# SUPPLEMENTARY MATERIAL: LEARNING SURFACE PARAMETERIZATION FOR DOCUMENT IMAGE UNWARPING

**Anonymous authors**
Paper under double-blind review

## OVERVIEW

In this supplementary material we include the following sections:

1. Usefulness of $L_{uv}$
2. Details of weighting function used in $L_z$
3. Training details of the UV prior network
4. Unwarping and texture editing details
5. Pre-processing details for the real scenes
6. Detailed ablation figure
7. More qualitative comparison with DewarpNet's (Das et al., 2019) best unwarped view
8. Qualitative comparison with Das et al. (2019) for different types of real documents
9. Qualitative comparison with You et al. (2017) on their test-set
10. OCR Evaluation
11. High-resolution results for texture editing
12. Limitations
13. Example of a failure case
14. Training time

## 1 USEFULNESS OF $L_{uv}$

In section 3.3 of the main submission, we define $L_{uv}$ (Eq. 10) to prevent non-uniform mapping between the 3D and the UV domain. Specifically, we constrain the output of $F_{uv}$ to be $\sim \mathcal{U}(0, 1)$ using $L_{uv}$. Without $L_{uv}$, $F_{uv}$ is prone to produce a mapping $\sim \mathcal{U}(a, b)$ where $a > 0$ or $b < 1$. Consequently, $F_z$ also learns an incorrect mapping between the texture and the 3D domain. As a result, the unwarped texture gets stretched or squeezed. We demonstrate two such examples in Fig. 1.

## 2 DETAILS OF WEIGHTING FUNCTION USED IN $L_z$

We define $L_z$ in Eq. 11 of the main submission:

$$L_z = \frac{1}{|P_{in}|} \sum_{p \in P_{in}} w_p (\hat{z}_p - \hat{z}'_p)^2 \tag{1}$$

where $P \in P_i n$ are the pixels for which ray-surface intersection is found and $M_p = 1$. $M_p$, denote the pixel in the document mask $M$. $M$ is a binary image, where $M_p = 1$ denotes the pixel $p$ is within the document region. $w_p$ is a pre-calculated per-pixel weight based on the document mask ($M$). $\hat{z}_p$ is the ray-surface intersection point obtained by sphere tracing, and $\hat{z}'_p$ is the ray-surface intersection point predicted by $F_z$. To derive the 2D texture map of a 3D surface, constraint optimization-based


Figure 1: Usefulness of $L_{uv}$: Examples trained **without** $L_{uv}$ show undesired stretches and squeezes in the unwarped texture.

techniques use user-defined keypoints (Tzur & Tal, 2009). The keypoints allow to constrain the 2D to 3D mapping estimation. For documents, we can consider the set of boundary points as the keypoints. From the application perspective, it helps to accurately map the texture boundary to the learned surface boundary (see Fig. 3(d) vs. (e) vs. (f)). Therefore, we employ a weighting function, which assigns a higher weight to the 3D surface points at the boundary. To implement $W(p)$ we use a Euclidean distance transform (Borgefors, 1986) on the document mask $M$, a binary image. Each pixel $p$, in the distance transformed image, $D$ encodes the distance to the nearest non-zero pixel. We first normalize and invert the distance transformed image:

$$D^{norm} = \frac{D - \min(D)}{\max(D) - \min(D)}$$
$$D^{inv} = 1 - D^{norm}$$

Here $\max(.)$ and $\min(.)$ denote the maximum and minimum value of $D_p$ over all the pixels. We assign the weights $w_p$ as follows:

$$w_p = \begin{cases} 10.0, & \text{if } D_p^{inv} > 0.8 \\ 0.3, & \text{otherwise} \end{cases} \tag{2}$$

## 3 TRAINING DETAILS OF THE UV PRIOR NETWORK ($\hat{F}_{uv}$)

Following L223 in the main paper, we provide more training details. We use an 8 layer MLP with a hidden layer of 512 units to learn the 3D to UV mapping prior for document shapes. Each hidden layer has a sine (Sitzmann et al., 2020) activation function. The final layer uses a HardTanh activation function. To train $\hat{F}_{uv}$ we utilize 10K UV mapped document meshes available in the Doc3D dataset. Each mesh is first registered with a $[-1, 1]$ uniform grid using a rigid transformation. Then the meshes are rendered in Blender (ble) to obtain the projected geometry image ($G$) and the UV image ($U$). In $G$, each pixel $p$ encodes the (X,Y,Z) coordinates. In $U$, $p$ encodes the corresponding UV coordinates. During training, we randomly sample 10K pixels from each $G$ as input to $\hat{F}_{uv}$ and use the corresponding pixels in $U$ as the ground-truth. We optimize the L1 loss for 150 epochs between the predicted and the ground-truth UV coordinates using the Adam optimizer with an initial learning rate of $10^{-5}$. The learning rate is halved every 50 epochs. Following NeRF (Mildenhall et al., 2020), we use a high dimensional Fourier mapping ($\chi_k : \mathbb{R} \rightarrow \mathbb{R}^{2k}$) to learn high-frequency details in the shape and the UV space. We empirically set the number of Fourier bands, $k = 10$.

## 4 UNWARPING AND TEXTURE EDITING DETAILS

To unwarp an input image, we determine a pixel at $p = (x, y)$ in the input image should be projected to $(u, v)$ in the unwarped image. Here the unwarped image refers to the texture space. The coordinates $(u, v)$ and $p$ are associated by $F_{\mathbf{z}}$ and $\tau$: For a $(u, v)$ coordinate, its corresponding point in 3D is obtained by $\hat{z}'_p = F_{\mathbf{z}}(u, v)$. Given the camera parameter $\tau$, $\hat{z}'_p$ is projected to $p$ in the input image. Thus, we can find its corresponding pixel in the input image for each pixel in the unwarped image, which is all we need for unwarping. More specifically, we use standard image projection and bilinear sampling (Jaderberg et al., 2015) to implement the unwarping step (see Fig. 2). The unwarping process can be realized as a grid sampling step from the warped document image to a 2D rectangular uniform grid. We can perform this sampling operation with a grid $G \in \mathbb{R}^{(H \times W \times 2)}$ and a bi-linear sampler. Here $H$ and $W$ denote the height and the width of the grid. Each location in $G$ encodes a pixel coordinate $\hat{p}$ of the input image.

At test time we sample $F_{\mathbf{z}}$ in a uniform grid and project using the known camera pose ($\tau$) to obtain the pixel coordinates. More specifically, sampling $F_{\mathbf{z}}$ in a uniform grid $\in [0, 1]$ yields a uniform 2D grid $R_z \in \mathbb{R}^{(H \times W \times 3)}$. Each $(u, v)$ in $R_z$ encodes a 3D coordinate of the document surface. The $R_z$ representation of the 3D shape is analogous to geometry images (Gu et al., 2002). We obtain $G$ from $R_z$ with a standard projection:

$$\hat{p} = K\,[R|T]\,\bar{\mathbf{z}} \tag{3}$$

Here, $\bar{\mathbf{z}}$ is the homogeneous coordinate representation of $\mathbf{z}$. $K \in \mathbb{R}^{3\times3}$, $[R|T] \in \mathbb{R}^{4\times4}$, denote the intrinsic and extrinsic parameters of the camera.

For the texture editing task, we first unwarp the image, then edit the texture, and finally warp each edited pixel $p$ back to the original position using the predicted texture coordinates ($t_p$). We can utilize the same bilinear sampling operation as the unwarping step.

