Segmentation for Digital Copying and Compression

ディジタルコピーおよび画像圧縮用領域分離

Abstract

Segmentation in digital copiers is designed to improve overall image quality by explicitly identifying significant elements in the input that require diverse and specialized treatment with respect to filtering, halftone generation, and marking strategies. Similarly, segmentation can greatly improve compression ratios while preserving image quality by selecting, for each region of the image, the compression method best suited to the content of that region. However, the performance requirements in digital copier and compression scenarios impose serious constraints on the extent and nature of feasible computation. In this paper we define the segmentation problem and discuss its implications for copier and compression tasks. We also examine the various tradeoffs involved with each task in terms of computation and image quality. We then show examples of two original segmentation methods: one achieving reasonable segmentation within the constraints of a low-cost copier product; the other producing high compression ratios while preserving image quality.

Introduction

Segmentation has emerged as a key technology in many areas of digital imaging and video, because it supports selective processing which can result in improved quality and performance. In digital copiers for example, segmentation can improve overall image quality by explicitly identifying significant elements in the input that require diverse and specialized treatment with respect to filtering, halftone generation, and even marking strategies. Similarly, segmentation can greatly improve compression ratios while preserving image quality by selecting, for each particular area of the image, the compression method best suited to the content of that region.

Historically, image segmentation has entailed a partition of the set of image pixels into a collection of equivalence classes—a set of contiguous subsets called regions. That is, the pixels of each region are connected and are roughly similar in terms of the particular defining image properties measured by the segmentation algorithm. Among the image properties typically used are: texture, color, edge information, and more recently wavelet signatures. Reed and Du Buf [12] offer a comprehensive survey of texture-based segmentation, and Ohta, et al. [10] do the same for color. For one specific treatment of wavelets, see Laine and Fan [9].

Segmentation may be computed by an enormous variety of algorithms. Haralick and Shapiro [5] have created a useful taxonomic survey of segmentation techniques including: thresholding, region growing, spatial clustering, and split-and-merge schemes. See also Chapter 7 of Gonzalez and Woods [3] for an overview. Additionally, there is a large literature on page or document segmentation. For specific examples, see Pavlidis and Zhou [11], Haralick [4], and Jain and Zhong [8].
The primary shortcoming of these "classic" approaches is that they pose the segmentation problem as a global optimization. This means that the data of the entire image must be available even if it is sub-sampled as in Ancin [1]. We will show that this global context is simply too expensive in terms of memory and in terms of computation, given the high throughput requirements of digital copier and compression scenarios.

1. Segmentation for Digital Copying

The advent of digital imaging technology in the field of printing and reprographics has brought with it the opportunity and the necessity to perform segmentation. Segmentation is necessary as a direct consequence of the various artifacts that digital sampling engenders; and segmentation is possible due to the discrete nature of the pixel mosaic, which provides the spatial context necessary for reliable feature discrimination.

A digital copier is comprised of a digital scanner at the front end, a digital print engine at the back end, and a sequence of digital transformations of color and spatial information in between. The scanner samples the input document as an image raster—a series of digital scanlines composed of discrete samples called pixels. This digital image raster is transformed in a sequence of processing steps that frequently entail resampling the image data. (Digital zooming is one example.) Finally the image is rendered to the digital printer at the back end of the copy pipeline. The printer is typically a bitonal device, which means that the output image must be halftoned—yet another sampling of the image data.

1.1 Objectives

The primary objective of segmentation in the context of digital copying is to improve copy quality by overcoming the adverse effects of digital sampling through selective enhancement of each type of element in the document.

