A Method on Tracking Common Boundaries of Color Regions in Function Approximation-based Image Coding

Tetsuo Sugiyama¹, Paul W.H. Kwan², Kazuo Toraichi³, Kazuki Katagishi³
¹Doctoral Program, Graduate School of Systems and Information Engineering, University of Tsukuba, JAPAN
²Core Research for Evolutional Science & Technology Program, Japan Science & Technology Corp., JAPAN
³Institute of Information Sciences and Electronics, University of Tsukuba, JAPAN
E-mail: {sugi, kwan}@wslab.is.tsukuba.ac.jp, {toraichi, katagisi}@is.tsukuba.ac.jp

Abstract - A boundary tracking method crucial to function approximation-based image coding is proposed that solves the problems caused by duplicate tracking the common boundaries of color regions when methods conventionally applied to bi-level raster images are inappropriately used. Experimental evaluation is performed to verify its effectiveness.

I. INTRODUCTION

Authors of this paper have been conducting research on automatic function approximation-based image coding techniques (also known as vector coding) since the 1980’s that had led to the development of function fonts for brush-written characters [1] and their utility in the world’s first brush-written word processor [2]. While in [1], the outlines of brush-written characters were approximated by two types of functions namely straight lines and curves in the form of quadratic B-splines, in [2] the outlines were approximated by a combination of three types of functions, representing straight lines, curves and arcs additionally.

Nowadays, a large percentage of images that are created and made available on the Internet are coded in commonly used raster or bitmap-based formats such as GIF and JPEG. In the process of image enlargement, the occurrences of jaggy noises along the contours of image objects is a problem commonly shared by these raster formats (Refer to Figure 1 for an example).

As an approach proposed to address the degradation of image quality caused by affine-transform operations on raster images, since mid-1990’s the authors of this paper have furthered their research on function approximation-based image coding that can be applied to digital documents comprising a combination of texts, images, and graphics [3]. Also, document images having a maximum of 256 colors (or 8 bits/pixel) were considered.

However, in this recent research, a conventional boundary tracking method applied to bi-level raster images was unsuitably used on document images that involve multiple color regions. Due to a major drawback in conventional methods that focus only on a single color region at a time during the tracking process (that is, a color region enclosed within a closed boundary is considered the object of interest while everything else is background), boundaries that are common to two color regions were being tracked in duplicate (Figure 2).

The occurrence of duplicate tracking the common boundary between two color regions led to two non-negligible problems. The first problem happened due to approximating the common boundary by two potentially different functions (that is, one for each of the two neighboring color regions), resulting in either overlaps or gaps appearing between the two regions in the reconstructed image (Figure 3). The second problem is related to the negative effects on efficiency caused by duplicate tracking that potentially leads to increase in processing time and size of coded data.

In this paper, a method on tracking common boundaries of color regions that is crucial to function approximation-based image coding is proposed. By applying this method, duplicate tracking is avoided leading to improved performance (in terms of processing time and coding efficiency) while maintaining the quality of reconstructed images, both original and resized.

The rest of this paper is organized as follows. In Section 2, the proposed boundary tracking method is described, with the major processing steps summarized in flowcharts. To facilitate understanding, the tracking result based on a similar image as Figure 2 is presented for illustration. In Section 3, experimental evaluation verifying the effectiveness of the proposed method is presented. Lastly, conclusions are drawn in Section 4.
II. PROPOSED BOUNDARY TRACKING METHOD

In Figure 4, the main processing steps of function approximation-based image coding from which this work originated are shown. Here, it is assumed that the initial stage of segmenting the input raster image into color regions has been previously performed. The work reported here focuses on the second processing step in which closed boundaries of color regions are traced and extracted. The third or last major processing step is the automatic selection of functions (straight line, curve, or arc) to approximate the segments that comprise these boundaries based on an approximation error tolerance, and is similar to the procedure adopted in [3]. As a result, the differences in processing time, compression rate and the visual quality of reconstructed images can be attributed solely to the proposed boundary tracking method.

