Shadow Removal on Colored Text Document

LO, Shih-heng
sloab@connect.ust.hk

CHAN Cheuk Wai
cwchanbm@connect.ust.hk

HSU Chia-hong
chsuae@connect.ust.hk

YU, Szu Ting
styuaa@connect.ust.hk

Abstract

In this paper, we re-visit the classical shadow removing for document pictures. Traditional algorithms regarding this topic share similar pipelines that possess a global background detection stage and a local background detection stage [1–3]. Recent image-processing-based algorithms transform the shadow removal task to a visibility detection problem by mapping the image intensity values to a 3D point cloud [4, 5]. Instead of proposing a new fancy algorithm, our proposed algorithm can generate enhanced colored results and binary results by using some simple pooling operations for shadow detection. We will show that our algorithm is intuitive, runtime efficient, and produce consistent results. We will evaluate our algorithm on the unshadowing task and the binarization task on document pictures. Lastly, we will conduct qualitative and quantitative benchmark experiments with previous algorithms and discuss some limitations of our proposed algorithm.

1. Introduction

The increase in smartphone usage has led to a significant increase in generating digital content with built-in cameras. Aside from capturing moments, the cameras also capture documents for digitization. Document (e.g., handwritten test papers, assignments, and projects) picturing is especially customary among students during the pandemic when most education is online. However, the quality of captured digital images may often be a concern. Shadows, or distorted illumination, are common forms of degradation in digital workflows, affecting the accuracy of the Optical Character Recognition (OCR) algorithm. Other defects such as stains, spots, ink bleeding, etc., may also affect the cleanness of the digitized result. To discuss the potential difficulties for document scanning, we will focus on two tasks in our experiments: 1) Text image binarization. 2) Image unshadowing. For the formal definition of the two tasks, please refer to section 2.

Earlier algorithms focus on removing shadows in the background by replacing the local backgrounds with the global background [1–3]. The global background can be seen as the general color of a document sheet that contains no text nor noise on the surface. Recent algorithms aim to solve the visibility problem of convex hulls by transforming the images to point cloud coordinates [4, 5]. In particular, the text regions result in steeper valleys on the 3D point cloud, whereas higher intensity pixels resemble mountains or ridges. Variations of this algorithm include the Water filling algorithm [6], which will be explained in more detail in Section 3.

In our study, we propose an explainable and effective image-processing- based algorithm to tackle this problem. We will show that our algorithm can remove the majority of shadows for most damaged images. To show that our algorithm is practical in real-life applications, we will also demonstrate the runtime advantage of our algorithm.

2. Problem

Our study focuses on removing shadows from document photos to enhance the readability of the defective image. Shadow is defined as all dark areas on the image that is caused by occlusion of one or multiple light source. However, there are more factors in real life that cause the quality of smartphone taken pictures to be reduced. This includes, but is not limited to folds, stains, inks, as well as soaked regions on the text sheet. In our study, the definition of a shadow is generalized to include all these damaging factors above.

There are two approaches for document readability enhancement: 1) Unshadowing, 2) Text image binarization. For the unshadowing task, we take an RGB document image as input that contains shadows and output the original RGB text image with all its shadowing removed. One of the biggest challenges of unshadowing is to preserve the color intensity values. Besides removing the shadows, we want to neutralize the color difference between regions contaminated with the clean regions. For text image binarization,

it transforms the digitized document into a classifying the differences between text and background. The input is an RGB image, and the output is a binarized image with only two intensity values representing the document content and the background.

3. Literature Review

We review the related works in the literature and classify their algorithms into the subsections as listed below.

3.1. Background Shading Estimation

One of the commonly used approaches for removing shading from RGB digitized documents is by estimating local and global background color. The algorithm, proposed by Bako et al. [1], consists mainly of two parts: 1) Find the local and global background colors, and 2) Compute the shadow map based on the background color reference. The algorithm has its basis on the assumption of digitized documents having constant colored backgrounds.

