

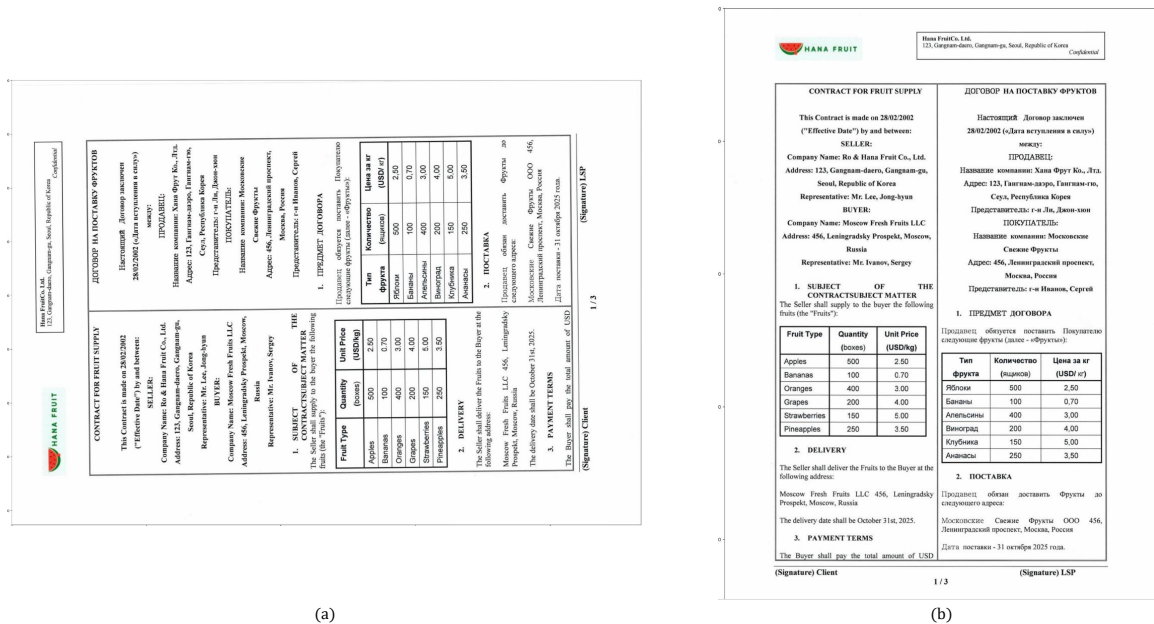
# Supplementary Material

## 1. Document change detection application whole process using TCD

In the supplementary material, we describe how our proposed method is applied to justify the necessity of our proposed model. We develop a contract document change detection (DCD) application using our proposed model, which is currently being used by the legal review team. Our application detects image pairs for comparison through several steps. The overall DCD process is as follows:

### 1.1. Preprocess

Typically, we receive the original contract document and its scanned version as inputs, where the scanned version may be rotated or skewed upon entry as illustrated in Fig. 1 (a). In addition, the aspect ratio may vary depending on the device settings. To address this issue, we adjusted the scanned version to align with the orientation and scale of the original document as depicted in Fig. 1 (b).



(a)

(b)

Figure 1. Pre-processing result of a scanned document

### 1.2. Layout detection

Contract documents often involve bilingual content, typically presented in a two-column layout. Since the documents contain different languages and layouts, we separate the layout to facilitate comparison. In our application, we classify the layout into distinct elements such as text, tables, headers, and footers, allowing for precise analysis. Furthermore, text and table layouts are divided into left, right, and center based on their position. The layout detection results for the original document and its scanned version are shown respectively in Fig. 2 as (a) and (b).



Figure 2. Layout detection results of original and adjusted scanned documents

### 1.3. Unit detection

We identify units within each layout element after detecting the layout. These units are typically detected at the word level, although variations may occur due to factors such as spacing. The result of this process is shown in Fig. 3, where (a) and (b) illustrate the results of table and text layout, respectively. Subsequently, we compare these detected units in pairs, and units that are differently detected in the original and scanned versions undergo a split-and-merge process for accurate comparison. Ultimately, this enables us to detect the final changed regions. The matching process only takes place between units belonging to the same layout element.

### 1.4. Two way segmentation

Our proposed method generates segmentation maps in both directions, *e.g.*, from the source to target and vice versa. Since the final decision is made through binary classification, we utilize the maximum values of these two segmentation maps. By analyzing the segmentation maps in both directions, we can identify areas where text has been inserted or deleted. Figure 4 illustrates the results of the proposed method using the test data samples in 4.1. Specifically, (a) and (b) represent the source (original) and target (scanned) pairs, respectively, while (c) and (d) denote the changes from the source to target segmentation map and vice versa. Furthermore, (e) displays the maximum value segmentation map of (c) and (d). In the third row, for example, since the existing characters do not change in (a), nothing is activated in (c). However, from the perspective of (b), the last character is removed, which activates the segmentation map containing the last character area in (d). In this case, we infer an insertion from source to target.

