**Handwriting character recognition based on LeNet-5**

**1. Pre-processing and character segmentation**

**Framework to mark the choice questions in the images of test paper**

**1.1 Pre-processing**

The most important step in this project was the preprocessing of the raw images, which involved different steps like filtering the red marks, image alignment, getting rid of irrelevant steps and segmentation to mention the least.

* + 1. **Filtering the Red Marks:**

The first and foremost step to recognize the characters was to get rid of irrelevant and irregular red marks. As these had no information and were a source of noise for the Optical Character recognition.

Method:

The red marks are removed by detecting the red pixels and then replacing them by nearby pixels or the white color.   
Consider if RGB represents red green and blue respectively in each pixel of the colored image. Pseudo code for detecting red marks:

*Go through each pixel and check   
If Red > (Green+20) or Red > (Blue+30)*

*then pixel is red*

*set red = white*

After removing all the red marks the image is converted from RGB to Greyscale.

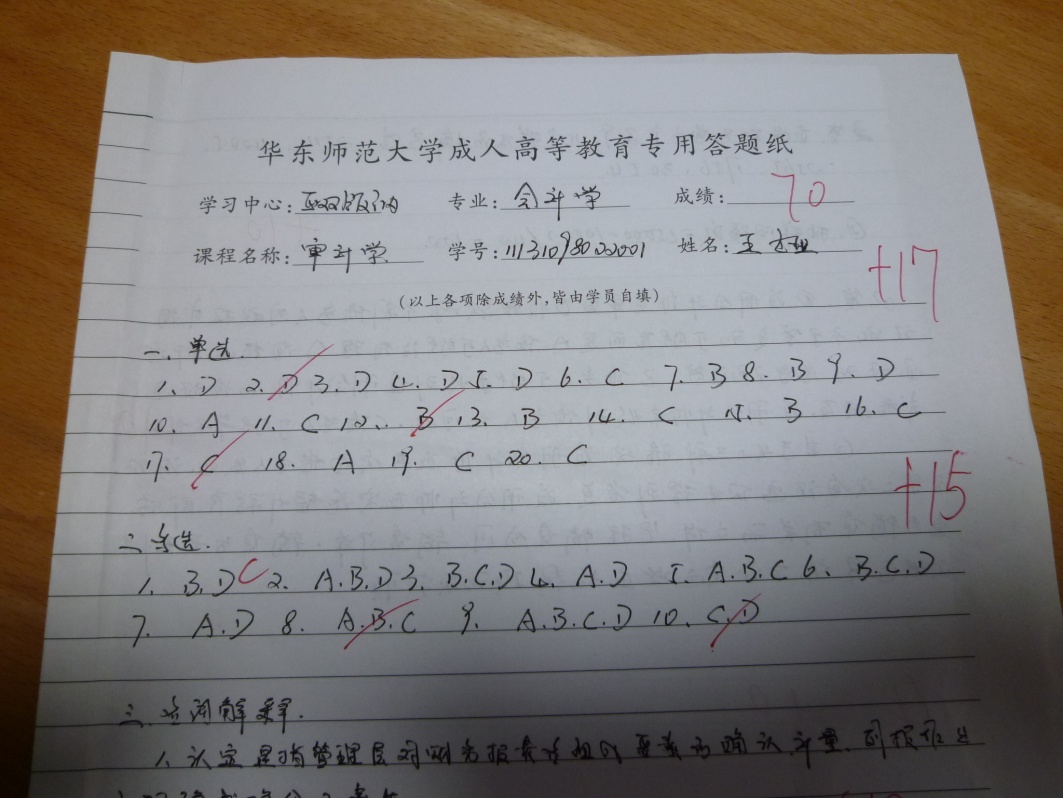


Fig. 1.1 Project Data image file sample with red marks

1.1.2 RGB to Grayscale:

To further the process the images, all the images were converted from RGB to Greyscale. As we are not concerned with the color in the images, our area of interest is the character. We can reduce the computing cost by converting all images into greyscale from RGB.   
  
Method:  
To achieve this, we simply used the built-in Matlab function “ *rgb2gray ”.* Fig. 2 shows an example of an RGB to grayscale converted image.

