CDD Model: A Cylinder Detector and Dewarper

Haoyun Qin
Shanghai Foreign Language School Affiliated to SISU
No.295, Zhongshan No.1 Rd., Shanghai
1247006353@qq.com

Abstract

Perspective transformation and edge detection are fundamental and important problems in the field of computer vision. This paper proposed a fast and adaptable method, CDD, to perform the detection and transformation of cylinders in images. In addition, in the cases when the cylinders are book pages, we optimized our model in the detection part, enabling the whole process to be automatic. Compared to other methods proposed, our model (CDD) has the features of shooting-angle-independent and content-independent. According to our experiment, after using CDD model, the OCR accuracy improved from 21.43% to 84.97% on average. Besides, the low computing power requirement of our model enabled us to implement the method on mobile terminals and embedded devices, and a demo app on iOS platform has been deployed.

Key words: CV; perspective transformation; page dewarp; machine learning; edge detection

1. Introduction

Transformation, restoration and detection of objects are very important tasks in computer vision. It can help us process and use images better. In the field of computer vision, this step usually belongs to the preprocessing work, and the quality of restoration is often related with the follow-up works and the complexity of manual processing. In the absence of image restoration, it would be time-consuming, laborious and costly to process these images manually, so a good image restoration algorithm is of great importance.

With the rapid development of information technology, image, as a very important information carrier, has been widely used in life and production. In our daily lives, we sometimes need to scan the documents or pages and transform them into rectangular shapes. Because not everyone has a scanner and it's not portable, this leads to the emergence of scanning softwares. Such software is of great significance, because it can greatly improve people's work efficiency and bring great convenience to people. But even so, there are still many problems in this kind of softwares nowadays, for example

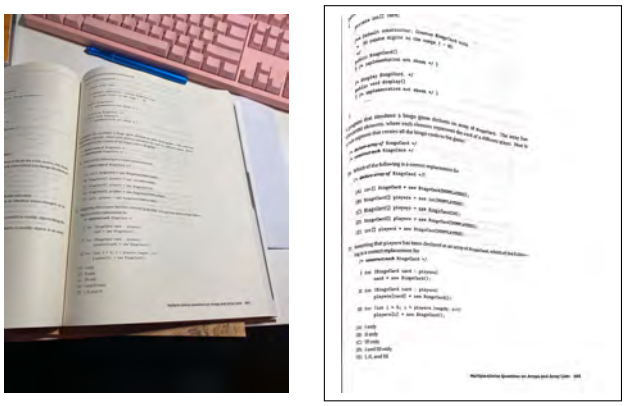


Figure 1. The software shown can only perform mere perspective transformation, and cannot deal with the curved edge

- curve edges cannot be handled correctly, see Fig. 1
- time-consuming
- edges cannot be correctly detected

In order to solve these potential problems, this paper proposed a more generalized solution, CDD model, which has solved the problem of cylinder recognition and restoration in image.[1] In addition, in the cases when the cylinders are book pages, we optimized our model in the detection part, enabling the whole process to be automatic. Generally speaking, our model contains three main parts, that is, edge detection, cylinder segmentation, and cylinder dewarp. Fig. 2 shows the procedure of the model.

After proposing the model, we implemented it on iOS platform (see Fig. 7 and Sec. 6). According to our small-sample experiment, after using our CDD model, the OCR accuracy improved from 21.43% to 84.97% on average (see Tab. 2). Compared with mzucker's model [16, 17], our model also perform well in severe conditions (see Fig. 16), and our timing cost is also significantly shorter (see Tab. 3).

[1] Cylinder is a curved surface formed by the parallel movement of a straight line along a fixed curve. The moving line is called the straight generatrix of the cylinder, and the fixed curve is called the directrix of the cylinder.

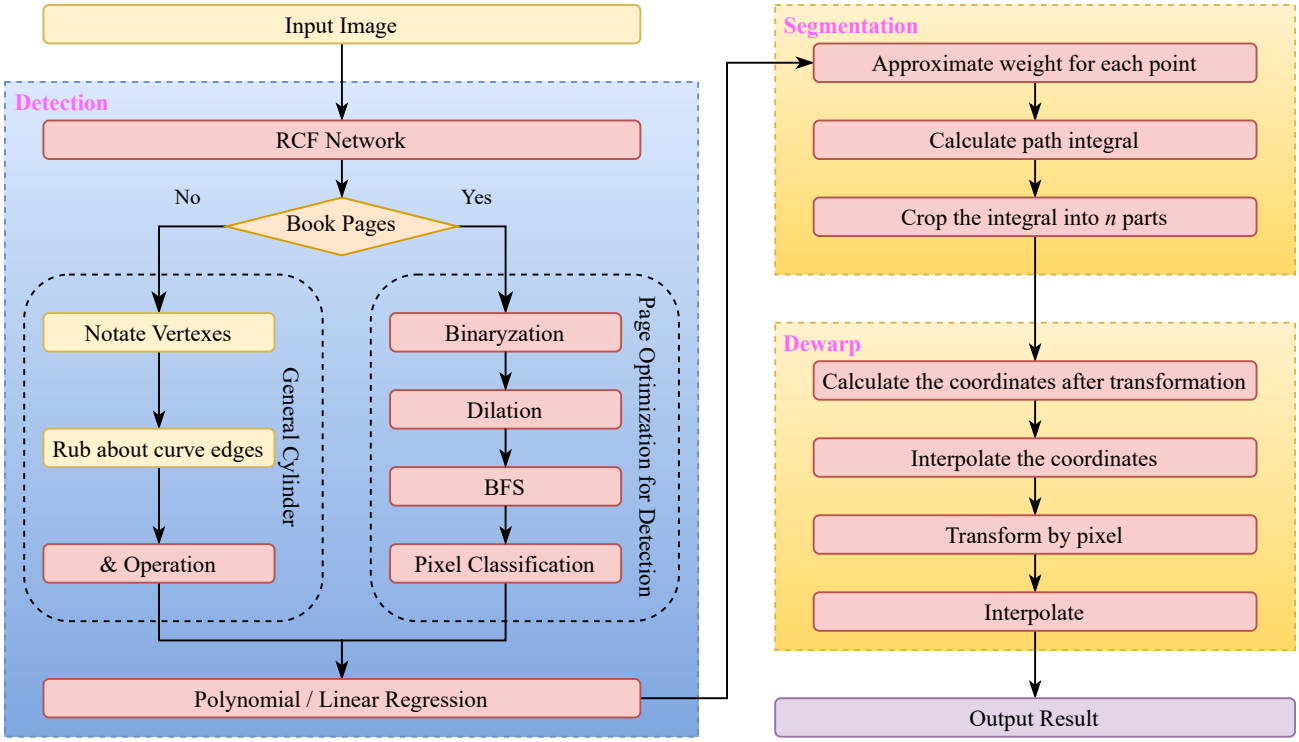


Figure 2. CDD Procedure

2. Related Works

As comparatively basic problems in computer vision, cylinder dewarp and book dewarp have been studied by researchers since last century. In fact, this can also reflect the importance of this problem and the value of studying it. For example, Liu Yifei's research on non-contact image shooting and restoration of ancient books based on edge information and shooting parameters at specific shooting angles [12], Michael P. Cutter *et al.* restored book pages by 3D cameras [5], Fengjun Guo and A. Ulges *et al.* reconstructed the surface by recognizing the text direction of the page to realize transformation [9, 24], and so on. Besides, many document dewarping networks have benn proposed these years, such as DocUNet, DewarpNet, etc. [15, 23, 2, 6]. Besides, we also found an open-source project on github discussing the simalar problem, which is the detection and dewarp of book pages [16, 17].

3. Cylinder Detector

Cylinder detection aims to distinguish cylinders in images. In fact, there are two approaches to solve this problem: object detection or edge detection.

