Scene Text Segmentation via Inverse Rendering

Yahan Zhou, Jacqueline Feild, Erik Learned-Miller, and Rui Wang
University of Massachusetts, Amherst
Email: yhzhou@cs.umass.edu, jfeild@cs.umass.edu, elm@cs.umass.edu, ruiwang@cs.umass.edu

***Abstract*—Recognizing text in natural photographs that contain specular highlights and focal blur is a challenging problem. In this paper we describe a new text segmentation method based on *inverse rendering*, i.e. decomposing an input image into basic rendering elements. Our technique uses iterative optimization to solve the rendering parameters, including light source, material properties (e.g. diffuse/specular reflectance and shininess) as well as blur kernel size. We combine our segmentation method with a recognition component and show that by accounting for the rendering parameters, our approach achieves higher text recognition accuracy than previous work, particularly in the presence of color changes and image blur. In addition, the derived rendering parameters can be used to synthesize new text images that imitate the appearance of an existing image.**

I. Introduction

Scene text recognition, which aims to recognize text in natural scenes, remains a challenging problem. Compared to scanned documents, which are usually taken under uniform lighting with sharp focus, scene text images often contain undesirable lighting effects, such as specular highlights, glossy reflections, and shadows; and the camera's focal and motion blur add additional difficulty to the recognition task.

In this paper we describe a new algorithm for scene text segmentation by exploiting inverse rendering, i.e. decomposing an input image into basic rendering elements. This technique assigns each pixel in the image to a foreground (text) layer or a background layer. Previous work [1] has shown that accurate segmentation can significantly increase the success of the subsequent text recognition step. However, segmentation in natural photographs presents a challenge because effects such as highlights, shadows, and focal blur can severely limit the segmentation accuracy.

As our main contribution, we introduce inverse rendering into the segmentation algorithm. Given an input image, our method automatically infers parameters such as the light source, material properties (e.g. diffuse/specular reflectance and shininess) as well as a blur kernel size. By accounting for these rendering parameters and illumination effects (which are often the cause of color changes), our method can greatly improve the accuracy of segmentation, and consequently lead to improved text recognition results.

Our method solves the rendering parameters using both closed-form solutions as well as analysis via synthesis. Starting from an initial set of parameters and foreground/background labels, we run a computer graphics rendering algorithm to compute a synthesized image. We then use the error of the synthesized image with the input image to drive the updates of the parameters and assigned label. We formulate this process as an iterative optimization problem and solve it using an Expectation-Maximization (EM) algorithm. The end result decomposes the input image into a foreground and a background layer, and the associated material parameters for each pixel. Results show that our method can greatly improve text recognition in images that contain challenging effects such as specular highlights and image blur. Besides text recognition, our method can be used to synthesize new text images that mimic the appearance of an existing image.

II. Related work

Scene Text Recognition. Scene text recognition is an important research problem in computer vision. Among existing solutions, one popular approach is to use sliding windows over the whole image to extract features for text detection and recognition, such as [2], [3] and [4]. Another approach is based on Maximally Stable Extremal Regions (MSERs). For example, in [5] MSER has been used for localizing text in images. In [6], the MSER-based method is further improved by constructing an MSER tree and applying an effective pruning algorithm. Instead of using MSER, in [7] a feature calculator is integrated into the process of computing Extremal Regions (ER), which enables efficient evaluation of all ERs.

Text Segmentation. Text segmentation aims to detect and split an image into regions of text (foreground) and background. A classic approach for segmentation is to use thresholding. A survey of such approaches can be found in [8]. Traditional thresholding-based algorithms are suitable for scanned documents, which are usually taken under uniform lighting with sharp focus. Thus the pixel values naturally form two clusters. However, such an approach is difficult to apply in scene text segmentation, due to the significant color variations and illumination effects in natural images.

There are also segmentation methods based on colors. For example, [9] runs a k-means algorithm to cluster pixels into several groups. This produces several possible segmentations, from which the best is selected using an SVM algorithm. In [10], the authors extract strokes and build a color distribution for embedded text segmentation tasks. A recent work in [1] introduced an MRF-based model, where they first use a Gaussian mixture model for clustering pixel colors, then use a graph cut algorithm for solving the MRF.

