Chapter 6

Preprocessing

As mentioned in chapter 4, the first stage in the HCR pipeline is preprocessing of the image. We have seen in earlier chapters why this is very important and at the same time a difficult task for Indian languages. A scan of the image corpus that we have collected gives us the feel of the complexity of the problem in terms of the nature of noise present, distortion and writing variations. In this chapter, we discuss the conventional preprocessing pipeline usually used for character recognition and analyze its performance against our corpus. Based on this, we propose a new preprocessing pipeline and report on its performance.

The corpus considered for the experiments is large. The isolated character images generally show wide variability with respect to size, ink, and pen and may be noisy due to paper texture, scanning imperfections and writer faults. The different types of noises posed a major challenge to preprocessing and this stage of HCR evolved throughout our research work. We started with a static preprocessing pipeline based on models adopted in literature and tried different variations of methods, shuffled the stages, analyzed the effect of each stage, modified some methods and finally built dynamicity into the system that can clean the image based on its size and intensity distribution.

As the image file format affects the quality of the image, in order to understand the effect of the quality of the input image, we considered two different file formats. Depending on the file type, two separate preprocessing pipelines are designed and developed. Both pipelines have the same sequence of stages with a different processing method.
256 color bitmap format: the images lost some information as shown in the figure 6.1(a). The dark background becomes greenish and after conversion to gray image it becomes dark gray that may clash with the intensity of the foreground. Each pixel is represented in 8 bits as compared to 24 bits required in the 24 bit format.

24 bit bitmap format: High quality image as shown in the figure 6.1 (b), but requires 3 times more space than the above format. The dark background retains the original color and after conversion to gray image it becomes dark gray that may result in denser noise.

Figure 6.1 Image quality based on file formats

6.1 Static Preprocessing Pipeline

The objective of preprocessing is to make the input character image suitable for feature extraction. In most cases, the binary thinned character images with smooth and continuous contour is required for feature extraction. To make it location and size invariant, the image is further confined to a bounded box and size normalized. To obtain such images, most researchers follow static preprocessing (SP) pipeline with a fairly standard set of filters in some what same sequence like smoothing, sharpening, thresholding, noise cleaning, thinning, bounding box extraction, and resizing. We initially implemented one such static preprocessing pipeline as shown in figure 6.2.

Figure 6.2 Static Preprocessing pipeline with Global Thresholding
In our pipeline we added two important processing stages – initial resizing of the image to 100x100 and an additional 5x5 median filter applied to gray image to remove denser noise. Each of the stages in the pipeline is explained briefly below.

Image resizing: The input images have huge size variations (figure 5.4). As there are some features that depend on the size (Geometric moments, density of information, etc.), to study the effect of resizing of the image on the features and hence on the recognition results, we considered image size normalization. The actual size (length and width) of the character is known only after the background is eliminated and the thinned image is confined to a bounding box. In resizing, if a bounding box image size is bigger than the desired size then the image is shrunk and if the size is smaller than the desired size, then zoomed. Shrinking of an image was not a problem but, zooming may result in shape deformation and broken edges. This is because thinning removes the information in the image that can help for zooming. Hence, there is a need for initial resizing of the input color image along with the background. Hence the original image is first resized to 100x100 (decided based on the training sample images’ average background surrounding the contour of the character and the minimum size of an image) and then after bounded box extraction, the processed image is resized to 64x64 (decided based on minimal shape deformation of the character contour). Figure 6.3(a) shows the original image of size 54x79 with background and (b) shows the effect of using first resizing to 100x100 operations.

![Figure 6.3 Effect of static 100x100 resizing on the images](image)

Color to gray conversion: The scanned images are colored ones and are first converted to gray image. Characters are defined in a two dimensional plane with two states usually a white background with black trace of the character and hence the character images can be best represented as a binary image. Still we cannot convert the color or gray image directly to binary due to noise. We need to first enhance the quality of the original image by sharpening
the character shape edges and by smoothing the noise. To do so we need good tonal resolution for the gray image. Hence the color image is first converted to gray image.