## 5 PRE-PROCESSING DETAILS FOR THE REAL SCENES

To train our proposed approach on the real scenes, we first obtain the camera poses using COLMAP (Schönberger et al., 2016). Each scene in the real data (You et al., 2017) has 5-10 views. We pre-process the camera poses to a spherical domain following (Mildenhall et al., 2019). Since all the training meshes used to train $\hat{F}_{uv}$ are aligned with a $[-1, 1]$ uniform grid, we apply a fixed pre-computed rigid-transformation on the estimated 3D shape during the joint training of $S$, $F_{uv}$, and

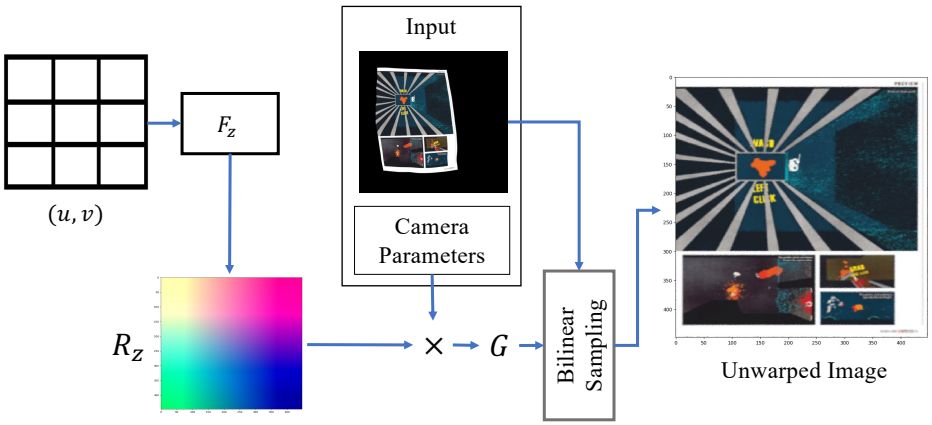



Figure 2: Unwarping steps at test time: $R_z$ denotes the flattened geometry in the texture space. Using the camera projection matrix for each view, we can obtain the unwarping grid $G$. $\times$ denotes matrix multiplication. $G$ can be used to sample (Jaderberg et al., 2015) the input image to get the unwarped image.

$F_z$. Specifically, we use a $6D$ rigid transformation, with two parameters for rotation (axis-angle representation), three for translation, and one for scale. We first train a vanilla IDR (Yariv et al., 2020) for 1000 epochs. Then we obtain a 3D point-cloud representation of the surface by sphere-tracing the IDR estimated SDF. Each point in the point cloud is a ray-surface intersection point. Note that we do not need a very accurate geometry at this step. Therefore it is not required to optimize the SDF until convergence. Now we obtain the desired rigid transformation by optimizing the Chamfer distance between the obtained surface point cloud and 10K points sampled from a 2D uniform regular grid $\in [-1, 1]$. We use SGD (Sutskever et al., 2013) with a learning rate of $0.001$ and momentum $0.9$ and optimize for 10K iterations. Later, At every iteration during the joint training, we apply the estimated rigid transformation on the sphere traced surface points ($\hat{z}_p$) and use the transformed points as an input to the $F_{uv}$.

## 6 Detailed ablation figure

We show a more detailed example of Fig. 6 of the main submission in Fig. 3, with zoomed-in regions to demonstrate the effect of the different components of $L_T$ (Eq. 12 in the main paper).

## 7 More qualitative comparison with DewarpNet's (Das et al., 2019) best unwarped view

In Fig. 4 we report the percentage of views that yield better quantitative results than the best result of DewarpNet (Das et al., 2019). We found that our approach yields better quantitative scores in $\sim 91\%$ of the views across 10 scenes. For a better illustrative comparison we show qualitative results of the 4 best (lowest LD) unwarped views using Das et al. (2019) in Fig.8, and 9. We choose scene 3 and 6 for Fig. 8, and 9. These scenes have a comparatively larger number of views with better LD scores than the proposed approach's average score. In Fig. 5, 6, 7, we show all the seven scenes where proposed approach yields worse average LD score compared to the best LD achieved by DewarpNet (Fig. 5 in main submission only shows six out of seven examples). Clearly in all of the cases we achieve better or comparative results. It's apparent that the quantitative metrics do not always reflect better visual quality. This discrepancy is due to the sensitivity of MSSIM and LD towards subtle global transformations such as a translation of few pixels. Furthermore, we can see that it is hard to predict which view will perform best for Das et al. (2019), and results vary significantly even if the views are reasonably frontal. Comparatively, being a multi-view method, our approach produces more consistent unwarping across all views.

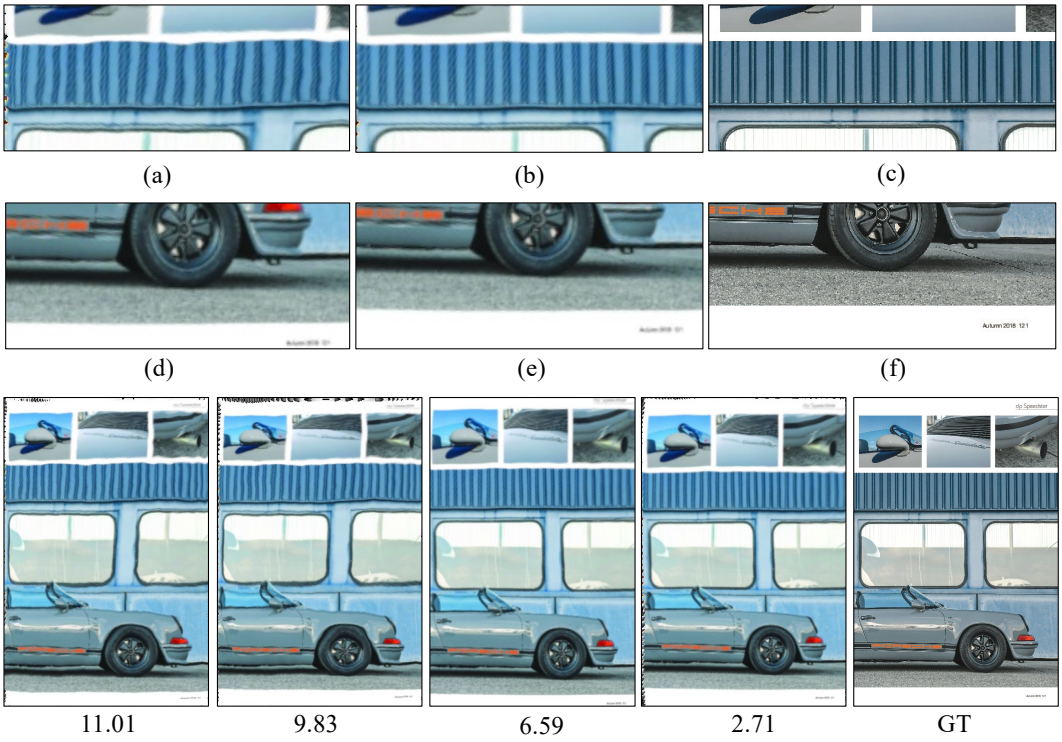



Figure 3: Weighted $L_z$, and conformality effects. Top and middle row: (a) without conformality constraints, (b) with conformality constraints, (d) $w_p = 1$ in weighted $L_z$, (e) weighted $L_z$ with $w_p$ calculated using Eq. 2, (c,f) ground-truth; bottom-row (left-to-right): without conformality constraints and weighted $L_z$; only with weighted $L_z$; only with conformality constraints; with conformality constraints and weighted $L_z$; ground-truth scan. Numbers in bottom denote the respective LD values.