Despite the success of previous work, the segmentation of images that contain significant specular highlights and focal blur still presents a great challenge. For example, highlights can change the color of a text region, making it difficult to distinguish a foreground (text) pixel from a background pixel. Using an appropriate lighting model will greatly help in such cases by accounting for illumination effects.

Physically-based Lighting Model. In computer graphics a standard lighting model is described by the integral of incident light with BRDFs (Bidirectional Reflectance Distribution Functions). The BRDF models have been well studied, ranging from the simplest Phong model [11], which gives good approximation of specular reflections, to more sophisticated models or even measured BRDFs. A detailed analysis and comparison of different types of BRDF models can be found in [12]. For text segmentation tasks, we apply the lighting model to infer the diffuse and specular reflectance as well as the shininess of the material on both foreground and background. Since input images are usually captured at low resolution with low dynamic range colors, a very precise BRDF model is unnecessary. Thus in our experiments we apply the simple Phong BRDF model [11], which has been shown to be sufficiently good for most images we have tested.

Inverse Rendering. Extracting illumination and BRDF information from images is known as inverse rendering and has been studied in previous work. While estimating BRDF from a single image in an unconstrained environment is generally intractable, researchers have proposed solutions based on multiple images, or with assumptions such as known lighting and/or 3D geometry [13] [14][15]. A survey of inverse rendering can be found in [16]. In our paper, the BRDF estimation problem is solved by making certain assumptions of scene text images, explained in Section III.

III. LIGHTING MODEL

This section explains our lighting model. Given an input image I, we simultaneously assign foreground / background (i.e. segmentation) labels for each pixel and derive the lighting parameters, such that a rendered image $\tilde{I}$ using these parameters matches the input as close as possible.

Assumptions. For the text segmentation problem, we assume the illumination comes from a single point light, text (foreground) pixels have the same material (BRDF), and similarly with the background pixels. In addition, we assume the image is taken under orthographic projection (with little perspective distortion), and pixels exist on the same planar surface (thus have the same normal). These assumptions are valid for most text image patches we tested. If necessary, the number of parameters can be increased to accomodate more complicated scenarios. Notice that we make these assumptions only for the segmentation part of the algorithm. The recognition part doesn't rely on them.

Rendering Equation. Denote a pixel $\mathbf{p}$'s screen coordinate as $\mathbf{p_s}$ and world coordinate as $\mathbf{p_w}$. The Phong BRDF model describes the observed color of pixel $\mathbf{p}$ as the sum of a diffuse and specular color:

$$\begin{aligned} \tilde{I}(\mathbf{p}) &= I_d(\mathbf{p}) + I_s(\mathbf{p}), & (1)\\ I_d(\mathbf{p}) &= \frac{C_d}{4\pi|\mathbf{S}-\mathbf{p_w}|^2}\max\left(0,\, \mathbf{L}\cdot\mathbf{N}\right), & (2)\\ I_s(\mathbf{p}) &= \frac{C_s}{4\pi|\mathbf{S}-\mathbf{p_w}|^2}\max\left(0,\, \mathbf{R}\cdot\mathbf{V}\right)^{\alpha}, & (3) \end{aligned}$$

where $L_d(\mathbf{p})$ and $L_s(\mathbf{p})$ denote the diffuse and specular components respectively, and $\cdot$ denotes vector dot product. The other symbols are listed as follows:

C_d	diffuse reflectance, determined by material
C_s	specular reflectance, determined by material
α	shininess of the material
$\mathbf{S}$	position of the light source
$\mathbf{L}$	direction from pixel to the light $\mathbf{L} = \frac{\mathbf{S}-\mathbf{p_w}}{\lvert\mathbf{S}-\mathbf{p_w}\rvert}$
$\mathbf{N}$	surface normal
$\mathbf{V}$	view direction
$\mathbf{R}$	reflected light direction $\mathbf{R} = 2(\mathbf{L}\cdot\mathbf{N}) - \mathbf{L}$