5x5 Median filter: It is used to remove the denser salt and pepper noise from the gray image. Figure 6.4 (a) shows the gray image and (b) shows the effect of 5x5 median filter on the gray image. It is observed that the dense noise around a pixel denser with neighboring 1 to 12 noisy pixels gets smoothened. We use the same gray image to illustrate every stage preprocessing effect.

![Figure 6.4 Effect of median filter](image)

Smoothing and sharpening: Due to various reasons, the edges are normally rugged and need to be smoothed to get a smooth continuous contour with minimal directional changes after thinning. Smoothing will blur the edges and hence edge sharpening is necessary. We used Gaussian filter for smoothing and unsharp masking filter for sharpening. The effect of these filters is shown in figure 6.5 (a) and (b) respectively.

![Figure 6.5 Effect of smoothing and sharpening](image)

Global thresholding: The smoothed and sharpened gray image is converted to binary using global thresholding. We tried different global threshold values and realized that a global threshold does not work out due to ink and background intensity variations in an image. A low threshold value resulted in edge fragments in some cases but eliminated background noise and a high threshold eliminated fragments but the noise was left behind. It is observed that
such preprocessing systems eliminate noise, provided the input images are size constrained and have similar ink spread, background etc, with less variations. As we have images with both light or dark background and also the intensity of the pen trace varies from image to image and also within an image, it cannot be eliminated using this pipeline. The effect of global threshold with threshold value $T = 200$ is shown in figure 6.6.

![Figure 6.6 Effect of Global Thresholding](image)

**3x3 Median Filter:** The binarized image is again passed through a 3x3 median filter to remove salt and pepper noise. This eliminates the noise of 4 or less pixel density. If the contour is thin and has foreground pixels less than 4 in the neighborhood, then, in some cases, this operation produced fragmented edges. To handle the noise denser than 4 pixels, we tried to smooth the original image using a 5x5 median filter.

**Thinning:** As the samples are written without any pen restrictions, we have images with different contour thicknesses. In some cases it is thin and in some cases it is thick. As the contour thickness does not convey any useful information, it is treated as noise. To get a uniform contour thickness and to reduce the computations associated with foreground information during feature extraction, the image is thinned. The thinning effect on the noisy thresholded image ($T=200$) and noiseless thresholded image ($T=140$) is shown in figure 6.7 (a) and (b) respectively. As in 6.7 (a), the noise also gets thinned and will be left in the image as foreground information. This noise will influence feature extraction.

![Figure 6.7 Effect of Thinning on (a) noisy thresholded image (b) noise less thresholded image](image)
**Bounding box extraction:** All white pixel rows and columns from the boundary are counted as they carry only background information and they are removed from the image. This makes the thinned contour of the character touch the boundary from all the four directions as shown in figure 6.8. Now actual size of the character in the image is actually known. The character is now resized to 64x64 making it touch the boundaries from all the four directions as shown in figure 6.8 (a) and (b) for both noisy and noiseless images respectively. Here the shape of the character is made square and the original shape of the character is lost.

![Figure 6.8 Effect of bounding box extraction (a) noisy image (b) noiseless image](image)

6.2 **Problems with Static Preprocessing Pipeline**

Each stage output of the static pipeline is as shown in figure 6.9. This static pipeline did not help to eliminate noise from the sample images as around 30% images were still noisy. We observed that, as the images did not have bimodal intensity distribution in some cases, the noise was left over after preprocessing. The 5x5 median smoothing, Gaussian smoothing and unsharp masking filters could not help to generate the bimodal image with the valley in the mid gray range. We observed that, in some cases, there is an overlap of foreground and background intensity information which cannot be removed by global thresholding, but, may be by local thresholding. Based on the observations we built a *modified static preprocessing pipeline*. 

100
6.3 Modified Static Preprocessing Pipeline

The images with dark background and the images with light ink and thin edges were mostly distorted. We designed a new local thresholding method to minimize the noise.