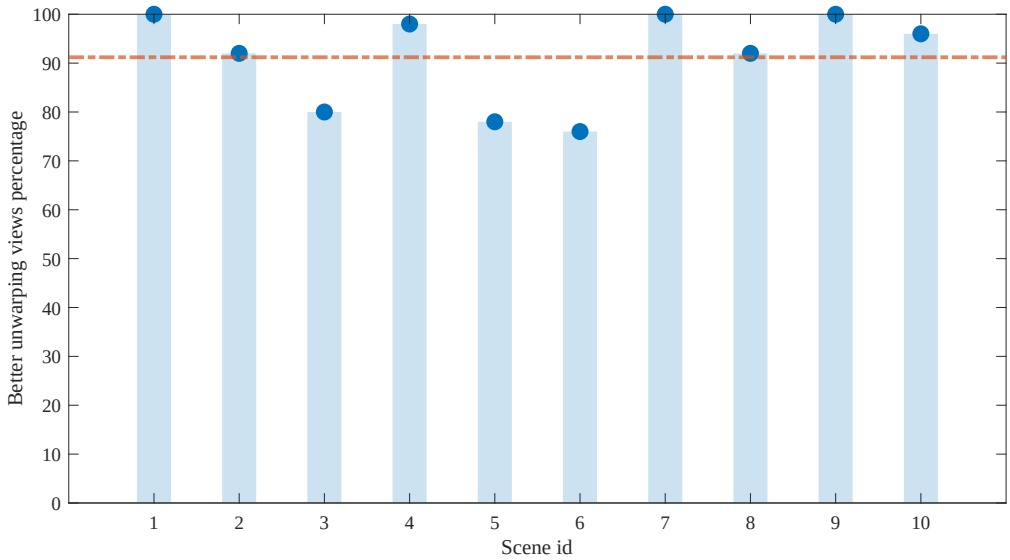
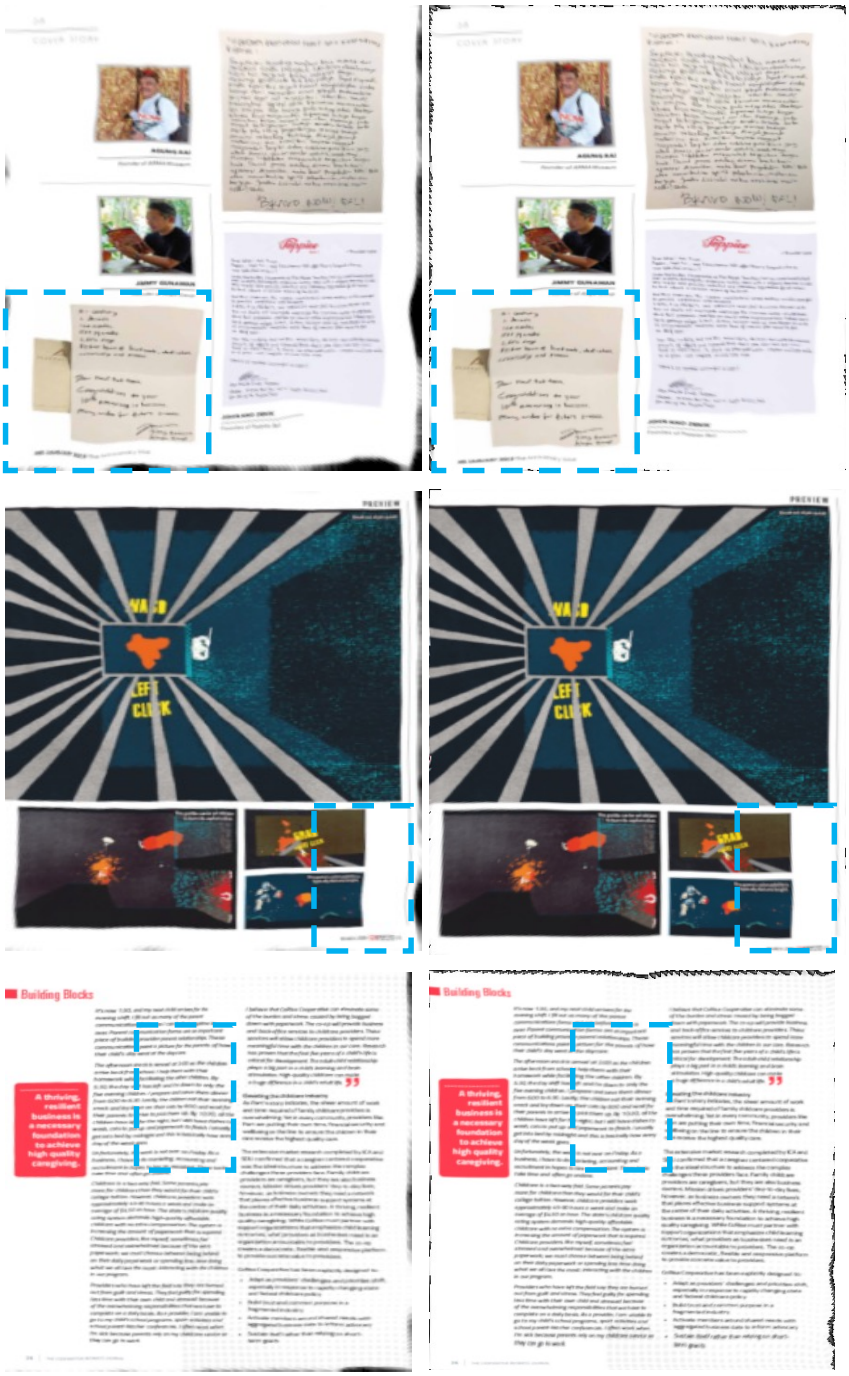



Figure 4: Percentage of unwarped views in each scene that are better than the best performing view of DewarpNet (Das et al., 2019). The number under each bar refers to the scene id presented in the same order as Table 1 in the main submission. In average (denoted by orange dashed line) 91.2% views unwarped by the proposed method produce better results than (Das et al., 2019).




(a) (b)

Figure 5: Comparison of the best view (i.e., view that yields lowest LD using DewarpNet) (a) unwarped by DewarpNet and (b) our approach. Our results are clearly better with straighter lines.