Since we assume foreground pixels have uniform material, they share the same set of parameters $C_{d,f}, C_{s,f}, \alpha_f$; similarly, background pixels share another set of parameters $C_{d,b}, C_{s,b}, \alpha_b$. In addition, since most images are taken from a direction right above the surface normal, we have $\mathbf{N} = -\mathbf{V}$. For simplicity, we set the world coordinates such as N is along the z-axis, thus $N = (0,0,1)$ and $V = (0,0,-1)$. Given the setup, equations 2 and 3 are now simplified to:

$$\begin{aligned} I_d(\mathbf{p}) &= \frac{C_d}{4\pi\left|\mathbf{S}-\mathbf{p_w}\right|^2}\max\left(0, \frac{\mathbf{S}.z-\mathbf{p_w}.z}{\left|\mathbf{S}-\mathbf{p_w}\right|}\right), & (4)\\ I_s(\mathbf{p}) &= \frac{C_s}{4\pi\left|\mathbf{S}-\mathbf{p_w}\right|^2}\max\left(0, \frac{\mathbf{S}.z-\mathbf{p_w}.z}{\left|\mathbf{S}-\mathbf{p_w}\right|}\right)^{\alpha}. & (5) \end{aligned}$$

In fact we can get rid of the $\max(\cdot)$ function in the above equations because $\mathbf{S}.z - \mathbf{p_w}.z > 0$ is always true for visible pixels. Thus rearranging the terms we have:

$$\begin{aligned} I_d(\mathbf{p}) &= \frac{\mathbf{S}.z-\mathbf{p_w}.z}{4\pi\left|\mathbf{S}-\mathbf{p_w}\right|^3}C_d, & (6)\\ I_s(\mathbf{p}) &= \frac{(\mathbf{S}.z-\mathbf{p_w}.z)^{\alpha}}{4\pi\left|\mathbf{S}-\mathbf{p_w}\right|^{\alpha+2}}C_s. & (7) \end{aligned}$$

Parameters. To summarize, in the simplified lighting model, the free parameters are the light source position $\mathbf{S}$ (relative to the depth of the text plane), the foreground/background label of each pixel, and the BRDF variables including the diffuse reflectance C_d, specular reflectance C_s and shininess α for foreground and background pixels separately. We will refer to them as the lighting parameters, denoted as θ. To simplify the equations further, we use a binary function $T(\mathbf{p})$ to represent foreground/background labels:

$$T(\mathbf{p}) = \begin{cases} 0 & \text{if } \mathbf{p} \text{ is a background pixel;}\\ 1 & \text{if } \mathbf{p} \text{ is a foreground pixel.} \end{cases}$$

And we denote the geometry terms in equations 6 and 7 as:

$$\begin{aligned} A_{d,t}(\mathbf{p}) &= \frac{\mathbf{S}.z-\mathbf{p_w}.z}{4\pi\left|\mathbf{S}-\mathbf{p_w}\right|^3}, & (8)\\ A_{s,t}(\mathbf{p}) &= \frac{(\mathbf{S}.z-\mathbf{p_w}.z)^{\alpha_t}}{4\pi\left|\mathbf{S}-\mathbf{p_w}\right|^{\alpha_t+2}}, & (9) \end{aligned}$$

where subscript t is the binary foreground/background label. This then allows us to combine the foreground and background pixels together into a single equation:

$$\begin{aligned} \tilde{I}(\mathbf{p}) = {} & T(\mathbf{p})A_{d,1}(\mathbf{p})C_{d,1} + (1-T(\mathbf{p}))\,A_{d,0}(\mathbf{p})C_{d,0}\\ & + T(\mathbf{p})A_{s,1}(\mathbf{p})C_{s,1} + (1-T(\mathbf{p}))\,A_{s,0}(\mathbf{p})C_{s,0}. \quad (10) \end{aligned}$$

This way the foreground or background lighting parameters will be automatically selected via the binary label $T(\mathbf{p})$.

