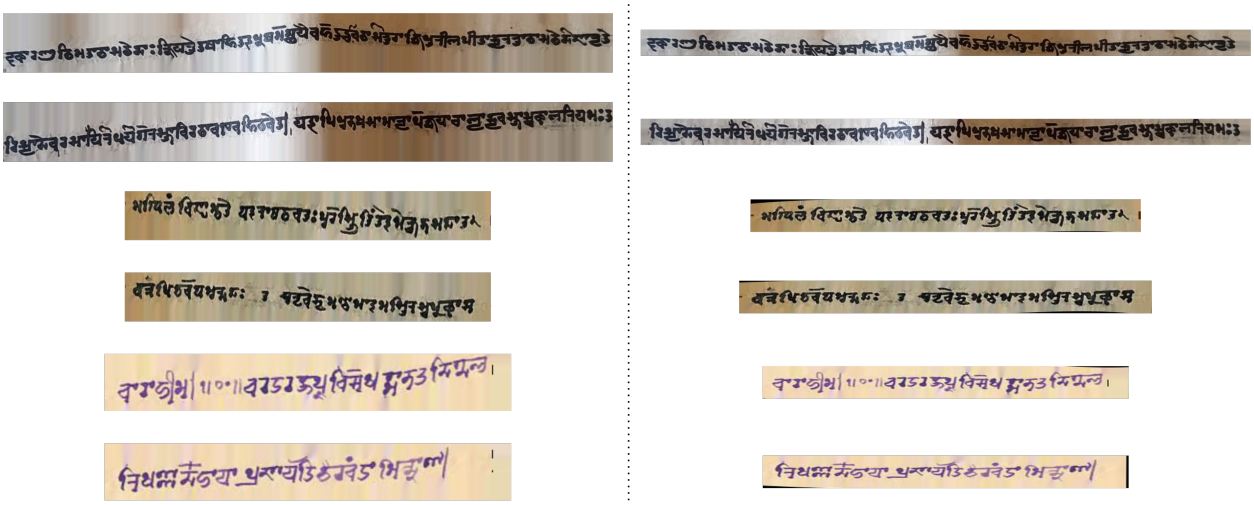


Figure 4. Qualitative effect of the rectification stage: lines are straightened while preserving fine diacritics and stroke detail.

Sanskrit Transliteration) [13], which we preserve during the initial stages of cleaning. Importantly, no text line from our real Sharada manuscripts is ever used for synthesis; the synthetic dataset relies entirely on the public corpora listed above.

The raw text is first subjected to a light cleaning pass. This involves removing non-content artifacts such as HTML or Markdown style tags and comments, deleting stray symbols, collapsing repeated spaces, trimming leading blanks, and reducing sequences of three or more empty lines to exactly two. We also drop short header-like fragments that are often inserted as page headers or navigation markers. The goal is not to aggressively re-shape the text, but simply to reduce noise while retaining the linguistic material. After cleaning, we transliterate IAST lines into the Sharada script using *Aksharamukha* [4, 8]. To ensure correctness, we employ a simple round-trip check on a random sample: each line is converted from IAST to Sharada and then back to IAST, and we verify that the recovered line matches the original line. Lines that fail this consistency test are excluded from further use, which provides a straightforward and effective guard against transliteration errors.

Because large text repositories often combine material from multiple editions, duplicate lines are common. We therefore apply a systematic de-duplication step. First, *exact duplicates* are removed by hashing the canonical form of each line. If two strings x_i and x_j produce the same hash value

$$h(x_i) = h(x_j),$$

they are considered identical and only one copy is kept. Next, we handle *near-duplicates*. Each line x is broken into a set of overlapping character Quintgram $F(x)$. For any two lines, we compute their Jaccard similarity

$$J(x_i, x_j) \;=\; \frac{|F(x_i) \cap F(x_j)|}{|F(x_i) \cup F(x_j)|}.$$

If $J(x_i, x_j) \geq 0.92$, we consider the two lines redundant and retain only one of them. This method ensures that trivial spacing or punctuation changes do not inflate the corpus with repeated material. After this filtering, the working corpus contains roughly 11 million unique Sanskrit lines.