3.1. Object Detection Method

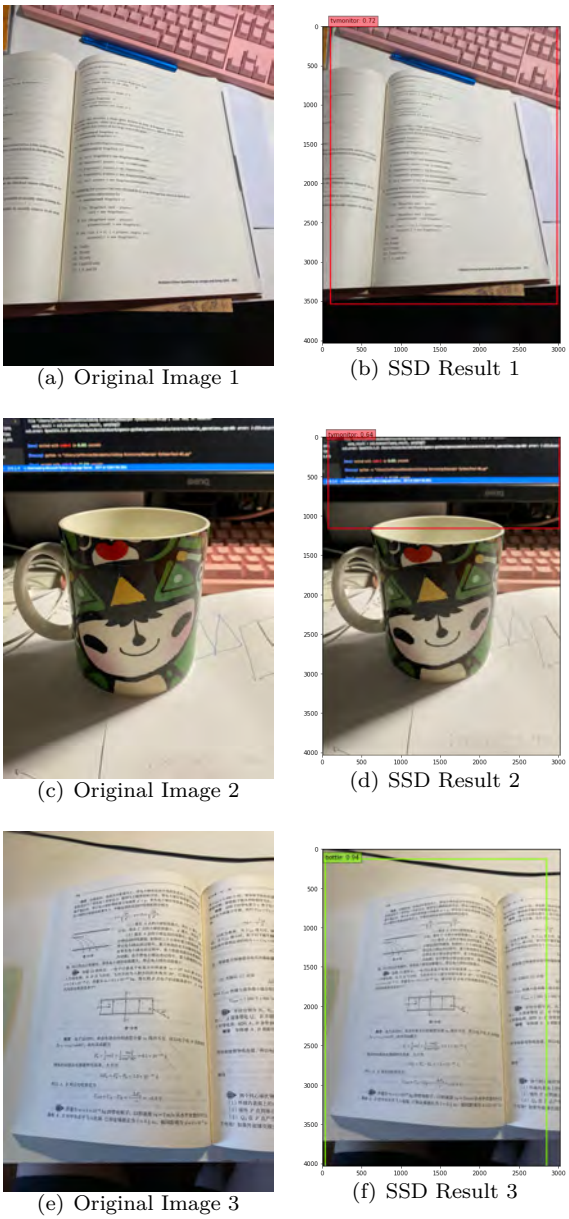
Though we refer to the problem as cylinder detection, when we apply the method to the specific cases, cylinders can be different objects, which makes the problem even harder. As a result, again, here we may focus on the special cylinders first, for example, book pages and cups. There are many models to recognize the objects in an image, such as R-CNN [8], Fast R-CNN [7], Faster R-CNN [22], Mask R-CNN [1], Joseph Redmon *et al.*'s YOLO [19, 20, 21], Wei Liu *et al.*'s SSD [11], etc. In this way, we can get the position information and the confidence probability of the cylinder in the image. However, after the experiment, I found that due to the complexity of the pattern on the page and cylinder, the traditional object detection method can not recognize well. Besides, the ambiguous classification "cylinder" itself contributes to the unideal result. The result of SSD Model is shown in Fig. 3, which is not ideal enough.

- In Fig. 3(b), book page is recognized as tvmonitor, with confidence probability of 0.72
- In Fig. 3(d), cup is not detected
- In Fig. 3(f), book page is recognized as bottle, with confidence probability of 0.94. The position is not accurate enough.

Based on the results above, we found that the cylinder detection method based on object detection is not reliable.

3.2. Two Methods of Edge Detection

As one of the most important problems in the field of computer science, numerous methods of edge detection have been proposed. From simple to complex, there are methods such as directly calculating the Sobel Operator, using Canny Edge Detection Algorithm (an extended version of Sobel), and applying deep learning: HED [26, 10], BDCN, RCF [13, 14]. Here, I will mainly introduce two representative methods: Canny Algorithm and RCF Model.



(a) Original Image 1 (b) SSD Result 1

(c) Original Image 2 (d) SSD Result 2

(e) Original Image 3 (f) SSD Result 3

Figure 3. SSD Results

3.2.1 Canny Edge Detection

Canny edge detection was first proposed by John canny in 1986 in his paper: A Computational Approach to Edge Detection [4]. It is a technique to extract useful structural information from different visual objects and can greatly reduce the amount of data to be processed. The algorithm can be divided into the following steps:

- **Applying Gaussian filter to smooth the image and reduce the noise**
  Canny edge detection is based on the difference in gray level of the image. Since in most cases, the input images are generally three-channel or four-channel color images, conversion is needed. The following formula is a typical one:

$$\text{Gray} = R * 0.299 + G * 0.587 + B * 0.114$$

  Gaussian filter is a kind of linear smoothing filter, which is suitable for eliminating Gaussian noise, especially for suppressing the noise obeying normal distribution. The filter can reduce the influence of noise in images, and it can also retain the edge information well. Applying Gaussian filter, we need to convolute the original image with the convolution kernel of Gaussian filter. The following is the expression of two-dimensional Gaussian kernel function and convolution kernel matrix (its essence is to calculate weighted average of the surrounding pixels according to two-dimensional normal distribution):

$$f(x,y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$$

$$H_{ij} = \frac{1}{2\pi\sigma^2} \exp\Big(-\frac{(i-(k+1))^2+(j-(k+1))^2}{2\sigma^2}\Big);$$

$$1 \le i,j \le (2k+1)$$

  The final expression of each pixel after filtering is expressed as

$$g_\sigma(m,n) = \sum_{1\le i\le 2k+1} \sum_{1\le j\le 2k+1} g_{ij}$$

$$g_{ij} = H_{ij} f(m+i-(k+1), n+j-(k+1))$$

- **Using Sobel operator to calculate the magnitude and direction of image gradient**
  The expressions of Sobel operators in the x direction and y direction are:

$$S_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, S_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

  To get the gradient, we have to convolute the original image with the Sobel Operator, and obtain the gradient G_x, G_y in the direction of x and y respectively.

$$G = \sqrt{G_x^2 + G_y^2}, \theta = \arctan\frac{G_y}{G_x}$$

- **Non maximum suppression is applied to refine the edges**
  Non maximum suppression is an edge sparsity method, which can refine the edge. After calculating the gradient of the image, the edge is still blurred only based on the gradient value. For each pixel, the gradient intensity of the current pixel is compared with the two pixels along the positive and negative gradient direction. If the gradient intensity of the current pixel is the largest compared with the other two pixels, the pixel is retained as edge.

- **Double Thresholding to determine true edges and potential edges**
  After non maximum suppression, the remaining pixels can more accurately represent the actual edge in the image. However, this still can not avoid the

occurrence of some edge pixels due to noise or color change. For this reason, double thresholds are used to filter out non-edge pixels. If the gradient is higher than the high threshold, the point is marked as a strong edge; if the gradient is lower than the low threshold, the point is rounded; if the gradient is in the middle, the point is marked as a weak edge.

- **Finally, the edge detection is completed by suppressing the isolated weak edge.**
  For each weak edge, if the weak edge is near the strong edge, then the point is also marked as a strong edge, otherwise the point is discarded.

Fig. 4 is the result of Canny Edge Detection. We found that a lot of irrelevant information were also included, because the essence of Canny edge detection is the gradient change of images. As a result, this leads to some cases when some non-edge points whose gradient change is obvious are also calculated as edges, or some edge points whose gradient change is not obvious can not be detected correctly. Therefore, Canny edge detection can not distinguish and discriminate some details effectively. For this reason, we will introduce RCF edge detection network, which would help us solve this problem.

3.2.2 RCF Edge Detection Network

1. Network Architecture RCF (Richer Convolutional Features for Edge Detection) is a convolution network proposed by Yun Liu *et al.* [14, 13], which is used to extract edge features of images. The network is based on VGG16. Fig. 5 shows the network architecture.

Compared with the original VGG network:

- RCF cut all the fully connected layers and the pool layer in the last stage. This is because the purpose of RCF is different from the original design of VGG network, that is, image classification. The network aims at edge detection, so the output of $1 \times 1 \times 4096$ from the last fully connected layer of VGG is meaningless. As a result, it was removed.
- In order to attain hybird fetures, each *conv* layer in VGG 16 is connected to a $1 \times 1 - 21$ *conv* layer. The dimension in the network is first increased and then decreased by a $1 \times 1 - 1$ *conv* layer.
- A *cross-entropy loss / sigmoid* layer is connected to the up-sampling layer in each stage.
- In each stage, a *deconv* layer is used to perform up-sampling, mapping the result to the original size of the image. Finally, all the up-sampling layers are concatenated in the *fusion* stage, and then an $1 \times 1 - 1$ *conv* layer is used to fuse feature maps from each stage, which enables the network to obtain a variety of mixed information.