Blur Estimation Accurate text segmentation needs to consider possible blur effects in images, caused by either camera's

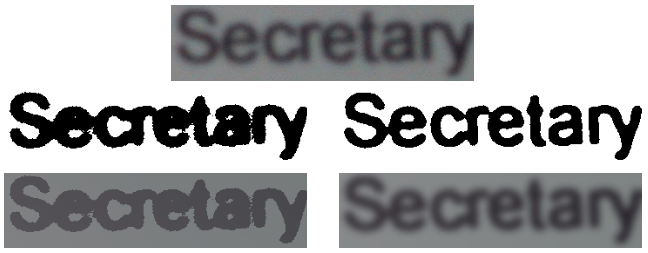

Fig. 1. Comparison of segmentation results with and without blur estimation. The top image is the input, the second row shows the segmentation results, and the third row shows the corresponding reconstructed images.

focal blur or motion blur. By incorporating blur estimation into our algorithm, we can greatly improve the segmentation results in natural images. An example is shown in Figure 1. We model the blur by convolving an image with a Gaussian kernel $\kappa(\cdot)$:

$$\tilde{I}_b(\mathbf{p}) = (\tilde{I} \otimes \kappa)(\mathbf{p}) = \int \tilde{I}(\mathbf{q})\kappa(\mathbf{p}-\mathbf{q})d\mathbf{q}. \tag{11}$$

In discrete form, it is:

$$\tilde{I}_b(\mathbf{p}) = \sum_{\mathbf{q}} \tilde{I}(\mathbf{q})\kappa(\mathbf{p}-\mathbf{q}). \tag{12}$$

Because $\tilde{I}(\mathbf{p})$ in equation 10 is expressed as the sum of four terms, we can associate the convolution with each term individually, thereby expressing $\tilde{I}_b(\mathbf{p})$ as:

$$\begin{aligned}\tilde{I}_b(\mathbf{p}) &= B_{d,1}(\mathbf{p})C_{d,1} + B_{d,0}(\mathbf{p})C_{d,0} \\ &+ B_{s,1}(\mathbf{p})C_{s,1} + B_{s,0}(\mathbf{p})C_{s,0},\end{aligned} \tag{13}$$

where $B_{d,1}$, $B_{d,0}$, $B_{s,1}$ and $B_{s,0}$ are the blurred foreground diffuse, background diffuse, foreground specular, and background specular components respectively (i.e. $B_{d,1}(\mathbf{p}) = [T(\mathbf{p})A_{d,1}(\mathbf{p})] \otimes \kappa$ and so on). Therefore, in addition to the lighting parameters and segmentation labels, we will also need to estimate the blur kernel size (refer to Section IV-C).

IV. Algorithms and Implementation

In this section, we describe an EM (Expectation Maximization) algorithm for estimating the parameters discussed in the above section. Given an input scene text image, our goal is to compute a set of parameters θ, such that a rendered image using these parameters is as close as possible to the input, measured by the L2 norm. In other words, we formulate this as an optimization problem, and the target error function is:

$$Err(\theta) = \sum_{\mathbf{p}} \left(\tilde{I}_b(\mathbf{p}) - I(\mathbf{p})\right)^2, \tag{14}$$

where $I(\mathbf{p})$ is the input and $\tilde{I}_b(\mathbf{p})$ is defined in equation 13. EM minimizes the error function iteratively by alternating between optimizing some parameters while keeping the remaining parameters fixed.

A. Solving the Lighting Parameters

We first discuss how to optimize the lighting parameters, assuming the segmentation labels and blur kernel size are fixed. To do so, we separate the lighting parameters θ into a linear set $\theta_L = [C_{d,0}, C_{s,0}, C_{d,1}, C_{s,1}]'$ and a non-linear set $\theta_N = [\mathbf{S}, \alpha_0, \alpha_1]$. These parameters are all initialized as uniform random numbers. The optimal linear parameters should each satisfy $\frac{\partial Err}{\partial \theta_L} = 0$. We demonstrate how to solve this by using the first parameter $\frac{\partial Err}{\partial C_{d,1}}$ as an example:

$$\begin{aligned}\frac{\partial Err}{\partial C_{d,1}} &= 2\sum_{\mathbf{p}} \left(\tilde{I}_b(\mathbf{p}) - I(\mathbf{p})\right) \frac{\partial \tilde{I}_b(\mathbf{p})}{\partial C_{d,1}} \\ &= 2\sum_{\mathbf{p}} \Big(B_{d,1}(\mathbf{p})C_{d,1} + B_{d,0}(\mathbf{p})C_{d,0} \\ &\quad +B_{s,1}(\mathbf{p})C_{s,1} + B_{s,0}(\mathbf{p})C_{s,0} - I(\mathbf{p})\Big) B_{d,1}(\mathbf{p}).\end{aligned}$$

