Towards Digitizing Filled Indic Handwritten Forms

Shaon Bhattacharyya, Ajoy Mondal, and C. V. Jawahar

CVIT, International Institute of Information Technology, Hyderabad, India
shaon.b@research.iiit.ac.in
{ajoy.mondal,jawahar}@iiit.ac.in

Abstract. The demand for daily form digitization requires extensive manual effort. This paper presents an efficient pipeline for digitizing Indic handwritten forms, minimizing human intervention. We validate the pipeline using *Hindi* and *Bengali* forms, creating a dedicated test set, *IIIT-Indic-Form-Test*. Our approach enables form capture and orientation alignment via smartphone, extracting printed and handwritten fields with OCR enhanced by template annotations. A predefined word list supports post-processing for fields like *name*, *state*, and *country*. We compare our pipeline with tools like Google Parser and Microsoft Azure and conduct ablation studies on form style, rotation, and handwriting variations. A GUI-based application for digitization is also developed. The code and model are publicly available at https://github.com/shaoncvit/Indic_Handwritten_Form_dataset.

Keywords: Handwritten · form · digitization · OCR · Indic languages · template.

1 Introduction

Imagine a bustling rural bank where individuals fill out handwritten forms for various tasks—a scene common across banks, university admissions, municipal permissions, and healthcare services. Forms are essential for data collection, record-keeping, compliance, and decision-making. Traditionally, physical records required extensive storage and were prone to misplacement. With the digital shift of the 21st century, there's been a push towards digitization to streamline archiving. Although electronic forms (E-forms) have become popular, they still demand significant manual input, leading to delays, errors, and a subpar user experience. This highlights the need for an automated pipeline to digitize Indic handwritten forms and reduce manual effort.

Document imaging and analysis have advanced with deep learning, making tools for word detection [1], layout analysis [10, 23, 24], and OCR widely available for digitizing content across domains. However, these techniques don't fully address the unique challenges of handwritten forms, which contain field-value pairs mixing handwritten and printed text. Effective form digitization requires

recognizing content in both fields and values, matching pairs, and handling challenging handwritten text, often with proper nouns like names and addresses. Diverse layouts, field types, fonts, and handwriting styles further complicate automated form digitization.

This work introduces a novel pipeline for digitizing handwritten forms, combining basic image processing techniques with state-of-the-art OCR methods. Initially, we utilize an empty template of each form to manually annotate *"Keys"* and their corresponding *"Values"* regions using annotation tools. This process is performed once for each template. Subsequently, our pipeline aligns the filled handwritten forms submitted by users and identifies the printed regions and their corresponding handwritten sections based on the pre-annotated templates. The OCR APIs from Bhashini[1] for Indic languages recognize text in printed and handwritten areas within the form fields. Further, we apply field-specific post-processing, addressing location and gender. It is important to note that our pipeline is designed to complement human effort rather than entirely replace it. Therefore, a human operator verifies the output for accuracy and rectifies any additional errors. We conduct an extensive user study to evaluate the effectiveness of different modules within our pipeline. To facilitate this evaluation, we compile and release a substantial handwritten form test set, *IIIT-Indic-Form-Test* encompassing two languages, including *Hindi, Bengali* along with manually annotated attributes.

In summary, our contributions are as follows,

- Develop an innovative pipeline for digitizing handwritten forms in Indic languages, integrating multiple language-specific OCR modules.
- Collect and publish a significant test set of Indic handwritten forms, titled *IIIT-Indic-Form-Test*, used for evaluation. This dataset can serve as a benchmark for future research in similar areas (see Fig. 5).
- Compare the performance of the proposed pipeline with commercial tools like Google Parser and Microsoft Azure.
- Create a web-based application for digitizing forms tailored to specific applications.

2 Related Work

Researchers have long been intrigued by various challenges in document imaging, particularly in tasks like word detection, which lays the foundation for subsequent Optical Character Recognition (OCR) processes. Recent advancements, such as East [25], PixelLink [6], CRAFT [1], and TextSnake [16], have provided robust solutions applicable to a wide range of documents. The ability to detect word boundaries naturally led to whether we could accurately recognize printed text.