It's worth noting that, with the same characteristics of VGG16, since all convolutional, pooling and deconvolutional operations do not have absolute requirements on size, RCF network can accept images of any size and get the same size output.

2. Annotator-robust Cost Function The RCF Model also introduced an annotator-robust cost function, which contributes much to the great performance.

Edge datasets in this community are usually labeled by multiple annotators using their knowledge about the presences of objects and object parts. Though humans vary in cognition, these human-labeled edges for the same image share high consistency. For each image, the paper average all the ground truth to generate an edge probability map, which ranges from 0 to 1. Here, 0 means no annotator labeled at this pixel, and 1 means all annotators have labeled at this pixel. Here, an hyper-parameter η is introduced. We consider the pixels with edge probability higher than η as positive samples and the pixels with edge probability equal to 0 as negative samples. Otherwise, if a pixel is marked by fewer than η of the annotators, this pixel may be semantically controversial to be an edge point, which is not calculated in the cost function. For each pixel, the cost function is given out as the following:

$$l(X_i; W) = \begin{cases} \alpha \cdot \log(1 - P(X_i; W)) & \text{if } y_i = 0 \\ 0 & \text{if } 0 < y_i \leq \eta \\ \beta \cdot \log P(X_i; W) & \text{otherwise,} \end{cases}$$

in which,

$$\alpha = \lambda \cdot \frac{|Y^+|}{|Y^+| + |Y^-|}$$

$$\beta = \frac{|Y^-|}{|Y^+| + |Y^-|}$$

$|Y^+|$ and $|Y^-|$ denote positive and negative sample set respectively. The hyper-parameter λ is introduced to balance positive and negative samples. The feature vector and ground truth edge probability at pixel i are presented by X_i and y_i respectively. $P(X)$ denotes the sigmoid function, and W denotes all the parameters in the network. Thus, the loss function can be formulated as

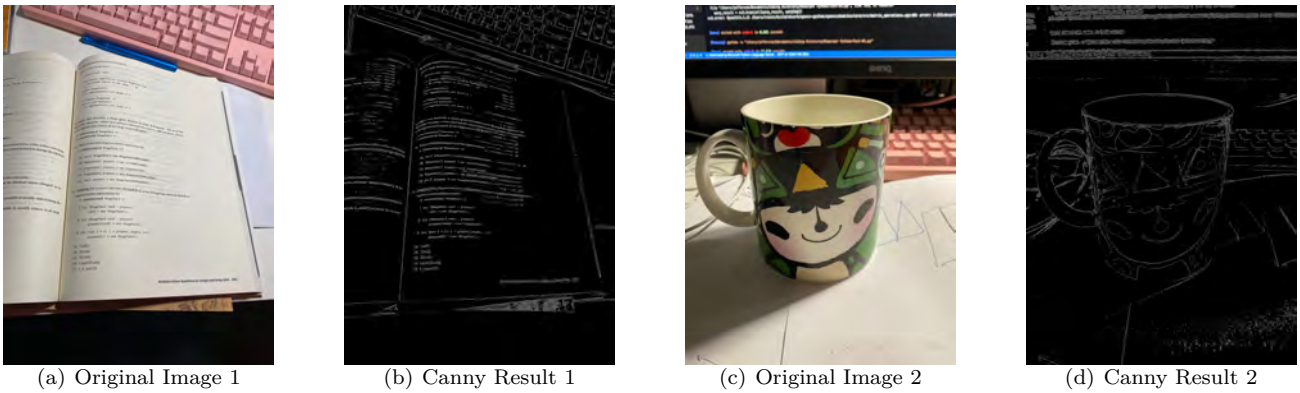
$$L(W) = \sum_{i=1}^{|I|} \Big(\sum_{k=1}^{K} l(X_i^{(k)}; W) + l(X_i^{\text{fuse}}; W) \Big)$$

in which $X_i^{(k)}$ is the feature vector of *stage* k, X_i^{fuse} is the feature vector of *stage fusion*, $|I|$ stands for the number of pixels in image I, and K is the number of stages (here is 5).

Fig. 6 shows the result of RCF Edge Detection. With such great performance, we can carry it to the next step of processing.

3.3. CDD: Semi-automatic Cylinder Edge Detection Model

According to the previous results from the experiments, we have already concluded that directly detecting cylinder edges can be very difficult (especially with the



(a) Original Image 1 (b) Canny Result 1 (c) Original Image 2 (d) Canny Result 2

Figure 4. Canny Edge Detection Results

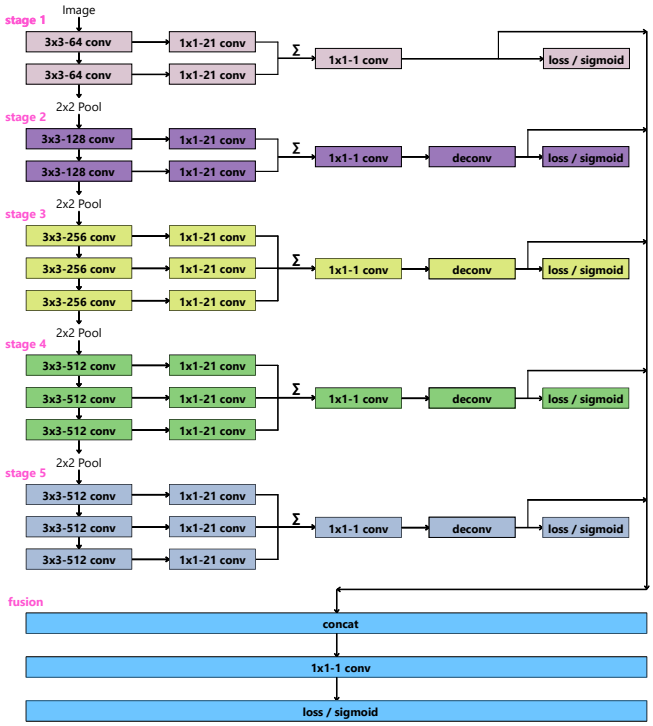


Figure 5. RCF Network Architecture, by Yun Liu *et al.* [14, 13]

restriction of light model size, as some instance segmentation models can indeed achieve this goal [1]). However, with the help of the RCF Network [14, 13] we mentioned in the last section, we proposed a semi-automatic way of edge detection with a relatively small model size. In our model, users are supposed to notate four vertexes, and the approximate edge rubbing. See Algorithm 1.

Fig. 7 shows the demo application I built on the iOS platform, using Objective-C, Swift, and C++. The implementation also used the OpenCV framework [18, 3], and the network part was implemented manually. The interface is user-friendly, and the whole process is swift and easy to implement.

3.4. CDD: Page Edge Detection Model

Pages are usually based on paper and printed text or graphics, so there will be a large area of background color. With this specialty, we can develop an automatic

Algorithm 1: Semi-automatic cylinder edge detection

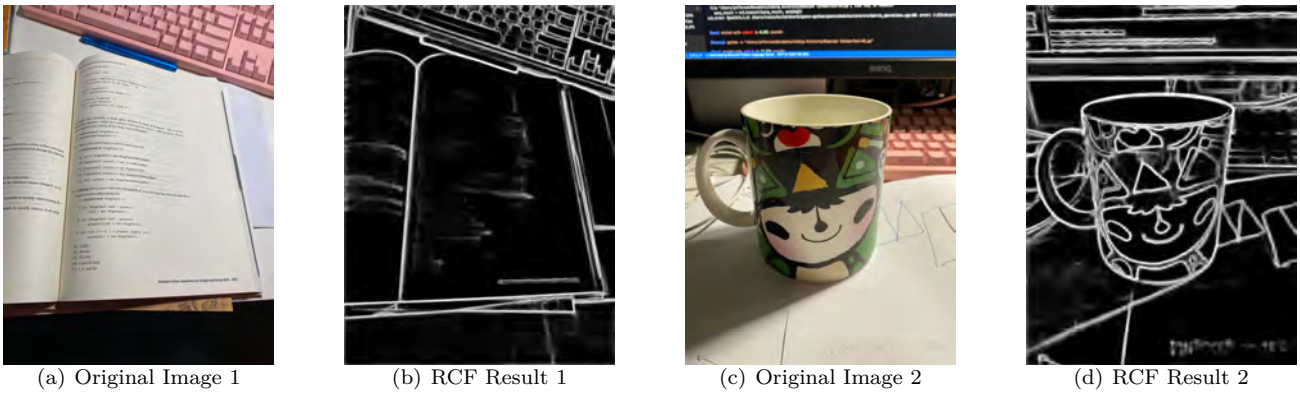
Input: 4 vertex coordinates, approximate edge rubbing
Output: expression of 4 edges

