Data Mining Project 2:

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

# Pre-processing

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

## 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.

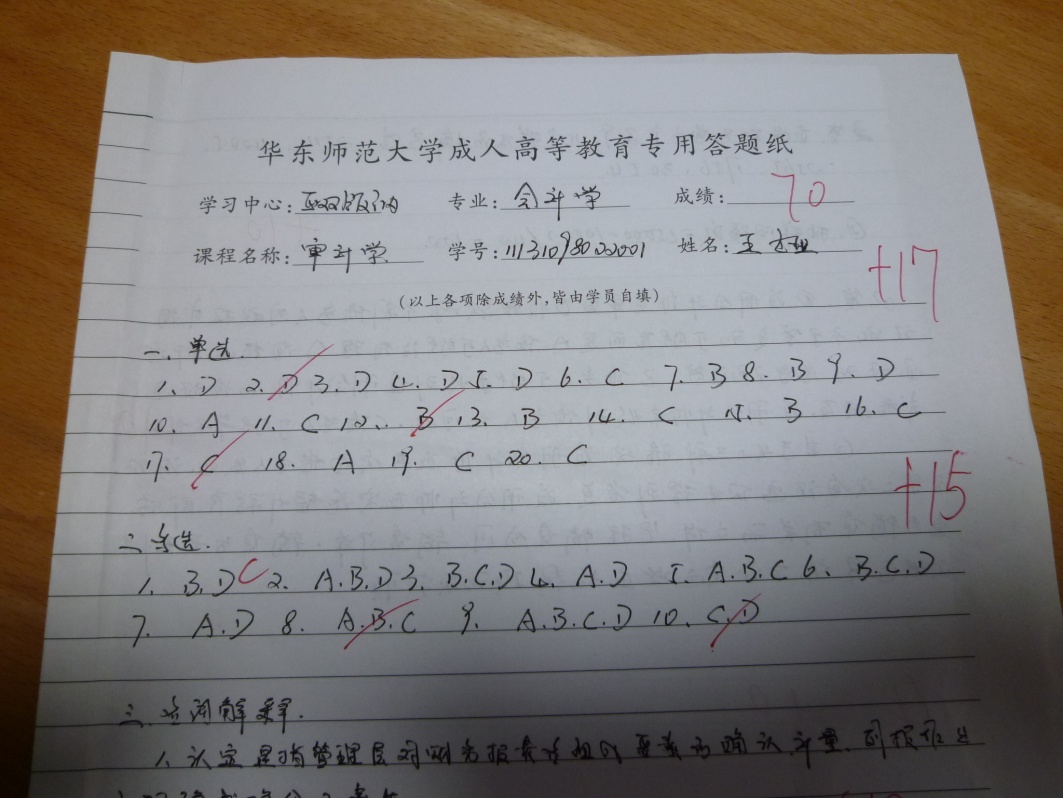


Figure 1. Project Data image file sample with red marks

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.

