

# Gradient Data Extraction

Gradients can be defined several different ways. As implemented by Concepts In Computing, Inc., a gradient simply defined as the rate of change in intensity from one pixel to the next. Simply put this is a slope value. Theoretically, gradient algorithms should be able to detect edges that any threshold algorithm can detect. Conversely, threshold algorithms cannot detect edges that gradient algorithms can. The trade-off for the enhanced edge detection is speed. Most commercially available gradient algorithms run two to five times slower than their threshold equivalent. In actual application, this kind of performance decrease is intolerable. For most OCR applications, speed is a critical component when trying to determine the efficacy of any solution.

## Performance

With the latest release of the utility library, Concepts In Computing, Inc. has introduced a gradient algorithm that reduces the time differential to less than two times the speed of thresholding. The table below lists typical times for the various threshold techniques implemented in the EconoCR library and the gradient algorithm applied to OCR. Figure 1 is the text that was used for this test.

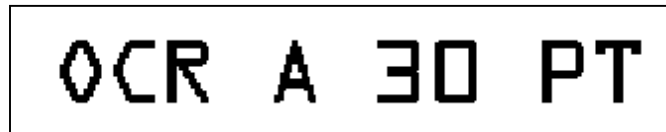


Figure 1.0

The characters are approximately 32 pixels high with an area of interest about 70 pixels high and 240 pixels wide.

Floating Point Algorithm		
Threshold Mode	No Font List	36 char list
Fixed	2.7	15.9 msec.
Computed	5.3	18.5
Linear	3.8	17.2
Non-linear	8.7	21.6
Gradient	7.3	<b>21.1</b>
Integer Algorithm		
Threshold Mode	No Font List	36 char list
Fixed	2.7	15.9 msec.
Computed	2.9	16.3
Linear	2.8	16.1
Non-linear	3.1	16.5
Gradient	7.3	<b>21.1</b>

Table 1.0  
Speed Comparison

The times shown in the table is the total time to perform a single read including data extraction from thresholding or gradient detection, segmentation, and classification.

As shown in Table 1.0, the performance penalty from a single computed threshold value to gradient detection is only 15% when using the floating-point algorithm. This increases to about 30% when compared to the integer algorithm. As the number of characters in the font list increases, the differences decrease since the extraction time is fixed and the classification time becomes a larger portion of the overall time.

## Application

Typical OCR applications that can benefit from gradient data extraction are:

- Can tops
- Cardboard packaging
- Metal cases

In many of these applications, threshold techniques work just fine and should be the preferred method because of the performance difference. Where threshold does not work reliably, gradient data extraction can be used.

### Metal Case

Figure 2.0 is a hard disk drive case that has both scratches in the metal and streaking from a sealant. To read this with threshold methods requires a positioning tool and seven readers, one for each group of characters. Although successful, considerable care was taken to setup and read this kind of image.

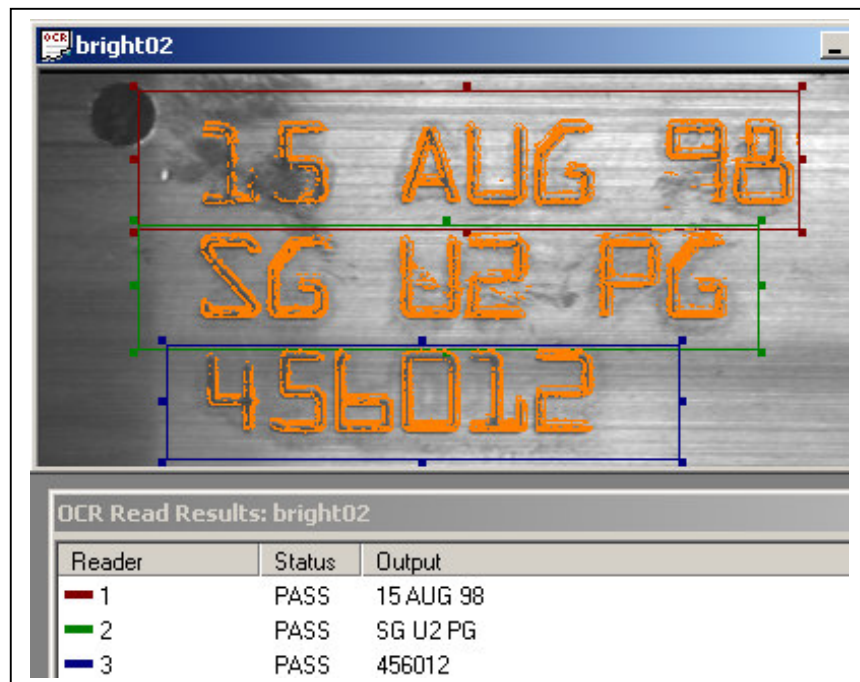


Figure 2.0  
Disk Drive Case

With gradient data extraction, the streaking through the 15 in row one, the hole in the upper left corner, and the darker areas to the left of the text are all ignored. Rather than seven readers and a positioning tool, three readers are sufficient.