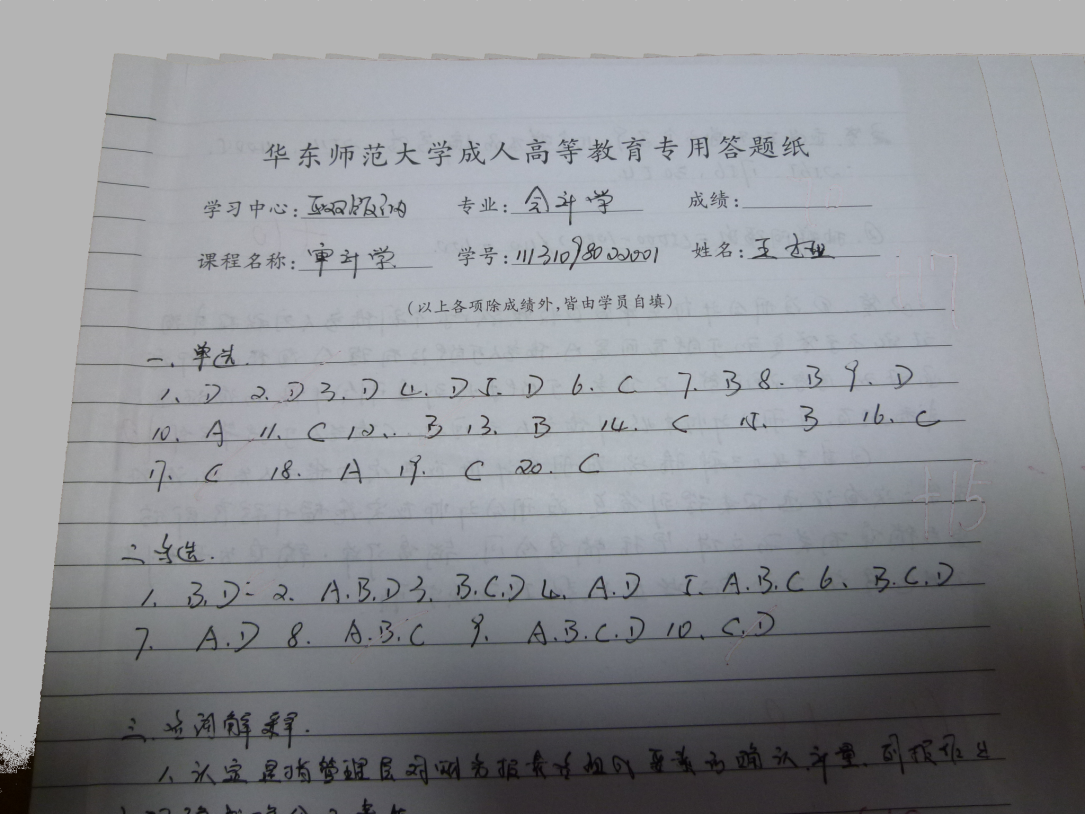


Fig. 1.2 Project Data image file sample with red marks filtered

1.1.3 Shadow Filtering using Histogram Equalization:

To further reduce the computing cost, we may convert greyscale images to binary images. As now we have only two levels of pixel intensity either black (0) or while (1). If we look at the greyscale image as shown in fig. 2 the grey scale intensity of the pixels is not uniform throughout the image. The upper region of the image in Fig 2 is bright while the lower region is dark. This may not allow us to use a global threshold value to convert greyscale images to binary images.

Method:  
To achieve uniform greyscale intensity throughout the image we applied “*histogram equalization*” function. Histogram Equalization function uses locally adaptive image threshold rather than global threshold. Locally adaptive image threshold is chosen using local first-order image statistics around each pixel. Fig 4 and Fig 5 show the effect of Histogram Equalization while converting the greyscale images, having non uniform greyscale pixel intensity, to the binary images.