Local thresholding: Two local thresholding methods are proposed. They use min, max and median ordered filters over 3x3 neighborhood giving Vmin, Vmax and Vmedian to change the center pixel value to foreground (black) or background (white). The gray image intensity range is 0 to 255 with 0 representing black (foreground) and 255 representing white (background). The min of 9 elements (3x3) helps to know the minimum intensity of a neighborhood pixel. Similarly max filter gives the maximum intensity of a neighborhood pixel and median gives the intensity of the 5th element when all 9 elements are arranged in order. These values help to know the pixel belonging. We tried two ways of using these parameters and different tolerable ranges. The two local thresholding methods are:

<table>
<thead>
<tr>
<th>Method</th>
<th>Condition</th>
<th>Pixel</th>
</tr>
</thead>
<tbody>
<tr>
<td>a.</td>
<td>If Vmedian &lt;= T</td>
<td>foreground</td>
</tr>
<tr>
<td></td>
<td>Pixel = foreground</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Else If Vmin &lt; T and (Vmax – Vmedian) &lt; P</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Pixel = foreground</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Else Pixel = background</td>
<td></td>
</tr>
<tr>
<td>b.</td>
<td>If Vmedian &lt;= T</td>
<td>foreground</td>
</tr>
<tr>
<td></td>
<td>Pixel = foreground</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Else If Vmax &gt; T and (Vmax – Vmedian) &lt; P</td>
<td>background</td>
</tr>
<tr>
<td></td>
<td>Pixel = background</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Else Pixel = foreground</td>
<td></td>
</tr>
</tbody>
</table>

Figure 6.9 Each stage outputs of Static preprocessing with Global thresholding
Where, $T$ and $P$ are decided based on the experiments and the final values chosen are $T = 200$ and $P = 30$. In both cases, first the majority of neighboring pixels is checked to see whether they belong to foreground. If not then, in the first case if there is at least one foreground pixel in the neighborhood, then the neighboring pixel variation is analyzed to make the center pixel value as foreground. In the second case, similar observation is done to decide the center pixel value as background. We found the second case more effective. The reason analyzed is that the background variation is less compared to foreground due to high writing, pen, and ink variations observed over the whole sample data. The trials with local thresholding showed that there is now, no need for median filter, Gaussian filter and Unsharp masking filter before thresholding and dropping them significantly reduced the computations. The modified static preprocessing pipeline is as shown in the figure 6.10.

![Figure 6.10 Modified static preprocessing pipeline](image)

Each stage results are shown in figure 6.11 (compare this with figure 6.9). The noise is now eliminated from the image with fewer stages of preprocessing.

![Figure 6.11 Each stage output of modified Static preprocessing with Local Thresholding](image)
6.4 Problems with Modified Static Preprocessing

We observed that still 10% of the images were noisy. In order to deal with these noisy images, further detailed analysis of the noise in preprocessed images was carried out to find the possible reasons for the deformation of the original shape. We analyzed the overall effect of static pipeline on the images, individual stage effects and also the effect of sequencing of these stages. We found that some characters suffered deformations due to the thinning method used and the size and the method chosen for size normalization. We also found that the cuts in the contours can be minimized if the resizing of the thinned image is avoided.

6.4.1 Effect of thinning algorithms

Thinning was one of the areas where we observed problems. We tried three popular thinning methods Laplacian of Gaussian (LOG), Canny edge detection and morphological thinning. The sample thinning effects of all three methods are shown in figure 6.12.

LOG is influenced by the width of the Gaussian hat to fit possible boundary thickness variations. This had the influence on the small objects. As shown in figure 6.12, with LOG, the small circle in the center is almost lost. Such small characteristics are important in Kannada, as we saw earlier.