Digital sampling differentially affects the elements in an input document and can seriously compromise the quality of the resulting copy. In digitally scanning an input document, text boundaries are subject to spatial aliasing and blurring, halftone areas may exhibit moiré, and contour (photographic) areas may experience quantization effects like contouring. The figure illustrates such problems in the text and halftone areas of a sample document. In the original scan shown in panel (a), the boundaries of the text are jagged and somewhat blurred—as seen in the close-up inset at the top right. At the same time, the halftone patches exhibit moiré. This figure also demonstrates the limited efficacy of applying a single global solution like simple filtering. In panel (b), moiré is reduced by smoothing but at the expense of text clarity. In panel (c) on the other hand, text legibility is enhanced by sharpening, but the cost is increased moiré in the halftone areas. Clearly, what is required is selective processing according to the element type—smoothing halftone areas and sharpening text areas. But to do so requires explicitly identifying the locations of...
each element type.

Segmentation addresses such problems by explicitly identifying different element types in the scanned document. This explicit information can support remedial processing specifically tuned to each type of error introduced in scanning. It can also be used to select, for each area of the image, the proper set of processing steps that will best enhance the image quality of the element type of that region—thereby avoiding additional problems, which might be introduced by the copier processing pipeline itself. For example, textual elements are processed specifically to increase their clarity and legibility. Finally, the explicit information provided by segmentation can help optimize the rendering process—e.g., to ensure that black text is printed with pure black colorant (K) rather than as a composite black (a mixture of C, M, and Y).

1.2 Constraints

The requirement for high throughput in a copier scenario mandates a specialized hardware solution, such as an ASIC or a DSP. This seriously constrains the design of the segmentation algorithm since there is often no floating-point support and no division—and, even if available, such operations are slow.

Since the selected algorithm must ultimately be mapped to hardware, complexity in both time and space are directly related to product cost. Complexity of time corresponds to the depth of the cascade of gates in an ASIC or the number of compute cycles in a DSP; complexity of space corresponds to the amount of image data and intermediate results that must be buffered for the algorithm to compute an output. In practice, it is feasible to buffer only a small number of image rows.

For segmentation, this means that many algorithmic strategies are simply not practical. In particular, any strategy requiring global context is excluded due to the prohibitively high cost of buffering the entire image—a standard letter-sized color document 8.5" × 11" sampled at 600 dpi would require buffering roughly 33 million pixels or 100 MB of 24 bit RGB image data. Thus most classic segmentation algorithms, which rely on accessing the entire image, are excluded out of hand.

1.2.1 Data stream

Because it is impractical to buffer the entire image, the data must be processed as a stream—in scanline order. A few rows of data are typically buffered in a queue or FIFO structure so that, although processing is limited to a narrow band of the image at any instant, a modicum of two-dimensional spatial context is provided. Such context is necessary for the sorts of neighborhood computations involved in feature discrimination.

A FIFO structure, by definition, flushes the oldest item (a row, in this case) as a new one is inserted. This imposes the further constraint that the algorithm must operate in a single pass over the image data stream, since it is impossible without rescanning to retrieve data which has been flushed. And, since rescanning entails significant additional time, this is simply not an option. Again, this excludes many classic segmentation techniques.

1.2.2 Small neighborhoods

Just as the cost of a FIFO severely limits the number of rows that may buffered, so the cost of gates (or cycles) limits the width of the neighborhood that may be reasonably considered. Simply put, larger neighborhoods mean higher cost—an $N \times N$ neighborhood entails $N^2$ operations per color channel per location.

Although cost dictates small neighborhood operations, the problem is that many useful computations like low-pass filtering require large spatial support, equating to large neighborhoods. More significantly, unambiguous discrimination of element types, such as between text and halftone, may require extensive spatial context. One must therefore balance the cost of spatial context against the accuracy of classification.

One solution—adopted here—is to focus processing primarily in a single color channel thereby permitting larger spatial neighborhoods for the same gate/cycle budget. However, no neighborhood (even the entire image) is sufficient to completely disambiguate every case. The conclusion, therefore, is that segmentation in the copier domain is inherently error-prone and processing that depends upon it must take this into account.

1.3 Segmentation Example

We illustrate our segmentation algorithm using three classes appropriate for low-end and mid-range color copiers. The algorithm partitions image pixels into three exclusive classes (TEXT, HALFTONE, and BACKGROUND) by combining the information of three distinct modalities: structure, statistics, and color. These modalities correspond to three computational modules, respectively: text localization, halftone identification, and dark pixel detection.