1 Use RCF Network to perform edge detection ;
2 Notate the 4 vertexes ;
3 Users are supposed to use input tools such as fingers, mouses or digital pencils to rub about in the edge upward and downward ;
4 Perform *and* (&) operation between RCF result and the part previously rubbed, serving as the mask layer ;
5 Perform polynomial regression to get the edge expression ;

edge detection model. In our model, images are first passed through the RCF network, then we will perform binaryzation (here thershold is 50, which is an empirical parameter) and dialation (kernel is a square, whose size is 5×5) process, and the process is iterated for five times. Now, we have already obtained relatively clear edges. Fig. 8 shows the results of the process.

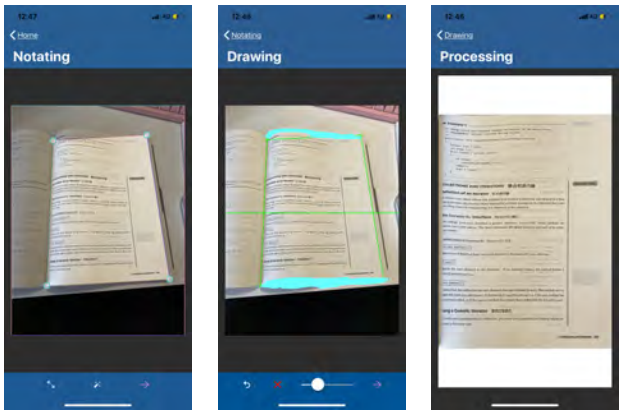
Now that the edge is clarified, we can start to extract the edges:

1. In order to locate the general area of edges, we might as well look for a large connected black block in the middle of the picture to locate the position of the page (we may either call the existing methods or simply implement it by BFS algorithm);
2. Search the edge pixels of the connected block;
3. Distinguish the outer edge and the inner edge, and the point set of the outer edge is reserved for the next operation;
4. Scan the point set row by row and column by column to determine which edge each pixel belongs to;
5. After classifying each outer edge pixel, find the nearest white pixel in the RCF result image as edge pixel;
6. After obtaining point set of each edge, perform polynomial regression or linear regression to get accruate edge expression.



(a) Original Image 1 (b) RCF Result 1 (c) Original Image 2 (d) RCF Result 2

Figure 6. RCF Result



(a) Notate vertexes (b) Rub edges (c) Output result

Figure 7. iOS Application Demo of Semi-automatic Cylinder Edge Detection Model

Fig. 9 shows the result of the model.

4. CDD: Cylinder Segmentation

Cylinder is a curved surface formed by the parallel movement of a straight line along a fixed curve. The moving line is called the straight generatrix of the cylinder, and the fixed curve is called the directrix of the cylinder. When we project a cylinder to a two dimensional surface, there would be two straight lines and two curves.

Before discussing about our segmentation model, here we first define the symbols involved. Suppose that after the original image is properly rotated, two curves can be expressed as two single valued function in the coordinate system of the rotated image (hereinafter referred to as *the coordinate system*): denote $F_U(x, y) = 0$ as the upper curve edge function, and $F_L(x, y) = 0$ as the lower curve edge function. Since pixels are discrete in images, the lower and upper curve edges are in fact point sets, and we express them as P_U and P_L respectively. Here, we also defined two other sets, X_U and X_L, which represents the set of horizontal coordinates of all pixels on the upper and lower edges.

$$P_U = \{(x, y)|(x, y) \in \text{upper edge}\}$$

$$P_L = \{(x, y)|(x, y) \in \text{lower edge}\}$$

$$X_U = \{x|x = P_U.x\}, X_L = \{x|x = P_L.x\}$$

a and b represent the start point and the end point of the curve edges, and notation $^{[u]}$ and $^{[l]}$ are used to distinguish upper and lower curveedges.

$$a^{[u]} = \min\{X_U\}, a^{[l]} = \min\{X_L\}$$

$$b^{[u]} = \max\{Y_U\}, b^{[l]} = \max\{Y_L\}$$

In order to simplify the model and improve the operation efficiency, we consider that the distributions of the upper and lower curve edges are approximately the same. $\forall x_i^{[l]} \in X_L$, $x_i^{[u]} \in X_U$, define the coordinates of *corresponding points*

$$x_i^{[u]*} = \frac{x_i^{[u]} - a^{[u]}}{b^{[u]} - a^{[u]}} \times (b^{[l]} - a^{[l]}) + a^{[l]}$$

$$x_i^{[l]*} = \frac{x_i^{[l]} - a^{[l]}}{b^{[l]} - a^{[l]}} \times (b^{[u]} - a^{[u]}) + a^{[u]}$$

from which, we are able to calculate the distance between the points and their corresponding points, $d_i^{[l]}$ and $d_i^{[u]}$. Because the sizes of the upper and lower point sets are different, in order not to confuse the notation, two sets of distances are defined here

$$d_i^{[l]} = \sqrt{(x^{[l]} - x^{[l]*})^2 + (y^{[l]} - y^{[l]*})^2}$$

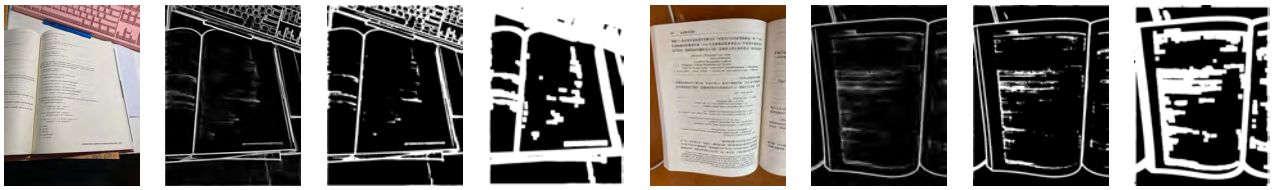
$$d_i^{[u]} = \sqrt{(x^{[u]} - x^{[u]*})^2 + (y^{[u]} - y^{[u]*})^2}$$

where $F_L(x^{[l]*}, y^{[l]*}) = 0, F_U(x^{[u]*}, y^{[u]*}) = 0$

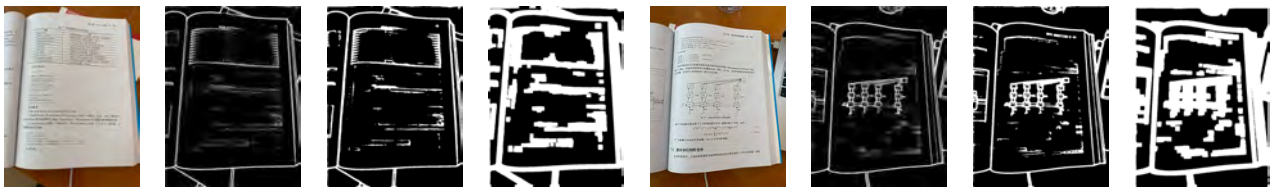
According to the perspective principle, if the shooting plane is parallel to the cylinder, the distance between the two points should be related to the reciprocal of the length on the image, so we can approximately estimate the *distance weight* W_i of each point relative to the shooting point by this method

$$W_i = \frac{\frac{1}{d_i}}{\sum_i \frac{1}{d_i}}, W(x_i) = \frac{\frac{1}{d_i|_{x=x_i}}}{\sum_i \frac{1}{d_i|_{x=x_i}}}$$

Using the weight obtained above, we may obtain the edge length by calculating the path integral of upper