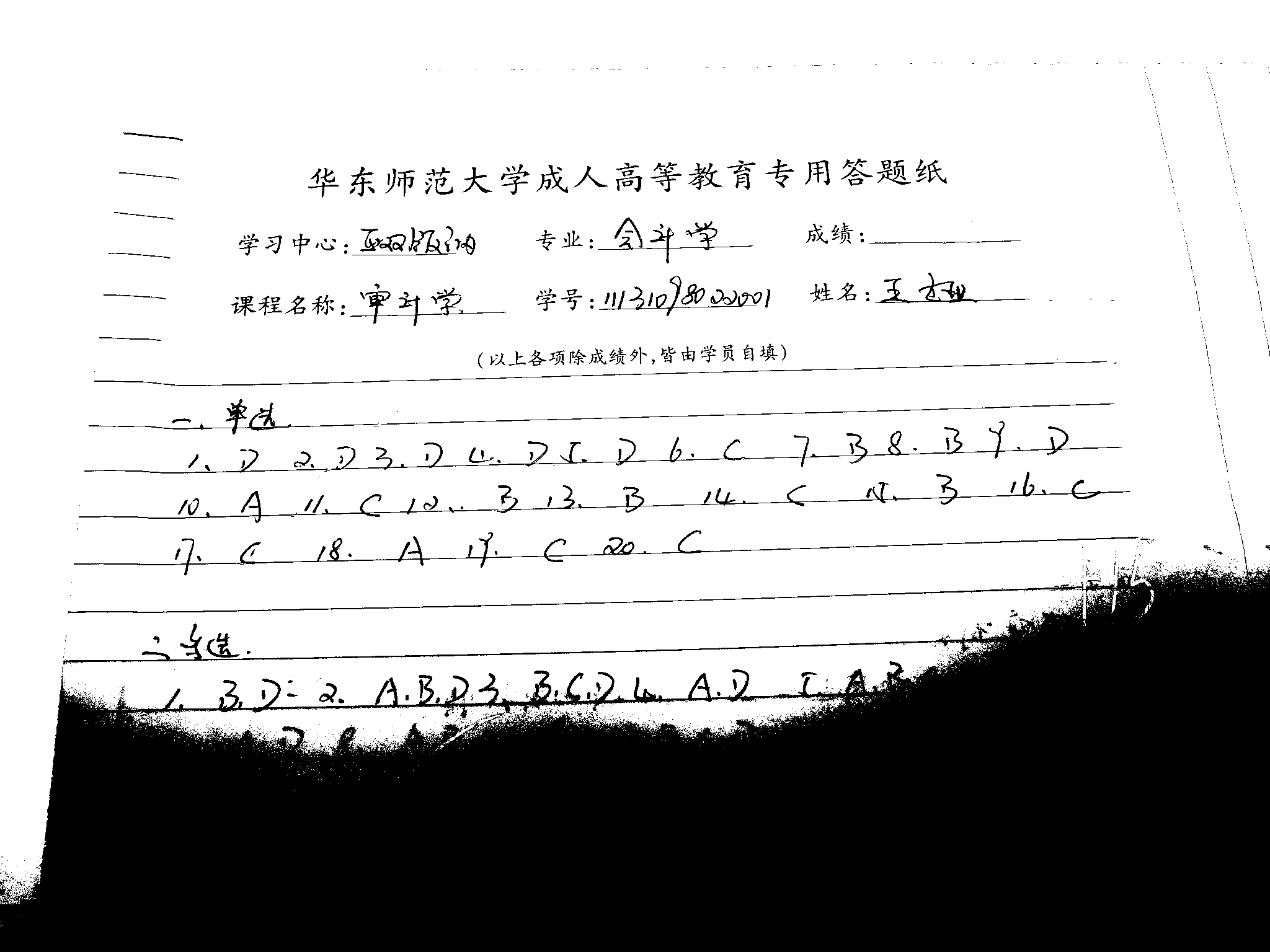


Fig. 1.3 Binary Image without removing the shadows

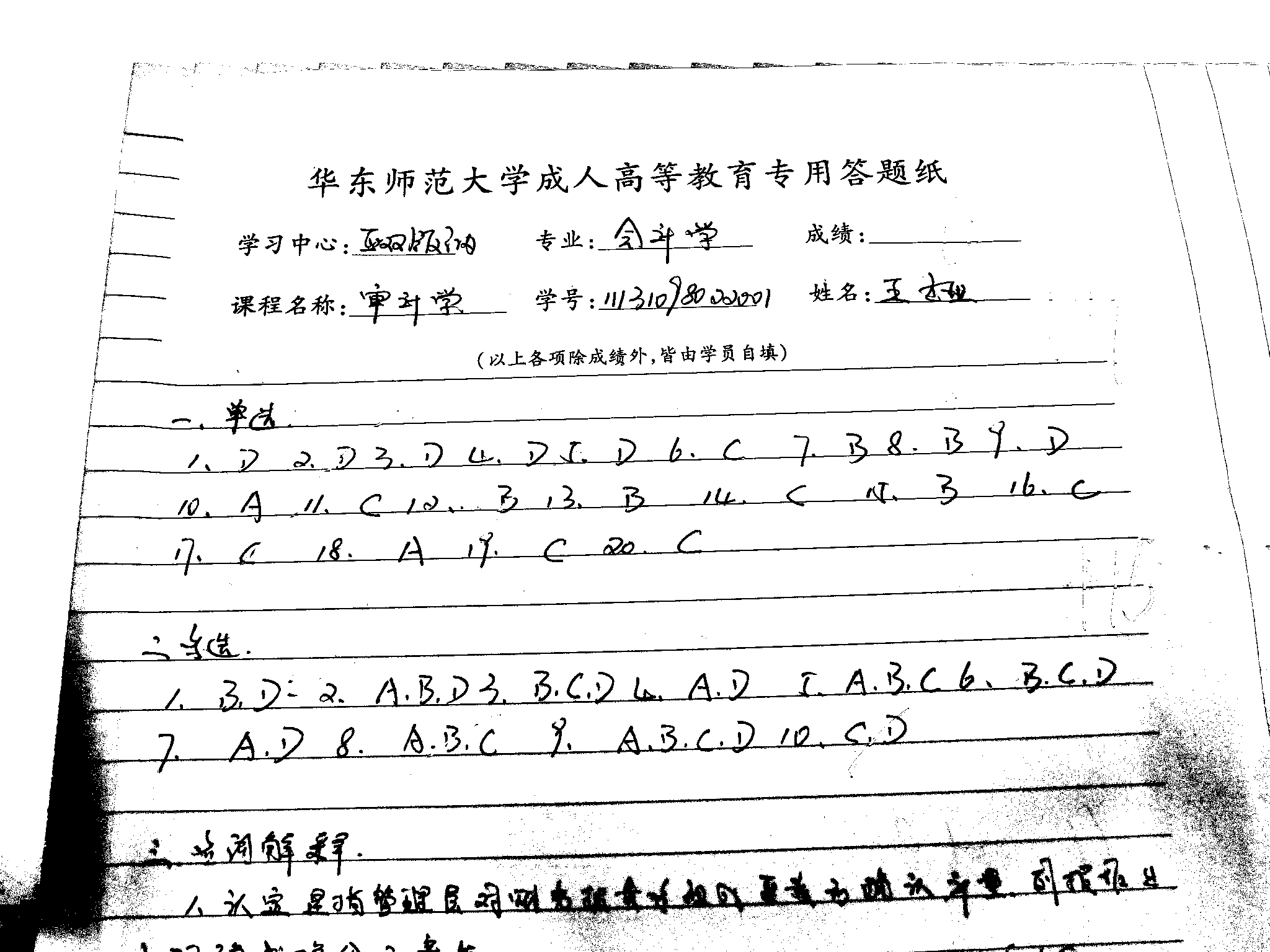


Fig. 1.4 Binary Image result after Histogram Equalization

1.1.4 GreyScale to Binary Image:

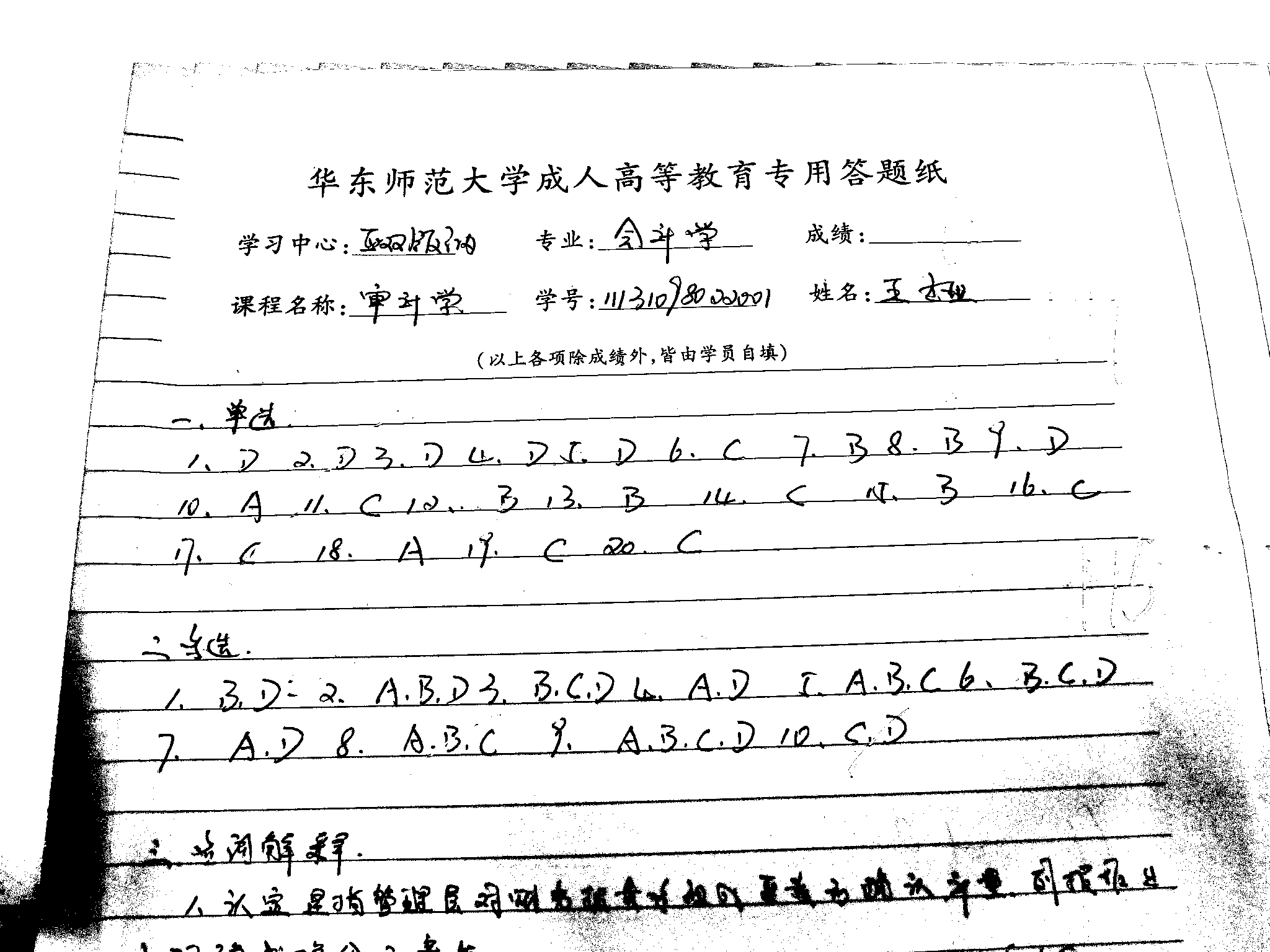
****

Fig. 1.5 Converted Binary Image

1.2 Segmentation

Segmentation is an image processing technique that helps to divide the image in to multiple segments so that we can get our area of interest. Further then, we applied the techniques of segmentation on all the binary images obtained.   
The process of segmentation was carried out in three steps:

* Locating the target area
* Removing the Lines
* Segment target characters of target choices

**Original Image**

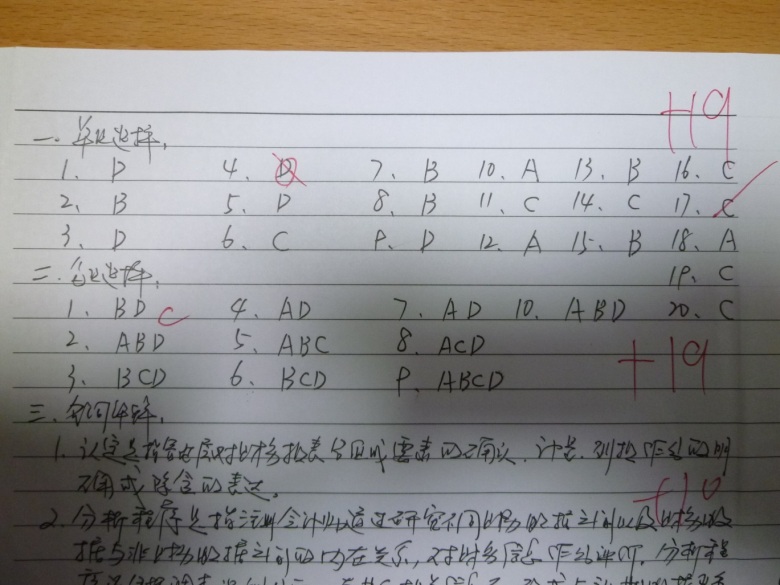


Fig. 1.6 Raw Image P1030503 from the data set

1.2.1 Locating the Target Area  
The foremost step for segmenting out the image was to locate the target area, since we have to mark single choice and multiple choice answers. We first segment our target area using the technique of horizontal projection of histogram. If we observe the images, we can clearly see that the first line is right above every single choice question. We may detect this first line to start our process of image segmentation. Later we detect the first blank line to mark the end of our segment for the single choice questions.

Method:  