Segmenting the image into small overlapping blocks and clustering the pixel intensities into two clusters, the authors find the local background colors of paper and text. Gaussian mixture models (GMM) with Expectation-Maximization (EM) and initialized with k-means clustering are used for clustering both the local and global data. The cluster with a higher mean is identified as the local background RGB color. The global reference background color is found by accounting pixel intensities of the entire image input and clustered into two categories, with the higher mean cluster labeled as background intensity. The pixels intensity closest to the background cluster means of the original input image is identified as the global background RGB reference. The authors then compute the ratio of local and global background colors to generate the shadow map, which is then mapped to each input pixel and produces an unshadowed result.

The limitation of this approach is that it assumes input images contain intensities corresponding only to either paper or text (i.e., no graphs, pictures, figures, etc.). The cluster mean will be biased and cause inaccurate background estimation if the additional regions' intensities are computed. In this case, an image-customized mask specifying these regions is taken to ignore them when computing the shadow map. Interpolation is taken to fill the hole after shadow mapping and generating the resulting image. The result is also often undesired if the document's background color changes throughout the document, causing the global background reference unreliable. Intense hard shadows will cause visible boundaries in the resulting output due to the local cluster mean being inaccurate. This approach, written in C++, takes roughly 2 seconds to process a 1024x1024 image.

3.2. Water-Filling Algorithm

Illumination distortion is caused not only due to occluding light sources by the camera itself or the camera holding hand of the user but also due to the paper documents having folds and causing uneven surfaces. In [6], Jung et al. proposed an illumination correction algorithm consisting of three stages: 1) Construct a topographic surface using the luminance values of pixels from the input digitized documents, 2) Model by diffusion equation for shading artifacts estimation, including two methods: *incrementally filling of catchment basins* method, and *flood-and-effuse* method, and 3) Digitized document reconstructed using the Lambertian surface model and remove the shades. The algorithm is based on the simulation of the dynamics of fluids on the topographic surface.

The main purpose of the modeling by diffusion equation is to restore and minimize the background of digitized documents affecting the overall structure. In *incremental filling of catchment basins*, the assumption of a continuous water supply source from a certain peak is made. Let *G(x,y,t)* be the overall altitude of water at a location *(x,y)* at time *t*. The value of *G()* after convergence is used to estimate the background layer of the original image. The *flood and effuse* method is further decomposed into two processes: *water effusion*, and *flood*. In the effusion process, it is considered that the dynamics of water flow are without an external water supply and that the effusion of water is only through the image border. After G() converges, it can be used to reconstruct the correct photometric of a digitized document. The Lambertian surface model is then applied for computing the foreground text layer and background shading layer. The final unshadowed image can be computed with the layers' value.

The performance of the water-filling algorithm takes an average time of 3 to 4 seconds to process a 756x600 image, using C++ running on our 2.3 GHz Intel Core i5 CPU. The limitation of this algorithm is that any text or photos which are connected to the image border are considered as shadows and are eliminated in the resulting image. The method also can not reconstruct digital images under overexposure, where the foreground layer is significantly damaged.

3.3. Visibility Detection

Kligler et al [4]. has proposed a method that transforms the original shadow removal problem to a visibility problem of image pixels. The algorithm consists of three stages: 1) Transform each pixel from $\mathbf{R^2}$ to a 3D cloud point in $\mathbf{R^3}$, 2) Detect visible/occluding points from a viewpoint, 3) Create the visibility based image, and 4) Apply an existing algorithm for the unshadowing task or binarization task on the visibility image. In the first stage, dark intensity values will be mapped to shallow valleys (text) or craters, whereas high-intensity values become ridges or

higher plateaus (global background). The second stage aims to apply different operators (HPR and TPO) that distinguish between narrow valleys and all other points. This can be defined as the following: given a point set $P \in \mathbf{R^3}$, and a viewpoint C, a point sampled from a surface S is considered to be visible to C on S. After some transformation that maps points in P to P' along the ray with respect to the viewpoint C, the algorithm searches for a convex hull that covers all subsets of P' such that any points residing on that convex surface is considered to be visible or occluding. The visible points will be used to construct the visibility image in stage 3. And during stage 4, we can apply shadow removal algorithms on the generated visibility image instead of the raw input.