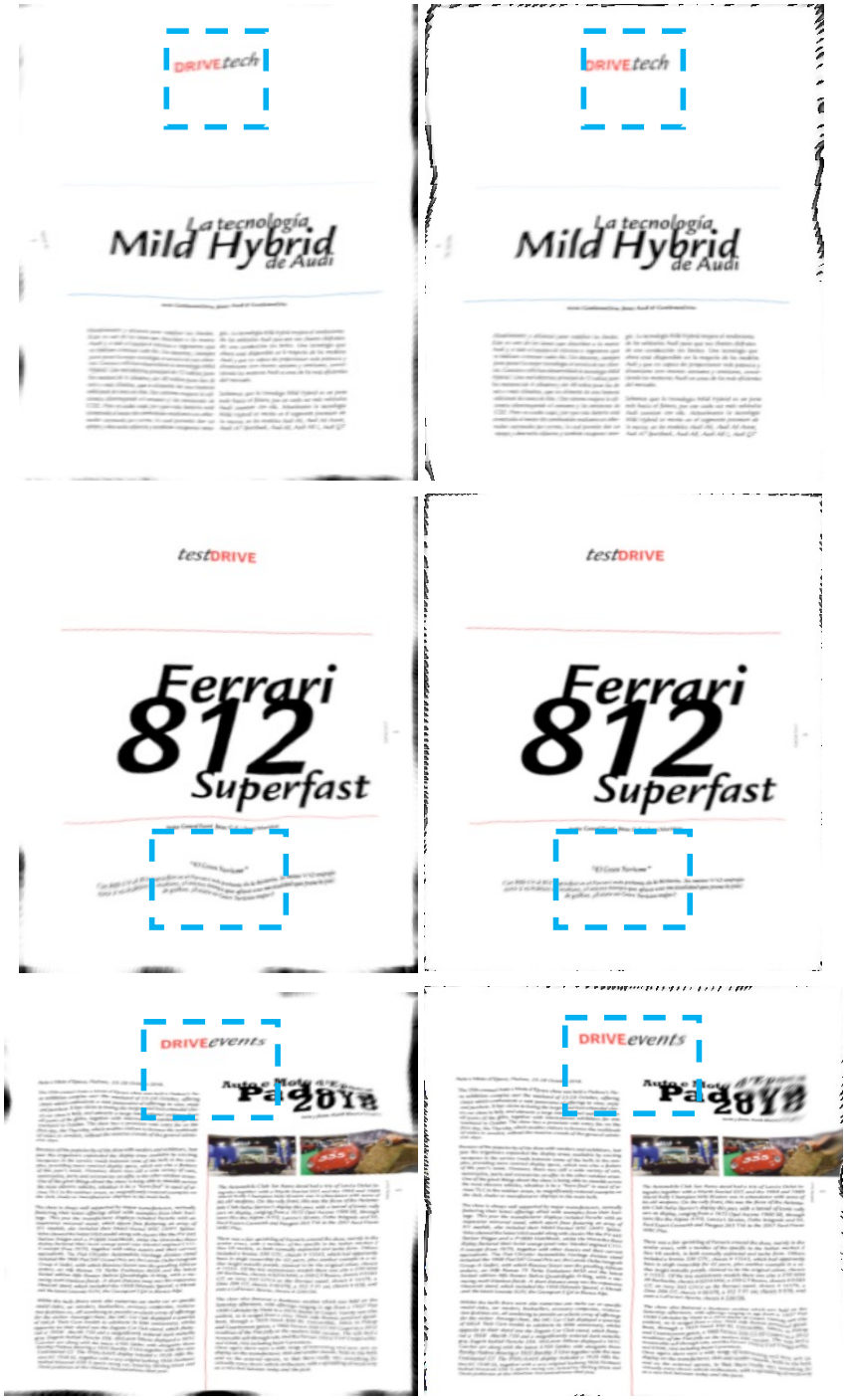
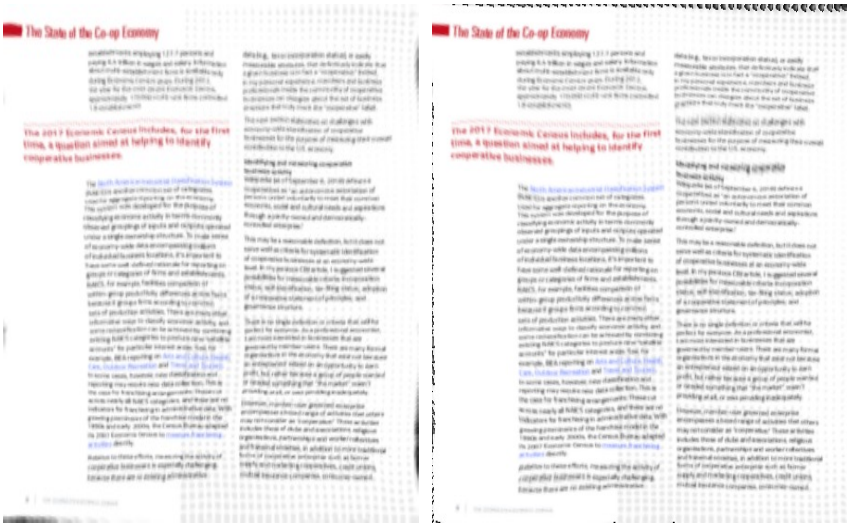



Figure 6: Comparison of the best view (i.e., view that yields lowest LD using DewarpNet) (a) unwarped by DewarpNet and (b) our approach. Our results are clearly better with straighter lines.

## 8 Qualitative comparison with Das et al. (2019) for different types of real documents

In Fig. 10, 11, 12, and 13, we show qualitative unwarping result for four different type of documents, e.g. book, receipt, flyer, and magazine. In all the views our method shows consistent and good quality unwarping results.



(a) (b)

Figure 7: Comparison of the best view (i.e., view that yields lowest LD using DewarpNet) unwarped by DewarpNet and our approach. Our results are clearly competitive with straighter lines.

| | DewarpNet | | | Proposed | | |
|---|---|---|---|---|---|---|
| | ED ↓ | CER ($std$) ↓ | WER ($std$) ↓ | ED ↓ | CER ($std$) ↓ | WER ($std$) ↓ |
| Mean | 798.30 | 0.2827 (0.12) | 0.4646 (0.17) | **600.78** | **0.2122** (0.10) | **0.3568** (0.11) |

Table 1: Comparison of OCR error metrics: We improve the OCR performance of Das et al. (2019) by ∼25% in terms of Edit Distance (ED), Character Error Rate (CER), and Word Error Rate (WER).

## 9 Qualitative comparison with You et al. (2017)

In Fig. 14, 14 we provide a qualitative comparison with 5 publicly available images from (You et al., 2017). The results are competitive and often produce better unwarping. Quantitative numbers couldn't be reported because the high-res/original unwarped results are not publicly available.

## 10 OCR Evaluation

Our OCR evaluation set contains 5 real documents with a total of 77 images. All these documents are text heavy and contains academic journals or write-ups. We use Edit Distance (ED) (Miller et al., 2009), Character Error Rate (CER) and Word Error Rate (WER) as our evaluation metrics. ED is defined as the total number of substitutions (s), insertions (i) and deletions (d) required to obtain the reference text, given the recognized text. The reference text is obtained by running the OCR algorithm on the scanned ground-truth image of each document. CER is defined as: $(s + i + d)/N$ where $N$ is the number of characters in the reference text. We use Tesseract 4.1.1 based LSTM OCR engine for this experiment.

In Table 1 we compare the proposed and the DewarpNet unwarped results in terms of OCR performance. Our unwarped results reduce the ED, CER and WER by ∼24%. This improvement proves our unwarped results are more suitable for downstream applications tasks like OCR.

## 11 High-resolution results for texture editing

In Fig. 16, 17 we show the examples of texture editing in higher resolution.


Figure 8: 4 best results (sorted in ascending order from top to bottom according to LD score [lower better]) of (b) DewarpNet compared to (c) proposed unwarping for scene 3 (Synth 3). For all the views proposed unwarping shows better and consistent visual results than DewarpNet. (a) is the input. Blue dashed boxes denote the discriminative areas in the unwarped results.