[1] https://bhashini.gov.in/

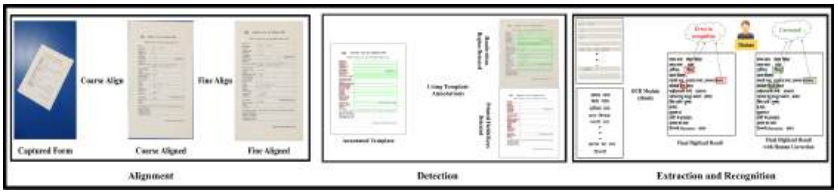

Fig. 1: Illustrates a pipeline designed to automate the digitization of handwritten forms in various Indic languages. The proposed approach includes multiple stages to digitize *key-value* pairs for a specific form, leveraging both the provided empty template and user-submitted images of completed forms. Best viewed in Zoom.

Recognizing printed text has been a focus of research for quite some time. Early works preceding the deep learning era [3, 12] marked the initial steps toward automating document digitization from images. Modern deep learning-based OCR systems like Tesseract [21], EasyOcr [11], and MMOCR [14] excel in processing diverse document images, encompassing various fonts, writing styles, and even handwritten text to a certain extent. However, human handwriting exhibits significant variability, prompting the creation of large datasets such as IAM Handwritten Forms Dataset [8], RVL-CDIP [18], and Tobacco3482 [9]. These datasets compile various handwriting styles and vocabularies to train models specifically for recognizing handwritten text [4, 7, 15]. Only a few datasets, such as FUNSD [13] and FUDGE[2], are publicly available for form understanding.

Despite the power of deep learning system, incorporating human expertise into deep learning system is not fallible and cannot guarantee 100% accuracy. Ensuring perfect digitization is crucial, especially for user-filled forms containing sensitive and confidential information. This demand for precision is also evident in other domains, such as medical imaging applications [2], where experts make final decisions while automated systems serve as supportive tools. Human-in-the-loop systems have proven effective in correcting OCR errors [5], particularly in tasks involving digitizing extensive document collections on the web. In our work, we embrace this concept to ensure the absolute accuracy of digitized forms, making them viable for real-world applications.

3 Indic Handwritten Form Digitization

Our handwritten form digitization pipeline comprises three primary steps — (i) form alignment, (ii) handwritten region detection, (iii) handwritten text detection and recognition, and finally, optional (iv) post-OCR error correction. Fig. 1 provides a visual representation of the pipeline. In this section, we will delve into each step in detail.

[2] https://github.com/LLNL/fudge

3.1 Aligning Handwritten Forms

Our pipeline initiates by aligning the user-captured handwritten form. This process involves a two-step algorithm: *Coarse Alignment* and *Fine Alignment*, ensuring standardized alignment with the form's template.

Coarse Alignment:

We begin with the user-captured form image in the alignment procedure, followed by separating the background and foreground from the image utilizing the U2-Net model [19]. Then, we detect the contours using the OpenCV function "*findContours*" after obtaining the masked image. Subsequently, we extract the four corner points from the contour and label them as top-left, top-right, bottom-left, and bottom-right. We then apply Algorithm 1 to obtain the coarse-aligned image, which proceeds to the next step of the alignment process. Fig. 2 visually illustrates the coarse alignment process.

Algorithm 1 The algorithm of course aligns the user-uploaded image of a handwritten form.

$pts_1 \leftarrow [(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4)]$
$i_p \leftarrow I_h$ ▷ Input Image
$t_p \leftarrow I_t$ ▷ Target Image
$h_t, w_t \leftarrow t_p$ ▷ Height and Width of Template
$pts_2 \leftarrow [(0, 0), (w_t, 0), (0, h_t), (x_t, h_t)]$
$M \leftarrow PerspectiveTransform(pts_1, pts_2)$
$o_p \leftarrow WarpPerspective(i_p, M, w_t, h_t)$

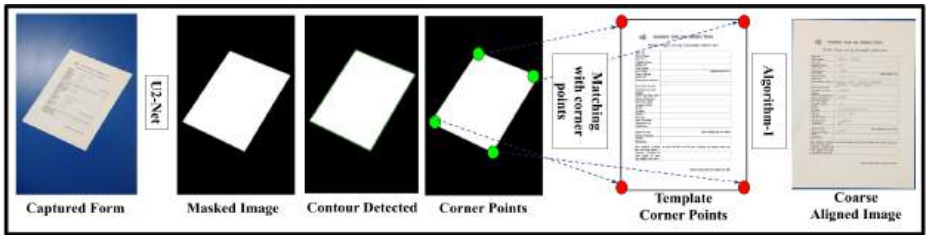

Fig. 2: Illustrates the intermediary steps of the coarse alignment procedure. Initially, the aim is to identify the corners of the recorded forms and subsequently compare them with the corners of the template. Best view in the Zoom.