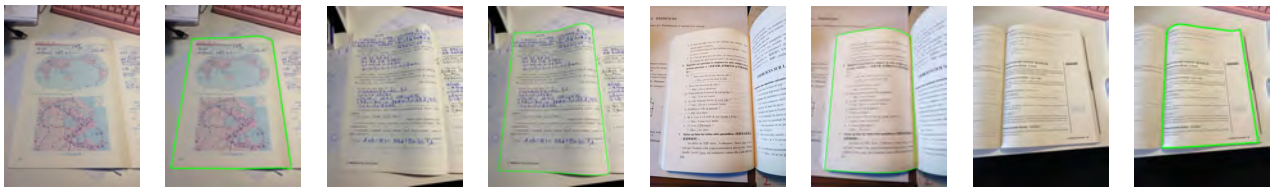


(a) Orig. 1 (b) RCF 1 (c) Bin. 1 (d) Dil. 1 (e) Orig. 2 (f) RCF 2 (g) Bin. 2 (h) Dil. 2



(i) Orig. 3 (j) RCF 3 (k) Bin. 3 (l) Dil. 3 (m) Orig. 4 (n) RCF 4 (o) Bin. 4 (p) Dil. 4

Figure 8. Binaryzation and Dilation Results



(a) Orig. 1 (b) Res. 1 (c) Orig. 2 (d) Res. 2 (e) Orig. 3 (f) Res. 3 (g) Orig. 4 (h) Res. 4

Figure 9. Page edge detection result

and lower curve edges

$$L_U = \int_{x=a^{[u]}}^{x=b^{[u]}} W(x)\mathrm{d}s$$
$$L_L = \int_{x=a^{[l]}}^{x=b^{[l]}} W(x)\mathrm{d}s$$

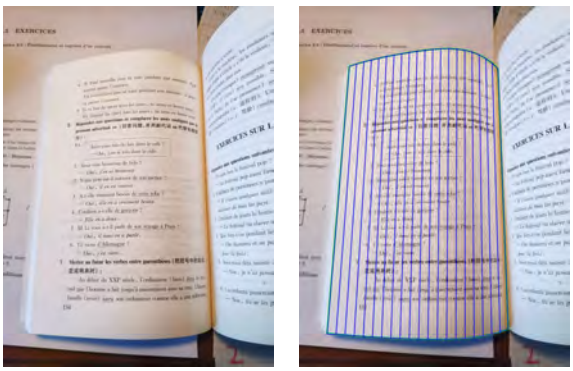
In the last section, we perform polynomial digression to obtain the curve edges. Since it is very inconvenient to calculate path integrals for functions of high degree, we may approximate the function by calculating Riemann sum.

$$\begin{aligned}\mathcal{L}_U &= \int_{x=a^{[u]}}^{x=b^{[u]}} W(x)\mathrm{d}s \\ &\approx \sum_i W(x_i)\sqrt{1+(y|_{x=x_i^{[u]}} - y|_{x=x_{i-1}^{[u]}})^2}\end{aligned}$$
$$\begin{aligned}\mathcal{L}_L &= \int_{x=a^{[l]}}^{x=b^{[l]}} W(x)\mathrm{d}s \\ &\approx \sum_i W(x_i)\sqrt{1+(y|_{x=x_i^{[l]}} - y|_{x=x_{i-1}^{[l]}})^2}\end{aligned}$$

Thus, the problem of cylinder segmentation is now converted to the problem of dividing the path integral to n equal parts.

Algorithm 2 shows the whole process of the algorithm. Note that the square here is the square of elements, and n can be understood as a hyper-parameter, which can be dynamically adjusted according to the result of restoration.

Fig. 10 shows the result of cylinder segmentation model.



(a) Orig. (b) Result

Figure 10. Cylinder Segmentation Result

5. Cylinder Dewarp

5.1. IAT (Image Affine Transformation)

In Euclidean geometry, an affine transformation, or an affinity, is a geometric transformation that preserves lines and parallelism (but not necessarily distances and angles). If X is the point set of an affine space, then every affine transformation on X can be represented as the composition of a linear transformation on X and a translation of X. [25]

For an image I, for a pixel $(x, y) \in I$, we introduce a *homogeneous coordinate* to cover translation: $\begin{bmatrix} x & y & 1 \end{bmatrix}^\top$. Denote the coordinate after transformation as $\begin{bmatrix} x' & y' & 1 \end{bmatrix}^\top$, and the affine transformation matrix as

Algorithm 2: CDD: Cylinder Segmentation

Input: $F_U, F_L, P_U, P_L, X_U, X_L, Y_U, Y_L$, #segmentation: n

Output: Segmentation Array

Calculate the corresponding coordinates ;

$X_U^* = \dfrac{X_U - a_U}{b_U - a_U} \times (b_L - a_L) + a_L$;

$X_L^* = \dfrac{X_L - a_L}{b_L - a_L} \times (b_U - a_U) + a_U$;

$D_U = \sqrt{(X_U - X_U^*)^2 + (Y_U - Y_U^*)^2}$;

$D_L = \sqrt{(X_L - X_L^*)^2 + (Y_L - Y_L^*)^2}$;

$W_U = \dfrac{1}{D_U} \Big/ \sum \dfrac{1}{D_U}$;

$W_L = \dfrac{1}{D_L} \Big/ \sum \dfrac{1}{D_L}$;

$L_U = \sum W_U \otimes \sqrt{1 + (Y_U[1:] - Y_U[:-1])^2}$;

$L_L = \sum W_L \otimes \sqrt{1 + (Y_L[1:] - Y_L[:-1])^2}$;

arc_length_u $= 0$, seg_pointer_u $= 1$, seg_array_u $= \{0\}$;

for $i = 0$ *to* $|P_U| - 1$ **do**

  arc_length_u = arc_length_u $+ L_U[i]$;

  if *arc_length_u* $\geq \dfrac{\textit{seg\_pointer\_u}}{n} \sum L_U$ **then**

    Apped $i + 1$ to seg_array_u ;

    seg_pointer_u = seg_pointer_u $+ 1$;

  end

end

Apped $|P_U|$ to seg_array_u ;

arc_length_l $= 0$, seg_pointer_l $= 1$, seg_array_l $= \{0\}$;

for $i = 0$ *to* $|P_L| - 1$ **do**

  arc_length_l = arc_length_l $+ L_L[i]$;

  if *arc_length_l* $\geq \dfrac{\textit{seg\_pointer\_l}}{n} \sum L_L$ **then**

    Apped $i + 1$ to seg_array_l ;

    seg_pointer_l = seg_pointer_l $+ 1$;

  end

end

Apped $|P_L|$ to seg_array_l ;

seg_array_u, seg_array_l ;

$\mathbf{M} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix}$, thus

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

which can also be broken into three steps: translation, shear, and rotation. There are six degrees of freedom, $(X, Y, s, \theta, \psi, \phi)$

$$\begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & X \\ 0 & 1 & Y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & \tan\phi & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ \tan\psi & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

In this transformation matrix, $\begin{bmatrix} a & d \end{bmatrix}^\top$ and $\begin{bmatrix} b & e \end{bmatrix}^\top$ are the new bases, $\begin{bmatrix} c & f \end{bmatrix}^\top$ is the new origin in the transformed coordinate system. In the OpenCV Library [3, 18], because its scaling is not necessarily *similarity transformation*, that is, its scaling may not be the same in the direction of two coordinate axes, so it may have higher degrees of freedom. In this passage, the affine transformation mentioned later all refers to the method in OpenCV Library.

5.2. IPT (Image Perspective Transformation)

Perspective transformation is a technique of projecting pictures to new planes, also known as projection, which is a three-dimensional coordinate transformation. In perspective transformation, the relationship of overlapping will remain unchanged, but the parallelism is not maintained.