The result of this algorithm is proven to be adaptive as it deals with many types of defects effectively. The creativity of this approach also attracted further works that extend this algorithm, for instance, Li et al. [5] proposed a loop to find more convex hulls, then takes the union of all visible points. However, the limitation of this algorithm is that it is too complex for practical applications as it takes $O(nlogn)$ during the convex hull step. The performance of the visibility detection takes an average time of 10 seconds to process a 756x600 image, using Matlab on 1.8 GHz Intel Core i7 CPU.

4. The Proposed Method

Generally, our proposed algorithm is composed of three stages as shown in figure 1. First, we perform shadow detection by $MaxMin$ algorithm on each channel of input image $I(c, x, y)$. Next, subtract the input image with the shadow mask by $ShadowRemove$. Eventually, restore the histogram by $HistogramRestore$ and merge the result of the three-channel back to the RGB domain. The simplified algorithm is shown as follow,

Algorithm 1 Image Shadow Removal

Input Shadowed RGB Text Image
Output Clean RGB Text Image

procedure SHADOWREMOVE3C(*img*)
B, G, R $= Split($*img*$)$
BlueRM $= ShadowRemove1C($*B*$)$
GreenRM $= ShadowRemove1C($*G*$)$
RedRM $= ShadowRemove1C($*R*$)$
Output $= Merge($*BlueRM*, *GreenRM*, *RedRM*$)$
$return$ *Output*

procedure SHADOWREMOVE1C(I)
$shadow = MaxMin(I)$
$I_{Diff} = ShadowRemove(I,$ *shadow*$)$
Output $= HistogramRestore(I, I_{Diff})$
$return$ *Output*

4.1. Shadow Detection

In Shadow detection, we aim to get a clear shadow map with a similar intensity of the shadow applied to our corrupted image. We apply a max filter with size 11*11 on each channel of $I(c, x, y)$ separately, followed by a mean filter with size 3*3 to the previous result. Then we get a rough shadow mask as figure6 shown. Notice that we can still see some white boxes appear near the position of the original text after this step.

In order to reduce the effect of text on our shadow map to the maximum extent, we apply a min filter with size 11*11 on each channel of the rough shadow mask separately. After that, a mean filter with size 3*3 is then applied to the previous result. As a result, we get a fine shadow mask which is the output of this Max-Min filtering. The white box, which is text initially, is now smoothly replaced.

Note that we chose the kernel size to be 11 and 3 because of the assumption that each stroke of characters is only several pixels wide. Otherwise, the kernel size needs to be chosen accordingly. More details will be discussed in the limitation part.

4.1.1 Reason for using max filter

Max filter is used for dilating the text into the background color, i.e., text in the shadow region will be dilated into the shadow background intensity, text in the bright region will be dilated into the bright region intensity

4.1.2 Reason for using min filter

Min filter is used for smoothing the image created by the max filter. It is also used for restoring the dilated shadow map closer to the original shadow map. The shadow has to be restored to ensure that the shadow image after max-min filtering will be subtracted from the shadow in the original image

4.1.3 Reason for using mean filter

Mean filter is used for denoising after each of the max and min filters. Despite the blurry effect of the mean filter, we want to extract the shadow mask, and thus, the effect is negligible. Doing so will help the *minMean* image extracted to be less noisy. Consequently, the background subtraction can be performed with a cleaner input.