The algorithm of course aligns the user-uploaded image of a handwritten form. Here, the pts_1 denotes the four corner points of the contour where (x_1, y_1) is the top-left corner point, (x_2, y_2) is top-right corner point, (x_3, y_3) is bottom-left

corner point and (x_4, y_4) is bottom-right corner point. Similarly, the pts_2 denotes a four-point representation of the form's template. Now, we utilize the function called $PerspectiveTransform$, which requires two arguments: $originalPoints$, a list of four corner points of the contour, and $transformedPoints$, corner points of the form's template. This function returns M, which is the transformation matrix. We apply this transformation matrix M, the input image captured by the user in the $WarpPerspective$ function, to get o_p, our coarse aligned image used in the second step of the alignment process.

Fine Alignment:

In the coarse alignment phase, matching corner points with the template was challenging due to variations in printer settings, which added extra white space around the template. This extra space caused pixel defects, miss-aligning detected corners with the form's actual corners, impacting the alignment of handwritten regions (see Fig. 4). Additionally, handling different form orientations added complexity. To address this, we extracted the four corner points, generated four permutations to cover all orientations, and selected the best-aligned image for the next phases.

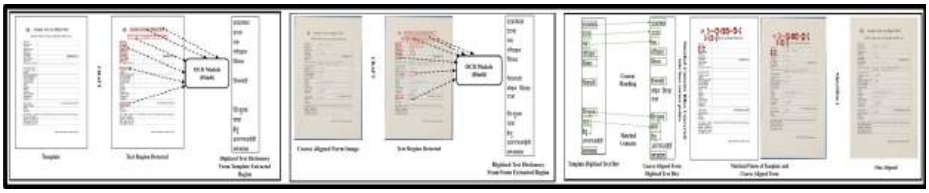

Fig. 3: Illustrates the intermediate steps of the fine alignment process. We extract content from the coarsely aligned form and the template to align the corresponding elements accurately. Best view in the Zoom.

Algorithm 2 The algorithm used to precisely align the user-uploaded image of the handwritten form.

$TemplateContentList \leftarrow list(DigitizeContentofthetemplate)$
$CoarseAlignedImageContentList \leftarrow list(DigitizeContentoftheimage)$
for $i \leftarrow len(CoarseAlignedImageContentList)$ **do**
    $score \leftarrow calBleu4(TemplateContentList, CoarseAlignedImageContentList[i])$
    if $score \geq 0.50$ **then**
        $MatchFound$
        $cnt \leftarrow cnt + 1$
if $cnt \geq 20$ **then**
    $SufficientMatchFound$

These challenges necessitated the introduction of an additional alignment step, termed fine alignment, as depicted in Fig. 3. The primary objective of fine alignment is to precisely align the image's content with the corresponding content in the template. Initially, we utilize the CRAFT [1] text detection module to identify the inner content of the template and the coarsely aligned image. Subsequently, optical character recognition (OCR) is employed to recognize the content within each. We then introduce Algorithm 2, designed to obtain matched contents corresponding to bounding boxes from both the template and the coarsely aligned image. We extract four points using the same Algorithm 1 employed in the coarse alignment procedure for each bounding box. This process enables us to obtain the final aligned image, which will be utilized in the subsequent phase of our pipeline.

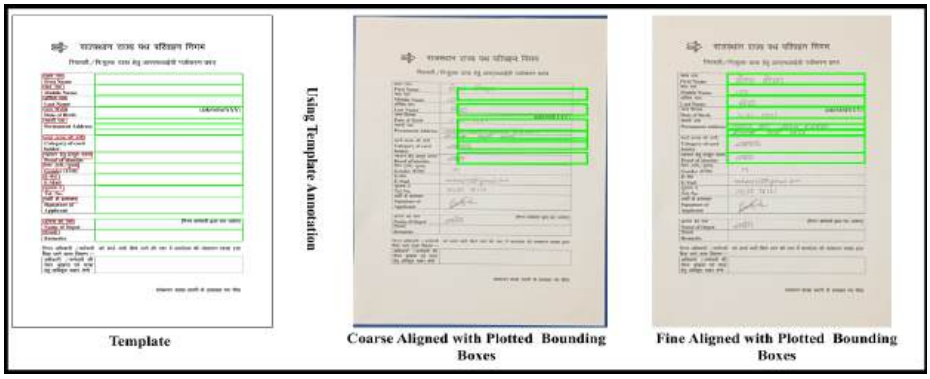

Fig. 4: Illustrates the disparities between the results obtained from coarse alignment and fine alignment techniques, providing a rationale for adopting fine alignment. Best view in the Zoom.

