Vol. 6, No. 3: 37–43

Contour Extraction of Compressed JPEG Images

Yair Wiseman
Bar Ilan University

Erick Fredj
The Jerusalem College of Technology

Abstract. We present a new algorithm that uses the JPEG compression format and provides a significant improvement in contour extraction for medical imaging, suc as X-ray photos. The algorithm searches for an object's contour in the compressed file. The complexity is reduced to $O(n)$ in the case of a high resolution picture, which is better than the state-of-the-art for contour extraction.

1. Introduction

Images are often stored in compressed form. The standard approach to image processing is first to decompress the image, and then to process the raw image data. For some image operations, however, we can use the compressed data directly. This has two advantages: first, we save the work of decompressing the image; second, we can use the frequency information embedded in the compressed data.

Our application focuses on contour extraction—finding a set of points on the boundary of an image—for X-ray images. The challenge for traditional radiologic interpretation is the inherent difficulty of reliably extracting contours in an X-ray image, since:

1086-7651/01 $0.50 per page

- X-ray images are noisy and have limited resolution (see Figure 2);
- X-ray images exhibit non-uniform exposure variation across the field of view, with contrast and exposure varying from shot to shot;
- X-ray images include irrelevant contours.

The three main approaches to contour extraction that are generally considered are edge detection, region growing, and active contours. The existing low-level techniques do not provide a satisfactory solution for automatic contour extraction.

Some work on searching patterns in a compressed file has been done in the past [Navarro, Raffinot 99], [Klein, Shapira 2000]. This article introduces a method of looking for objects in an X-ray picture compressed in standard JPEG compression form.

2. The JPEG Standard

JPEG [Wallace 91] is a well-known standard for image compression which has also been used in medical applications [Wittenberg 93]. Although JPEG loses some data in the compression process, the human eye usually cannot distinguish the difference between the original images and the JPEG picture.

There are three stages in the JPEG compression process:

- First, the image is split into blocks of 8×8 bytes. Each block is used as input to the Forward Discrete Cosine Transform (FDCT) [Rao, Yip 90].
- The second stage of the processing is quantization. The DCT coefficients created in the previous stage are reduced to smaller integers. (The image creator determines the degree of reduction.) Rounding to an integer causes information loss, but this loss allows the image to data to be stored using less space. If the numbers are reduced significantly, the information loss will be large; if the numbers are not reduced significantly, the information loss will be less. One usually tries to find a compromise which provides a significant reduction in storage space with the least amount of damage to the picture quality. The reduction method and some experiments done with it are described in [Hwang, Yang 95].
- The last stage is the entropy encoder, in which the data from the previous stage is compressed. The JPEG standard [ISO/IEC 93] defines two methods for this compression: one is a version of Huffman coding [Huffman 52], and and the other is a version of arithmetic coding [Witten et. al. 87]. (Baseline JPEG uses Huffman coding.)

Decompression, on the other hand, is an inverse process. First, the Huffman codes are decompressed, and then the inverse of the quantization process that was completed during compression is done. In this stage, the small numbers are increased to be close to the original DCT coefficients. The last stage in decompression is an Inverse Discrete Cosine Transform (IDCT) performed on the data from the previous stage.

3. JPEG Contour Extraction Algorithm

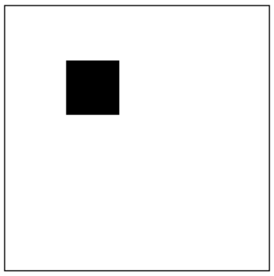
This section describes an edge detection algorithm that uses the JPEG format. As mentioned above, the JPEG standard is based on the DCT paradigm, which changes an image into a frequency space. The frequency coefficients that are of low magnitude are rounded to zero, and quantization reduces the larger coefficients. When most of the coefficients in a given block are zero or have very low magnitude, the compression algorithm will assign a very short bit sequence to the block. Zero sequences are treated very efficiently by JPEG compression. However, when there is a drastic change in an 8×8 block many frequency coefficients will be high, and the sequence will require many more bits. In the JPEG standard a block must be 8×8 pixels, but the algorithm obviously will be good for other small $N \times N$ pixels sizes, too.

