Some Possible Book Scanning "Gutter Math"

Don Lancaster
Synergetics, Box 809, Thatcher, AZ 85552
copyright c2009 pub 12/09 as GuruGram #103
http://www.tinaja.com
don@tinaja.com
(928) 428-4073

The main goal of a book or bound document scanning system is no longer to simply capture the page images. In addition, you will want to provide for reliable **OCR** recognition. The latter both dramatically reduces file sizes and allows full text searching. While greatly easing web distribution. Plus possible later editing, text reading, or other handicapped aides.

All the while minimizing stress and damage to the book itself.

There are a number of high end solutions to content capture emerging. **Google** is rumored to use a pair of 45 degree optical systems. These project an infrared grid onto the pages and then use that sensed grid to provide for image **deskewing** and distortion correction. What is also needed are **effective scanning solutions** for individuals who, at reasonable costs, only wish to occasionally capture a few dozen to a few hundred bound legacy documents.

New Opportunities

Three recent developments have greatly improved "home" book scanning. The first of these is Adobe's **ClearScan** included in **Acrobat 9**. ClearScan creates new custom fonts that are a best match to the existing typography. While accurately preserving justification and column widths. Sadly, the process is not yet perfect. The fonts look a tad on the mangy side at lower resolutions and there is not yet any font editing ability in place. Nonetheless, ClearScan is a major leap forward for OCR. File sizes can approach 6K per page.

Second are the **dramatic improvements in digital camera pixel counts**. To the point where they are reasonably competitive with scanning solutions. And offer "to the page edge" opportunities that elude most scanners.

Because of the **Bayer Filtering** used, the fancy algorithms, and the geometry factors, any advertised camera resolution is nowhere near its effective resolution. But a 16 Megapixel camera might approach 12 Megapixels of effective grayscale resolution. Approximating 450 DPI even for larger documents.

There may even be enough excess resolution to allow the third opportunity of **elaborate image post processing**. The **PostScript** language is especially adept at reading and writing most any file in most any format. And especially in rectifying **bitmap images**.

Tasks such as reversal (because of a mirror in the optic path), anamorphic scaling (if the mirror is not at a 45 degree angle), and image rectification should now be easily accomplished. Deskewing is already well done in ClearScan, so we can leave it in the "ain't broke" category.

Many examples of image post processing appear in **this link**. What I thought we might do here is look at some possibly useful math to deal with...

The "Gutter Problem"

Bound books do not tend to lie perfectly flat. Especially if opened all the way. Words near the gutter (the bound center portion) tend to appear very cramped when scanned or photographed. Besides looking awful, OCR becomes much more difficult with much worse error rates. One workaround, of course, is to **not open the book that far**. Use a 45 degree mirror or prism or other optical path to minimize any curvature. But some guttering may remain.

Can we mathematically define guttering? Can the math be made simple enough, general enough, and fast enough to allow usable correction? Especially when combined with mirroring to minimize its effects?

Let's **initially consider only a left facing page**, gutter to the right. Further, let us **normalize** our math so the leftmost area of interest is zero and the rightmost a one. Often, this will coincide with the text margins.

We might at first consider using a **cubic spline** approach. But something that is more **maximally flat** such as a sparse **Taylor Series** is rather likely to end up a lot more useful.

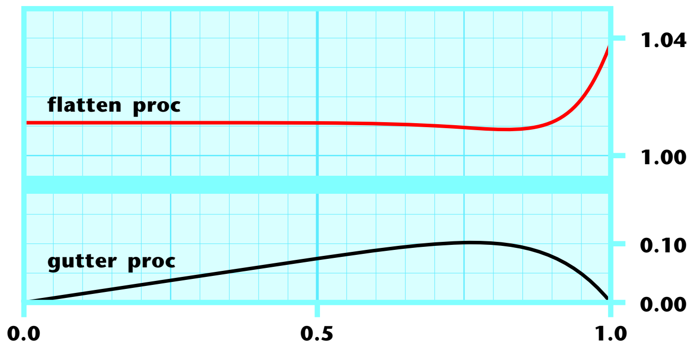
Consider this function...

y = ax + bx^n

Now, **x** will vary from **0** on the left to **1** on the right. **y** will vary with the page curvature. **a** sets the overall linear page slope. **b** sets the depth of the gutter. And our "magic" **n** determines how "curvey" and how wide our gutter is.

Any **x^n** with larger **n** will be **1** at **x=1** regardless of **n**. And the higher the value of **n**, the lesser its effect becomes progressively to the left.