Since the gradient algorithm was designed for OCR, there is significant character fill as shown in figure 2.0. Because of that, font files created from gradients or thresholding can be interchanged and intermixed.

### Cardboard

Figure 3.0 is a marked piece of cardboard. Although the image quality is good, the cardboard itself has darker areas around the right side of the text. This creates problems when trying to use threshold methods.

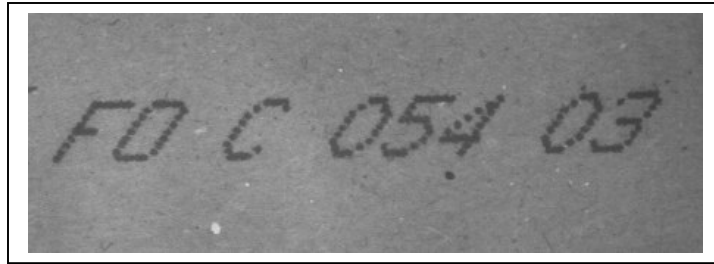


Figure 3.0  
Cardboard Image

Figure 3.1 shows the effect of that darker area when trying to threshold and read this image.

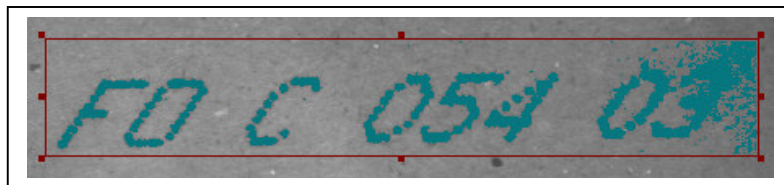


Figure 3.1  
Thresholded Image

With the large amount of area on the right side within the threshold range, this image is unreadable. The only way to read this image with Thresholding is to use multiple readers for each group of text.

Figure 3.2 is the same image using gradient data extraction.

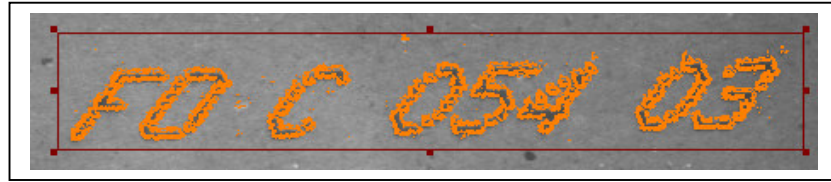


Figure 3.2  
Gradient Image

As can be seen, each character is clearly defined using this method. The only remaining task before actually reading would be de-slanting of the text. Figure 3.3 shows the image after de-slanting.

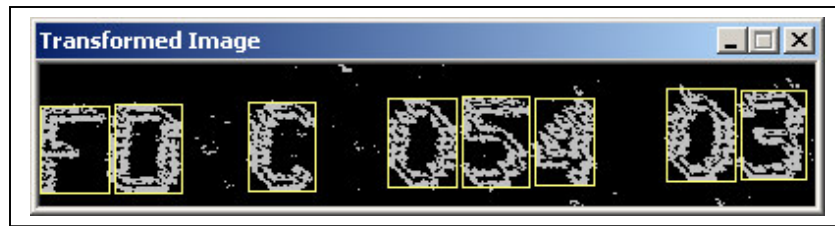


Figure 3.3  
De-slanted Image

With the characters straight, the reader can be setup to obtain one tightly bound box for each character. This is the key to obtaining repeatable and reliable reading.

## Summary

For applications that have difficulty using threshold data extraction techniques, gradient data extraction can be a viable alternative. Although there is a small time penalty for this technique, it should not be prohibitive in most applications. Additional benefits include interchangeability of font files and reader settings. Thus, switching from threshold mode to gradient mode and back is transparent.