The template and image content are initially digitized and stored in lists in Algorithm 2. Subsequently, the algorithm assesses the similarity between the image's content and the template by computing the BLEU [17] score. If the similarity score for any image content exceeds 0.50 (a hyper-parameter), it is deemed a match, and the counter is incremented. Finally, the algorithm searches for an additional 20 matches (a hyper-parameter). If the count meets or exceeds this threshold, the algorithm determines that a *"Sufficient Match"* has been identified. Therefore, the algorithm selects the one that fulfills all its criteria as the final aligned image among the four coarsely aligned images.

3.2 Handwritten Region Detection

The subsequent step involves detecting the handwritten regions of the aligned form once the user uploads the form aligned. This process relies on the template annotation of the particular form. Each key and its corresponding value are

tagged in the template, facilitating the matching of digital keys and values and ensuring accurate interpretation of the data.

Algorithm 3 The algorithm to detect handwritten region.

Initialize variables:
    $j \leftarrow 0$
    $c \leftarrow []$
    $counter \leftarrow False$

for i in range(gray.shape[1] − 100) **do**
    Extract region i_m of width 100 pixels from column i
    if Number of white pixels > $\frac{\text{Number of black pixels}}{10}$ in i_m **then**
        $counter \leftarrow True$
    else
        if $counter == True$ **then**
            if Region does not represent a continuous straight line **then**
                Append (j, i) to c
            $j \leftarrow i$
        $counter \leftarrow False$
if $counter == True$ **then**
    if Last region does not represent a continuous straight line **then**
        Append $(j,$ gray.shape[1]$)$ to c

3.3 Handwritten Text Detection and Recognition

After identifying handwritten regions, we isolate them carefully, though residual white space can reduce OCR accuracy. To mitigate this, we introduce a secondary cropping algorithm (Algorithm 3) designed to extract handwritten areas from cropped images. This algorithm scans the image columns, isolating regions of a set width and checking for continuous handwritten lines. The output is a list of tuples with the start and end coordinates of these detected lines.

After obtaining these finely detailed handwritten regions, we transcribed them using optical character recognition (OCR) technology. For Indic languages (*Hindi and Bengali*), we employ APIs from Bhashini[3]. Subsequently, following the annotation-specified OCR on the *keys* and *values* within the handwritten regions, we store the identified *key-value* pairs in a text file for further processing and review. Finally, human operators are engaged to rectify errors in the recognized text.

3.4 Automatic Post-OCR Error Correction

Inaccuracies in the recognized content are expected to be encountered following the digitization of keys and values. We introduce an automated post-processing

[3] https://bhashini.gov.in

technique for error detection to mitigate this. Specific fields like *name and state* consistently contain particular values in many forms. Then we compile lists of names and states to enhance accuracy, especially given the diversity of Indian names and states. With the help of this list, we automatically correct the wrongly predicted names and states in the forms based on Edit distance [20] among the list of words and predicted words.

4 Experiments

4.1 IIIT-Indic-Form-Test Dataset

We create the *IIIT-Indic-Form-Test* dataset to evaluate the proposed algorithm. The dataset includes three form templates for *Hindi* and five for *Bengali*. Forty participants (aged 20-27) proficient in Hindi completed the Hindi forms, while fifty participants (of the same age range) filled out the Bengali forms. To preserve privacy, participants were instructed not to include personal information. The forms are captured using various mobile cameras, including iPhones, with different camera resolutions. Photos are taken against different backgrounds to prevent interference with the handwritten content and with random rotations between 0 and 360 degrees. We also vary the distance of the camera while capturing the images. Each form is duplicated ten times with different rotations, resulting in 1200 *Hindi* forms (from 40 users) and 2500 *Bengali* forms (from 250 users), totaling 3700 forms with structural and rotational variations. We manually annotate all forms, and the dataset will be a valuable resource for future research on handwritten form analysis. Fig. 5 shows a few samples of filled forms from our dataset.