The peaks in histogram projection tell us about the existence of a line. Intensity level between two lines tells us whether it’s a blank line or not. A blank line can tell us about end of our segment of single choice question. The same process is repeated for multiple choice questions. In this way we are able to segment out the two target areas from each image.

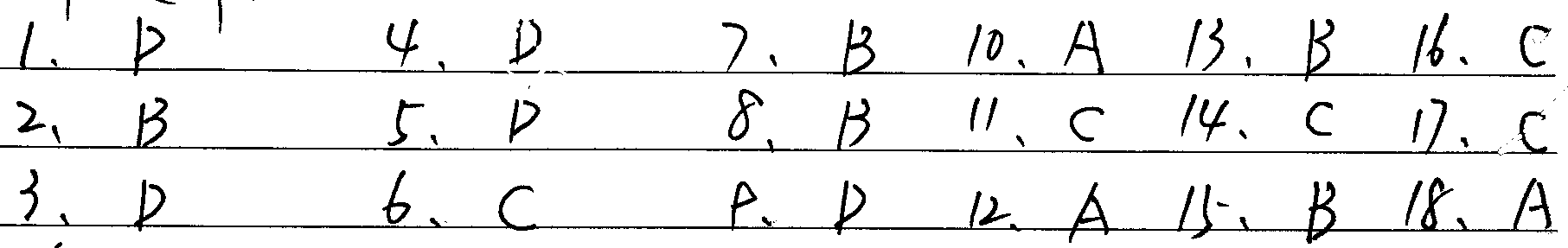


Fig. 1.7 The Target area segmented out of the raw image

1.2.2 Removing the Lines:  
  
Next, the target lines were located and were removed so that each character can be easily segmented out.

**Method:**

To remove the lines we consider no. of consecutive zeros in the rows of each image. If the no. of zeros is greater than a certain number (greater than approximate width of a single character) which cannot be part of a character, then it’s considered a line. And we replace those consecutive zeros of the binary image with the ones. This helps us to remove the lines to a greater extent.

Since not all of the lines are perfectly horizontal and may have some irregularities, we may have some leftover traces of the lines after performing the above technique. To remove these traces, we detect the no. of connected zeros (not necessarily horizontal) and replace them with ones if their connectivity is below certain number (less than the number of the connected pixels in the smallest character in the image).