Since we want $\frac{\partial Err}{\partial C_{d,1}} = 0$, we can re-write the above equation in matrix form,

$$\begin{bmatrix}(B_{d,1} \cdot B_{d,1}) \\ (B_{d,0} \cdot B_{d,1}) \\ (B_{s,1} \cdot B_{d,1}) \\ (B_{s,0} \cdot B_{d,1})\end{bmatrix}^T \begin{bmatrix}C_{d,1} \\ C_{d,0} \\ C_{s,1} \\ C_{s,0}\end{bmatrix} = (B_{d,1} \cdot I), \tag{15}$$

where $\cdot$ denotes the dot product between two images (i.e. by treating an image as a long vector).

Clearly the equations for $\frac{\partial Err}{\partial C_{d,0}}$, $\frac{\partial Err}{\partial C_{s,1}}$ and $\frac{\partial Err}{\partial C_{s,0}}$ will have a similar form. Therefore we can put them together as:

$$B^T B\, C = B^T I, \tag{16}$$

where $B^T = [B_{d,1}, B_{d,0}, B_{s,1}, B_{s,0}]$ and similarly $C^T = [C_{d,1}, C_{d,0}, C_{s,1}, C_{s,0}]$. To solve the linear parameters C, we fix the non-linear parameters, compute matrix B and multiply the psudo-inverse of $B^T B$ to both sides of equation 16.

Solving Non-linear Parameters After the linear parameters are optimized in the previous step, we will then move on to optimize the non-linear parameters θ_N. Since we do not have a closed-form solution for θ_N, we use a Levenberg-Marquardt algorithm with the objective function defined in equation 14. The details of this algorithm can be found in [17].

B. Foreground/Background Label Assignment

Now we discuss how to compute the labels assuming the other parameters are fixed. This problem can be formulated as minimizing the error function by tuning the segmentation labels $T(\mathbf{p})$. The labels are initialized by performing a k-means clustering on the pixel colors, resulting in two clusters. We begin by describing the case where there is no blurring.

Label Assignment with No Blur Suppose for now that there is no blur. In this case from Equation 10 we have:

$$\begin{aligned}\tilde{I}_b(\mathbf{p}) &= \tilde{I}(\mathbf{p}) \\ &= \begin{cases} A_{d,1}(\mathbf{p})C_{d,1} + A_{s,1}(\mathbf{p})C_{s,1} & \text{if } T(\mathbf{p}) = 1; \\ A_{d,0}(\mathbf{p})C_{d,0} + A_{s,0}(\mathbf{p})C_{s,0} & \text{if } T(\mathbf{p}) = 0. \end{cases}\end{aligned}$$

Thus $|\tilde{I}_b(\mathbf{p}) - I(\mathbf{p})|$ is minimized if for every pixel $\mathbf{p}$ we pick its label T that results in the smallest error:

$$T(\mathbf{p}) = \begin{cases} 1 & \text{if } |A_{d,1}(\mathbf{p})C_{d,1} + A_{s,1}(\mathbf{p})C_{s,1} - I(\mathbf{p})| \\ & \quad < |A_{d,0}(\mathbf{p})C_{d,0} + A_{s,0}(\mathbf{p})C_{s,0} - I(\mathbf{p})|; \\ 0 & \quad otherwise. \end{cases}$$

Label Assignment with Blur Now consider the case with blur. The above solution cannot apply directly, because in general $\tilde{I}_b(\mathbf{p}) \neq \tilde{I}(\mathbf{p})$. However, we can replace the input