4.2 Evaluation Metrics

The *Structural Similarity Index* (SSIM) [22] measures the similarity between two images, focusing on structural aspects rather than pixel-wise differences like *Mean Squared Error* (MSE) or *Peak Signal-to-Noise Ratio* (PSNR). The SSIM score ranges from -1 to 1, with values close to 1 indicating high similarity. We use the SSIM score in Eq. (1) to assess the effectiveness of the form alignment algorithm.

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}, \tag{1}$$

where μ_x, μ_y, μ_x^2, μ_y^2, and σ_{xy} are the pixel mean, variance, and covariance of images x and y, respectively.

We utilize two widely recognized evaluation metrics, namely, *Character Error Rate (CER)* and *Word Error Rate (WER)*, to evaluate the performance of the model. Error Rate (ER) is defined as:

$$ER = (S + D + I)/N, \tag{2}$$

Fig. 5: Displays sample filled forms from our *IIIT-Indic-Form-Test* dataset, captured with varying mobile camera configurations, orientations, and scales. Zoom in for a clearer view.

where S represents the number of substitutions, D denotes the number of deletions, I signifies the number of insertions, and N indicates the total number of instances in reference text. In the context of *CER*, Eq. (2) operates at the character level, and while of *WER*, Eq. (2) operates at the word level.

4.3 Result Analysis

While our main goal is to assist with digitizing forms, we also evaluate each module within our pipeline to understand their contributions. Additionally, we compare the final output of our pipeline with the results from Google Form Parser and Microsoft Azure Form Processing tools. Fig. 6 showcases the digitization of two handwritten Indic forms in *Hindi* and *Bengali*, both captured using a mobile phone.

Evaluation on Alignment of Forms:

We use SSIM to evaluate the effectiveness of our alignment process. Our primary goal is to align the content and corner points of the captured forms with the corresponding template during alignment. Ideally, each content region in

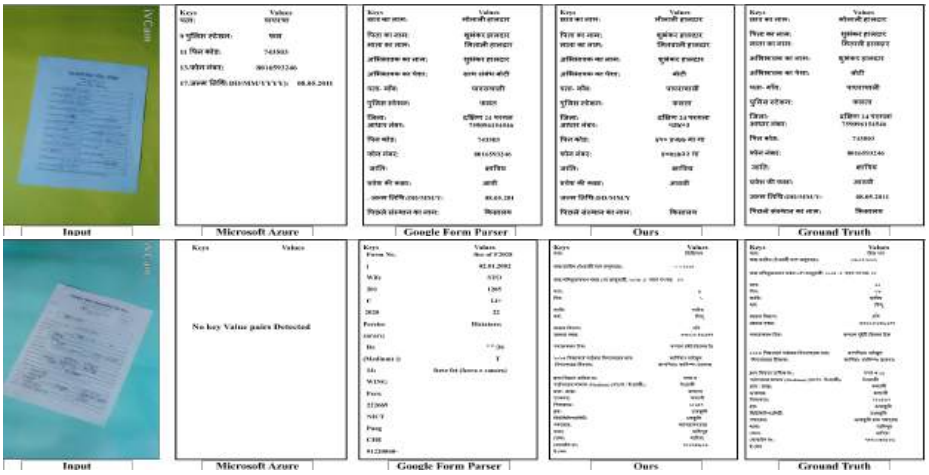

Fig. 6: Displays the final results of digitized handwritten forms in Hindi and Bengali, processed through our pipeline, Microsoft Azure form processor, and Google form parser. For optimal viewing, zoom in on the images.

Language	#Forms	#Styles	#Users	Alignment	Recognition	
				SSIM↑	WER↓	CER↓
Hindi	1200	3	40	0.75	0.67	0.55
Bengali	2500	5	50	0.76	0.78	0.65

Table 1: Shows the language-wise average quantitative scores over all results obtained by our pipeline. ↑ indicates that a higher value corresponds to better results, while ↓ indicates that a lower value corresponds to better results.