![Figure 6.12 Thinning effects](image)

It is observed that double edge responses of LOG and Canny technique depend on the edge thickness and therefore they are influenced by the ink spread. We also observed that very crucial small objects were lost making the image more vulnerable to misclassification. This resulted in reduction of recognition results of some of the characters differentiated by such small objects and the average results decreased by 3% when tested under similar conditions. So we used morphological thinning in further experiments.
6.4.2 Effect of the image size

The acquired images are of varying sizes and are finally normalized (resized) to 64x64. A number of experiments are done to find the suitable size and normalization method to minimize the character shape deformation. In the previous static pipeline, we resized the image to 64x64 by making the character touch the boundary from all the directions. This actually deformed the original shape of the character by stretching it in all directions (see figure 6.3). We, therefore, performed size normalization by not loosing the shape of the character. Experiments are done with 32x32, 50x50 and 64x64 image sizes and the results are as shown in the figure 6.13. We found that with smaller size, the small objects are deformed or lost and a 64x64 pixel image maintains the shape of the character without deformation.

![Resizing effects](image)

Figure 6.13 Resizing effects

6.4.3 Effect of size normalization method

As the samples are taken in an unconstrained environment, the original image size has huge variations. The image should be normalized to a fixed size. Initially we considered the resizing to a fixed size 100x100. This made the image to loose its shape. Also in the process of making the image a square one, the shape may get stretched with some unwanted edges added as shown in the examples (a) and (b) of figure 6.14 (i). The literature analysis showed that the effect of some abnormally elongated strokes in the character images is around 5% on the recognition result [124]. But in the process of normalization, the algorithms try to normalize all the strokes in such a way that the overall shape tends to take square shape (refer section 2.2.2) and hence the original shape of the character image is lost.
As some Kannada character shapes are long, some wide and some square, this itself becomes one of the important features and hence we carried out size normalization by maintaining the original shape as much as possible. The effect is as shown in figure 6.14(ii). The rectangle with the thick lines indicates the occupied area of the normalized image within a square 64x64 image. Here the M x N character shape is proportionately stretched by making either m or n take a size of 64 depending on which is larger and the other one is padded with background to make it 64 as shown with examples (a) and (b) in figure 6.14 (ii).

6.4.4 Effectiveness of static preprocessing pipeline

The static pipeline fails to produce noiseless images in some cases as shown in the figure 6.15. The system could not adequately eliminate the ink spread, background noise, and resulted in the removal of small and thin edges or light ink objects.

As shown in figure 6.15(a), the static pipeline did not eliminate the noise adequately when –

a. the background is darker
b. the ink spread is not uniform

It is observed in figure 6.15(b) that the wanted edges or part of edges are eliminated when –

a. the edges are too thin, thinner than filter size
b. the size of the image is small, that edge thickness is less than filter size
c. the pen ink flow is not proper
d. pressure exerted while writing is less, so that the ink impression is lighter

It is also observed in figure 6.15(c) that some new edges are generated when –
a. the size of the image is small
b. two or more edges are very close by
c. the ink spreads making the edges nearer

In order to overcome these problems, we proposed a dynamic preprocessing pipeline that is capable of handling these variations. This is outlined in next section.

![Figure 6.15 Static preprocessing effects for some special case](image)

### 6.5 Dynamic Preprocessing Pipeline

The dynamic preprocessing (DP) pipeline proposed is shown in figure 6.16 and has 3 stages: initial stage processing, categorized processing and final stage processing. The main change is the introduction of categorized processing where different variants of an algorithm is applied to an image depending on some property of the image; that is why the pipeline is called dynamic. The three stages are described below. As compared to static pipeline,
different filters and methods are used in some stages and re-sequencing of some stages is also done.

![Dynamic Preprocessing pipeline](image)

**Figure 6.16 Dynamic Preprocessing pipeline**

### 6.5.1 Initial stage processing

The initial stage processing shown in figure 6.17 makes the image suitable for categorized processing by producing the smooth and sharp contours of the character image that are dynamically resized if smaller than 70x70.