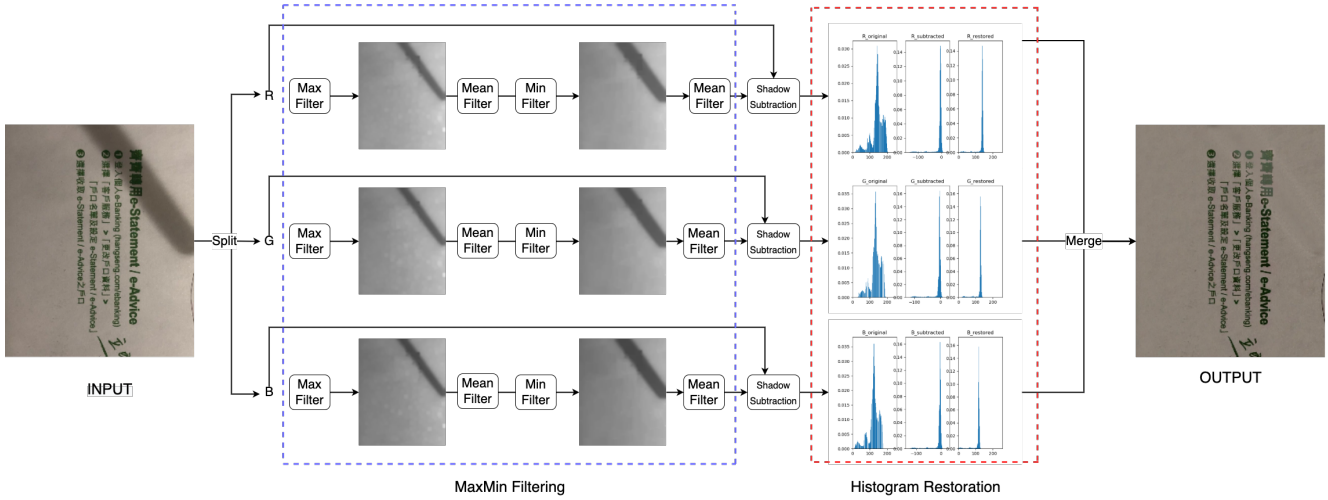
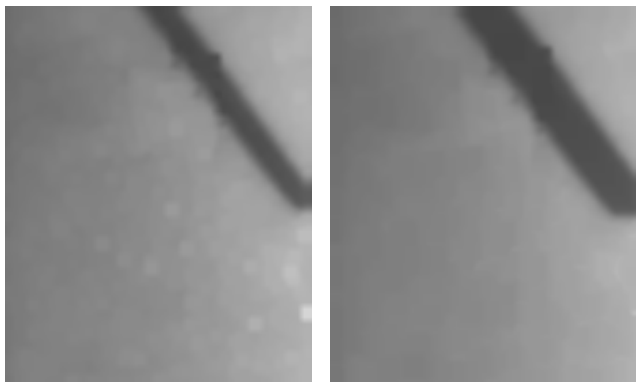


Figure 1. Flow of our algorithm

Algorithm 2 Max-Min Filtering

Input Single Channel Image
Output Shadowed Map of Input

procedure MAXMIN(*I*)
max $\leftarrow MaxFilter(I)$ ▷ 11 x 11 Kernel
maxMean $\leftarrow MeanFilter(max)$ ▷ 3 x 3 Kernel
min $\leftarrow MinFilter(maxMean)$ ▷ 11 x 11 Kernel
minMean $\leftarrow MeanFilter(min)$ ▷ 3 x 3 Kernel
$return$ *minMean*



(a) maxMean (b) minMean

Figure 2. Shadow Detection using MaxMin Filtering

4.2. Shadow Subtraction

We subtract $I(c, x, y)$ with the result of previous step minMean and get a result $I_s(c, x, y)$. Note that most of the intensity of pixels after subtraction will be close to zero, and pixels of texts will have negative values depending on the initial intensity. For example, pixels of pure background of $I(c, x, y)$, which is close to $RGB(255, 255, 255)$, minus pixels of pure background of $I_s(c, x, y)$, which is also close to $RGB(255, 255, 255)$, will have a result close to $RGB(0, 0, 0)$. The same situation happens on pixels of pure shadow. The text should be the only area with a strong negative value. As a result, the subtracted image should be a differential image in which the shadow and background area is close to black and text with negative values.

Algorithm 3 Channel Shadow Removal

Input Single Channel Image
Output Differential Image

procedure SHADOWREMOVE(I, *Shadow*)
Output $\leftarrow I - Shadow$
$return$ *Output*

4.3. Histogram Restoration

First we define $H_a(I)$ as a histogram distribution of a image I in specified area a.