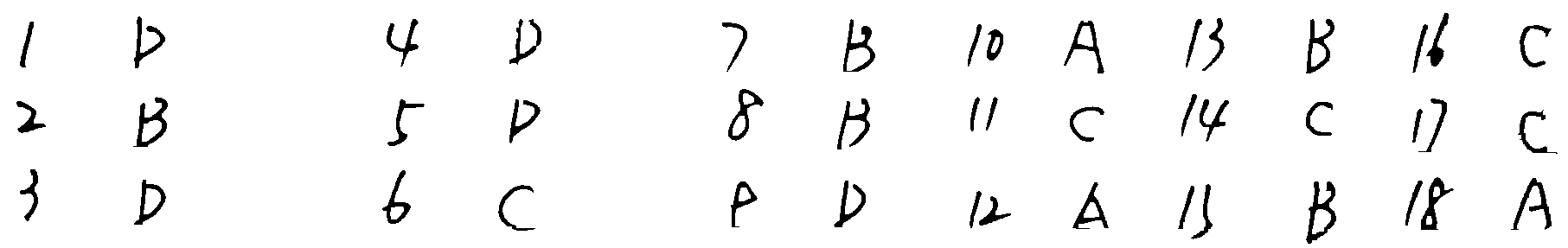


Fig. 1.8 Lines removed from the target area

* + 1. Segmenting the Target Characters:

The last step in the segmentation process was to segment out the targeted characters by performing the following two steps:

1. Cutting the segment horizontally

Using the low values of horizontal projection of histogram, we can detect the blank space between two lines of characters and segment the big one segment into segments containing only one line of characters as shown in fig. 1.9.

1. Cutting the segment vertically

Using the low values of Vertical projection of histogram of each single line character segment , we can detect the blank space between two characters and segment the line of characters into single characters as shown in fig. 1.9.



Fig. 1.9 Characters segmented out

Challenges in Preprocessing:

* Completely different perspective of photos
* Different arrangements in answers
* Different colors of handwritings
* Ambiguous characters, etc.

**Matlab Code until removing lines:**

|  |
| --- |
| close all;  for img=345:345  myimg=['E:\study material\SJTU\2nd Semester\DataMining\Projects\Project 2\Data\P1030',num2str(img),'.jpg'];  I=imread(myimg);  [ni,nj,nk]=size(I);  for i=1:ni  for j=1:nj  if(I(i,j,1)>(I(i,j,2)+20)|| I(i,j,1)>(I(i,j,3)+30))  I(i,j,:)=255\*ones(1,1,1);  end  end  end  grsc=rgb2gray(I);  BW = imbinarize(grsc,'adaptive','ForegroundPolarity','dark','Sensitivity',0.4);  BW = im2bw(BW);  [r,c] = size(BW);  se = zeros(1,35); %Structuring element for lines detection  sel = size(se,2); %structuring element length  for k = 1:r  for l = 1:c-sel  if (BW(k,l:l+sel-1)== se)  BW(k,l:l+sel-1)= 255\*ones(1,sel);  end  end  end  BW = 1-BW;  BW = bwareaopen(BW,60);  BWs = bwareaopen(BW,500,4);  BW = BW-BWs;  se2 = strel('cube',4); %Structuring element for dilation  BW1 = imdilate(BW,se2);%dilation to fill the gap if any lines cuts through a character  se3 = strel('cube',4); %Structuring element for erosion  BW = imerode(BW1,se3); %Undoing the effect of dilation  BW = 1-BW;  myconvimg=['E:\study material\SJTU\2nd Semester\DataMining\Projects\Project 2\Binary images segmented\',num2str(img),'.jpg'];  imwrite(BW,myconvimg)  clearvars -except img  end |

**Matlab Code for segmentation:**

|  |
| --- |
| clc;  clear all;  close all  BW = imread('E:\study material\SJTU\2nd Semester\DataMining\Projects\Project 2\Binary images segmented\416\_a.png');  BW = im2bw(BW);  figure;  imshow(BW);  I=im2bw(BW);  [h0,w0]=size(I);  name\_net=Pic\_name(1:end-4);  counter\_1=zeros(h0);  for i=1:1:h0  counter\_1(i)=0;  for j=1:1:w0  if I(i,j)==0  counter\_1(i)=counter\_1(i)+1;  end  end  end    j=1;  row = ones(h0);    for i=1:1:h0  if counter\_1(i)<3  j=j+1;  row(j)=i;  end  end  row(j+1)=h0;    num\_1=0;  for k=1:1:j  if row(k+1)-row(k)>40  num\_1=num\_1+1;  a=row(k);  b=row(k+1);  eval(['R\_' ,num2str(num\_1),'=I(a:b,1:w0);']);  end  end    for m=1:1:num\_1  R=eval(['R\_',num2str(m),';']);  [h1,w1]=size(R);  counter\_2=zeros(w1);  for y=1:1:w1  counter\_2(y)=0;  for x=1:1:h1  if R(x,y)==0  counter\_2(y)=counter\_2(y)+1;  end  end  end  col=zeros(w1);  n=1;  for y=1:1:w1  if counter\_2(y)==0  col(n)=y;  n=n+1;  end  end  col(n)=w1;  num\_2=0;  for s=2:1:n  if col(s)-col(s-1)>14  num\_2=num\_2+1;  a1=col(s-1);  b1=col(s);  str=[name\_net,'\_',int2str(m),'\_',int2str(num\_2)];  imwrite(imresize(R(1:h1,a1:b1),[40,40]),['C:\Users\Javed\Documents\MATLAB\',str,'.jpg']);  end  end  end |

**2 Recognition of handwritten characters**

**2.1 Structure of LeNet-5**

Since the aim of this project is to recognize handwritten alphabetic characters and number characters in given images, i.e, answer sheets, the algorithm of LeNet-5, a kind of convolutional neural network, which has been widely used to recognize handwritten checks for its high accuracy, might be useful for our work.

In terms of the structure, it is specialized for handwriting recognition with 7 layers of neural networks, as shown in figure 2.1.

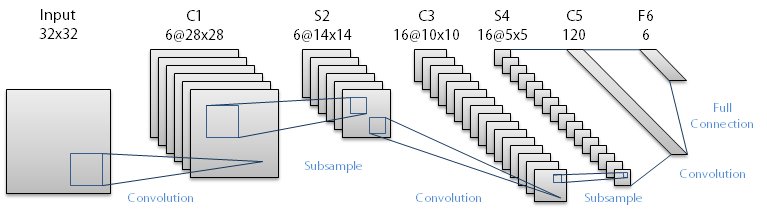


Fig. 2.1 Structure of LeNet-5

The first layer is the input layer. Images size as 32 by 32 pixels are input in the network from this layer. C1, C3 and C5 in the figure represent conventional layers, which can enhance the features of the input signal and decrease noises. S2 and S4 are down-sampled layers, which can remove some useless information and reserve important information. The output layer consists with Euclidean Radial Basis Function neurons.