![Initial stage processing of DP](image)

**Figure 6.17 Initial stage processing of DP**

*Dynamic Resizing*: This first step helps to retain the original shape of the character and to make the original image size suitable for resizing at the end. Resizing the image to 100x100 makes the image zoom to this fixed size. For example, a 40x60 image is stretched to 100x100 by increasing 40 rows to 100 rows and 60 columns to 100 columns. That means, rows are more stretched than the columns. This may affect the original shape of the character by stretching it non-uniformly. We can observe the vertical stretch deformation of the original character shape in the resized image in figure 6.3. Dynamic resizing does proportionate resizing. Based on the observation, small size images (MxN with M rows and N columns)
with size either M or N less than 70 are resized. It is observed that images smaller than 70x70 along with background may result in a small size (smaller than 64x64 – depends on amount of background surrounding the shape of the character) boundary touching thinned image and the resizing and median filtering of such small images results in distorted images. So we initially resized the original image to a new dynamic size M+(70-min(M,N)) x N+(70-min(M,N)). For example, 30x50 size image is proportionately resized to 70x90 by uniformly stretching the rows and columns by 40. The adjustment constant 70 is chosen so that the minimum size after resizing is 70 along with background. The figure 6.18 (b) illustrates the effect of dynamic resizing of the image in (a). This helped in preserving small size character shapes and also eliminated size related problems after final size normalization.

![Figure 6.18 dynamically resized image](image)

**Wiener filtering:** Wiener filter does both smoothening and sharpening of the edges and hence replaces Gaussian and unsharp masking filters of static pipeline. It fills the ink gaps and lightens the noise information and sharpens the edges. It filters a gray scale image that has been degraded by constant power additive noise. It uses a pixel-wise adaptive method based on statistics mean $\mu$ and variance $\sigma^2$ estimated from a local neighborhood of each pixel given by equation 6.1

$$g(x, y) = \mu + \frac{\sigma^2 - \nu^2}{\sigma^2} (f(x, y) - \mu)$$  \hspace{1cm} (6.1)

Where, $f(x,y)$ is the input gray level at $x,y$, $g(x,y)$ is the output gray level at $x,y$, and $\nu^2$ is the noise variance. If noise variance is not given, the filter uses the average of all the local estimated variances. The Gaussian noise spreads the edges and by removing such noise, the edges are smoothened. The effect of Wiener filter on 6.19 (a) is shown in figure 6.19 (b).
Figure 6.19 Wiener filtered image

*Histogram analysis:* If the intensity distribution (spread) is less then the character image is of good quality and if the spread is more then the quality of the character image is poor as shown with examples along with their histograms in figure 6.20 (a) and (b) respectively.

(a) ![Original Image](image1)
(b) ![Wiener filtered image](image2)

Figure 6.20 Histogram of a character image (a) good (b) bad

Histogram of the wiener filtered image and the original image is done in order to perform the background and foreground intensity distribution analysis for the next stage. The histogram spread is the spread of the pixels taking gray values in the range 0 to 255. The ideal character image histogram should have a wide valley with mid gray values not used in both foreground and background. The foreground histogram spread represents the foreground intensity distribution (FID) and the background histogram spread represents the background intensity distribution (BID).
6.5.2 Categorized processing

The histogram analysis shows that the foreground and background intensity distribution varies from image to image. The FID varies due to pen type, ink quality, ink color, ink flow, paper quality and the pressure exerted while writing. Similarly the BID varies due to paper quality, non-uniform paper illumination, scanner imperfections, etc. A good quality character image is one whose distribution is close to a binary image (all the pixels mapped to either black or white) as the characters are represented in two states with a single color pen trace on a usually white paper.

We observed that if the ink color is dark black or dark blue then image FID is close to black (good). On the other hand, if the ink is light black or light blue or if there is a lot of pressure variations while writing or improper ink flow then FID is close to mid gray (bad). Similarly if the background is noiseless then BID is close to white (good) and if background is noisy then BID is close to mid gray range (bad). For a good FID or BID, the histogram spread is less and the distribution concentrates close to black and white. For bad FID or bad BID the histogram spread is more and the distribution of either FID or BID spreads towards mid gray. If both FID and BID are bad, they overlap in the mid gray range. This makes the separation of foreground and background difficult. Based on this intuition, we propose categorized processing based on FID and BID. Accordingly, the character image can be of one of the 4 categories:

- FID good and BID good – images can be binarized using simple global thresholding.
- FID good and BID bad – images need to be contrast adjusted before binarization with less stretch towards black and more stretch towards white.
- FID bad and BID good – the contrast adjustment should be done with more stretch towards black and less stretch towards white.
- FID bad and BID bad – there will be overlapping of FID and BID, and the separation of character contour from the background is difficult and we do not consider such images.