## 2. Supplementary experimental results

In Sec.4, we only conduct experiments with text image data, because its objectives diverge from those of traditional change detection approaches. However, we also conduct additional experiments, and the results are included in the supplementary

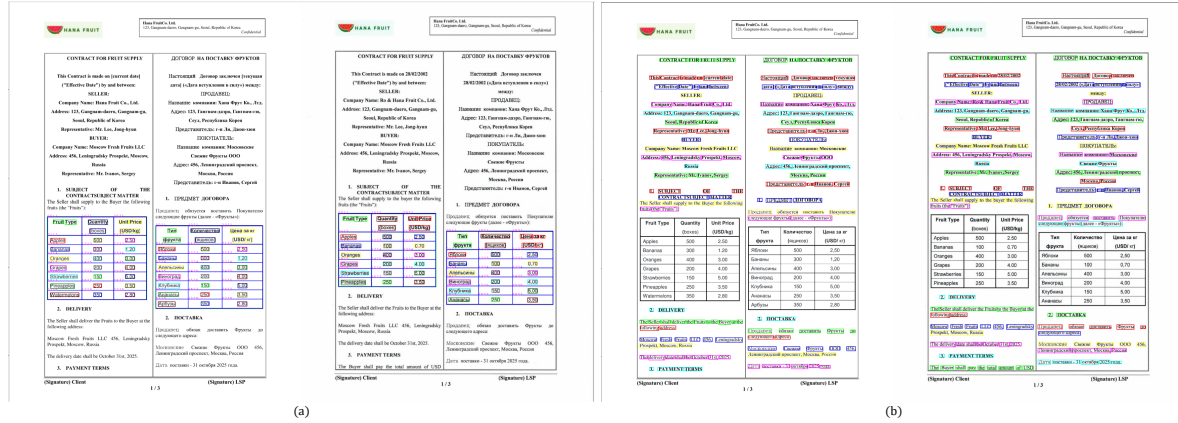


Figure 3. Unit detection results

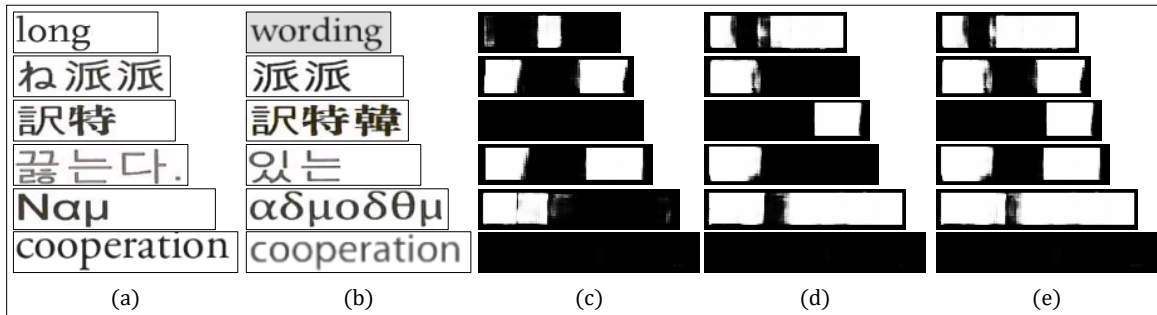


Figure 4. Two way segmentation map

materials due to page length constraints in the paper.

## 2.1. Segmentation results for Hindi and Arabic

To compare the performance of our model with SotA change detection models on text images, we generate document images in multiple languages through printing and scanning, and then create word-level image pairs using a unit detector. However, the unique linguistic properties of Arabic and Hindi render it exceedingly challenging to produce image pairs using the unit detector. Due to this limitation, we are unable to create segmentation benchmark datasets for these languages. Instead of quantitative evaluation, we utilize OCR test datasets and conduct experiments to qualitatively evaluate the segmentation results for these languages. The results are shown in the Fig. 5.

We conduct experiments comparing the performance of our proposed approach against SotA segmentation models, including BIT-CD and SARAS-net. Notably, our proposed model produces a two-way segmentation map, yielding both source to target and vice versa. To facilitate a comprehensive evaluation, we display the maximum values of these two maps in Fig. 5. The BIT-CD model exhibits sensitivity to even slight variations in character ratio and position, often misclassifying such cases as different. This limitation is evident in the results shown in Fig. 5 rows 1, 6, and 8. In contrast, SARAS-net demonstrates a converse behavior, struggling to identify discrepancies in different pairs, as observed in Fig. 5 rows 4, 5, and 9. Notably, our proposed model demonstrates robustness to character ratio variations and minor truncations, accurately identifying modified regions. As evidenced by Fig. 5, only our model successfully detects changes in rows 2 and 7. Moreover, across most cases, our model excels at identifying modified regions through its two-way analysis, providing the most comprehensive detection of modified regions from both source and target image perspectives.