The advantage of LeNet-5 is that it can extract useful features through the process of machine learning. The many times it learns, the better result can be got. What we have to do is just to prepare training sets and test sets for it. It can get the recognition results after training.

**2.2 Building data sets**

For the general sense of pattern recognition, the process requires both a training set and a test set for the implementation of LeNet-5. The available character database was provided by the tutor of Course Data Mining (see Fig. 2.2). In the database, the number of Character ‘A’ – Character ‘D’ was 55 respectively.

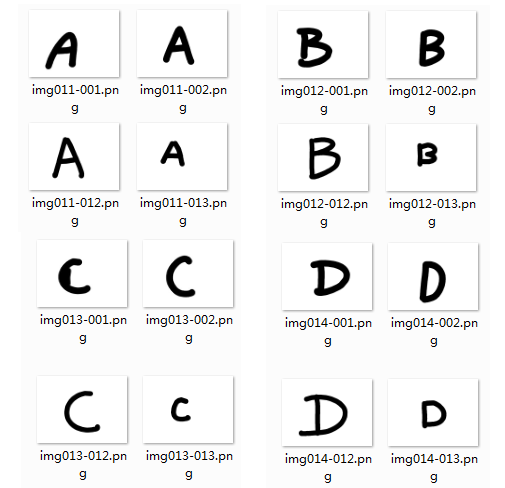


Fig. 2.2 Character database comprising Character ‘A’ – Character ‘D’

There were several steps to compile the training set and the test set.

1. Turn color images into gray-scale images;
2. Centering characters (see Fig. 2.3);
3. Normalize images into 32-by-32 pixels;
4. Separating dataset into a training dataset and a test dataset;

|  |  |  |  |
| --- | --- | --- | --- |
| D:\Curriculum\数据挖掘\PROJECT\Project 2\CODING\CLASSIFICATION\CHARACTERS\A\img011-001.png | D:\Curriculum\数据挖掘\PROJECT\Project 2\CODING\CLASSIFICATION\CHARACTERS\B\img012-017.png | D:\Curriculum\数据挖掘\PROJECT\Project 2\CODING\CLASSIFICATION\CHARACTERS\C\img013-001.png | D:\Curriculum\数据挖掘\PROJECT\Project 2\CODING\CLASSIFICATION\CHARACTERS\D\img014-033.png |
|  |  |  |  |
| D:\Curriculum\数据挖掘\PROJECT\Project 2\CODING\CLASSIFICATION\CHARACTERS\A_1\1.png | D:\Curriculum\数据挖掘\PROJECT\Project 2\CODING\CLASSIFICATION\CHARACTERS\B_1\17.png | D:\Curriculum\数据挖掘\PROJECT\Project 2\CODING\CLASSIFICATION\CHARACTERS\C_1\1.png | D:\Curriculum\数据挖掘\PROJECT\Project 2\CODING\CLASSIFICATION\CHARACTERS\D_1\33.png |
|  |  |  |  |

Fig. 2.3 Examples of centering characters by cropping

In practice, Step 2 were found to have exerted an important influence over recognition accuracy. If the position of a character was on the edge of the image, it could hardly be recognized correctly by CNN. Therefore, the position of each character was adjusted by cropping to the center after detecting the distance from the edge of each character to the edge of this image. Relevant source code was shown as below.

|  |
| --- |
| up=10000;down=0;left=10000;right=0;  Pic\_name=Pic\_list(i).name;  I=imread(strcat(file\_path,Pic\_name));  I=rgb2gray(I);  I=256-I;  for j=1:size(I,1)  for k=1:size(I,2)  if I(j,k)>30  if j<up;up=j;end  if j>down;down=j;end  if k<left;left=k;end  if k>right;right=k;end  end  end  end  I=I(up:down,left:right); |

Source codes are shown as follows (example of samples of Character ‘A’).

|  |
| --- |
| clear;clc;  char\_num=55;  char\_size=32;  A=zeros(char\_size,char\_size,char\_num);  file\_path='CHARACTERS\A\';  file\_path\_1='CHARACTERS\A\_1\';  Pic\_list=dir(strcat(file\_path,'\*.png'));  for i=1:char\_num  up=10000;down=0;left=10000;right=0;  Pic\_name=Pic\_list(i).name;  I=imread(strcat(file\_path,Pic\_name));  I=rgb2gray(I);  I=256-I;  for j=1:size(I,1)  for k=1:size(I,2)  if I(j,k)>30  if j<up;up=j;end  if j>down;down=j;end  if k<left;left=k;end  if k>right;right=k;end  end  end  end  I=I(up:down,left:right);  % I=imerode(I,strel('disk',20));  I=imresize(I,[char\_size,char\_size]);  mkdir(file\_path\_1);  imwrite(I,strcat(file\_path\_1,num2str(i),'.png'));  A(:,:,i)=I;  end |

For the overall dataset, we have 55 samples for each character, 30 of which were allocated for the training dataset. In other words, the test dataset encompassed 25\*4 = 100 of the rest samples of characters in total. Both datasets were packed up as follows. For each dataset, there were two variables, img\_X and lab\_X; Variable img\_X was a 32\*32\*N matrix for store of handwriting images while Variable lab\_X records the class of the N images, sequentially.

|  |
| --- |
| num\_train=30;  num\_test=char\_num-num\_train;    A=permute(A,[1,3,2]);  B=permute(B,[1,3,2]);  C=permute(C,[1,3,2]);  D=permute(D,[1,3,2]);    img\_train=[A(:,1:num\_train,:),B(:,1:num\_train,:),C(:,1:num\_train,:),D(:,1:num\_train,:)];  img\_train=permute(img\_train,[1,3,2]);  lab\_train=[zeros(1,num\_train)+1,zeros(1,num\_train)+2,zeros(1,num\_train)+3,zeros(1,num\_train)+4];  img\_test=[A(:,num\_train+1:end,:),B(:,num\_train+1:end,:),C(:,num\_train+1:end,:),D(:,num\_train+1:end,:)];  img\_test=permute(img\_test,[1,3,2]);  lab\_test=[zeros(1,num\_test)+1,zeros(1,num\_test)+2,zeros(1,num\_test)+3,zeros(1,num\_test)+4];    A=permute(A,[1,3,2]);  B=permute(B,[1,3,2]);  C=permute(C,[1,3,2]);  D=permute(D,[1,3,2]);    images=img\_train;  labels=lab\_train;  save('train\_data.mat','images','labels');  images=img\_test;  labels=lab\_test;  save('test\_data.mat','images','labels'); |