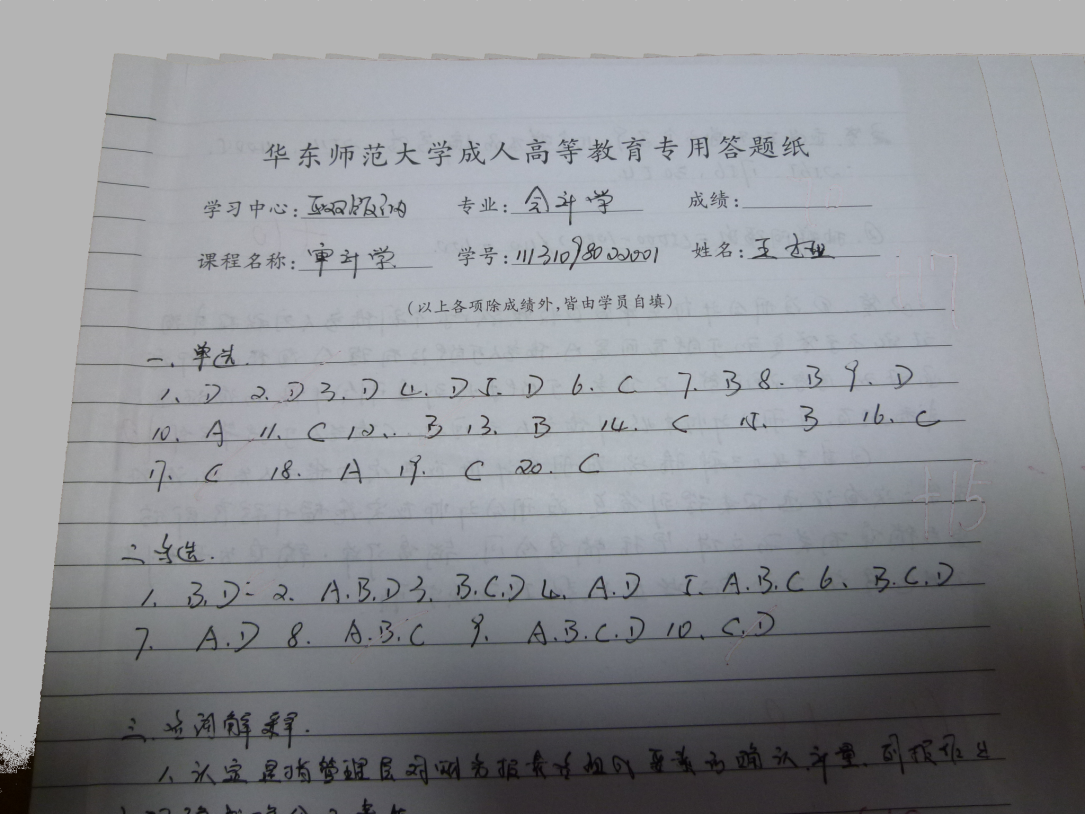


Figure 2. Project Data image file sample with red marks filtered

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.