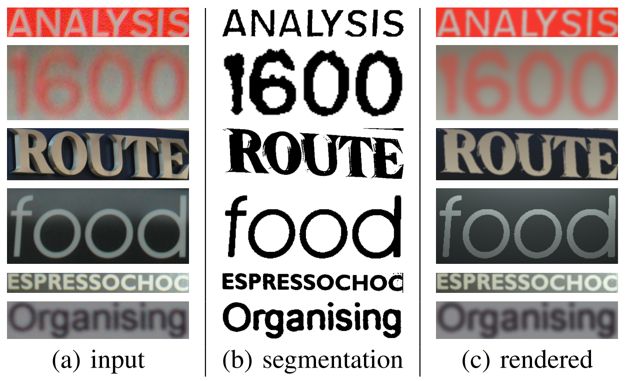

Fig. 2. Text segmentation results. (a) input images; (b) segmentation results; (c) reconstructed (i.e. rendered) images using extracted parameters.

image I with a deblurred version I_{deblur}, and run the same label assignment process as above. In other words, instead of minimizing $|\tilde{I}_b(\mathbf{p}) - I(\mathbf{p})|$ we are now trying to minimize $|\tilde{I}(\mathbf{p}) - I_{deblur}(\mathbf{p})|$, where $\tilde{I}$ is the non-blurred version of the rendered image, and $I_{deblur}(\mathbf{p})$ is approximately the non-blurred ground truth of the input $I(\mathbf{p})$. To compute I_{deblur} we use blind deconvolution with a maximum likelihood algorithm. Experimental results show that our approach works well in practice. Notice that this is different from running the deblurring algorithm as a pre-process to the whole pipeline, which will degrade the performance of our algorithm because of the ringing effects introduced by directly deblurring.

C. Blur Kernel Size

As described above, we model blur using a Gaussian kernel. Since we don't know the kernel size (i.e. σ in the Gaussian function) a priori, we take a few samples of σ in uniform steps, then run the EM algorithm to calculate the lighting parameters and segmentation labels for each sampled σ, and finally compare the reconstruction error E. Usually the lowest reconstruction error is achieved when the kernel size is close to ground truth. Therefore taking a sequence of increasing σ values, we keep running the computation until we observe an increase in the reconstruction error. This is analogous to how the camera's auto-focus works. To reduce the overall computation cost we pass the output labels and lighting parameters from the previous sampling step to the next one, and use them for initialization.

D. Summary

To summarize: our algorithm starts by assuming no blur, then gradually increases the σ of the Gaussian kernel from 1 to 10 in integer steps. During each step, we perform 3 EM iterations for updating the segmentation labels and estimating lighting parameters, except for when $\sigma = 0$ we perform 10 iterations of EM to allow for better convergence.

V. Results

In this section we present experimental results. We ran the experiments on a single core of an Intel i5 3.2GHz CPU. It took a few seconds to a few minutes to segment one image, depending on the image size.

To begin, we present the segmentation results by using inverse rendering. Figure 2 shows selected examples. Note that since our algorithm extracts the lighting parameters and blur kernel size, we can reconstruct a rendered image. Figure 2 column (c) shows the reconstructed images. Note that they look very similar to the input images in column (a).

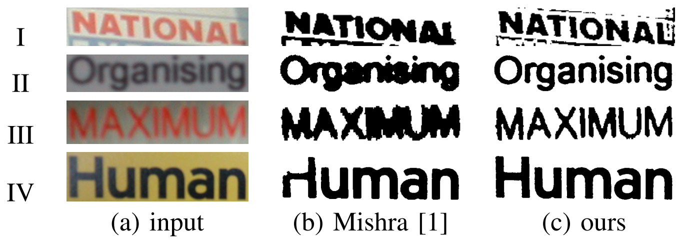

Fig. 3. Comparing segmentation results between Mishra et al.'s method [1] (b) and our method (c). Column (a) shows the input images.

	ICDAR(FULL)	ICDAR(50)
Mishra et al. [1]	67.30%	75.07%
Our method	69.29%	76.78%

TABLE I. Comparison of word spotting accuracy between Mishra et al's method [1] and our method.