**2.3 Training & testing data**

To implement the algorithm of LeNet-5 for character recognition, we referred to the work of myCNN by Nikolay Chumerin, on whose homepage, a toolbox with a quantity of matlab implementation for convolutional neural network was generously provided (<http://sites.google.com/site/chumerin> ). In our work, the matlab implementation of LeNet-5 was enormously established based on this toolbox called myCNN.

The main body of the source code started with parameter setup, followed by the initiation of LeNet-5, the optimization of the network to best fit the training dataset and lastly the test of test dataset. The running process was based on the packed-up datasets built up as aforementioned. The source code of the main body was shown as below.

|  |
| --- |
| close all force  clear all force    classes\_to\_select = '1234'; % character classes  train\_samples\_per\_class = 30; % how many samples per class should be used for training  test\_samples\_per\_class = 25; % how many samples per class should be used for testing  inp\_rows = 32; % number of rows in training images  inp\_cols = inp\_rows; % number of columns in training images  n\_iterations = 30; % number of iterations of training algorithm  recom\_Hes\_it = 15; % recompute diagonal Hessian every recom\_Hes\_it iterations  show\_perf\_it = 1; % show performance plots every show\_perf\_it iterations [inf - don't show]  use\_LeCun\_eta = true; % use the eta sequence, proposed by LeCun  resize\_samples = false; % resize or pad the input images flag  samples\_coeff = 1; % rescaling coefficient for the input data  max\_subsamples = 500; % maximal number of samples for Hessian estimation  max\_samples = 60000; % maximal number of samples for training    n\_classes = numel(classes\_to\_select);  n\_train\_samples = n\_classes \* train\_samples\_per\_class;  n\_subsamples = min(min(max(ceil(n\_train\_samples\*1.0), 20), max\_subsamples), n\_train\_samples);    % set the path to MNIST dataset...  if ispc,  mnist\_dir = '/'; % Windows  else  mnist\_dir = '/'; % \*nix  end  % read and preprocess data for training from MNIST dataset  disp('Loading train data from MNIST')  [train\_samples train\_targets] = get\_MNIST\_data([mnist\_dir 'train\_data.mat'], inp\_rows, inp\_cols, ...  classes\_to\_select, train\_samples\_per\_class, resize\_samples, samples\_coeff);    % read and preprocess data for testing from MNIST dataset  disp('Loading test data from MNIST')  [test\_samples test\_targets] = get\_MNIST\_data([mnist\_dir 'test\_data.mat'], inp\_rows, inp\_cols, ...  classes\_to\_select, test\_samples\_per\_class, resize\_samples, samples\_coeff);    % create a new LeNet5 network  myLeNet5 = newMyLeNet5(inp\_rows, inp\_cols, n\_classes);    % show some crucial info  fprintf('Number of classes: %g\n', n\_classes)  fprintf('Train samples: %-6g (%gx%g)\n', train\_samples\_per\_class\*n\_classes, n\_classes, train\_samples\_per\_class)  fprintf('Test samples: %-6g (%gx%g)\n', test\_samples\_per\_class\*n\_classes, n\_classes, test\_samples\_per\_class)  fprintf('Number of subsamples for diagonal Hessian estimation: %g\n', n\_subsamples)    % train the network  [myLeNet5 E\_train MCR\_train E\_test MCR\_test] = train\_LM(myLeNet5, train\_samples, train\_targets, test\_samples, test\_targets, show\_perf\_it, ...  use\_LeCun\_eta, max\_subsamples, recom\_Hes\_it, n\_iterations);  save('myLeNet5\_data.mat','myLeNet5'); |

As the preliminary attempt, it was first tested on the given database offered by the tutor of Course of Data Mining, with 55 samples for each character respectively. The number of characters concerned was 4, i.e, Character A – Character D.

Figure 2.4 shows the change of root mean square error (RMSE) and that of misclassification rate of training dataset (with blue dotted line) and test dataset (with red dotted line) as the number of iteration increased. There was a stable convergence in value of RMSE of both the training dataset and the test dataset after Iteration 25 while the tendency line of misclassification rate of both dataset converged a mite earlier at around Iteration 20. The optimized network of LeNet-5 perfectly fit the training dataset with a misclassification rate of 0% at Iteration 30, i.e., all 120 characters in the training dataset were correctly fit, while the misclassification rate of the test data went to 2%, i.e., 2 in 100 tested handwritten characters were misrecognized.

Considering generalization, we applied the approach of 5-fold cross validation over method of LeNet-5 on the given database. All handwritten characters (N=55\*4=220) were evenly grouped into 5 folds, each of which included 11 samples of each class, i.e., 44 in total. For each time, one fold was chosen to become the test dataset while all other 4 folds were combined to be the training dataset. Repetitions were ended up until each fold alone has been tested as the test dataset. Results of the change of RMSE and misclassification rate of datasets in one situation were plotted in Fig. 2.5. The average of misclassification rate of test dataset was approximately 0.91%.