Language	Azure		Google OCR		Ours	
	WER↓	CER↓	WER↓	CER↓	WER↓	CER↓
Hindi	0.86	0.75	**0.42**	**0.28**	0.67	0.55
Bengali	-	-	-	-	**0.78**	**0.65**

Table 2: Presents the performance comparison of form digitization across different systems. ↓ indicates lower value corresponds to better result.

the aligned forms should closely match the template. We use printed field annotations as keys within the templates to achieve this. After aligning the forms using the CRAFT model [1], we identify each region in the aligned forms, ensuring that regions with matching keys align closely with the annotated keys in the templates. Table 1 presents the SSIM scores for filled forms by language, showing minimal differences in IoU scores between Hindi and Bengali. The SSIM scores indicate robust alignment between the regions of interest in the captured forms

and their corresponding areas in the template, despite the complex orientation of the forms.

Evaluation on Form Digitization:

We evaluate our pipeline's OCR module for handwritten text recognition using WER and CER scores, shown in Table 1 for Bengali and Hindi. Results indicate better accuracy for Hindi forms than Bengali. We also compare our pipeline with commercial tools like Google Form Parser and Microsoft Azure Form Recognizer (Table 2). Google's parser outperforms ours on Hindi forms but struggles with Bengali, likely due to its training on Hindi-specific data, while our OCR is trained on more generalized datasets. In contrast, our pipeline consistently surpasses Azure's for both languages, demonstrating robustness with complex scripts like Bengali. These results show our method's effectiveness for multilingual form digitization.

4.4 Ablation Study

Different DPI of Handwritten Forms:

DPI	SSIM↑	WER↓	CER↓
143	0.75	0.79	0.58
137	0.74	0.75	0.58
125	0.75	0.78	0.57
120	0.77	0.75	0.56
107	0.77	0.78	0.57
78	0.70	0.82	0.67
74	0.73	0.78	0.57
71	0.59	0.94	0.88
65	0.55	0.95	0.88

Table 3: Displays the performance of alignment and digitization in our pipeline across various DPIs.

We evaluate the performance of our end-to-end pipeline at various camera distances, using mobile phones in natural lighting. Photos are taken with an iPhone and other high-megapixel devices, capturing forms at different DPIs. We assess alignment accuracy with SSIM and digitization performance with WER and CER. Results in Table 3 show that low DPI (71 or less) correlates with poor alignment (low SSIM) and sub-optimal digitization (higher WER and CER). In contrast, a DPI range of 74 to 143 yields better SSIM scores and improved digitization results.

Orientation	SSIM↑	WER↓	CER↓
0	0.75	0.79	0.58
45	0.74	0.77	0.56
90	0.75	0.79	0.55
135	0.77	0.78	0.62
180	0.77	0.77	0.59

Table 4: Performance of our pipeline's alignment and digitization at various orientations.

Form Template Style	SSIM↑	WER↓	CER↓
Dotted Line	0.75	0.68	0.56
Straight Line	0.75	0.87	0.58
Only Boxes	0.77	0.71	0.34
Straight Line + Boxes	0.78	0.57	0.62
Table	0.75	0.67	0.57

Table 5: Performance of our pipeline's alignment and digitization across various styles of handwritten form templates.

Different Orientation of Handwritten Forms:

We randomly select ten filled forms in Bengali and Hindi and capture images at different angles ranging from 0 to 180 degrees (as shown in Fig. 5 to evaluate the performance of our alignment algorithm and digitization process. The digitization results of these oriented forms are presented in Table 4. The table indicates that varying the orientation significantly does not affect performance.

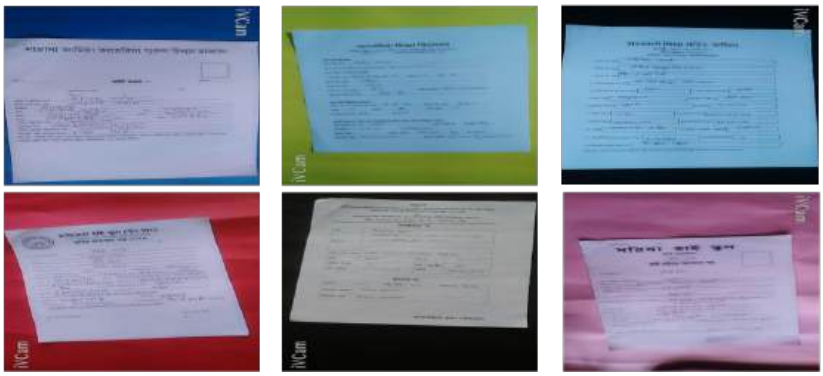

Fig. 7: Displays various styles of form templates completed by users in their respective languages. The first column includes — (i) dotted lines, (ii) straight lines, and (iii) only boxes. The second column features — (i) a straight line with boxes and (ii) a table, while the last column again shows a straight-line template.