## 2.2. Segmentation results for conventional change detection dataset

Our proposed model also uses change detection methods. Since the main target of change detection is in the field of remote sensing, we also evaluate its performance on the publicly available remote sensing benchmark dataset. We conduct

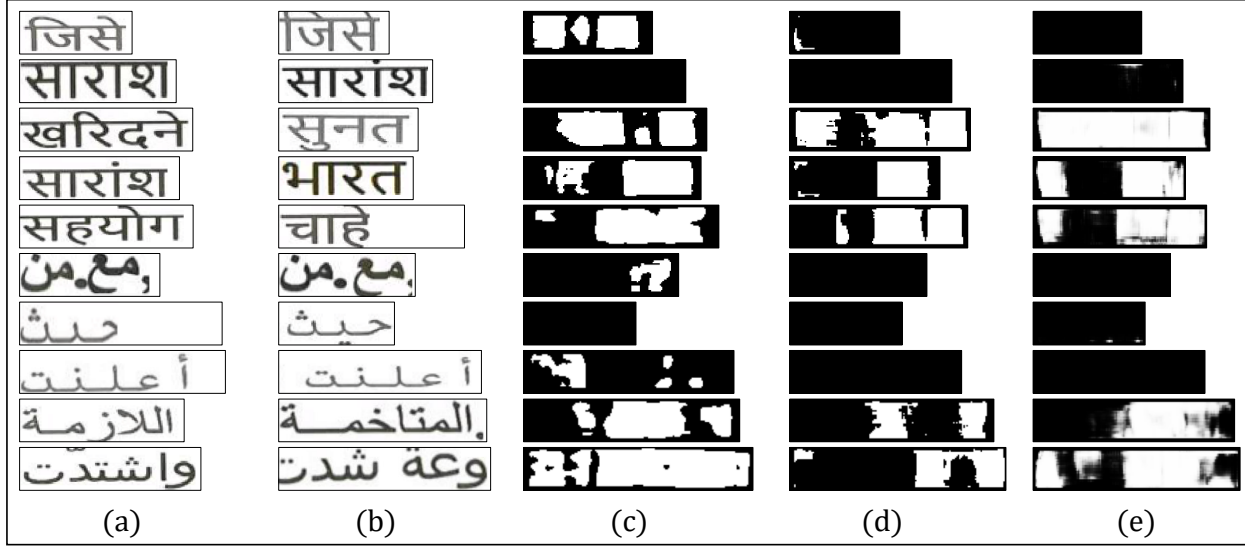


Figure 5. Qualitative comparison of our approach with state-of-the-art methods on Arabic and Hindi dataset: (a) source image, (b) target image, (c) BIT-CD, (d) SARAS-net, and (e) TCD(ours)

Model	Pre.	Rec.	F1	IoU	OA
BIT-CD	74.8	71.9	73.3	57.9	97.3
SARAS-Net	<b>90.8</b>	62.9	74.3	59.1	<b>97.5</b>
<b>TCD(ours)</b>	76.4	<b>77.6</b>	<b>77.0</b>	<b>62.6</b>	96.5

Table 1. Quantitative results of the segmentation benchmark on the LEVIR-CD dataset.

comparative experiments using the LEVIR-CD dataset, a public remote sensing change detection dataset. We train BIT-CD, SARAS-Net, and our TCD model on the LEVIR-CD train set from scratch for 200 epochs. The test results are presented in the Tab. 1 and Fig. 6. Experimental results show that BIT-CD and SARAS-Net produce fewer false positives, but they fail to detect many actual changes. In contrast, although this results in relatively more false positives, our model detected most of the changes and demonstrated the highest performance in F1 score, IoU, and OA. This is because our primary goal is to detect infrequently changed content within a large volume of text in the contract, leading us to prioritize improving recall, even at the expense of increased false positives.