A. Method Description

The accurate tracking of boundaries of regions is crucial to the overall quality of function approximation-based image coding. In the case of bi-level raster images, boundary tracking methods based on the conventional chain code technique have often been used [4]. The result of tracking, which is common to all these methods, is sequences of pixels that each constitutes the closed boundary of an image region. During function approximation, the array of $(x,y)$ coordinates for every sequence of pixels are used to produce a pair of parametric representations, $(t_k, x_k)$ and $(t_k, y_k)$, where $k = 0, 1, \ldots, N-1$ and $N$ is the length of the sequence. Based on these parametric representations, a pair of functions $S_x(t)$ and $S_y(t)$ can be established, which are approximated simultaneously by determining a combination of line segments (straight line, curve, and arc) that comprise the boundary within the preset approximation error tolerance.

In our recent research that targets color document images, a boundary tracking method for bi-level raster images was simply applied [3]. Due to a major drawback in conventional methods that focus only on a single color region at a time during the tracking process, boundaries that are common to two neighboring color regions were being tracked in duplicate.

**Figure 4.** Main processing steps of function approximation-based image coding

Duplicate tracking led to two non-negligible problems. First, overlaps or gaps might occur at common boundaries due to different approximation functions. Second, increase in processing time and size of coded data could happen.

To address these problems, a novel boundary tracking method is proposed that avoids duplicate tracking the common boundaries of neighboring color regions. The main difference between the boundary tracking method applied in [3] and the one proposed here lies in the processing that follows after a point on the boundary at which three or more colors meet is detected. At that moment, the current position and other necessary tracking information are registered. From then, a boundary common to two of these color regions is selected and tracked, while ensuring that it will be tracked only once. As the tracking continues, additional points at which three or more colors might be reached, resulting in additional information be recorded and the tracking process repeated in a recursive manner. Figure 5 illustrates the result of tracking by the proposed method.

Furthermore, in order for two neighboring color regions to share a common boundary, here the corners of a pixel (rather than its center) are used to denote the coordinates along a tracked boundary as illustrated in Figure 6.

Together, Figure 7(a)-(d) and Table 1 summarize the overall processing of the proposed tracking method.

**Figure 7(a)** The “Main” Routine
Figure 7(b) Routine “Track One Common Boundary”

Repeat until tracking position and direction match initial position

Register tracking position as part of current boundary

If tracking direction is up
Mark the front left pixel in order to avoid duplicate tracking

If tracking direction is down
Mark the front right pixel in order to avoid duplicate tracking

Search lookup table to decide the next tracking direction

Advance one pixel in the current tracking direction

If tracking is finished
Return the tracked boundary and branches at its end point to caller

Return the tracked boundary and no branch to caller

Table 1. Lookup table to decide the next tracking direction

<table>
<thead>
<tr>
<th>Right (A=C=D)</th>
<th>Forward (A=C and B=D)</th>
<th>Left (B=C=D)</th>
<th>Finish* (neither of other patterns)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Current position

Forward

Left

(*) Determine the next direction using surrounding pixels is preferable than terminate the tracking. For example, if more surrounding pixels have the color of A than of B, the next direction may be “Right”. In the converse, the direction could be “Left”.

Call “Track One Common Boundary” with tracking position

If the tracked boundary has no branch
Create a new region, and return both the tracked boundary and the region to the caller

Mark the backward branch of the tracked boundary as part of an outside region

Register all branches at end point of the tracked boundary in non-tracked branch list

Repeat until non-tracked branch list becomes empty

Remove a branch from non-tracked branch list, and set tracking position and direction to this branch

Call “Track One Common Boundary” tracking position

If the backward branch is in the non-tracked branch list
Remove this backward branch from the

Register branches at end point of the tracked boundary in non-tracked branch list, except the backward branch

Register both the tracked boundary and its

Call “Compose Regions from Boundaries”

Return to caller with the local boundary list and region list

Figure 7(d) Routine “Track Common Boundaries”

B. A Simple Illustration

The image in Table 3 is composed by three color regions (the background is also counted as one color), sharing a single common boundary. The small letters \{a,b,c,d,e,f\} denote the branches together with their tracking directions at the point where three or more colors meet is detected. The capital letters \{A,B,C\} denote the three boundaries and \{Arev, Brev, Crev\} denote them in reverse direction.