We also observed that small size of the character image, small nearby objects in the character contour and the ink spread makes the edges nearer and need to be processed carefully to avoid formation of any new joints.
Based on these observations we proposed Categorized processing shown in figure 6.21. The decision making is based on two factors. The first factor is the foreground intensity distribution (FID) and background intensity distribution (BID) and the second factor is the pen ink spread and original image size.

The categorized contrast adjustment is one of the important dynamic step in this pipeline. We used imadjust function of matlab for contrast adjustment given by:

\[
\text{A1} = \text{imadjust}([\text{org}, 15]) \\
\text{A2} = \text{imadjust}([w, 10])
\]

\[
\text{A1} = \text{imadjust}([\text{org}, 12]) \\
\text{A2} = \text{imadjust}([w, 3])
\]

\[
\text{A1} = \text{imadjust}([\text{org}, 15]) \\
\text{A2} = \text{imadjust}([w, 5])
\]

\[
\text{org} = \text{original image} \\
w = \text{wiener filtered image}
\]

Add (A1, A2)  
Global Threshold  

Is edge thickness ?

Too thin / thick with spread

Normal

3x3 median filter  

2x2 median filter  

Categorically processed image
\[ J = \text{Imadjust}(I, [\text{low\_in} \ \text{high\_in}], [\text{low\_out} \ \text{high\_out}], \text{GAMMA}) \]

It is a function that maps the values of image \( I \) to new values in \( J \) such that values between \( \text{low\_in} \) and \( \text{high\_in} \) of \( I \) map to new values between \( \text{low\_out} \) and \( \text{high\_out} \). The values below \( \text{low\_in} \) and above \( \text{high\_in} \) are clipped to \( \text{low\_out} \) and \( \text{high\_out} \) respectively. GAMMA is a nonnegative real value and specifies the shape of the curve describing the relationship between the values in \( I \) and \( J \). If GAMMA is less than 1, the mapping is weighted toward higher (brighter) output values. If it is greater than 1, the mapping is weighted toward lower (darker) output values. That is, higher the values of GAMMA the weighting towards darker values increases. If it is 1 then, it gives linear mapping.

We considered full input gray range mapping to full output gray range with \( [\text{low\_in} \ \text{high\_in}] = [0 \ 1] \) and \( [\text{low\_out} \ \text{high\_out}] = [0 \ 1] \). GAMMA is varied above 1 to map \( I \) to darker values to produce \( J \). Both original image (org) and wiener filtered image (w) are contrast adjusted under categories with more stretch used for original image than the wiener image. The reason is to produce more dark pixels in one of the image so that the foreground values are near to black gray level. This also makes the background dark.

In the next step, both original and wiener filtered contrast adjusted images are added. The addition results in 0 to 510 ranges of the gray levels by adding two images of 0 to 255 range gray levels. As we are adding 2 images with each pixel value in unsigned integer 8 bit format, the output after addition will also be in unsigned integer 8 bit format and so the output pixel values that exceed 255 are all truncated to 255. That means, all values above 254 are all now white representing background and the values till 254 will retain their values as it is giving a wide gray range for the foreground. In the process, the near to black gray values of the original image when added with the less adjusted wiener filtered image, for example \( 24+60 = 84 \), the values still remain close to black w.r.t 0-510 range. But when dark background values when added with the bright background of the less adjusted wiener filtered image, for example \( 24+240=264 \), then 264 will be truncated to 255(maximum gray value representing white). This improves the valley between the foreground and background. The histograms and the effect of these operations are shown in figure 6.22.