While we could directly compare the segmentation results of our approach to ground truth, in practice such ground truth is difficult to obtain. Moreover, the effectiveness of segmentation is ultimately measured by the accuracy of the final text recognition. Therefore we compare our approach to other segmentation methods in terms of the text recognition accuracy. We do so by using a complete scene text recognition system and varying the segmentation method used. In the following experiments we focus specifically on *word spotting* [18], where we are given a pre-specified lexicon guaranteed to contain the ground truth, and the goal is to choose a word label from it.

Our recognition system uses a segmentation image to find an initial word label and then finds the closest word to that label in the lexicon. To find an initial label, we consider each connected component in a segmentation image to be a character, and we model the sequence of characters with a linear-chain conditional random field (CRF) model. For each character, we extract one histogram of oriented gradients (HOG) descriptor, centered and covering the character as the appearance features. We estimate the parameters of the CRF model using maximum likelihood training by minimizing the negative log-likelihood of the objective function. Then, we use this model to find an initial word label using the Viterbi decoding algorithm [19]. We use a software package for graphical models by Mark Schmidt [20]. Next, we calculate the edit distance between the initial label and each word in the lexicon. The edit distance is the minimum number of insertions, deletions or substitutions required to transform one string into another. We choose the lexicon word with the smallest edit distance as the final label.

We compare our text segmentation method to the state-of-the-art text segmentation method by Mishra et al. [1] on the ICDAR 2003 scene data set [21]. We use the scene version of this data set because we were provided segmentations for direct comparison. We compute the accuracy using a lexicon containing all words in the test set (ICDAR FULL), and a lexicon with the ground truth word for the image plus 50 random words (ICDAR 50). The word-spotting accuracy is shown in Table I. From the results we can see that our method achieves higher accuracy in both cases.

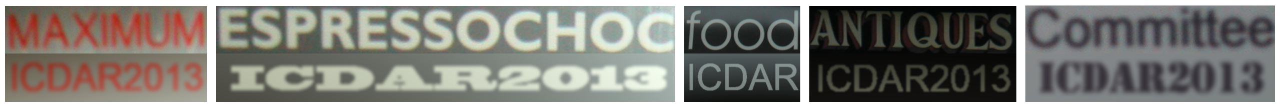

Fig. 5. Our method can be used to replace text in existing images. This is done by applying the lighting parameters extracted from the existing image to novel text, producing new images (second row) that look similar in appearance to the input images (first row). Here the novel text we applied is 'ICDAR2013'.

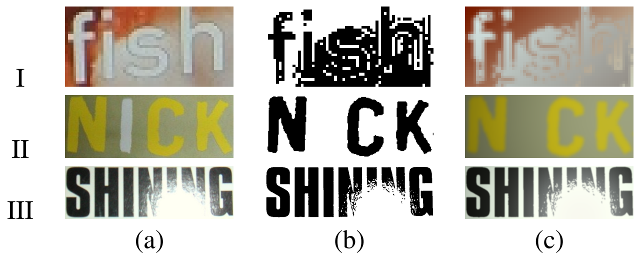

Fig. 4. Failure cases. Column (a): input images; Column (b): segmentation results; Column (c): reconstructed images.

Some selected results for comparison are shown in Figure 3. Note that Mishra et al's method may fail to segment small features (like the hole in the capital 'A' in Row I) while our method can handle these cases well. Since we consider blur estimation, our method is particularly effective in segmenting blurred images, as shown in Row II and III. Row IV shows an image where there is a smooth color transition due to lighting effects. This leads to incorrect segmentation in their method, while our method is able to handle it well.

Our method also has limitations in challenging scenes. Figure 4 shows a few failure cases. Rows I and II show cases where the foreground contains multiple colors, some of which are similar to the background. This causes our algorithm to incorrectly classify some foreground pixels into background. Row III shows a case where there is very strong highlight, which again causes some foreground pixels to have similar color with the background and lead to incorrect segmentation.

In addition to text recognition, our method can be used to synthesize new text images that look similar to existing images. To do so, we simply replace the label image with novel text, and render a new image with the same lighting parameters. This allows us to replace text in place, as is shown in Figure 5.

VI. Conclusion and Future Work

In this paper we introduce a new method where an inverse rendering approach is integrated into a text segmentation pipeline. We compare our new segmentation algorithm with previous methods by evaluating its accuracy on a word-spotting task. We demonstrate that our model is effective for modeling scene text images with single foreground material, single background material, and blur effect.