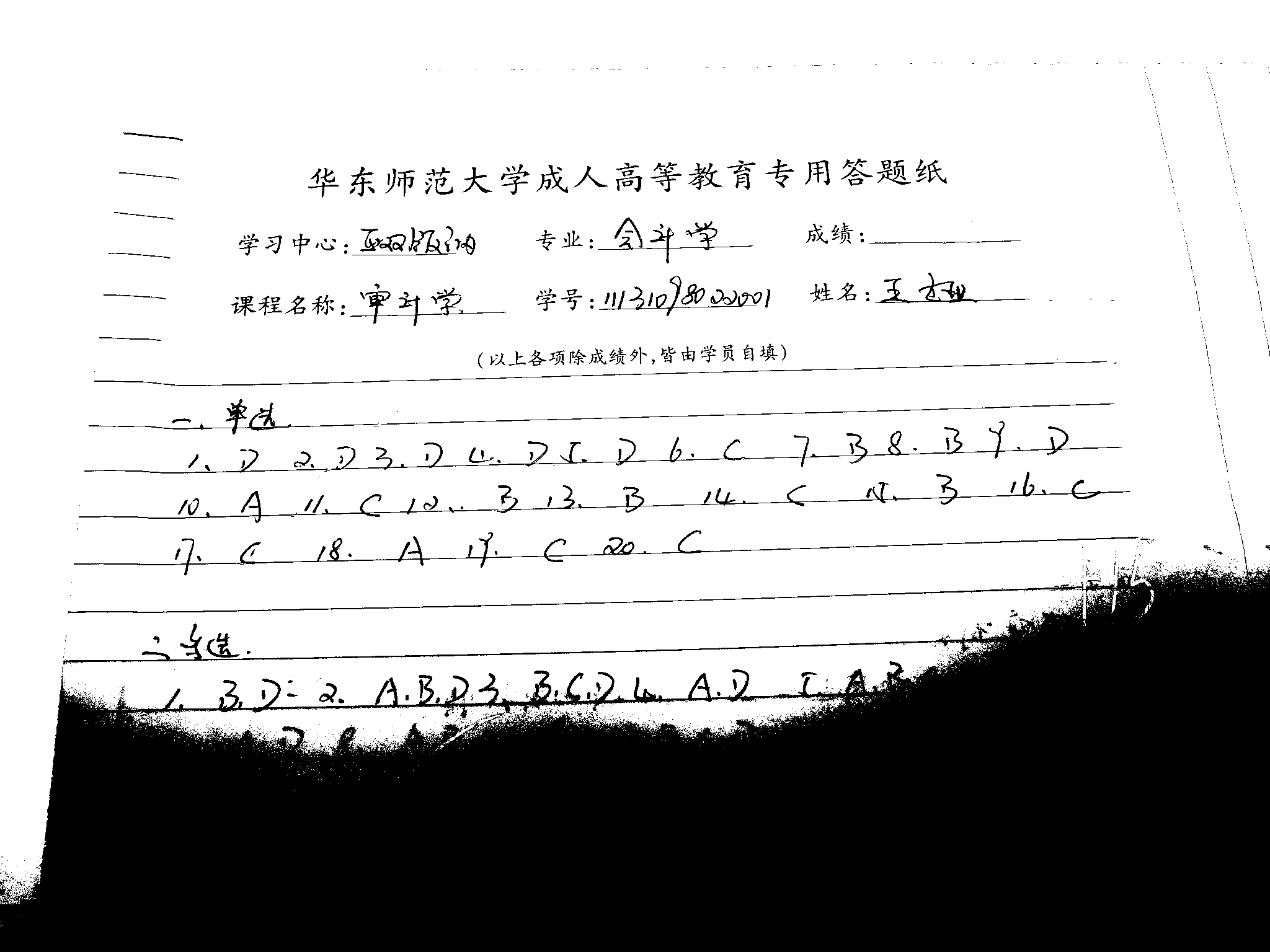


Figure 4. Binary Image without removing the shadows