Assume that the difference between the histogram peak of the background $max(H_{bg}(I))$ and the text $H_{text}(I)$ remains the same after Shadow Mask and Shadow Subtraction operations. Then, we restore the histogram of each channel separately by linearly shifting the overall intensity by the amount of the difference between $max(H_{bg}(I_{orig}))$ and $max(H_{bg}(I_{res}))$. Specifically,

Algorithm 4 Histogram Restoration

Input Single Channel Differential Image
Output Restored Single Channel Image
procedure HISTOGRAMRESTORE(I_{orig}, I_{diff})
offset $\leftarrow max(H_{bg}(I_{orig})) - max(H_{bg}(I_{diff}))$
$I \leftarrow I_{diff} +$ offset
$return\ I$

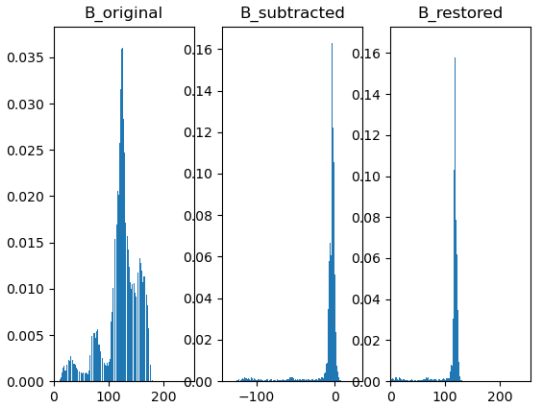


Figure 3. Intensity histogram of each step

Finally, we merge each restored channel to an RGB image. The shadow removal is then done.

5. Monochromatic Method

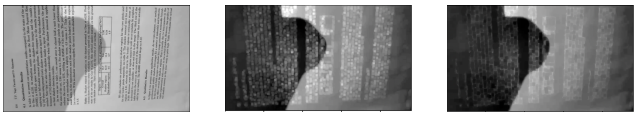
All of the approaches listed above assume you're working with a multicolored text document. The vast majority of the documents we read, on the other hand, are monochrome.

In fact, we've chosen to provide an alternative method that focuses mostly on grayscale or white and black RGB documents.

To remove the shadow from the document, there will be three processes, namely shadow detection, background subtraction and normalization, and binary mask

5.1. Shadow Detection

Just as the Same as dealing with RGB images, we will have to do min-max filtering on our monochromatic image. This time the image does not have to be split into the 3 channels. Instead, we can directly filter the image on all the 3 channels together.



(a) Input (b) Max-Mean Image (c) Min-Mean Image

Figure 4. Monochromatic Shadow Mask

5.2. Shadow Subtraction and Normalization

Getting the min-mean image allows us to execute shadow subtraction mentioned in *part 4.2*. After the shadow subtraction, some of the pixel intensity will be below 0 and that is not appropriate. Thus, in order to reset the representation of the intensity to what it normally would be. We will have to normalize the intensity from the background-subtracted value to the range of 0 to 255.

For our team, this operation is done by using the OpenCV function *normalize*. The effect of doing this is shown as the following.

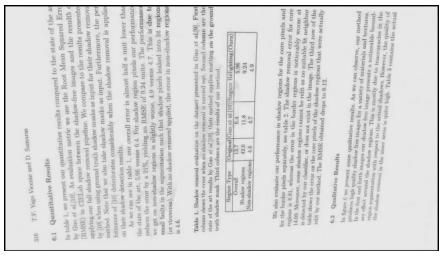
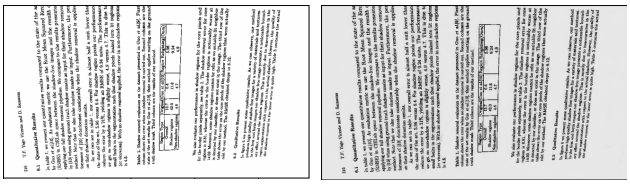


Figure 5. Normalization

5.3. Binary Mask

Even though after the background subtraction and normalization, the shadow is already removed. The text colors will often have a lower intensity and are harder to read. To counter this effect, we have generated a binary mask using a threshold to differentiate between the background and the text. After tuning the threshold values, we have found that pixels with an intensity lower than 180 should be treated as the background, and pixels with an intensity higher than the background should be treated as the text.