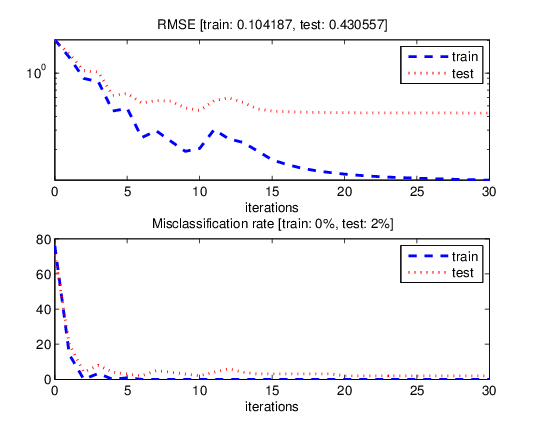


Fig. 2.4 Root mean square error (RMSE) and misclassification rate of training dataset (N=30, with blue dotted line) and test dataset (N=20, with red dotted line)

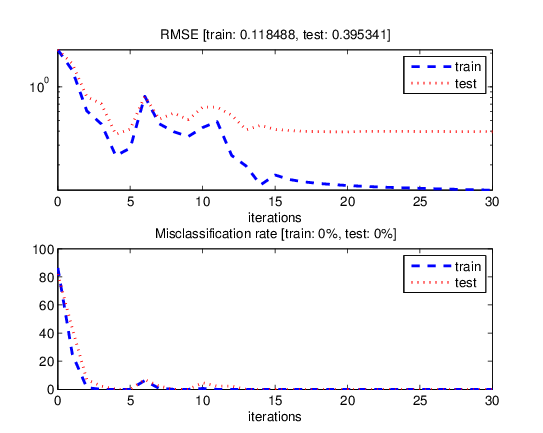


Fig. 2.5 Root mean square error (RMSE) and misclassification rate of training dataset (N=44, with blue dotted line) and test dataset (N=11, with red dotted line) in one situation of 5-fold cross validation

For simplicity in testing new database, a new version of source code was developed, which only required an extant optimized network of LeNet-5 (in .mat) form and an established folder with separated images (in .png form) of single handwritten characters to test. It would satisfy the situation with more than the aforementioned four characters (‘A’-‘D’) as long as the extant network of LeNet-5 has already been optimized based on the situation. However, due to lack of time, it was still not tested on the extracted characters from images of answer sheets.

|  |
| --- |
| clear;clc;    char\_num=4;  char\_size=32;  file\_path='CHARACTERS\test\';  labels=1:4;    Pic\_list=dir(strcat(file\_path,'\*.png'));  for i=1:char\_num  up=10000;down=0;left=10000;right=0;  Pic\_name=Pic\_list(i).name;  I=imread(strcat(file\_path,Pic\_name));  I=rgb2gray(I);  I=256-I;  for j=1:size(I,1)  for k=1:size(I,2)  if I(j,k)>30  if j<up;up=j;end  if j>down;down=j;end  if k<left;left=k;end  if k>right;right=k;end  end  end  end  I=I(up:down,left:right);  I=imresize(I,[char\_size,char\_size]);  images(:,:,i)=double(I);  end    save('tmp.mat','images','labels');  load('myLeNet5\_data.mat');  out\_lab=run\_test\_data('tmp.mat',myLeNet5);  % run\_test\_data.m  function out\_lab=run\_test\_data(mnist\_file,net)  load(mnist\_file);  for i=1:size(images,3)  img=images(:,:,i);  img=img-mean(img(:));  img\_std=std(img(:));  if img\_std>0  img=img/img\_std;  end  images(:,:,i)=img;  end  [~,out]=propagate(net,images);  maxima=double(repmat(max(out),[4,1])==out);  out\_lab=zeros(1,size(maxima,2));  for i=1:size(maxima,2)  [~,tmp]=max(maxima(:,i));  out\_lab(i)=tmp;  end  out\_lab=[labels;out\_lab];  disp(out\_lab);  end |

**2.4 Character recognition**

In this session, we use the LeNet-5 referred above to recognize the characters on the image, which are segmented into 50 by 50 pixels in chapter 1. As shown in figure 3.6.



Fig. 2.6 The results of the segmentation

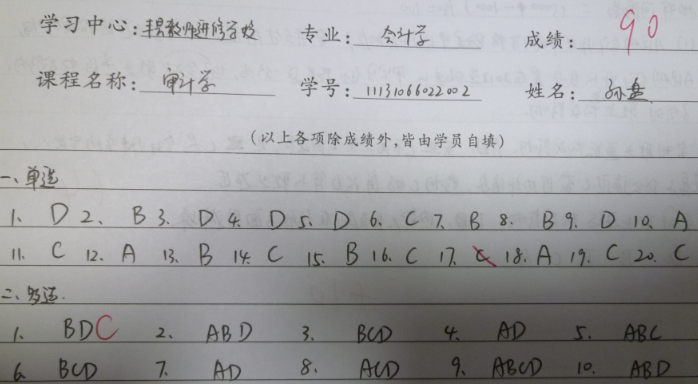


Fig. 2.7 Original image

In the program, the A, B, C and D are recorded as 1, 2, 3 and 4. The result of the recognition is show as following.

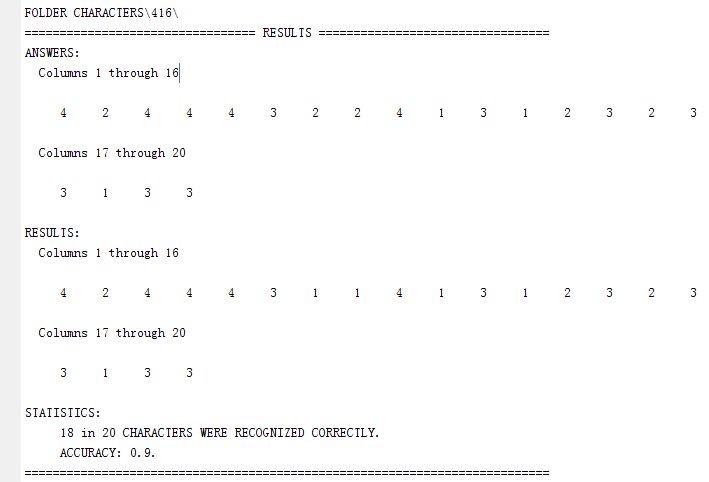


Fig. 2.8 Recognition result

In the fig. 2.8, you can notice that the accuracy is as high as 90 percent. It is reasonable to say that our method can be used to recognize the characters in the given images.

**References:**

OCR binarization and image pre-processing for searching historical documents

[Maya R. Gupta](http://www.sciencedirect.com/science/article/pii/S0031320306002202),, [Nathaniel P. Jacobson](http://www.sciencedirect.com/science/article/pii/S0031320306002202), [Eric K. Garcia](http://www.sciencedirect.com/science/article/pii/S0031320306002202)