Let's try a plot of **y = 0.15*x + 0.15*x^9.2**...



NORMALIZED PAGE POSITION

Now that black bottom "gutter proc" curve sure looks like an open book page to me. How well it matches your book page will depend on the page size, the book thickness, and how stressed the binding is. But with only partial page openings, the differences should end up minor and may actually converge.

Wider book pages may have a narrower gutter, so higher powers of **n** (perhaps in the **11** to **14** range) may be needed. The slope of the "flat" portion can often be eliminated by adjusting the optical path to the camera. The gutter shape may also vary with how far into the book the scanned page will be. All these factors should minimize if you **make the page as flat as possible before image capture**.

Flattening the Page

Useful OCR pretty much demands a perfectly flat page. The next crucial question is to find out what adjustment is needed to the gutter proc to make each letter appear in its true position. We can call this process **finding a flatten proc**.

To do this, start at the left end and make a very small change in **delta X**, our horizontal position. Then find out how much the true distance **delta S** changes. This will be a vector sum of the horizontal and vertical changes, as found by the **Pythagorean Theorem**. Perhaps looping code something like this…

```
0 0.001 1 {/xcur exch store
            /ycur posn a mul posn n exp b mul sub store
            /scur scur xcur xold sub dup mul ycur yold sub
               dup mul add sqrt add store
            /xold xcur store
            /yold ycur store
                    % do stuff with scur here
            } for
```

A plot of the true page distance versus the x distance appears in red above. In this example, the "guttermost" pixels need progressively moved to the right up to four percent, or around **80** pixels on a **2000** pixel line.

While a tad nonobvious, our function dips slightly before increasing. Which suggests a simplification that leaves "most" of the pixels right where they were all along. **We can make a = 0 by aligning any mirrors or whatever to end up with a camera "film plane" that is parallel to most of the page**.

A second major simplification is possible by sending...

Table Lookup to the Rescue

The math behind our red flatten proc curve can be rather nonobvious and a tad ugly, being an integral of the inverse of a nonlinear square root. Yes, we could find several **Taylor Series** terms or even **Cubic Splines**, but **we would very much like to avoid complex pixel-by-pixel calculations**.

My favorite solution to avoiding repeated complex math calculations is to use **table lookup**...

Distance correction is best done as a one-pixel-per-entry table lookup to minimize individual pixel calculations.

If we try this using a "forward correcting" lookup, we might end up with some image dropouts during our post proc. Instead, we can use an **inverse table** that tells each pixel in the final image how far to go to the left for its actual data.

Except for our leftmost pixels, **image interpolation** may be needed to find your intermediate values between pixels. Since these may degrade image quality, the **bicubic** interpolation is probably better used than a **bilineal** one.

Additional details are found **here**.

A typical table for a flattened gutter correction might look like this...

```
/fixgut [ 1.000000 1.000000 1.000000 1.000000 1.000000
               .  .  .  .  .
          1.000000 1.000000 1.000000 0.999999 0.999995
               .  .  .  .  .
          0.961136 0.960588 0.960034 0.959473 0.958907
          0.958334 0.957754 0.957169 0.956577 0.955979
        ] store
```

Some more detailed code and table lookup examples appear in the **sourcecode** to this **GuruGram**

A Plan, sort of

I don't quite have all the details worked out on all this just yet. The long term goal is to get many (and preferably all) of my "selectric era" sourcecode free articles and reprints available online. Along with many of my out-of-print books. Several dozen easier to find construction projects are already available as searchable single files **here**.

Some interesting third party approaches to reasonably priced DIY book scanners are found **here**.

My present thinking is to **open the book only 45 degrees** using an unframed quality mirror. This should minimize the gutter problem beyond the approaches others are using. The book could further be clamped using some **potato chip bag closures**.

Again with a goal of dramatically reducing guttering. Photographing the page images with a high resolution digital camera (preferably 16 Megapixels) should then produce images. A one pass picture post processing should be able to rectify, degutter, and reverse. Perhaps using improvements on **these routines** and **this code** in particular.

More to follow as time permits.

For More Help

Many more application programs and utilities appear on our **PostScript Library** page. A guide to our **PostScript Gonzo Utilities** and additional links appear **here**.

Additional consulting services are available per our **Infopack** services and on a contract or an hourly basis. Additional **GuruGrams** are found **here**. Seminars also available.

Further **GuruGrams** await your ongoing support as a **Synergetics Partner**. For details, you can email **don@tinaja.com**. Or call **(928) 428-4073**.