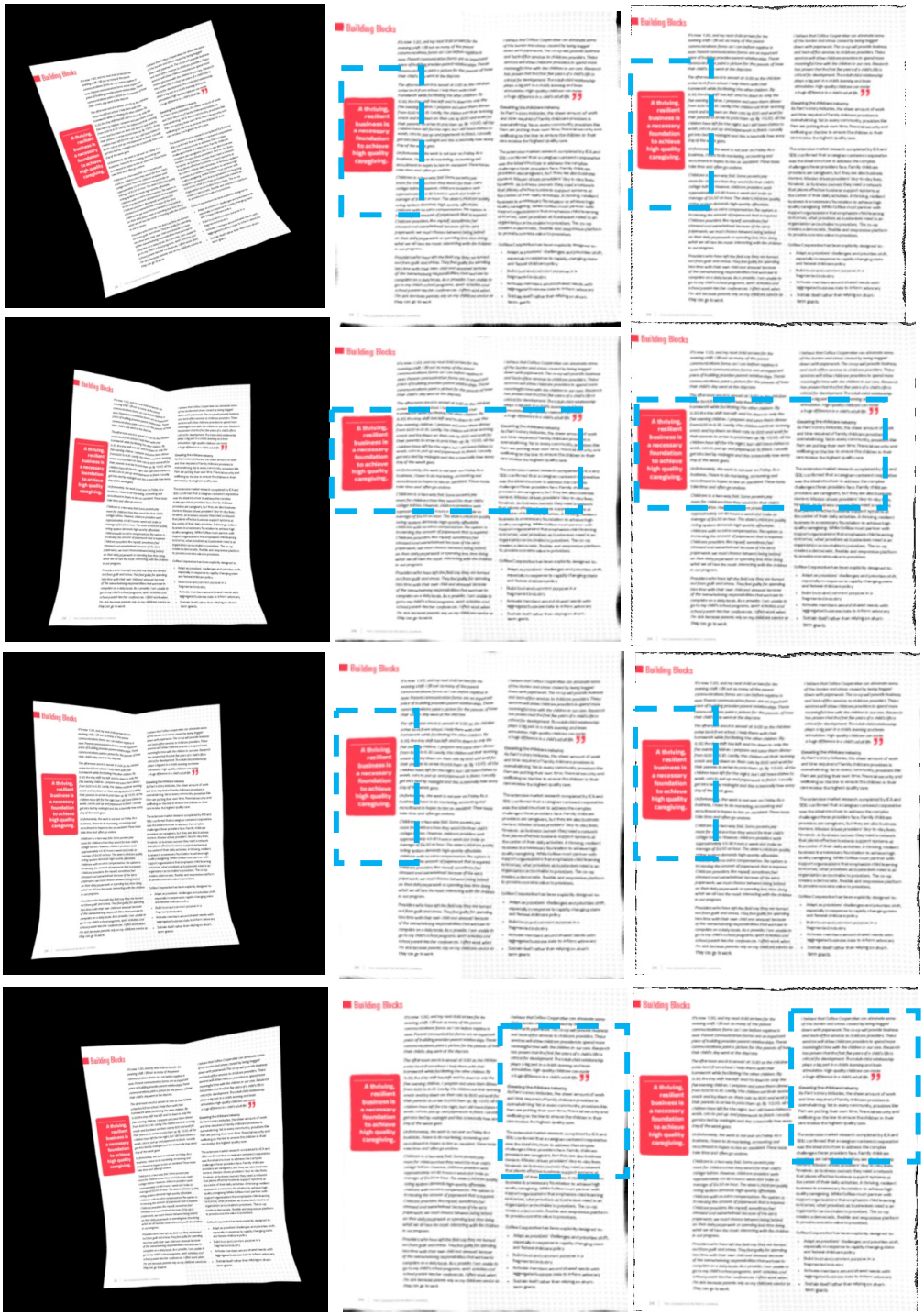



Figure 9: 4 best results (sorted in ascending order from top to bottom according to LD score [lower better]) of (b) DewarpNet compared to (c) proposed unwarping for scene 6 (Synth 6). In all the views proposed unwarping shows better and consistent visual results than DewarpNet. (a) is the input. Blue dashed boxes denote the discriminative areas in the unwarped results.

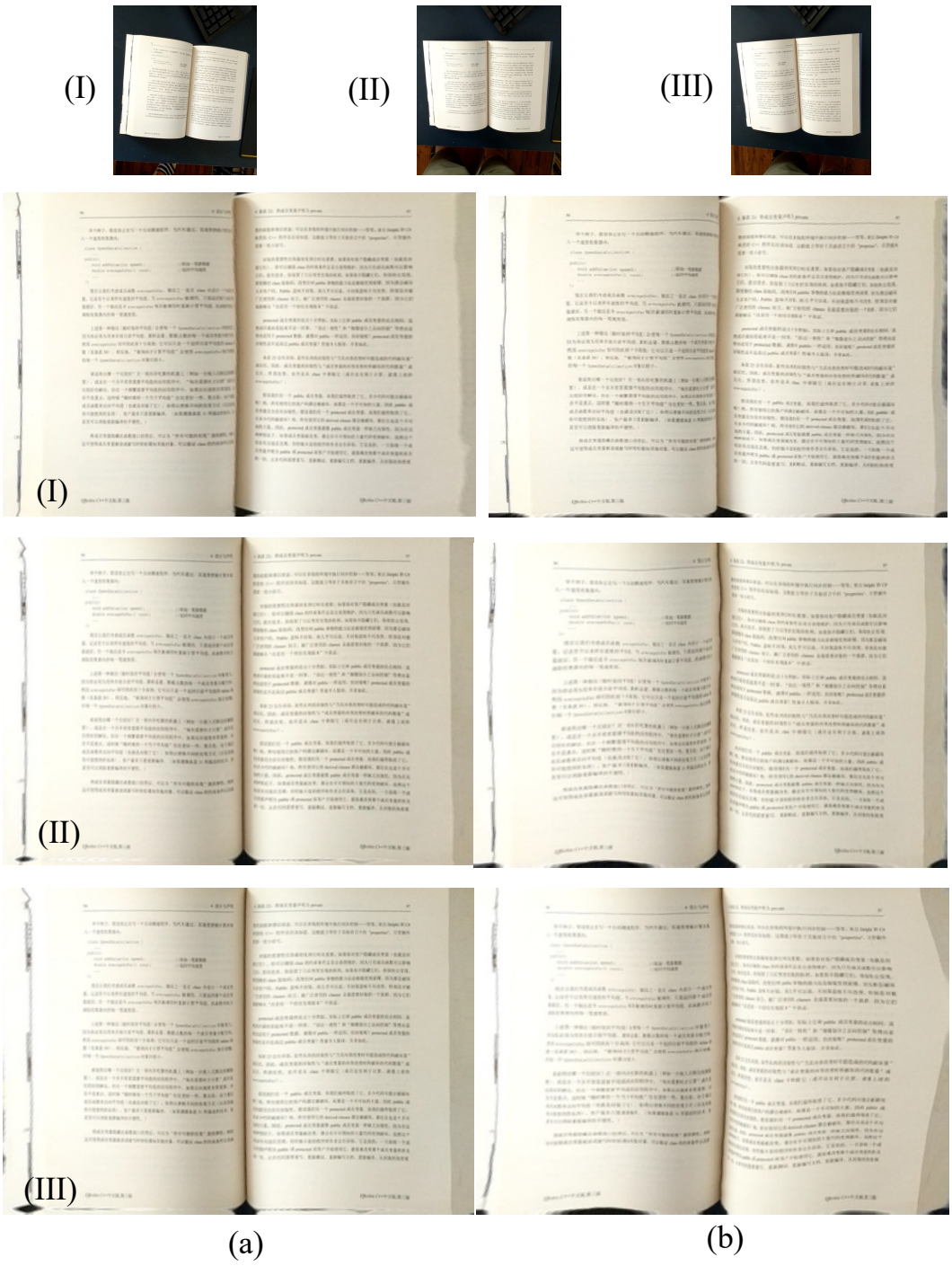



Figure 10: Unwarping results on different views of a book. Top row shows the inputs. (a) Proposed, (b) DewarpNet. Our method generates good quality unwarping results with straighter text-lines.


Figure 11: Unwarping results on different views of a receipt. Top row shows the inputs. (a) Proposed, (b) DewarpNet. Our method generates good quality unwarping results with straighter textlines.