For an image I, for a pixel $(x, y) \in I$, whose homogeneous coordinate is $\begin{bmatrix} x & y & 1 \end{bmatrix}^\top$. Denote the coordinate after transformation and the perspective transformation matrix as $\begin{bmatrix} x' & y' & 1 \end{bmatrix}^\top$ and $\mathbf{M} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$ respectively,

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} X/Z \\ Y/Z \\ Z/Z \end{bmatrix}$$

from which, we can calculate the coordinate expression after transformation,

$$x' = \frac{X}{Z} = \frac{a_{11}x + a_{12}y + a_{13}}{a_{31}x + a_{32}y + a_{33}}$$
$$y' = \frac{Y}{Z} = \frac{a_{21}x + a_{22}y + a_{23}}{a_{31}x + a_{32}y + a_{33}}.$$

It is not difficult to find that, the result will not be affected if the numerator and denominator are divided by a_{33}, so let us let $a_{33} = 1$. As a result, the transformation matrix $\mathbf{M}$ has 8 degrees of freedom, which can be obtained given 4 pairs of coordinates.

5.3. CDD: Cylinder Dewarp Model

Cylinder is a special kind of surface whose curvature is zero in a certain direction, which can help us simplify our model. Different from other researches such as [12], our model has no limitation on shooting-angle. This

section mainly describes the development process of our CDD Model.

Model 1 (Discrete IPT[1] Model)

Cylinder Segmentation
↓
IPT respectively
↓
concate

Model 2 (Discrete IAT[2] Model)

Cylinder Segmentation
↓
IAT respectively
↓
concate

Model 3 (Interpolated IPT Model)

Cylinder Segmentation
↓
Calculate IPT coordinates on seperation edges
↓
Perform interpolation for coordinates on the image
↓
Transform by pixel
↓
Interpolate again

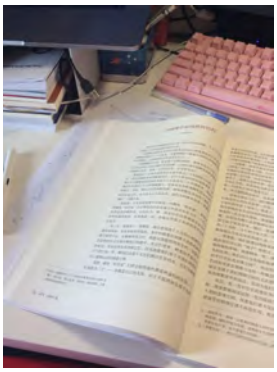
[1]Image Perspective Transformation
[2]Image Affine Transformation

5.3.1 Discrete IPT Model

Using the idea of infinitesimal method, we came up with the following idea: we can divide the picture into several blocks, then perform perspective transformation respectively, and finally concatenate them together. This method has the advantage of low computational requirement and easy implementation, which can be an ideal method. However, after experiments, problems occurred (see Fig. 11). Though the transformation of each block is performed perfectly, the image would be discontinuous after concatenation (see Fig. 11(b), 11(d)).

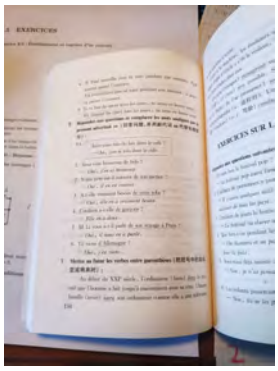
5.3.2 Discrete IAT Model

One of the goals of our model is to achieve the feature of shooting-angle-independent. Thus, we may as well analyze the pictures qualitatively and quantitatively. Since the unfolded surface of the cylinder is a rectangle, and the cylinder segementation process in the last section has already reduced the effect of perspective distortion

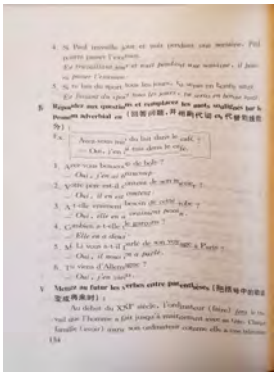


(a) Orig. 1

(b) Discrete IPT 1



(c) Orig. 2



(d) Discrete IPT 2

Figure 11. Discrete IPT Model Result

in one direction, we only have to consider the distortion in the other direction, which is perpendicular to the former one. Fig. 12 shows the diagram of a possible method. After the segmentation of the cylinder, we will first perform affine transformation for each block, which will transform them into a trapezoid, and then perform perspective transformation for the entire image, restoring it into the rectangular shape.

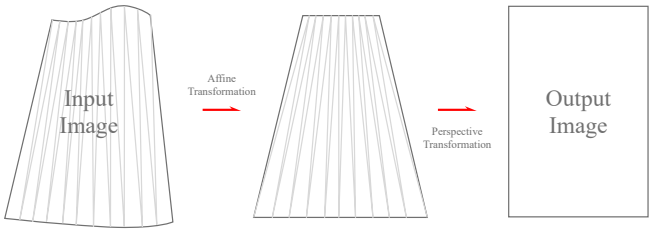


Figure 12. Discrete IAT Model Diagram

Fig. 13 is a sketch map for the model (though the upper and lower edges are straight in the diagram, this might not be true in most cases).

Tab. 1 shows the parameters and results of the model.

Fig. 14 shows the dewarping result of the model.

5.3.3 Interpolated IPT Model

According to Tab. 1, in terms of time, the performance of Discrete IAT model is not great enough, mainly because my implementation generated multi mask layers,

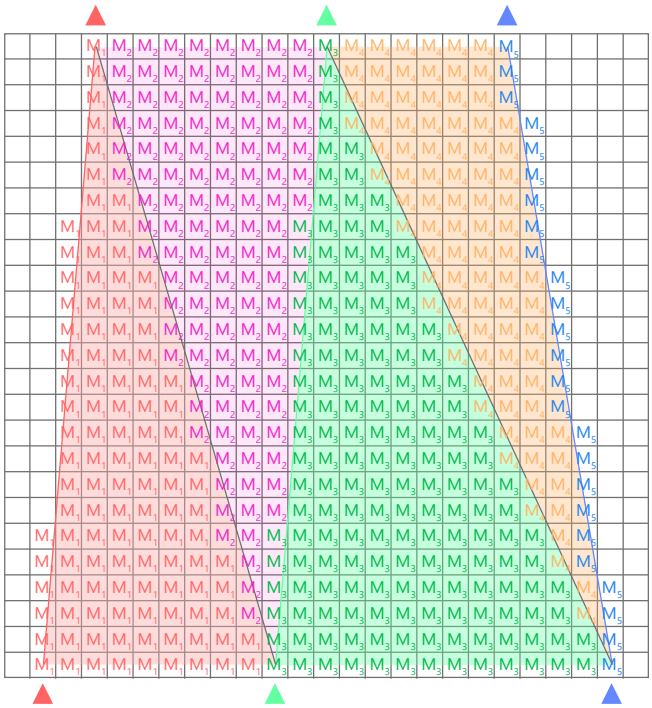


Figure 13. Discrete IAT Sketch Map

image size	3024×4032
output size	2100×2850
CPU	AMD Ryzen 9 3900X
multi-thread	False
avg. time (second)	22.6545
#segmentation	100

Table 1. Discrete Affine Transformation Model Result

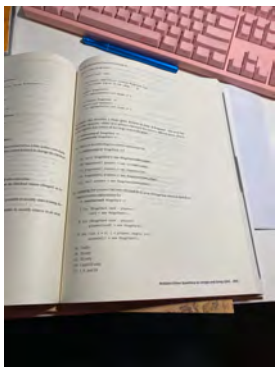
preventing image from overlapping. Also, we discovered that the wrinkles caused by affine transformation are difficult to refine, and the perspective distortion still had great impact on the results. Thus, a more effective method had to be propsed, which is the **Interpolated IPT Model**, see Algorithm 3.

Algorithm 3: Cylinder Dewarper: Interpolated IPT

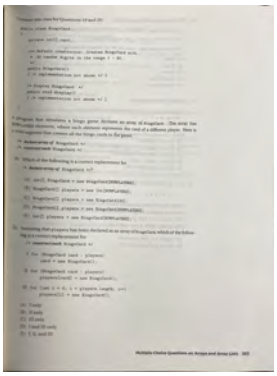
Input: Orig. Image, Cylinder Segmentation, Cylinder Edges
Output: Dewarped Image
1 Calculate IPT coordinates on seperation edges ;
2 Perform interpolation for coordinates on the image ;
3 Transform pixel by pixel ;
4 Interpolate the whole image ;

In the algorithm, we need to fill vectors size of $\mathbb{R}^2$ into each point of a line on the two-dimensional plane. For the implementation, the line can be found by **Bresenham Algorithm**, and the vector value can be filled on that basis.