Finally, to make the synthetic lines resemble the density of real manuscript lines, we perform length-matched sampling. For each real reference line of length ℓ_{ref}, we select a corpus line whose character length falls within

$$\ell \in [\,\ell_{\text{ref}} - 5,\ \ell_{\text{ref}} + 5\,].$$

The size of the corpus ensures that suitable matches are always available. In this way, every synthetic line used in our pipeline (i) comes from a public source, (ii) is cleaned and free of markup, (iii) is not a duplicate or near-duplicate, (iv) has been transliterated IAST→Sharada with a round-trip check, and (v) is length-matched to a real reference line to preserve realistic character density.

5. Font Color Estimation

To characterize the ink appearance of manuscript lines, we estimate a single representative font color from each line image. This is challenging because of uneven illumination, variable ink density, and anti-aliasing artifacts along stroke edges. We therefore employ a three-stage pipeline: (i) text segmentation, (ii) refinement of core-ink pixels, and (iii) robust color estimation via median statistics.

Text segmentation : Among several tested methods, adaptive Gaussian thresholding provided the most stable separation of text from background in our manuscripts. We use it to generate an initial mask M_{init} that marks the candidate text pixels.

Core ink refinement : The initial mask may still include lighter edge pixels from anti-aliasing or faded ink, which can bias color estimation. To focus only on the darkest and most stable stroke regions, we gather all grayscale values of pixels marked as text and compute a percentile threshold T_{core}. We retain only the darkest part (about 25%) of these pixels to obtain a refined mask M_{refined}.

Robust font color : Using M_{refined}, we extract the corresponding pixel values from the original RGB image. The font color is estimated as the channel-wise median

$$\text{Color}_{\text{font}} = \big(\,\text{median}(C_R),\ \text{median}(C_G),\ \text{median}(C_B)\,\big),$$

where C_R, C_G, C_B are the sets of red, green and blue values for all refined pixels. The median is chosen instead of the mean for robustness against outliers such as faint pixels or residual noise. However, in practice, exact median values do not correspond to a single fixed color across all samples. To stabilize the estimate, we discretize each channel into small buckets of width 5 and report the representative value of the bucket containing the median. This coarse quantization ensures that color estimates are robust and comparable across different lines. The resulting values yield a reliable representative ink color for each line image.

Note: This procedure was not described in the main paper, as subsequent analysis revealed that the majority of manuscript text lines fall into a narrow set of shades, most often black, navy blue, or gray, limiting the benefit of the method for Sharada manuscripts.

6. Warping Text

Baseline construction. To bend straight text so that it follows a handwritten curve, we first need a clear description of that curve. From the rectification stage we obtain a smoothed scribble, which we treat as a baseline curve $\mathbf{c}(s)$. Here s is the distance along the curve and the total length of the curve is L. At each point on the curve we know two directions: the tangent $\mathbf{t}(s)$, which tells us where the curve is heading, and the normal $\mathbf{n}(s)$, which points straight out from the curve. The straight text image is written as $I(u, v)$, with width W (columns u) and height H (rows v). In this straight version, the baseline row v_0 is the place where the characters are rendered, given by the font metrics.

Mapping straight to curved. The next step is to decide how the horizontal coordinate u of the straight image should map to a distance s along the curve. A simple proportional mapping would be:

$$s(u) = \tfrac{u}{W}\,L, \qquad u(s) = \tfrac{s}{L}\,W,$$

which evenly stretches the text so that the left edge of the image ($u = 0$) maps to the start of the curve ($s = 0$) and the right edge ($u = W$) maps to the end of the curve ($s = L$).

In our case, however, we have access to the actual font advances $a_1, a_2, \ldots, a_N$, which give the true widths of each character and space. Using these values, we build a more accurate mapping by placing cumulative anchors:

$$(s_k, u_k) = \left(\frac{\sum_{i=1}^{k} a_i}{\sum_{i=1}^{N} a_i}\,L,\ \frac{\sum_{i=1}^{k} a_i}{\sum_{i=1}^{N} a_i}\,W \right).$$

This ensures that letter spacing and word gaps from the straight image are preserved exactly when the text is bent along the curve, rather than being evenly stretched.