Different Form Style:

The form templates are categorized into five types — table-like, boxed, dotted line, straight line, and straight line with boxes (see Fig. 7). We collect twenty filled forms in Bengali and Hindi from different users. We evaluate the align-

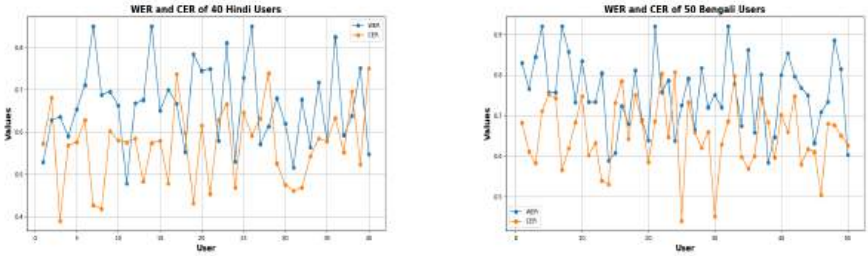

Fig. 8: Shows the WER and CER change in the different handwriting styles of Hindi and Bengali users.

ment and digitization performance of our pipeline on them. The performance results are presented in Table 5.The table shows that SSIM values remain fairly consistent across different styles. However, templates with straight lines exhibit notably lower OCR accuracy compared to others. This issue arises from fixed line annotations, which lead to character cropping when handwriting extends beyond these lines. The complexity of Hindi and Bengali scripts exacerbates this challenge.

Fig. 9: Shows examples of handwritten forms with distorted edges and occluded sections, highlighting scenarios where our pipeline fails to generate accurate results.

Different Writing Style:

Fig. 8 illustrates OCR performance for Bengali and Hindi forms across different writing styles, based on data from 40 Hindi and 50 Bengali users. We assess the effectiveness of our digitization pipeline using WER and CER. The plots in Fig. 8 reveal how different writing styles impact OCR accuracy, highlighting areas for improvement and refinement in our pipeline to handle various handwriting styles more effectively.

Practical Use Case:

We tested our end-to-end form digitization pipeline with forms from the ASHA foundation, used by ASHA workers for cancer patients. We work with two forms:

one in Bengali and the other one bilingual in Bengali and English. Our pipeline digitizes these forms, detects tick marks, and automatically classifies the language. We also create a FastAPI[4] interface for public access[5].

5 Limitations

Our pipeline has limitations, particularly during the coarse alignment phase, where accurately pinpointing corner points can be challenging. Issues such as corrupted or obscured corner points (see Fig. 9) can cause alignment failures. The pipeline's effectiveness also depends on having well-annotated templates. If a form lacks a corresponding template and annotation, extracting handwritten regions becomes problematic. Despite these challenges, we continue to refine our technique, achieving strong accuracy in our digitization efforts.

6 Conclusion

This study presents a novel end-to-end pipeline for digitizing handwritten forms, which minimizes manual intervention while achieving high accuracy. The pipeline employs corner point detection and content-matching techniques in a two-stage process to align images with their corresponding templates. Handwritten regions are extracted using template annotations and specialized OCR modules for English, Hindi, and Bengali. Human input is also incorporated to refine accuracy and support post-processing. Additionally, we introduce a new handwritten dataset featuring multiple users and languages, including Bengali and Hindi, to support future research. We also develop a web-based application for digitizing forms filled in Indic languages.

References

1. Baek, Y., Lee, B., Han, D., Yun, S., Lee, H.: Character region awareness for text detection. In: CVPR. pp. 9357–9366 (2019) 1, 2, 6, 10
2. Bansal, H., Khan, R.: A review paper on human computer interaction. IJARCSSE **8** (2018) 3
3. Brunelli, R.: Template matching techniques in computer vision: theory and practice. John Wiley & Sons (2009) 3
4. Coquenet, D., Chatelain, C., Paquet, T.: End-to-end handwritten paragraph text recognition using a vertical attention network. IEEE Trans. on PAMI **45**, 508–524 (2020) 3
5. Das, D., Philip, J., Mathew, M., Jawahar, C.V.: A cost efficient approach to correct ocr errors in large document collections. In: ICDAR. pp. 655–662 (2019) 3
6. Deng, D., Liu, H., Li, X., Cai, D.: Pixellink: Detecting scene text via instance segmentation. In: AAAI (2018) 2

[4] 10.4.16.81:8001/docs/
[5] More visual results can be found in supplementary material.