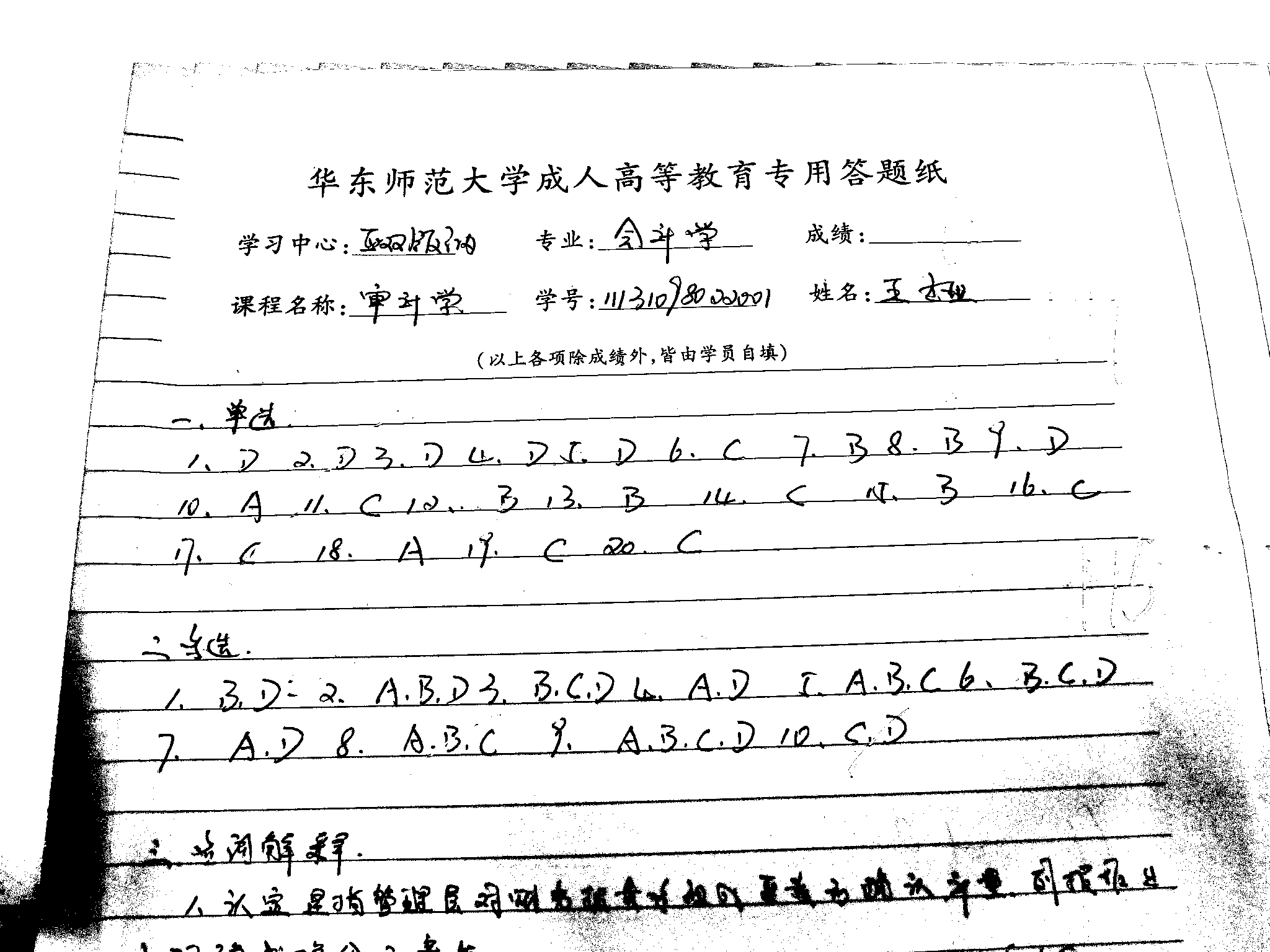


Figure 5. Binary Image result after Histogram Equalization

GreyScale to Binary Image:

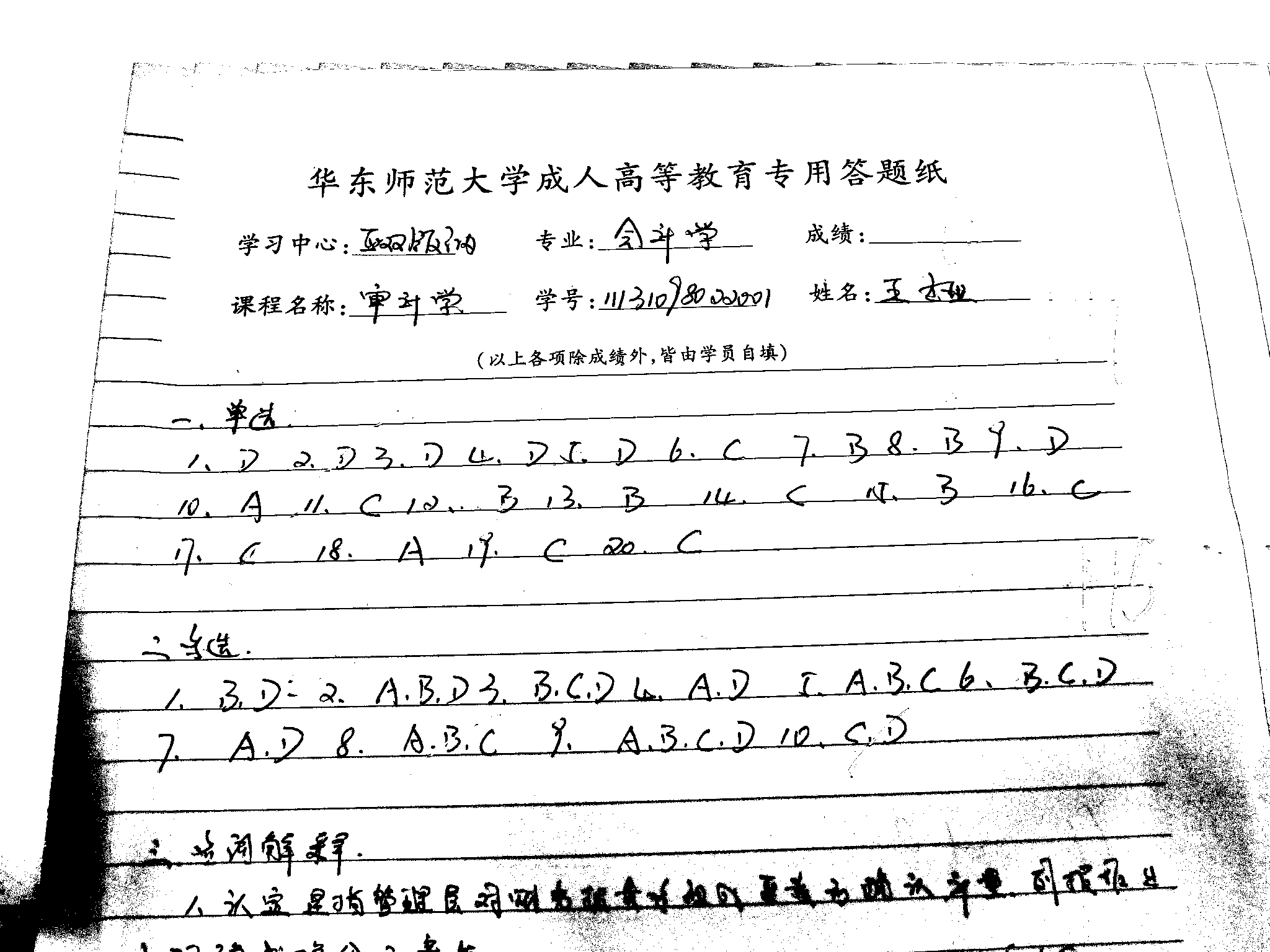
****

Figure 3. Converted Binary Image

## 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

Results

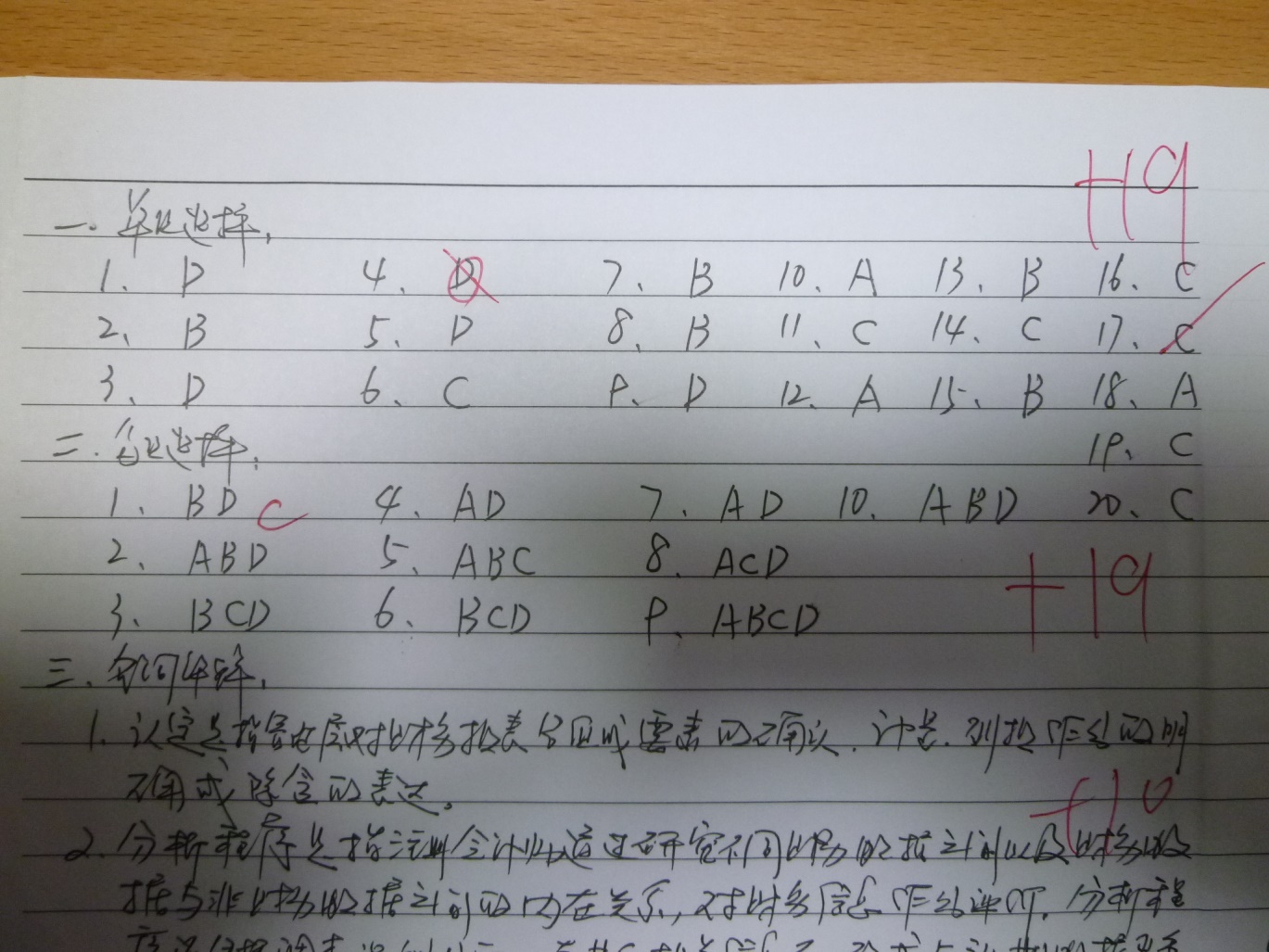
Original Image

Figure 6. Raw Image P1030503 from the data set

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.

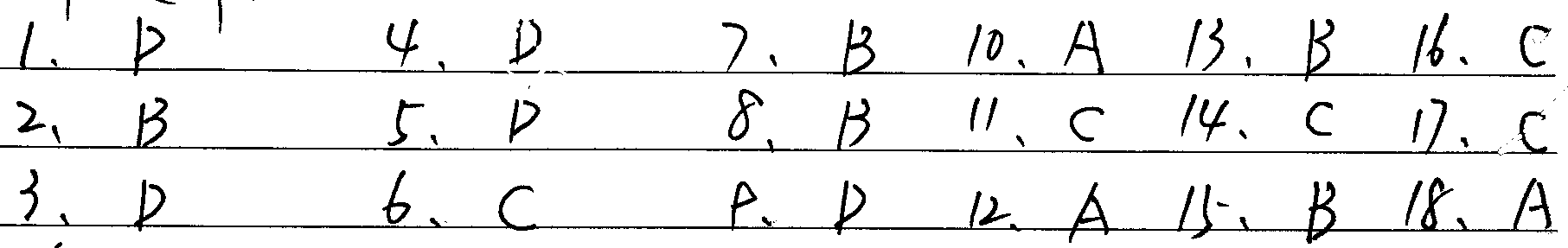


Figure 7. The Target area segmented out of the raw image

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).

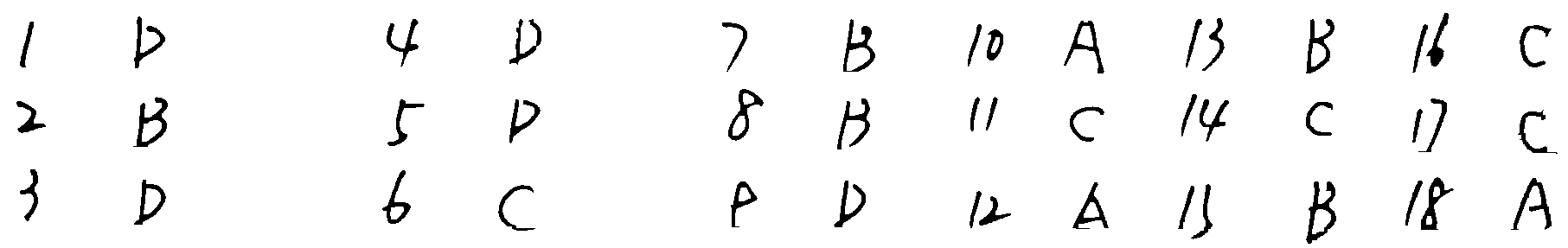


Figure 8. Lines removed from the target area

### Segmenting the Target Characters:

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

#### 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. 9

#### 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. 9

Figure 9. Characters segmented out

## Challenges in Preprocessing:

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

# Appendix:

## 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

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)