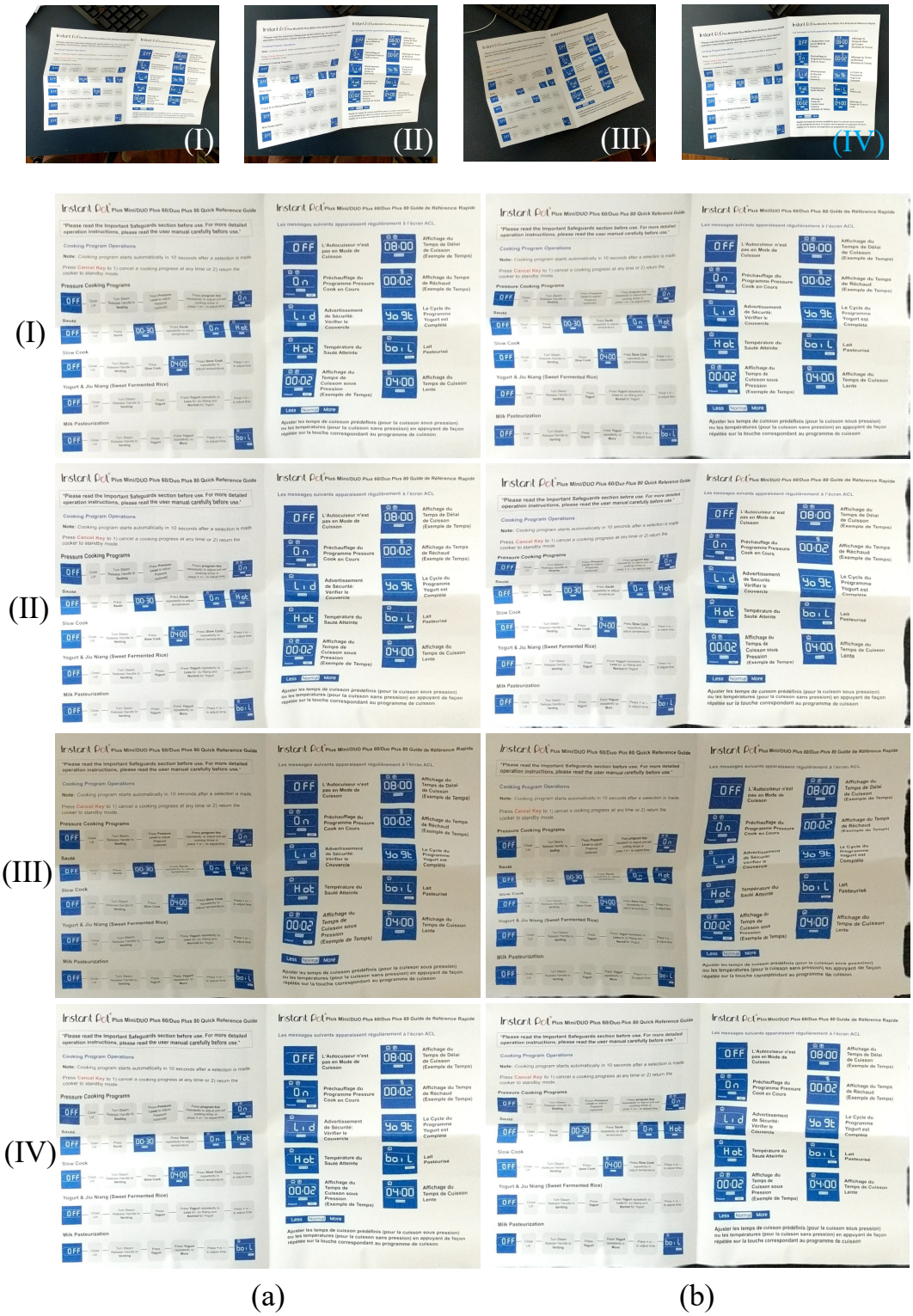



Figure 12: Unwarping results on different views of a flyer. Top row shows the inputs. (a) Proposed, (b) DewarpNet. Our method generates good quality unwarping results with straighter text-lines.

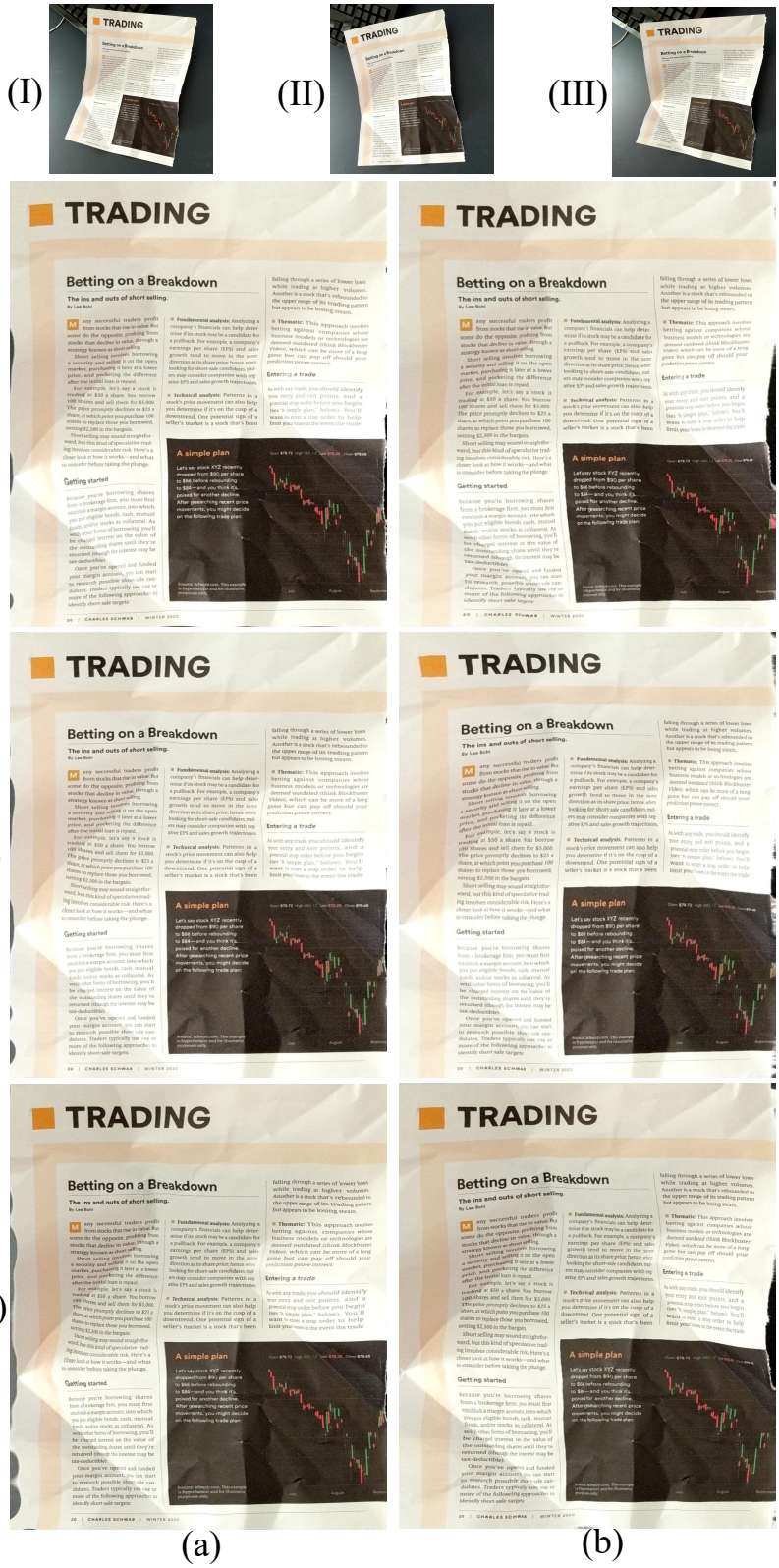



Figure 13: Unwarping results on different views of a magazine page. Top row shows the inputs. (a) Proposed, (b) DewarpNet. Our method generates good quality unwarping results with straighter text-lines.