Warping by sampling. Finally, we actually "bend" the text. Instead of pushing each pixel of the straight image forward (which can cause overlaps or holes), we *pull* the correct pixel value from the straight image for each location along the curve. For a given output point $\mathbf{p}$ near the curve, we:

1. Find the closest point $\mathbf{c}(s)$ on the baseline.
2. Measure how far above or below the curve this point lies, using the normal direction:

$$d = \langle \mathbf{p} - \mathbf{c}(s), \mathbf{n}(s) \rangle.$$

3. Read the pixel value from the straight image at coordinates $(u(s), v_0 + d)$.

This procedure means every pixel in the final curved text image has a well-defined source. Letters keep their shapes and thickness because displacement is only in the normal direction. Since each output pixel is written exactly once, no gaps or overlaps appear in the warped image.

7. Background Extraction

We remove foreground text from line-level manuscript images with a three-stage procedure designed to get good quality backgrounds for synthetic data creation ,while preserving paper texture:

Figure 5. Example of the input image (left), the HiSAM mask (middle), and the extracted background (right).

Text segmentation using Hi-SAM. To obtain line masks, Hi-SAM [18] is configured to dense point sampling. High-confidence detections are retained, their contours converted to polygons, and each polygon expanded by a small outward offset to include halos and thin ascenders/descenders without touching clean background. A light elliptical dilation then provides a conservative margin. The resulting masks filled regions that fully cover the text, preserve line margins, and remain stable under downsampling—yielding unambiguous replace areas for diffusion inpainting.

As an alternative, we tested LineTR-derived line polygons as the inpainting mask. However, these polygons often spill from the strokes into the blank margins of the line crop (Fig. 6 middle), causing diffusion to repaint the margins, alter paper texture, and introduce seams. By contrast, Hi-SAM masks stay confined to the text region (Fig. 6 bottom), preserving margins. This margin preservation motivate using Hi-SAM for text removal, while reserving LineTR for warping text lines

Background synthesis (Stable Diffusion Inpainting). We remove text by inpainting the masked regions with a Stable Diffusion [14] inpainting model, guided by short texture-oriented prompts that encourage paper-like fills and explicitly suppress text. In practice, we used a positive prompt for background and paired with a strong negative prompt to remove text, this reliably produced natural substrates. The negative prompt is critical: it consistently prevents letter-like hallucinations and reduces the need for aggressive parameter tuning.

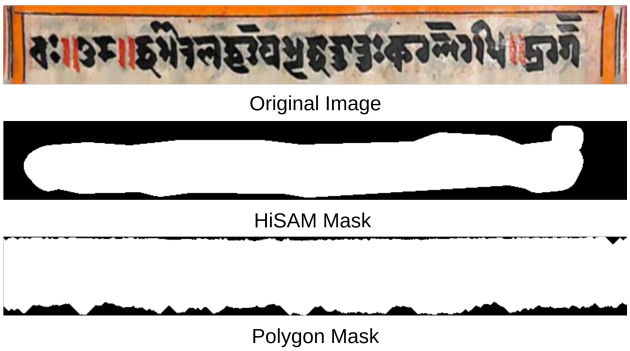


Figure 6. Line image (top), HiSAM mask (middle), and polygon mask (bottom). The polygon mask covers the left, right, and top margins, making it unsuitable for text inpainting.

Positive: `No text, clean plain background, natural surface`
Negative: `text, letters, words, characters`

Residual artifact cleanup (classical post-processing). Diffusion can leave speckles or faint shadows. We detect these with adaptive (Sauvola) thresholding [16], lightly dilate to ensure coverage, and keep only small connected components as a residual mask. A final, deterministic TELEA [17] inpainting pass is applied exclusively to this residual mask, to get the background without invoking diffusion again. In practice, this "diffusion + light post-processing" pipeline reliably removes text while preserving the subtle fiber and tonal variation characteristic of historical paper Fig. 5.

