HOWTO: Improve Barcode Reading Speed/Accuracy

This article discusses some settings and techniques you can use to improve barcode reading.

The first and most obvious advise is making sure you have a clean, deskewed, and bitonal (black and white) image to barcode. We have that `BinarizeCommand` to gracefully threshold an image to black and white. Once it is `Pixel1bppIndexed` you can use the `AutoDeskewCommand` to deskew it. If you have noisy images the `DocumentDespeckleCommand` will remove small specks. The `MorphoDocumentCommand` can be used to close small gaps or lighten dark areas.

When you run in to issues barcoding I would recommend saving the images out (after any processing) and inspecting them visually. If the barcode is blurry or lines run together or is otherwise hard for you to see then it will be hard for the barcoding to recognize it as well.

Example:

```
if(image.PixelFormat != PixelFormat.Pixel1bppIndexed)
{
  BinarizeCommand bc =
newBinarizeCommand(BinarizeMethod.AdaptiveThreshold);
  AtalaImage bwImage = bc.Apply(image).Image;

   if(!bc.InPlaceProcessing)
   {
      image.Dispose();
     image = bwImage;
   }
}

AutoDeskewCommand adc = newAutoDeskewCommand();
AtalaImage deskewed = adc.Apply(image).Image;
if (!adc.InPlaceProcessing)
{
   image.Dispose();
   image = deskewed;
}
```

The second is using the ScanInterval setting on the ReadOpts object. The barcode reader scans across a line looking for barcode content. Setting the ScanInterval higher means it ends up having to check fewer lines which will speed up overall processing.

Example:

```
BarCodeReader reader = newBarCodeReader(image);
ReadOpts opts = newReadOpts();
opts.ScanInterval = 10;
BarCode[] codes = reader.ReadBars(opts);
```

The last suggestion is that if you know what region a page a given barcode will be you can set the RectOfInterest property on the ReadOpts object to only check a smaller area. This will have a significant impact on your processing speed. The fewer pixels the barcode reader needs to check the faster it will be. If you know a barcode will even be roughly the upper/leftquarter of an image and can set the RectOfInterest = new Rectangle(0, 0, img.Width / 2, img.Height / 2) that will mean the barcode reader is doing 1/4 of the work (meaning it will take roughly 1/4 of the time) while only skipping areas you already know won't have barcodes.

Example:

```
BarCodeReader reader = newBarCodeReader(image);
ReadOpts opts = newReadOpts();
opts.RectOfInterest = newRectangle(0, 0, image.Width / 2,
image.Height / 2); //example only. Your barcode may not be in the
top/left quadrant.
BarCode[] codes = reader.ReadBars(opts);
```

Original Article:

Q10438 - HOWTO: Improve barcode reading speed/accuracy

HOWTO: Improve Barcode Reading Speed/Accuracy

Atalasoft Knowledge Base
https://www.atalasoft.com/kb2/KB/50062/HOWTO-Improve-Barcode-Reading-SpeedA...