Algorithm for finding skew angle and then de-skewing the image

14th June 2008

by Debayan Banerjee

Background: The Tesseract OCR engine does not support connected script such as Devnagri. The maatraa clipping code added to the engine makes this a possiblility now. The maatraa clipping code requires the page to be perfectly straight, ie, un skewed to work. To this effect, this document describes the algorithm used to find the skew angle from the document image, and then the algorithm to deskew it.

Modus Operandi: There are several very good ways to find out skew angles in documents and it is a thoroughly researched topic. The method in vogue is using Hough transforms, using scan lines at different angles over the image. However, paucity of time forced me to resort to an ad-hoc algorithm, which is not very robust, but works.
The aim is to find the angle of inclination of any maatraa. We take several slope readings for a particular maatraa eliminating values that vary wildly, and then average it.
The de-skewing algorithm uses trigonometric relations between a skewed and un-skewed image to read corresponding pixel data into the new image. This method introduces key stoning effect (jagged edges) in the image, and it can be overcome by using rotation by shear algorithm instead.

Input to the algorithm: The thresholded, unskewed/skewed and noise free image to be OCRed.
Output of the algorithm: The de-skewed image.

Algorithm for finding out skew angle:

1. Let the line AA' be a horizontal scanning line that starts from the top of the image and proceeds to the bottom of the image.
2. Store the co-ordinates of the first black pixels encountered. Ideally, this is the tip of a skewed maatraa. (There may be exceptions to this. It may be a part of an alphabet that rises above a maatraa, in which case the angle returned will be false.)
3. If the x co-ordinate of the point thus found is >width/2 , we assume that the page is tilted towards the right, otherwise left. To check of the page is un-skewed, we drop projections from the top of the page to the top most maatraa. If the height of any two projections are found to be same, the page is straight already and no de-skewing is required.
4. If the page is tilted, we proceed to find the angle of tilt. We try to find an end of the top most (or any) maatraa. Then we find the angle of tilt of the maatraa as shown in Fig x. We then

eliminate wild values (tilt>10 degrees) and keep averaging the values found.

5. We use tan inverse to find the angle of tilt and then return the value.

Algorithm for de-skewing the image:

1. Using the dimensions of the skewed image, and the angle of tilt, derive the dimensions of the new image using these relations:

```
float Point1x=(srcheight*sine);
float Point1y=(srcheight*cosine);
float Point2x=(srcwidth*cosine-srcheight*sine);
float Point2y=(srcheight*cosine+srcwidth*sine);
float Point3x=(srcwidth*cosine);
float Point3y=(srcwidth*sine);

float minx=min(0,min(Point1x,min(Point2x,Point3x)));
float miny=min(0,min(Point1y,min(Point2y,Point3y)));
float maxx=max(Point1x,max(Point2x,Point3x));
float maxy=max(Point1y,max(Point2y,Point3y));

int DestWidth=(int)ceil(fabs(maxx)-minx);
int DestHeight=(int)ceil(fabs(maxy)-miny);
```

   Here Point 0,1,2,3 are the 4 corners of the source image.

2. Use the following relations to create the new image:

```
int Srcx=(int)((x+minx)*cosine+(y+miny)*sine);
int Srcy=(int)((y+miny)*cosine-(x+minx)*sine);
```

   Where x and y are the pixel co-ordinates of the new image, and SRCx and SRCy are the co-ordinates of the source image.

Images and Figures

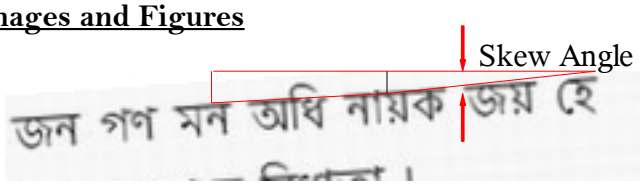

Fig 1

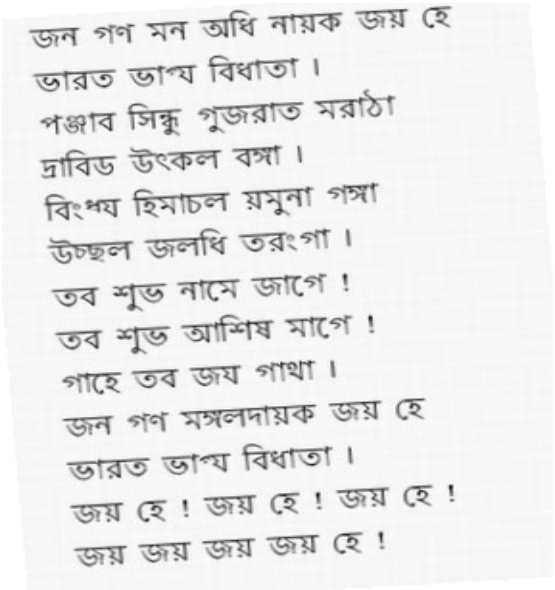

Fig 2

Tilted scanned image

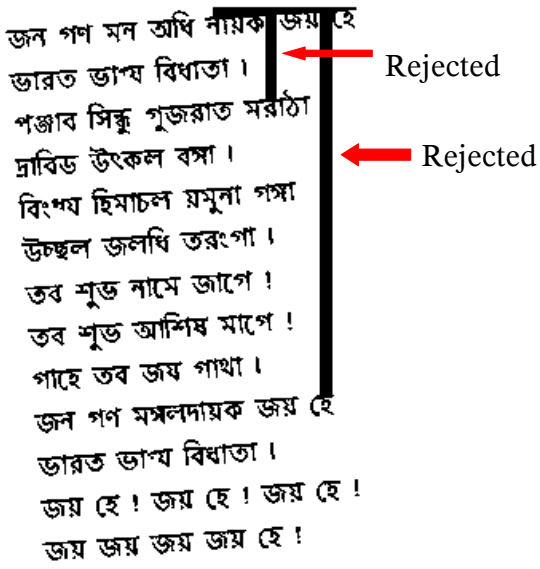

Fig 3

This image has been generated by tracing the execution of the algorithm by using black pixels. The long black pixels have been rejected for angle calculation because the slope the give us are too large (>10 degrees).

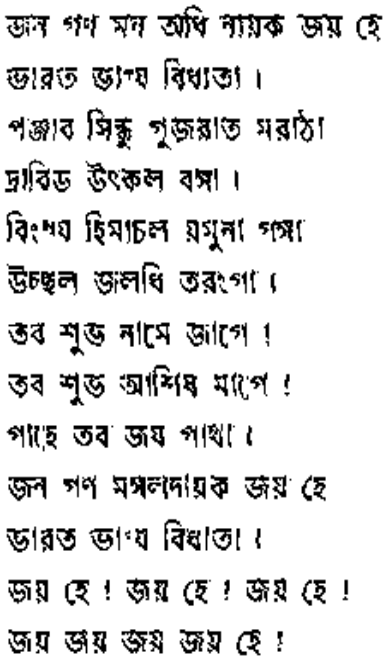

Fig 4

This is the final de-skewed +clipped image. The image quality is significantly reduced, and shows jagged edges. Theres a key stoning effect prevalent. It occurs due to inaccurate deskewing operation due to the inherrent discrete nature of pixels. We can improve it by using rotation by shear. The clipping is fine, except the first line, which is an anomaly which can be fixed easily.