After obtaining the binary mask, we are replacing the intensity of all the pixels which are regarded as texts into the lowest intensity among them. This operation is done on the normalized image so that the background color can still be retained.



(a) Binary Mask (b) Final Output

Figure 6. Binarization Mask

5.4. Runtime Evalutation

As this application is supposed to be implemented on a mobile application, we want the runtime of this method to be as low as possible. Therefore, we are evaluating the runtime of our method.

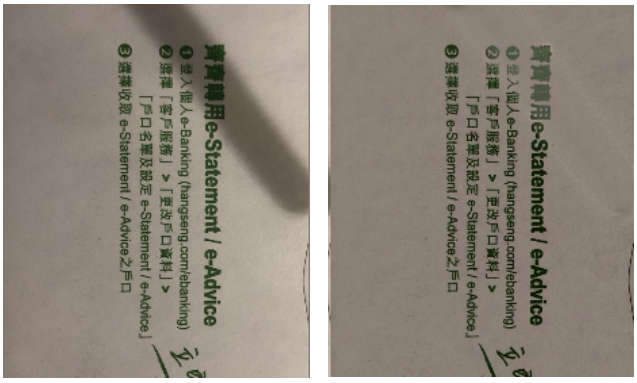
Since all the operations, including shadow mask extraction, shadow subtraction, and histogram restoration, are in $O(n)$ time asymptotically, the time complexity of our proposed method is in $O(n)$ time.

6. Experiment Result

Our Shadow removal method is implemented in OpenCV-Python. For a 460*375*3 image, it takes 4.46 seconds to remove shadow and restore color. These timings were measured on an AMD Ryzen 5800X CPU @3.0GHz with 32GB RAM.

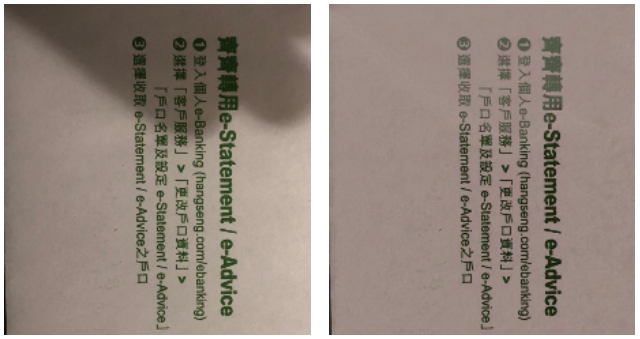
As Figure 7 has shown, our method can effectively remove shadows on colored text without severe color distortion. However, it is still visible that the color being covered by shadow is slightly shallower than the color it should be.

The problem comes from assuming that the background intensity is almost identical distributed and close to its mean. However, the lighting cause may unbalance background color, and our histogram restoration problem only considers linearly shifting. So, instead, we should find a more realistic restoration model in the future.