Table 2 and Table 3 summarize the steps involved in tracking the sample image by the proposed method. The end result is that boundary ‘B’ is determined as a common boundary and duplicate tracking was avoided.

Table 3. Sample image and tracked boundaries

<table>
<thead>
<tr>
<th>Tracked boundaries</th>
</tr>
</thead>
<tbody>
<tr>
<td>A(outside), Arev</td>
</tr>
<tr>
<td>A(outside), Arev, B, Brev</td>
</tr>
<tr>
<td>A(outside), Arev, B, Brev, C, Crev</td>
</tr>
</tbody>
</table>
III. EXPERIMENTAL EVALUATION

Two Microsoft bitmap images in 8bit/pixel are used for experimental evaluation and are shown in Table 4. The software for image coding was written in C, while the computer used for the experiments is a Pentium III 1.1GHz PC with 632MB main memory.

Table 5 shows the outlines of two reconstructed images for Test image #1 being enlarged by 3 times the original size, with the one on the left tracked by the proposed method while the one on the right by the conventional method as applied in [3]. It is clear that overlaps and gaps are observed in the reconstructed image on the right which do not occur in the one on the left.

Similarly, Table 6 shows an area in the reconstructed images for Test image #2 being enlarged to 3 times the original size. By comparing these two reconstructed images, overlaps and gaps are clearly observed at the places that are highlighted in the reconstructed image on the right which do not occur in the one on the left.

Next, the proposed method is compared with the conventional method based on the amount of reduction in processing time and the reduction in total length of tracked boundaries. The results are summarized in Table 7.

From Table 7, it is clear that for the two images, the processing time of image coding when the proposed method is applied is less than when the conventional method is used. The reductions are 66% and 59% respectively. In terms of coded data size, the experimental results show that the total length of tracked boundaries is smaller when the proposed method is applied than when the conventional method is used. Based on these results, it is verified that the proposed boundary tracking method contributes to the reductions in both processing time and size of coded data in image coding.

IV. CONCLUSIONS

A boundary tracking method that solves the problem of duplicate tracking common boundaries of color regions in function approximation-based image coding is proposed. Effectiveness of the method is confirmed by experiments.

REFERENCES


Table 4. Two bitmap images used in experimental evaluation

<table>
<thead>
<tr>
<th>Test image #1</th>
<th>Test image #2</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Image 1]</td>
<td>![Image 2]</td>
</tr>
</tbody>
</table>

Table 5. Compare proposed and conventional method on reconstructed outlines of Test image #1 (3x enlarged)

<table>
<thead>
<tr>
<th>Proposed method</th>
<th>Conventional method</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Image 3]</td>
<td>![Image 4]</td>
</tr>
</tbody>
</table>

Table 6. Compare proposed and conventional method on an reconstructed area in Test image #2 (3x enlarged)

<table>
<thead>
<tr>
<th>Proposed method</th>
<th>Conventional method</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Image 5]</td>
<td>![Image 6]</td>
</tr>
</tbody>
</table>

Table 7. Compare processing time and total length of tracked boundaries between the proposed and conventional method

<table>
<thead>
<tr>
<th>Method</th>
<th>Processing Time (seconds)</th>
<th>Total length of tracked boundaries</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Image #1</td>
<td>Image #2</td>
</tr>
<tr>
<td>Conventional</td>
<td>1.260</td>
<td>10.037</td>
</tr>
<tr>
<td>Proposed</td>
<td>0.426</td>
<td>4.158</td>
</tr>
</tbody>
</table>

855