8. Synthetic Data Samples

Our synthetic dataset follows the pipeline described in the main paper and is both *curvature-aligned* and *background-aware*. We warp straight Sharada renderings along rectified baseline scribbles to match real line geometry, and composite them onto text-free paper substrates obtained via background extraction. This yields lines that mirror real curvature, spacing, and paper tone while remaining text-safe. No real manuscript text is used to construct the synthetic corpus Fig. 7.

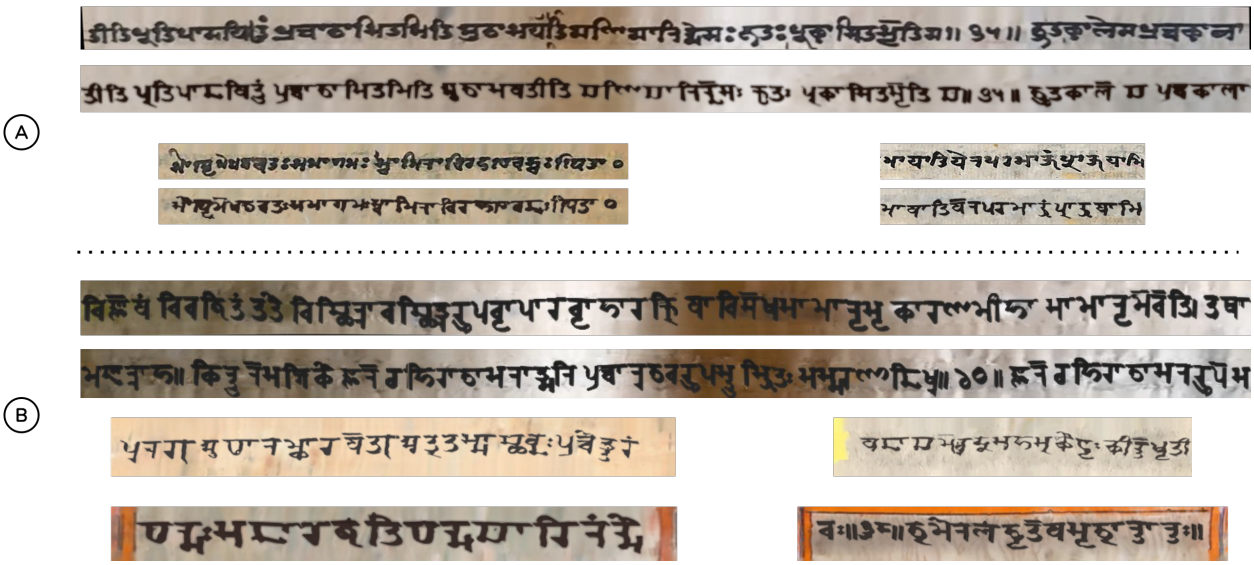


Figure 7. **Synthetic data examples. A** Real Sharada line segments from UUV/SLA (top of each pair) and synthetic counterparts (bottom) generated by our pipeline using the rectified baseline scribble and the extracted paper background of the reference. For illustration only, pairs reuse the same transcription to highlight geometry/appearance matching; the synthetic corpus itself does *not* use any real manuscript text. **B** Additional samples produced by our curvature- and background-aware generator, showing varied paper substrates and line shapes. *No real manuscript text is used to construct the synthetic corpus.*

9. Extensions

9.1. Rectification module to other scripts

Our pipeline runs directly on real manuscript pages across scripts *without any code modifications*. The end-to-end flow—line detection, curvature-profile estimation, and rectification—is fully script-agnostic. We demonstrate this on Telugu Manuscript from India subcontinent to illustrate cross-script extensibility; see Fig. 8. The exact same executable used for Sharada is applied. We do not report quantitative OCR here due to the lack of ground-truth transcriptions for these manuscripts; instead, we include qualitative outputs highlighting robustness under high curvature and dense ligatures. We expect that the proposed curvature-aware processing will lower the barrier to future OCR efforts in low-resource scripts by enabling scalable pre-processing and line normalization even in the absence of labeled data.