There are several directions for future research. First, in the current work we assume that the text image only contains two uniform materials, which limits the use of our algorithm to scenes with multiple materials. Thus an extension would be to segment pixels into multiple groups and have one set of reflectance parameters for each group. Another future research direction would be to incorporate a global language model into the pipeline, which will help in challenging scenes such as those with similar foreground/background colors, or those with a strong specularity that washes out part of the text.

Acknowledgments This work was supported in part by NSF Grant IIS-0916555 and NSF Grant CCF-0746577.

References

[1] A. Mishra, K. Alahari, and C. V. Jawahar, "An mrf model for binarization of natural scene text," in *ICDAR '11*, 2011, pp. 11–16.

[2] K. Wang, B. Babenko, and S. Belongie, "End-to-end scene text recognition," in *ICCV '11*, 2011, pp. 1457–1464.

[3] A. Coates, B. Carpenter, C. Case, S. Satheesh, B. Suresh, T. Wang, D. Wu, and A. Ng, "Text detection and character recognition in scene images with unsupervised feature learning," in *ICDAR '11*, 2011, pp. 440 –445.

[4] A. Mishra, K. Alahari, and C. Jawahar, "Top-down and bottom-up cues for scene text recognition," in *CVPR '12*, 2012, pp. 2687–2694.

[5] L. Neumann and J. Matas, "A method for text localization and recognition in real-world images," in *ACCV '10*, 2011, pp. 770–783.

[6] ——, "Text localization in real-world images using efficiently pruned exhaustive search," in *ICDAR '11*, 2011, pp. 687–691.

[7] ——, "Real-time scene text localization and recognition," in *CVPR '12*, 2012, pp. 3538–3545.

[8] P. Stathis, E. Kavallieratou, and N. Papamarkos, "An evaluation technique for binarization algorithms." *J. UCS*, vol. 14, no. 18, pp. 3011–3030, 2008.

[9] K. Kita and T. Wakahara, "Binarization of color characters in scene images using k-means clustering and support vector machines," in *ICPR '10*, 2010, pp. 3183–3186.

[10] X. Wang, L. Huang, and C. Liu, "A novel method for embedded text segmentation based on stroke and color," in *ICDAR '11*, 2011, pp. 151–155.

[11] B. T. Phong, "Illumination for computer generated pictures," *Commun. ACM*, vol. 18, no. 6, pp. 311–317, 1975.

[12] A. Ngan, F. Durand, and W. Matusik, "Experimental analysis of brdf models," in *Proceedings of the Eurographics Symposium on Rendering*, 2005, pp. 117–226.

[13] Y. Yu, P. Debevec, J. Malik, and T. Hawkins, "Inverse global illumination: recovering reflectance models of real scenes from photographs," in *SIGGRAPH '99*, 1999, pp. 215–224.

[14] R. Ramamoorthi and P. Hanrahan, "A signal-processing framework for inverse rendering," in *SIGGRAPH '01*, 2001, pp. 117–128.

[15] M. Chandraker and R. Ramamoorthi, "What an image reveals about material reflectance," in *CVPR '11*, 2011, pp. 1–8.

[16] G. Patow and X. Pueyo, "A survey of inverse rendering problems," *Comput. Graph. Forum*, vol. 22, no. 4, pp. 663–688, 2003.

[17] K. Levenberg, "A method for the solution of certain non-linear problems in least squares," *Quarterly Journal of Applied Mathmatics*, vol. II, no. 2, pp. 164–168, 1944.

[18] K. Wang and S. Belongie, "Word spotting in the wild," in *ECCV '10*, 2010, pp. 591–604.

[19] A. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Transactions on Information Theory*, 1967.

[20] M. Schmidt, "UGM: Matlab code for undirected graphical models," http://www.di.ens.fr/ mschmidt/Software/UGM.html, 2013.

[21] L. P. Sosa, S. M. Lucas, A. Panaretos, L. Sosa, A. Tang, S. Wong, and R. Young, "Robust reading competitions," in *ICDAR '03*, 2003, pp. 682–687.