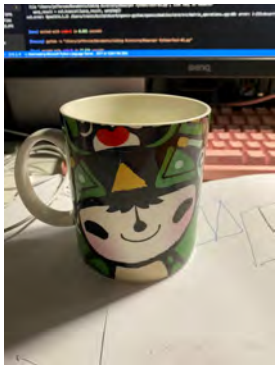
Fig. 15 shows a sketch of this model, in which



(a) Original Image 1



(b) Discrete IAT Result 1



(c) Original Result 2

(d) Discrete IAT Result 2

Figure 14. Discrete IAT Model Result

✓represents the pixels whose transformation coordinates are known, and question marks ? represents the pixels whose transformation coordinates are to be known by interpolation.

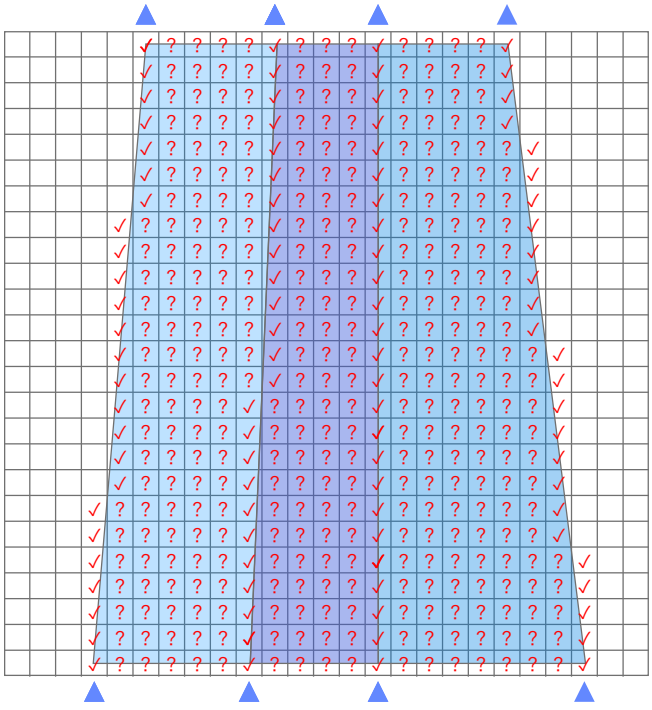
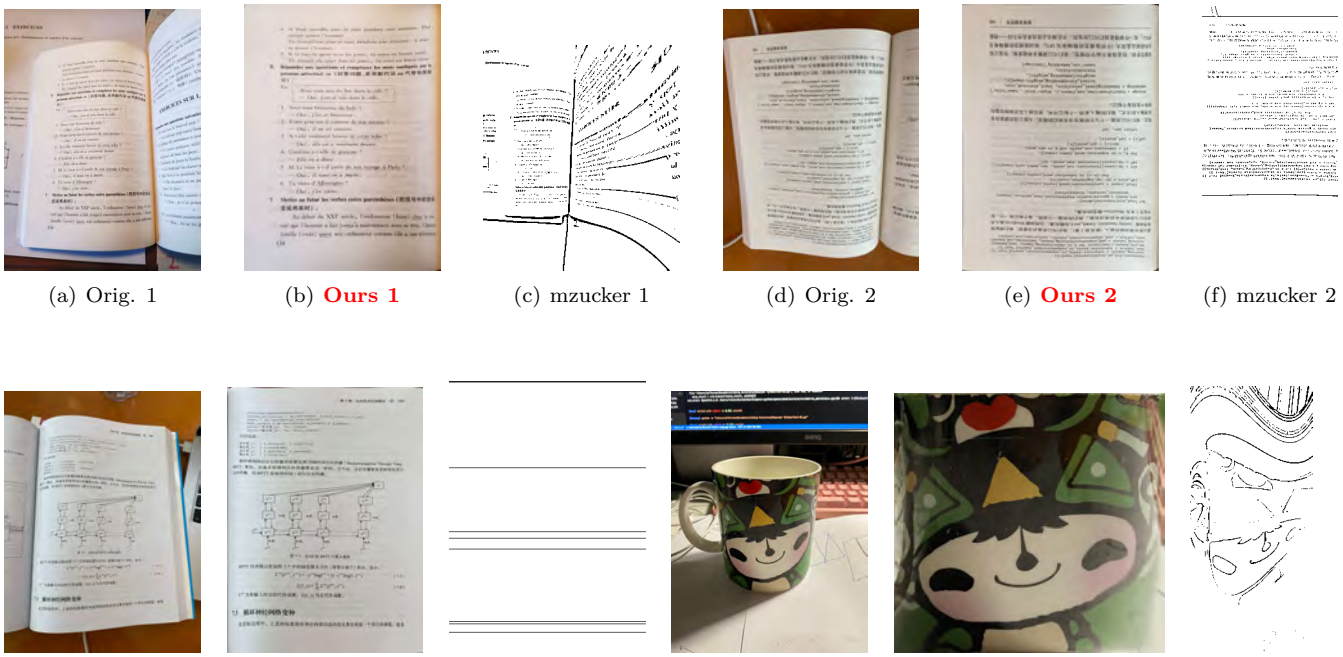


Figure 15. Interpolated IPT Model Diagram



(a) Orig. 1 (b) **Ours 1** (c) mzucker 1 (d) Orig. 2 (e) **Ours 2** (f) mzucker 2

(g) Orig. 3 (h) **Ours 3** (i) mzucker 3 (j) Orig. 4 (k) **Ours 4** (l) mzucker 4

Figure 16. Comparison between our model and mzucker's model

6. Experiment

Since we did not find any datasets that fit our problem, we only experimented with dozens of photos and tested the OCR accuracy before and after applying the model. Tab. 2 shows the OCR accuracy of CDD Model, in which "Mere IPT" means that the images are processed only by perspective transformation.

Table 2. OCR Result

	Original	Mere IPT *	Our Model
Accuracy	21.43%	38.64%	**84.97%**

* Image Perspective Transformation

Also, we compared our model with mzucker's model [16, 17] in some severe cases to test our features, shooting-angle-independent and content-independent. As his code was written in `python 2`, which is no longer supported now, we converted the code into the version of `python 3`, and updated the `opencv` library to the latest one. Results are shown in Fig. 16. We can see that our model achieved better performance in all these four cases. Compared to mzucker's model, our model have higher robustness towards environment (see the first and fourth group), and towards content (see second and third group, in which there are dense text lines and oblique lines). In addition, our model's timing cost is significantly shorter than mzucker's, see Tab. 3

Table 3. Timing Result

	mzucker's Model	Our Model
avg. time (s)	24.25	**3.79**

Though originally only designed to dewarp cylinder surfaces, our model does a surprisingly good job on document dewarp tasks. In this regard, we performed experiments on some document images, and compared our work with DocUNet [15], DewarpNet [23], and another gated U-Net network proposed last year [2]. While other models are specificly designed for document dewarping, our cylinder detection method cannot adapt all the cases in the DocUNet dataset. As a result, we only performed a small scale experiment to show the robustness and performance on the document dewarping problem, see Fig. 17, 18. We can see that our model outperforms the other models in some certain cases, and it has better stability.