(a) Input (b) Output

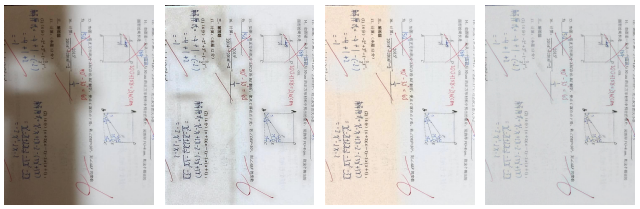
Figure 7. Result



(a) Input (b) Output

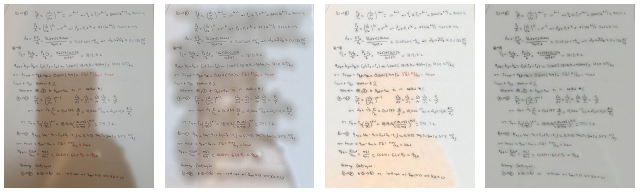
Figure 8. Another result

6.1. Benchmark



(a) Input (b) Visibility [4] (c) Water [6] (d) Ours

Figure 9. Benchmark 1

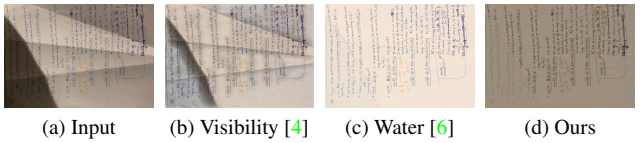


(a) Input (b) Visibility [4] (c) Water [6] (d) Ours

Figure 10. Benchmark 2

(a) Input (b) Visibility [4] (c) Water [6] (d) Ours

Figure 11. Benchmark 3



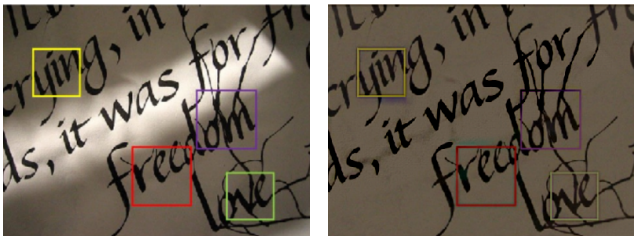
(a) Input (b) Visibility [4] (c) Water [6] (d) Ours

Figure 12. Benchmark 4

6.2. Limitation

1. Text Size: As we use min, max, and mean filter during the removal process, the kernel size indirectly influences the output as it affects if a text will be blurred after those filters. For example, if the strokes of text are more than several pixels and we apply an 11*11 max and min filter, the text will not be blurred entirely and result in a noisy shadow map.

2. Shadow Area: When doing histogram restoration, we treat the peak of the intensity as the intensity of background with the assumption that the shadow area will not be larger than half of the image. If the shadow area is larger than half of the image, our algorithm will mistreat its peak as the background color and shift the whole image intensity to the background color as shown in Figure 5.



(a) Input (b) Output

Figure 13. Limitation: Shadow area too large

7. Conclusion

In this paper, we introduce this simple but effective method to remove shadow while retaining the original color of the text. Our method stands out from other algorithms by the accuracy of color recovering and low run-time.

In the future, we plan to solve the current limitations of our method. First, we will find an adaptive method to decide our kernel size. We believe that can be achieved by statistical analysis of the text. Moreover, we hope that we can find some way to solve the problem in which the shadow area is too large for our algorithm to detect the actual background correctly.

Acknowledgments: Our initial idea comes from this article "Shadow Removal based on OpenCV" [7] written by Zhang Kanghui, which proposed a shadow removal method targeting gray-scale text documents. We follow the idea and modify the algorithm and extend it to an algorithm that works on colored text documents.

Teamwork Contribution

- CHAN Cheuk Wai designed the proposed algorithm
- LO, Shih-heng designed the proposed algorithm
- HSU Chia-Hong did the literature review
- YU, Szu Ting did the literature review

References

[1] S. Bako, S. Darabi, E. Shechtman, J. Wang, K. Sunkavalli, and P. Sen, "Removing shadows from images of documents," in *Asian Conference on Computer Vision*. Springer, 2016, pp. 173–183. 1, 2

[2] B. Wang and C. P. Chen, "An effective background estimation method for shadows removal of document images," in *2019 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2019, pp. 3611–3615. 1

[3] J.-R. Wang and Y.-Y. Chuang, "Shadow removal of text document images by estimating local and global background colors," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 1534–1538. 1

[4] N. Kligler, S. Katz, and A. Tal, "Document enhancement using visibility detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2374–2382. 1, 2, 6

[5] J. Li, Y. Chen, and S. Liu, "Document image binarization using visibility detection and point cloud segmentation," in *Chinese Conference on Pattern Recognition and Computer Vision (PRCV)*. Springer, 2021, pp. 92–104. 1, 3

[6] S. Jung, M. A. Hasan, and C. Kim, "Water-filling: An efficient algorithm for digitized document shadow removal," in *Asian Conference on Computer Vision*. Springer, 2018, pp. 398–414. 1, 2, 6

[7] Z. Kanghui. (2020) Opencv. [Online]. Available: https://zhuanlan.zhihu.com/p/335777554?fbclid=IwAR01E-dVKF6-Ts9r-Bn8P8Y6fNU5BP-pIum4LQJpupzjYJ7RD4LAhnzwls0 7