### 2.3. Processing time for real contract documents

We develop a DCD application based on our proposed TCD model to compare draft and scanned-signed versions of contract documents. To test the application, we create a test set consisting of real-world contracts. Our dataset consists of 344 contracts, comprising 4044 pages of original documents and 4231 pages of scanned documents. Of these, 72 contracts are bilingual, while the remaining are monolingual. The contracts feature texts in multiple languages, including English, Vietnamese, Turkish, Hindi, Korean, Russian, Chinese, Latvian, Arabic, and Spanish. Due to data confidentiality issues, we can only report numerical results without sharing the actual experimental samples. We evaluate the performance of our TCD model on an NVIDIA RTX 2080 Ti, which utilize 2.7GB of memory and achieve an average processing time of 2.82 seconds per page. This result includes the entire processing pipeline, from inputting the two documents to generating output after completing preprocessing, layout detection, unit detection, and the proposed TCD model. We evaluate our approach using the entire dataset, and some of the evaluation results along with the average value of the whole data are presented in Tab. 2. Finally, we created a demo document in a mock contract format, printed and scanned it to simulate a real-world contracting process, and generated a pair of draft and scanned-signed documents. We provide a demo video showcasing the DCD application’s functionality using this demo data.

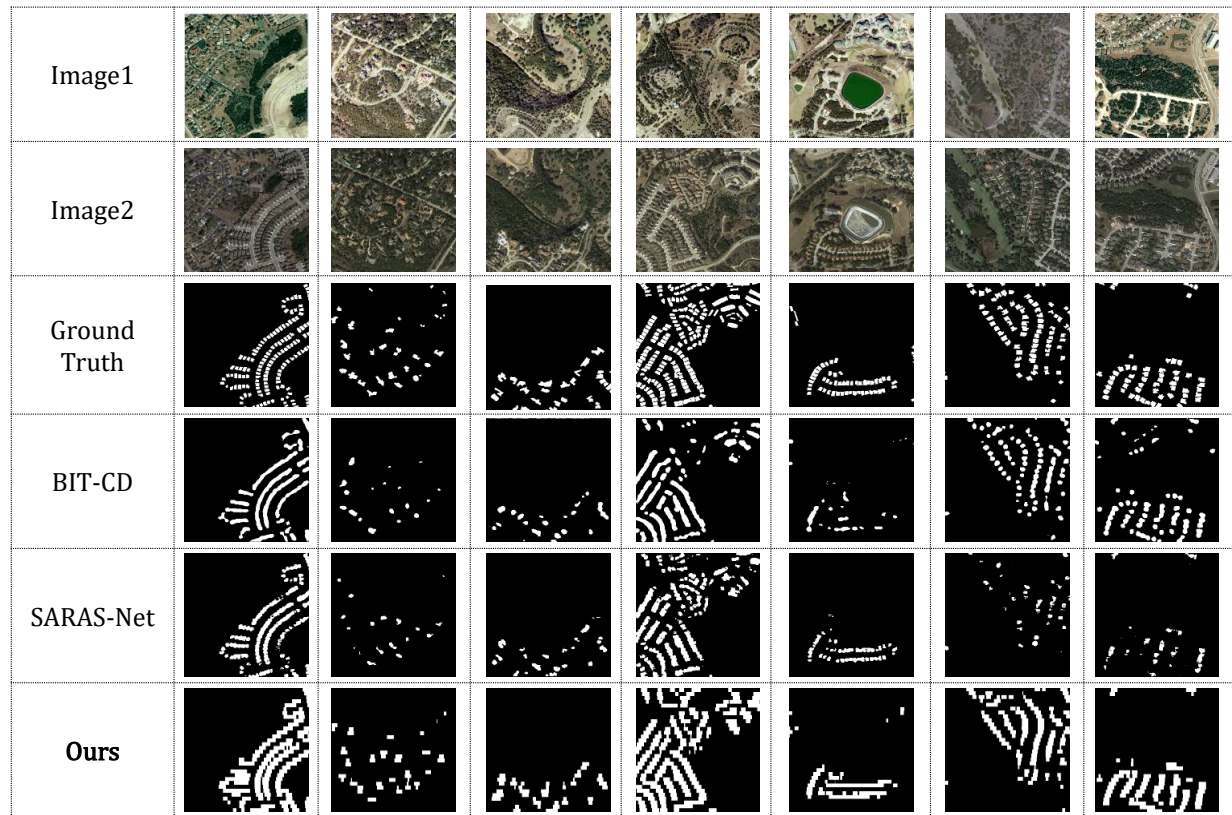


Figure 6. Qualitative results of different segmentation models on the LEVIR-CD dataset.

No.	language	page(org)	page(scan)	time(sec)	time/page(sec)
1	English,Spanish	73	73	303.76	2.08
2	English,Hindi	8	8	47.81	2.99
3	English,Turkish	4	4	19.26	2.41
4	English,Latvian	10	15	60.03	2.40
5	English,Russian	15	16	77.66	2.51
6	Korean	5	5	16.54	1.65
7	English	14	14	68.81	2.46
8	Spanish	11	13	46.10	1.92
9	Arabic	5	6	23.20	2.11
10	Chinese	42	42	129.933	1.55
Total(344)	All	4044	4231	25008.04	<b>2.82</b>

Table 2. Quantitative results of document comparison time performance.