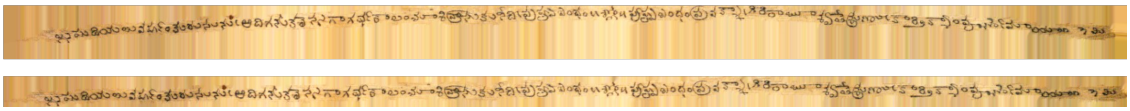


Figure 8. Rectification pipeline on telugu manuscript

9.2. Background extraction to pagelevel

We extend our background-extraction approach from lines to full pages, enabling page-level substrates that act as realistic canvases for compositing curvature-aligned synthetic text. This generalizes our synthetic pipeline beyond isolated lines and opens a direct path toward page-level OCR training and evaluation (Fig. 9), without changing much of the pipeline.

Figure 9. Manuscript leaves and corresponding extracted backgrounds

References

[1] Gretil: Göttingen register of electronic texts in indian languages. https://gretil.sub.uni-goettingen.de/, 2020. Accessed 2025-08-11. 4

[2] Vivritti commentary of ishvar pratyabhijna by utpal: Sharada folios with devanagari transcript. https://archive.

org/details/dsor_utpal-vivritti-final-transcript-sharada-to-devanagari-sarayu-foundation-trust-and-e-gangotri/page/n27/mode/2up, 2023. Accessed 2025-08-11. 3

[3] Sharadalipi magazine archive. https://www.shardalipi.com/, 2025. Accessed 2025-08-11. 3

[4] Shreevatsa A., Vinodh Rajan, and contributors. Aksharamukha: Script converter – about. https://www.aksharamukha.com/about, 2025. Accessed 2025-08-11. 5

[5] Vaibhav Agrawal, Niharika Vadlamudi, Muhammad Waseem, Amal Joseph, Sreenya Chitluri, and Ravi K. Sarvadevabhatla. Linetr: Unified text line segmentation for challenging palm leaf manuscripts. In *Proceedings of the International Conference on Pattern Recognition (ICPR)*, pages 217–233. Springer, 2024. 3

[6] M. Bertalmio, A.L. Bertozzi, and G. Sapiro. Navier-stokes, fluid dynamics, and image and video inpainting. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, pages I–I, 2001. 3

[7] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000. 3, 4

[8] Aksharamukha developers. Aksharamukha python package. https://pypi.org/project/aksharamukha/, 2025. Accessed 2025-08-11. 5

[9] Claude E. Duchon. Lanczos filtering in one and two dimensions. *Journal of Applied Meteorology and Climatology*, 18(8): 1016 – 1022, 1979. 4

[10] frotms. Curve-text-rectification-using-pairs-of-points. https://github.com/frotms/Curve-Text-Rectification-Using-Pairs-Of-Points, 2020. Accessed 2025-08-11. 4

[11] Oliver Hellwig. Digital corpus of sanskrit (dcs). http://kjc-fs-cluster.kjc.uni-heidelberg.de/dcs/index.php, 2015. Accessed 2025-08-11. 4

[12] Ligeia Lugli and Luis Quiñones. Mangalam corpus of buddhist sanskrit literature. https://doi.org/10.5281/zenodo.15739330, 2025. Zenodo dataset, CC-BY 4.0. 4

[13] Monier Monier-Williams. A sanskrit-english dictionary, 1899. 5

[14] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10684–10695, 2022. 7

[15] Sanskrit Documents. Sanskrit documents. https://sanskritdocuments.org/. Accessed 2025-08-11. 4

[16] J. Sauvola and M. Pietikäinen. Adaptive document image binarization. *Pattern Recognition*, 33(2):225–236, 2000. 8

[17] Alexandru Telea. An image inpainting technique based on the fast marching method. *Journal of Graphics Tools*, 9(1):23–34, 2004. 8

[18] Maoyuan Ye, Jing Zhang, Juhua Liu, Chenyu Liu, Baocai Yin, Cong Liu, Bo Du, and Dacheng Tao. Hi-sam: Marrying segment anything model for hierarchical text segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 47(03):1431–1447, 2025. 7