7. Diaz, D.H., Qin, S., Ingle, R.R., Fujii, Y., Bissacco, A.: Rethinking text line recognition models. ArXiv (2021) 3
8. Grieggs, S., Shen, B., Li, P., Short, C., Ma, J., McKenny, M., Wauke, M., Price, B.L., Scheirer, W.J.: Measuring human perception to improve handwritten document transcription. IEEE Trans. on PAMI **44**, 6594–6601 (2019) 3
9. Harley, A.W., Ufkes, A., Derpanis, K.G.: Evaluation of deep convolutional nets for document image classification and retrieval. ICDAR (2015) 3
10. Huang, Y., Lv, T., Cui, L., Lu, Y., Wei, F.: LayoutLMv3: Pre-training for document ai with unified text and image masking. In: ACM International Conference on Multimedia. pp. 4083–4091 (2022) 1
11. JaidedAI: Easyocr. https://github.com/JaidedAI/EasyOCR (Year) 3
12. Jain, A., Duin, R., Mao, J.: Statistical pattern recognition: a review. IEEE Trans. on PAMI **22**(1), 4–37 (2000) 3
13. Jaume, G., Ekenel, H.K., Thiran, J.P.: FUNSD: A dataset for form understanding in noisy scanned documents. In: International Conference on Document Analysis and Recognition Workshops (ICDARW) (2019) 3
14. Kuang, Z., Sun, H., Li, Z., Yue, X., Lin, T.H., Chen, J., Wei, H., Zhu, Y., Gao, T., Zhang, W., Chen, K., Zhang, W., Lin, D.: Mmocr: A comprehensive toolbox for text detection, recognition and understanding. ACM Multimedia (2021) 3
15. Li, M., Lv, T., Cui, L., Lu, Y., Florêncio, D.A.F., Zhang, C., Li, Z., Wei, F.: Trocr: Transformer-based optical character recognition with pre-trained models. ArXiv (2021) 3
16. Long, S., Ruan, J., Zhang, W., He, X., Wu, W., Yao, C.: Textsnake: A flexible representation for detecting text of arbitrary shapes. In: ECCV (2018) 2
17. Papineni, K., Roukos, S., Ward, T., Zhu, W.J.: BLEU: a method for automatic evaluation of machine translation. In: ACL. pp. 311–318 (2002) 6
18. Pramanik, S., Mujumdar, S., Patel, H.: Towards a multi-modal, multi-task learning based pre-training framework for document representation learning. ArXiv (2020) 3
19. Qin, X., Zhang, Z., Huang, C., Dehghan, M., Zaiane, O.R., Jägersand, M.: U2-net: Going deeper with nested u-structure for salient object detection. PR **106** (2020) 4
20. Ristad, E.S., Yianilos, P.N.: Learning string-edit distance. IEEE Trans. on PAMI **20**(5), 522–532 (1998) 8
21. Smith, R.: An overview of the tesseract ocr engine. In: ICDAR. vol. 2, pp. 629–633 (2007) 3
22. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: from error visibility to structural similarity. IEEE trans. on image processing **13**(4), 600–612 (2004) 8
23. Xu, Y., Xu, Y., Lv, T., Cui, L., Wei, F., Wang, G., Lu, Y., Florêncio, D.A.F., Zhang, C., Che, W., Zhang, M., Zhou, L.: LayoutLMv2: Multi-modal pre-training for visually-rich document understanding. In: Annual Meeting of the Association for Computational Linguistics (2020) 1
24. Xu, Y., Li, M., Cui, L., Huang, S., Wei, F., Zhou, M.: LayoutLM: Pre-training of text and layout for document image understanding. In: ACM SIGKDD international conference on knowledge discovery & data mining. pp. 1192–1200 (2020) 1
25. Zhou, X., Yao, C., Wen, H., Wang, Y., Zhou, S., He, W., Liang, J.: East: An efficient and accurate scene text detector. CVPR pp. 2642–2651 (2017) 2