As the contrast enhancement creates a deep valley between the foreground and background histogram, the global thresholding is now adequate for binarization.
Median filter is a very essential filter to remove left over salt and pepper noise in the binarized image. This filter makes foreground as background if edges are thinner than the filter dimension. If there is spread of ink, the edges will be close by within the filter dimensions and median filter joins the edges. So based on the original image size and the ink spread we chose median filter size to be either 3x3 or 2x2.

The median filtered image is the stage 2 output and is further processed through final stage processing.

![Image](image.png)

*Figure 6.22 Contrast enhancement*

### 6.5.3 Final stage processing

The final stage processing is as shown in the figure 6.23. After bounded box extraction, we performed resizing followed by thinning of the image. This shuffling is done to avoid the stretching of the binary thinned images which result in cuts in the edges while resizing. The resizing is done by maintaining the original shape of the character so as to distinguish the
characters based on their shapes. The processing of these steps is as explained in static processing.

![Diagram](image)

Figure 6.23 Final stage processing of DP

Hence dynamic preprocessing converts the original color / gray image into a thinned, size normalized, boundary touching, noiseless character image with original shape maintained. Output of each stage for one sample image is as shown in figure 6.24.

![Image](image)

Figure 6.24 Dynamic Preprocessing stage outputs

### 6.5.4 Results of dynamic preprocessing pipeline

Some sample comparative results of preprocessing using the static pipeline and dynamic pipeline are shown in figure 6.25. Original images here have some kind of abnormality or deformation like close by strokes, ink spread, small size, background noise, improper ink flow, very small size objects in the character image and long and tall character shapes, respectively given in columns 1 to 8. For these images, the static preprocessing results
in noisy and/or deformed images where as dynamic preprocessing produces relatively noiseless results with shape and small objects preserved.

<table>
<thead>
<tr>
<th>Images</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original</td>
<td>📷</td>
<td>📷</td>
<td>📷</td>
<td>📷</td>
<td>📷</td>
<td>📷</td>
<td>📷</td>
<td>📷</td>
</tr>
<tr>
<td>Static preprocessing</td>
<td>📷</td>
<td>📷</td>
<td>📷</td>
<td>📷</td>
<td>📷</td>
<td>📷</td>
<td>📷</td>
<td>📷</td>
</tr>
<tr>
<td>Dynamic preprocessing</td>
<td>📷</td>
<td>📷</td>
<td>📷</td>
<td>📷</td>
<td>📷</td>
<td>📷</td>
<td>📷</td>
<td>📷</td>
</tr>
</tbody>
</table>

**Figure 6.25 Sample comparative results**

Overall dynamically preprocessed results of 200 samples of a character are shown in figure 6.26. The original samples are as in figure 5.3. The image of each sample shows that the shape of the character is intact with only noise eliminated from the images. The feature values extracted from such images will have less intra class variations and more inter class variations which increases the recognition efficiency of the characters.
Based on the overall preprocessing experiments, we observe:

- There is no significant difference between the outputs from images of two different file formats. The 256 color bitmap image file format is also suitable to represent the input images for the experiment. This reduces the high storage space requirement. We, hence suggest this format for scanned images.

- As the handwritten characters vary with respect to ink trace quality, background noise, and character size randomly, we realize that static pipelines can handle such random variations under certain limitations. The steps set for one kind of problem may not be suitable for another kind of problem. If the variations are huge, then, based on the variations, if appropriate steps are followed to handle the specific kind of deformations observed, the results can improve further. This is the kind of dynamicity we tried to build in our dynamic preprocessing.

- The proper sequencing of the two operations: resizing of the image and thinning, is required to minimize any cuts in the contours of resized image.

- Dynamic resizing at the beginning will allow the HCR system to handle huge size variations without deformation of original shape.

- The proper selection of image thinning algorithm, size normalization method, image size etc, is important to reduce the distortions that may happen due to the preprocessing stage itself.

- Wiener filter is efficient to remove the Gaussian noise usually added by the scanning devices.

We have adopted the dynamic preprocessing pipeline in the rest of our experimentation to produce noiseless binary thinned character image with smooth and continuous contour that is confined to a standard bounding box and is size normalized; this will be used for feature extraction in our experiments, (to be discussed in next chapter).