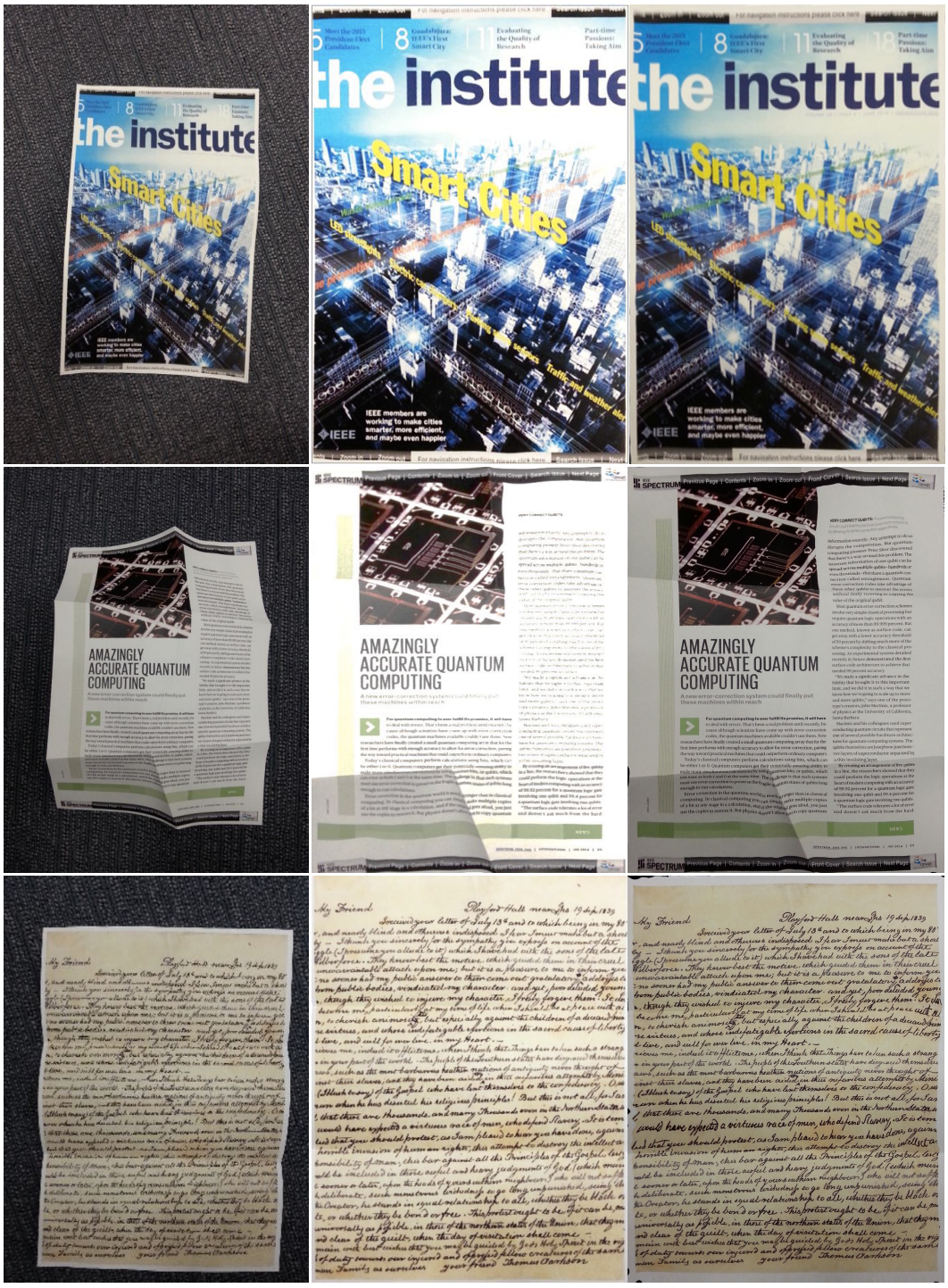
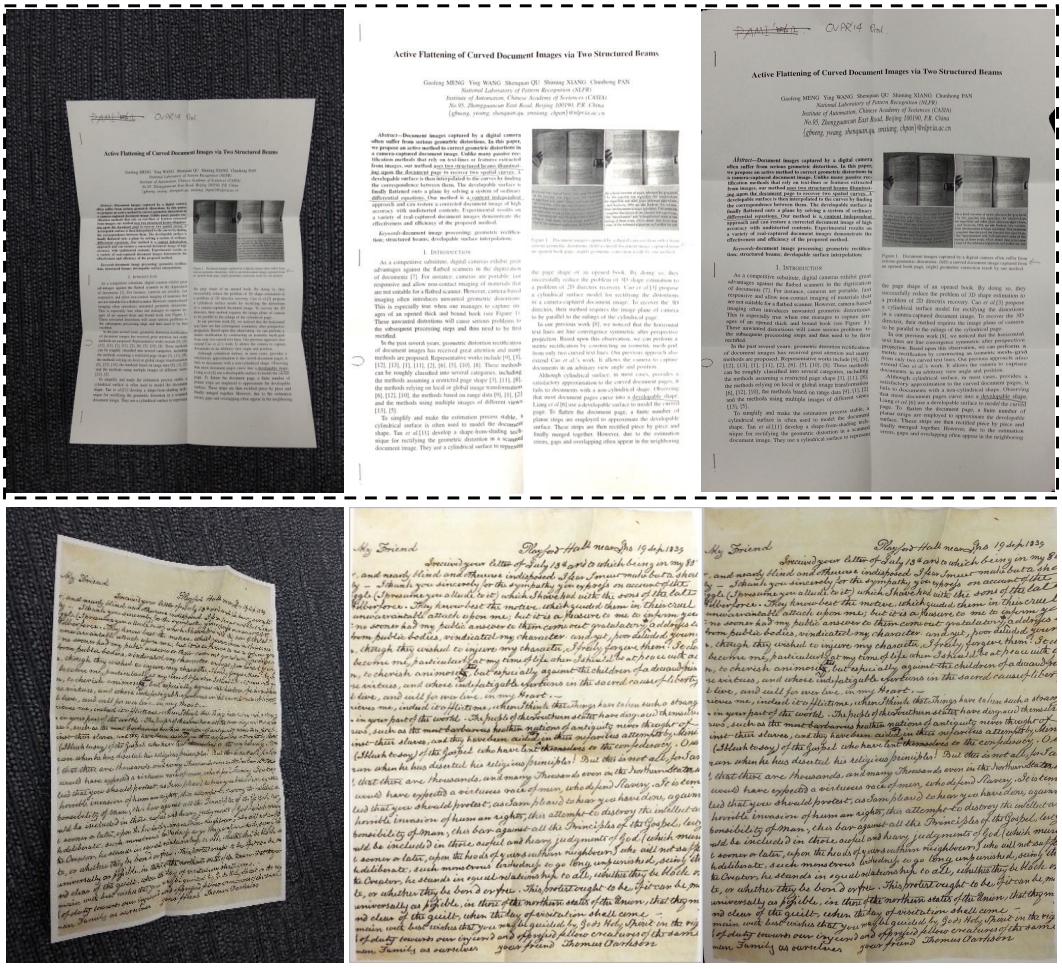



Figure 14: Comparison with You et al. (2017): We show competitive unwarping results compared to a prior multi-view unwarping approach. A quantitative comparison could not be performed because high-res/original unwarped results are not publicly available.



Input You et al. Proposed

Figure 15: Comparison with You et al. (2017): We show competitive unwarping results compared to a prior multi-view unwarping approach. A quantitative comparison could not be performed because high-res/original unwarped results are not publicly available. The example with the dashed outline shows a failure case of our method: 'Data 6'(see figure 18).