Besides, Due to the high efficiency of our model, we also implemented the model on iOS platform (see Fig. 7). We used the MVC architecture to organized the app. Though the interface part was mainly written in Swift, we wrote the algorithm part in Objective-C and C++ to enhance the efficeincy. The project is completed open source. Here is the main page of the project, https://github.com/JeffersonQin/cdd-page-dewarp-ios

7. Conclusion

The rectification and restoration of curved surface is a very important work in image processing. It can help us to read better, make the OCR technology more accurate in recognizing the text on the image, promote the digitization of data, and improve the efficiency of production and work. In this regard, the current research generally depends on more information than images, including: depth of field, shooting angle, shooting focal length, object distance, text curves, and so on. In this paper, we proposed the CDD models with following features

- content-independent

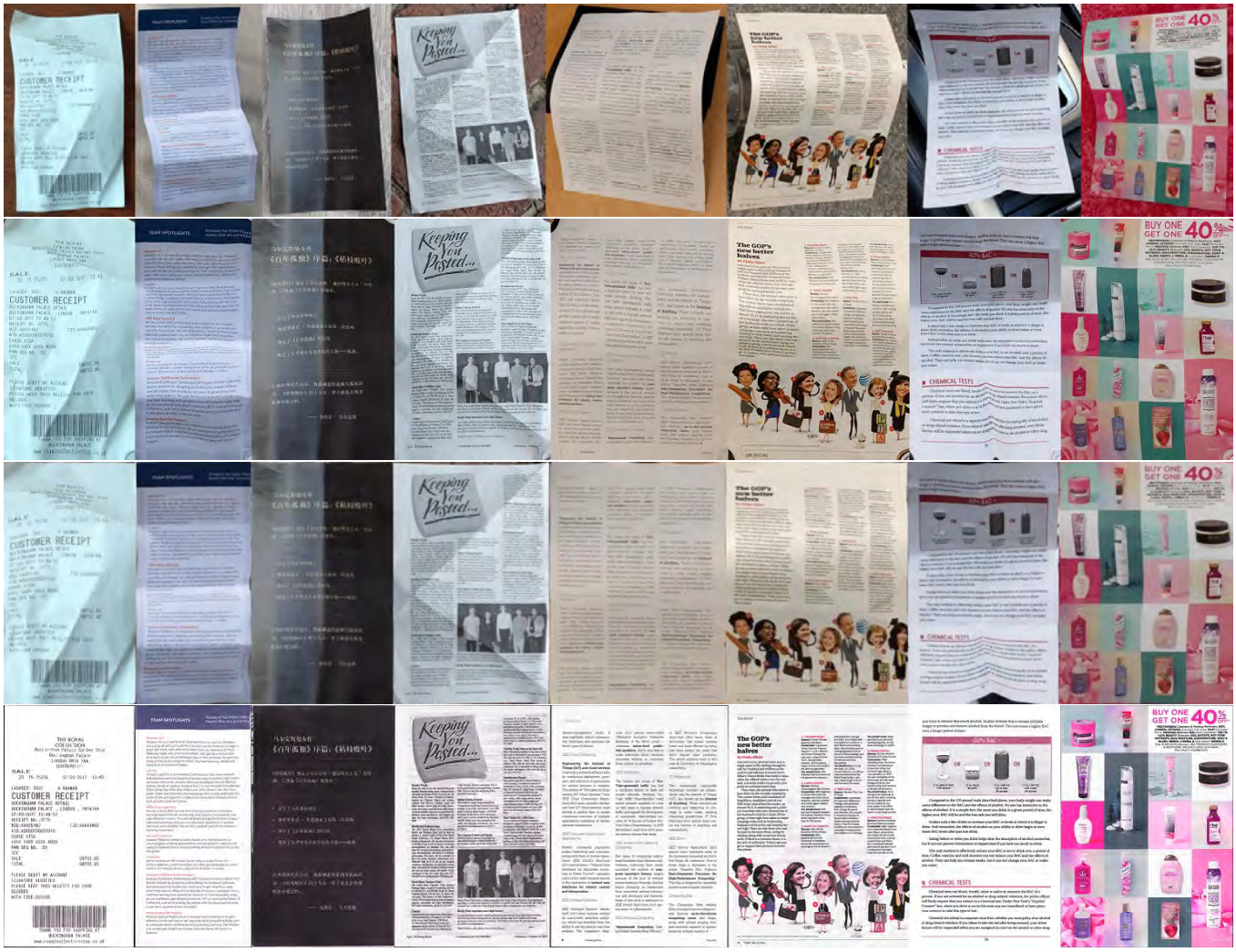


Figure 17. Comparison of document dewarping with DocUNet [15]. Rows from top to bottom: Input Images; DocUNet Results [15]; Our Results; Ground Truth

- shooting-angle-independent
- low computing power requirement.

This method can also be widely used in production, including non-contact shooting and restoration of curved cultural relics, defect detection of products in industry, etc.

In addition to cylinder dewarp, this paper also discussed the edge detection method of generalized cylinder and the automatic detection of page edge. Our model is based on feature extraction network, RCF [14, 13], which contributes to the high speed and light size. With these features, mobile applications can be easily implemented, since the number of parameters are significantly less than large networks such as YOLO [19, 21, 20], and SSD [11].

At present, our model still has certain requirements for the quality of the image, and can not eliminate the shadow caused by shooting, tricky shooting-angles may contribute to the failure of restoration. At the same time, due to the small number of datasets in for this problem, we are not able to carry out large-scale experiments. In the future, we will also use OCR, MS-SSIM and other indicators to measure and compare results of experiments as long as datasets are available.

References

[1] W. Abdulla. Mask r-cnn for object detection and instance segmentation on keras and tensorflow. https://github.com/matterport/Mask_RCNN, 2017.

[2] H. Bandyopadhyay, T. Dasgupta, N. Das, and M. Nasipuri. A gated and bifurcated stacked u-net module for document image dewarping, 2020.

[3] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[4] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698, 1986.

[5] M. P. Cutter and P. Chiu. Capture and dewarping of page spreads with a handheld compact 3d camera. In *2012 10th IAPR International Workshop on Document Analysis Systems*, pages 205–209, 2012.

[6] S. Das, G. Mishra, A. Sudharshana, and R. Shilkrot. The common fold: Utilizing the four-fold to dewarp printed documents from a single image. In *Proceedings of the 2017 ACM Symposium on Document Engineering*, DocEng ' 17, page 125–128, New York, NY, USA, 2017. Association for Computing Machinery.

Figure 18. Comparison of document dewarping. Rows from top to bottom: Input, DewarpNet Results [23]; Gated and bifurcated stacked u-net [2]; Our Results; Ground Truth

[7] R. Girshick. Fast r-cnn. In *International Conference on Computer Vision (ICCV)*, 2015.

[8] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Computer Vision and Pattern Recognition*, 2014.

[9] F. Guo, Y. Li, and P. Liu. A fast page outline detection and dewarping method based on iterative cut and adaptive coordinate transform. In *2019 International Conference on Document Analysis and Recognition Workshops (ICDARW)*, volume 4, pages 1–6, 2019.

[10] Y. Jia. Caffe: An open source convolutional architecture for fast feature embedding. http://caffe.berkeleyvision.org/, 2013.

[11] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In B. Leibe, J. Matas, N. Sebe, and M. Welling, editors, *Computer Vision – ECCV 2016*, pages 21–37, Cham, 2016. Springer International Publishing.

[12] Y. Liu. *Image processing method of curved pages and its application in digitization of ancient books*. PhD thesis, Capital Normal University.

[13] Y. Liu, M.-M. Cheng, X. Hu, J.-W. Bian, L. Zhang, X. Bai, and J. Tang. Richer convolutional features for edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(8):1939–1946, 2019.

[14] Y. Liu, M.-M. Cheng, X. Hu, K. Wang, and X. Bai. Richer convolutional features for edge detection. In *IEEE conference on computer vision and pattern recognition (CVPR)*, pages 3000–3009, 2017.

[15] K. Ma, Z. Shu, X. Bai, J. Wang, and D. Samaras. Docunet: Document image unwarping via a stacked u-net. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[16] mzucker. page dewarp. https://github.com/mzucker/page_dewarp, 2016.

[17] mzucker. Text page dewarping using a "cubic sheet" model. https://mzucker.github.io/2016/08/15/page-dewarping.html, 2016.

[18] OpenCV. Open source computer vision library, 2015.

[19] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[20] J. Redmon and A. Farhadi. Yolo9000: Better, faster, stronger. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[21] J. Redmon and A. Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[22] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. *arXiv preprint arXiv:1506.01497*, 2015.

[23] D. Sagnik, M. Ke, S. Zhixin, S. Dimitris, and S. Roy. Dewarpnet: Single-image document unwarping with stacked 3d and 2d regression networks. In *Proceedings of International Conference on Computer Vision*, 2019.

[24] A. Ulges, C. H. Lampert, and T. M. Breuel. Document image dewarping using robust estimation of curled text lines. In *Eighth International Conference on Document Analysis and Recognition (ICDAR'05)*, pages 1001–1005 Vol. 2, 2005.

[25] Wikipedia contributors. Affine transformation — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Affine_transformation&oldid=1011191615, 2021.

[26] S. "Xie and Z. Tu. Holistically-nested edge detection. In *Proceedings of IEEE International Conference on Computer Vision*, 2015.