As mentioned before, one strategy to optimize the compute and memory budget for color images is to focus the majority of processing in a single, color channel. The raw
RGB output of a typical scanner is not a good choice since information is fairly evenly distributed across all three channels. However, by first transforming the image to a luminance-chrominance-chrominance (LCC) color space, most of the spatial information can be isolated in the luminance channel while the color information is contained in the two chrominance channels. This means that the spatial image processing components of the segmentation algorithm can concentrate on the L channel alone. Figure 3b shows a color document example and the corresponding L channel data.

The impact of segmentation errors on copy quality depends on subsequent processing. Typically, in digital copying the segmentation result is used to select filters and to control black generation—i.e., the use of pure versus composite black in generating neutral tones (grays). In a low-end copier, BACKGROUND and HALFTONE areas are often treated identically, and the analysis here is given in this context. Similar shows typical output of our segmentation algorithm for the example image of Figure 3. Here purple indicates background; green indicates halftone; and black indicates the boundaries of text.

Representative errors are illustrated in Figure 3b, where T+ indicates a text false positive, T- a text false negative, H+ a halftone false positive, and H- a halftone false negative. Here is a brief analysis of each category of error.

- The circled area labeled H- and others like it have been mistaken for BACKGROUND. Such errors have a negligible effect on copy quality since BACKGROUND and HALFTONE are treated similarly. In addition, such
areas can be recovered by simple post-processing like a morphological closing.

- The circled area labeled H+ and most other examples are small isolated false positives which will also have little effect on copy quality as long as the overlaying text has been correctly identified.

- The circled area labeled T+ has been misclassified as TEXT. It is on the border of HALFTONE and a very dark region of the same halftone image. Most other text false positives are of a similar nature. The result is to sharpen the edge in this area and to paint a curve of K. The overall perceptible impact is hard to judge in the abstract.

- The circled area labeled T- is at the intersection of the vertical and horizontal strokes of a large-sized lower case "t." In this case, a smoothing filter is applied so text boundaries in this area will be blurred. However, because the type is bold face and the type size is large (about 24 pt), decreased legibility is not really a factor. Nor is the density of black in the interior of the character adversely affected.

Based on this example, the potentially most serious errors are text false positives, since they may result in perceptible artifacts. Whereas, post-processing may help to eliminate other types of errors. And finally, a blending function can be used to transition between filters and thereby to minimize the apparent impact of transitions between TEXT and HALFTONE regions.

One final note: it is precisely because of the size and thickness of the strokes in the text false negative example that it is not properly classified as text. This is the price of restricting the size of the processing neighborhood—it fails to provide sufficient spatial context to resolve such areas as text. More to the point, within the processing window this part of the character looks identical to a dark uniform patch in a halftone or contone area.

In the following section, the issues and objectives surrounding segmentation for digital compression are discussed, and some results of our current work in this area are offered.

### 2. Segmentation for Digital Compression

The large size of files containing scanned or electronically generated images makes compression an absolute necessity for storage and/or transmission of such information. However, typical image compression algorithms, like JPEG and JBIG, are inadequate because they are designed to treat one specific image class (natural images and binary images, respectively), and most material is not simple in terms of image content. Often, the images are compound images that contain a mixture of text, graphics, line drawings, photographs and other natural images. Therefore, using one compression algorithm will not achieve the best compression ratio or retain image quality in regions outside the design scope of the algorithm. For example, images that have been compressed using JPEG contain annoying ringing artifacts around high frequency material, especially text. A well-designed compression scheme that treats portions of the image differently based on content will eliminate these artifacts while improving the compression ratio. One important part of such schemes is image decomposition or segmentation.

There are several ways in which an image can be decomposed. One natural way is to label different image regions spatially. A three-layer model shows such a segmentation. Note however, that it is difficult to label the region in the lower one-third of the image. This is clearly a text region, but underlying the text is a natural image. The image format described in the following alleviates this problem.