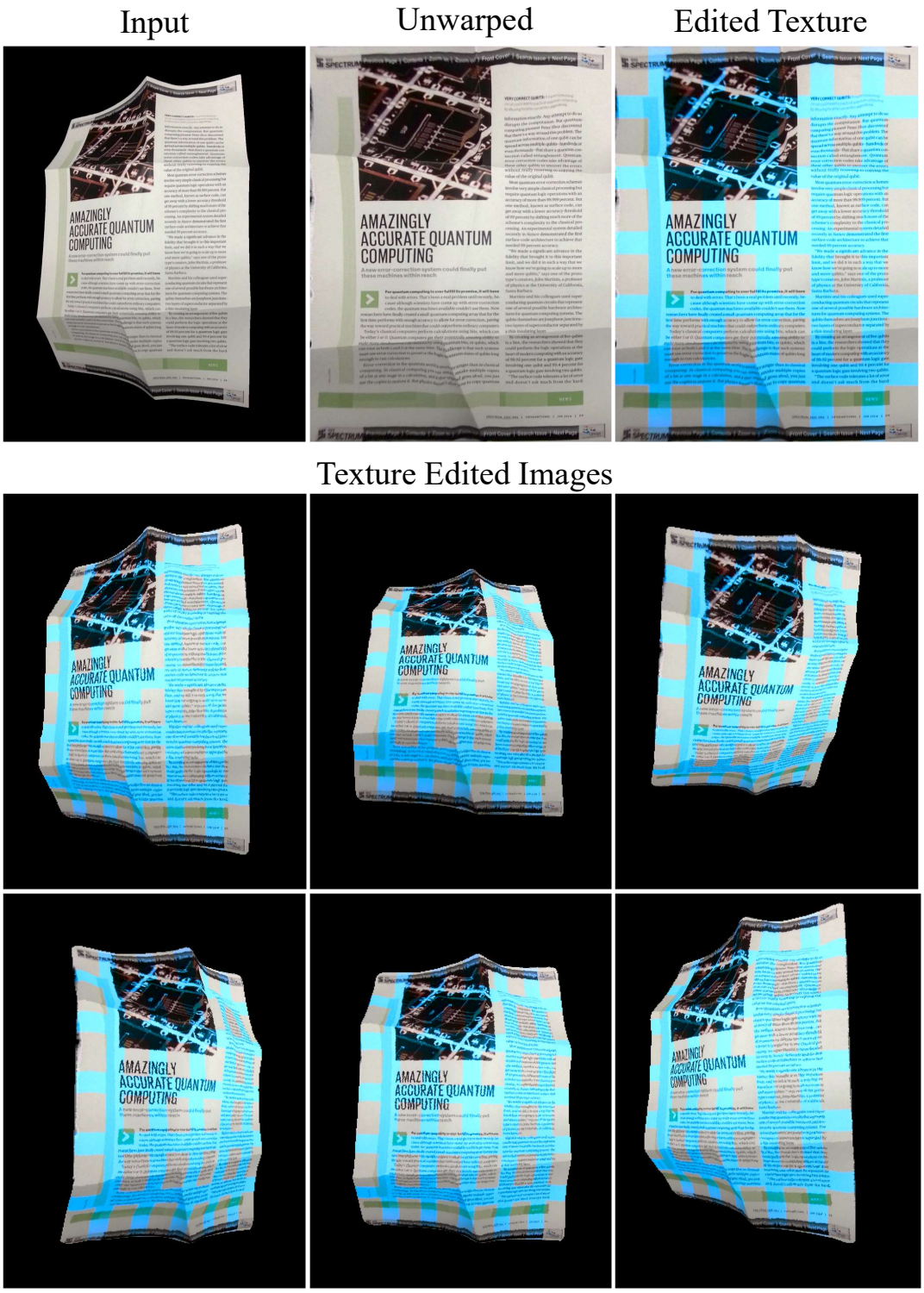



Figure 16: Example of texture edited images from different views. Note the perspective changes and deformation on the edited texture due to the complex shape of the paper.


Figure 17: Example of texture edited images from different views. Note the perspective changes and deformation on the edited texture due to the complex shape of the paper.

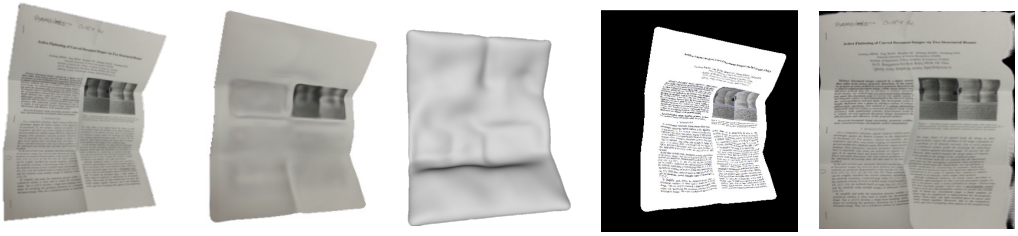



Figure 18: Shows a failure case of our method due to inferior 3D reconstruction. This happened due to fewer available views for the scene and insufficient texture.

## 12 LIMITATIONS

In the following, we discuss few potential limitations of our method:

- **3D reconstruction:** The main limitation of our method follows from IDR (Yariv et al., 2020). Inadequate number of images of a scene with large texture-less regions lead to inferior 3D reconstruction which affects our unwarping result (see section 13).
- **Training time:** The current approach takes $\sim$18 hours to train a model and separate models must be trained for every scene which makes it unsuitable for real time applications. Runtime improvement is considered as out-of-scope of this paper and will be addressed in future (see section 14).
- **Need for masks:** We assume masks are available for every image. Although masks are currently provided as manual inputs, we believe it's fairly straightforward to train a foreground-background document segmentation model to automate the task.

## 13 EXAMPLE OF A FAILURE CASE

Our method might fail due to imperfect 3D reconstruction. We show one such case for "Data 6" (You et al., 2017) in Table 2 of the main submission. Mainly, there are two reasons for failure cases: first, fewer views (only 5), and second, insufficient textured documents. IDR has insufficient information to reconstruct the 3D shape. As a result of the poor 3D shape, our texture parameterization network produces an inferior unwarping result. For illustration, we show the reconstructed 3D shape, warped texture, and unwarped texture in Fig. 18.

## 14 TRAINING TIME

Our proposed method for a scene can be trained in approximately 18 hours for $448 \times 448$ resolution images using a single Titan Xp GPU. The current training time per scene is very high compared to DewarpNet's inference time which makes it unsuitable for real time applications. However, this is a fast growing field and there are multiple other works that are focusing on improving the speed and generalization abilities (Garbin et al., 2021; Bergman et al., 2021) of neural rendering. Obtaining a faster training scheme is out of the scope of this paper. We believe a few things can be done to improve the current system such as using a faster rendering based shape reconstruction framework which we leave as a future work.

## REFERENCES

Blender - a 3D modelling and rendering package.

Alexander W. Bergman, Petr Kellnhofer, and Gordon Wetzstein. Fast training of neural lumigraph representations using meta learning, 2021.

Gunilla Borgefors. Distance transformations in digital images. *Computer vision, graphics, and image processing*, 34(3):344–371, 1986.

Sagnik Das, Ke Ma, Zhixin Shu, Dimitris Samaras, and Roy Shilkrot. DewarpNet: Single-image document unwarping with stacked 3D and 2D regression networks. In *Proceedings of the International Conference on Computer Vision*, 2019.

Stephan J. Garbin, Marek Kowalski, Matthew Johnson, Jamie Shotton, and Julien Valentin. Fastnerf: High-fidelity neural rendering at 200fps, 2021.

Xianfeng Gu, Steven J Gortler, and Hugues Hoppe. Geometry images. *ACM Transactions on Graphics (TOG)*, 21(3):355–361, 2002.

Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. In *Advances in Neural Information Processing Systems*, 2015.

Ben Mildenhall, Pratul P. Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Transactions on Graphics (TOG)*, 2019.

Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. In *Proceedings of the European Conference on Computer Vision*, 2020.

Frederic P. Miller, Agnes F. Vandome, and John McBrewster. *Levenshtein Distance: Information Theory, Computer Science, String (Computer Science), String Metric, Damerau?Levenshtein Distance, Spell Checker, Hamming Distance*. Alpha Press, 2009. ISBN 6130216904.

Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision (ECCV)*, 2016.

Vincent Sitzmann, Julien N.P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. In *arXiv*, 2020.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pp. 1139–1147. PMLR, 2013.

Yochay Tzur and Ayellet Tal. FlexiStickers: Photogrammetric texture mapping using casual images. In *Proceedings of the ACM SIGGRAPH Conference on Computer Graphics*, 2009.

Lior Yariv, Yoni Kasten, Dror Moran, Meirav Galun, Matan Atzmon, Basri Ronen, and Yaron Lipman. Multiview neural surface reconstruction by disentangling geometry and appearance. *Advances in Neural Information Processing Systems*, 33, 2020.

Shaodi You, Yasuyuki Matsushita, Sudipta Sinha, Yusuke Bou, and Katsushi Ikeuchi. Multiview Rectification of Folded Documents. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.