The International Telecommunications Union (ITU) has developed a new image representation standard, ITU-T Recommendation T.44 [7], also referred to as "Mixed Raster Content (MRC)," to permit efficient compression of pages containing compound-image content. An MRC page consists of one or more layers, and each layer is coded independently with one of the ITU-approved compression algorithms enumerated in the T.44 Recommendation. Therefore, in the MRC approach, different compression algorithms can exist within a page.

A three-layer model is the basis of the MRC Recommendation. The three layers are referred to as: the background layer, which contains contone color (continuous-tone and/or palletized color) imagery; the foreground layer, which contains the colors of text, graphics, and line art; and the mask layer, which is bi-level and selects between the foreground and background layers for pixel reproduction. Illustrates this three-layer image decomposition. The white pixels in the mask indicate foreground, and the black pixels indicate background. While T.44 provides for the processing, interchange, and archiving of the multiple separate layers, the segmentation or decomposition of an image into the layers is not prescribed by the Recommendation and, thus, remains an encoder issue. The layered image decomposition is not unique to T.44. AT&T uses the same decomposition in their compression scheme,
DjVu [2], for images consisting of mixed content. However, while T.44 has a selection of approved compression algorithms for each layer, AT&T uses their own proprietary compression algorithm for coding each layer. While any segmentation may be used, in order to be T.44 compliant only those compression algorithms enumerated in the Recommendation may be used for the compression of the layers.

Xerox also uses the MRC approach in their color document image representation DigiPaper [6], and Xerox has been a major proponent and contributor to the development of the T.44 Recommendation.

An important aspect of compound-image coding is the interaction between the segmentation and the choice of the compression algorithm for each respective layer. There must exist an appropriate compression algorithm for the different layers into which an image is decomposed. The existence of robust, effective algorithms for compression of bi-level images (JBIG and JBIG2) and natural images without high intensity, high frequency content (JPEG and JPEG 2000) makes the foreground/background/mask layers a natural choice for decomposition. If new compression algorithms are developed that are particularly effective on images with specific attributes, then layers which reflect those attributes would be desirable.

2.1 SLA Activity

At SLA, we have an ongoing research program in the areas of segmentation and improved compression algorithms for the coding of compound images. The constraints described in Section 2.2 are relevant to the application of these algorithms to printer/copier products, and as such, limit the computations and algorithmic steps we employ in our methods.
shows a portion of an image compressed and decoded using both SLA's method and AT&T's DjVu method. In our method, the mask layer was compressed using JBIG, the background layer using JPEG-2000, and the foreground layer using a palette reindexing method developed at SLA [13] and JPEG-LS. The compression ratios are 8.6:1 and 7.1:1 for SLA and AT&T, respectively.

Conclusions

Segmentation has emerged as a key technology in many areas of digital imaging and video, because it supports selective processing which can result in improved quality and performance. In digital copiers for example, segmentation can improve overall image quality by explicitly identifying significant elements in the input that require diverse and specialized treatment with respect to filtering, halftone generation, and even marking strategies. Similarly, segmentation can greatly improve compression ratios while preserving image quality by selecting, for each particular area of the image, the compression method best suited to the content of that region.

However, the performance requirements in digital copier and compression scenarios impose serious constraints on the extent and nature of feasible computation. Typically, computation is severely restricted in terms of memory, operation count, arithmetic support, etc. Conversely, segmentation can involve a suite of rather elaborate algorithm–search requiring significant computational expense.

The problem is further complicated by the fact that perfect segmentation is unattainable. Memory restrictions mean that the spatial context of processing neighborhoods is generally insufficient to unambiguously classify features. On the other hand, larger processing neighborhoods entail increased computation; and, even so, it is not clear that this additional computation would produce an error-
free result. Moreover, depending on the particular components of the image-processing pipeline, certain errors can be catastrophic while others may be hardly perceptible. The goal then is to balance the added cost of an inevitably flawed segmentation against the potential improvements in image quality that it supports.

References