When determining the contour of an object, the goal is to find the object's border. The compressed file is broken into its original blocks and then searched for long bit sequences. The blocks then are presumed to be the object's border. The technique can be summarized as follows: (t is a threshold value, b is a block number and sb is the block length.)

```
for each decoding block $b$
  set $sb$ =size_of_block ($b$)
  if ($sb > t$)
    then $b$ is an edge and paint it according to $sb$
  if not at end (EOB) return to step 1
```

If we have no idea what the threshold value t should be, we can examine the probability density function (PDF) of the block representation to select a suitable value. In the uncomplicated case, the PDF should be mono-modal and we set the value to the inflection point.

A weakness of this algorithm is that it does not use the spatial information in the blocks, and so identifies any sequence of bits as a contour of the object. Our assumption is that the resolution is sufficient to describe the shape's character.



0	0	1	0	1	0

JPEG format of uniform area.

-12	18	-8	-2	3	-1	-1	1
22	-25	8	2	-3	0	1	-1
-18	20	-7	-2	2	0	-1	1
16	14	4	1	-1	0	1	1
-11	10	-2	-1	0	-1	-1	0
6	-5	1	-3	0	0	0	0
-2	2	-1	0	0	0	0	0
1	-1	0	0	0	0	0	0

JPEG format of the upper left corner of the square block.

Figure 1. Demonstration of JPEG used for contour extraction. The left image shows the original image, which is a high-resolution picture of 1000 × 1000 pixels. The upper-right table shows the JPEG format in the white or black area. The lower-right table shows the JPEG format in the upper left corner of the black square. The size of the black square is 200 × 200 pixels, and the square is not aligned relative to JPEGs 8 × 8 blocks.

4. Examples and Discussion

Figure 1 shows a sample image and how JPEG can be used for contour extraction. The original image is compressed in a grayscale baseline JPEG format with 75% quality.

The JPEG file reports the difference of magnitude between the discrete cosine coefficients of a previous block relative to the current block. For the case of a white or black area there are no changes in the magnitude of the coefficients. This type of block is encoded with six bits by the JPEG standard [ISO/IEC 93], as shown in Figure 1. The first two bits "00" indicate that there are no differences between the discrete cosine coefficients values of the previous and the current block; the last four bits "1010" indicate the end of the block.

If there is a difference in the intensity of the DC coefficients, the encoding block will be slightly larger. For example, a block that encodes a sharp change from white to black is represented by a wide range of frequency coefficients. It is easy to select a threshold that delimits the edge of the shape from the rest of the image. Figure 1 shows a sample of the block containing the upper left corner of the black square. In JPEG standard, 243 bits are needed in comparison to the 6 bits required for a uniform area. The difference between 6 and 243 is obviously significant. By using three parameters—the length of the block, its magnitude, and the number of consecutive blocks—the threshold

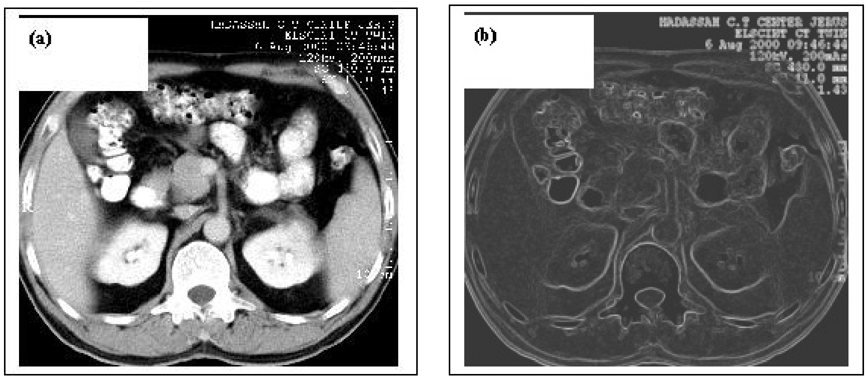


Figure 2. (a) Abdomen high-resolution picture of 1600 × 1200 pixels courtesy of the Hadassah CT center Jerusalem; (b) this image illustrates the JPEG extraction algorithm.

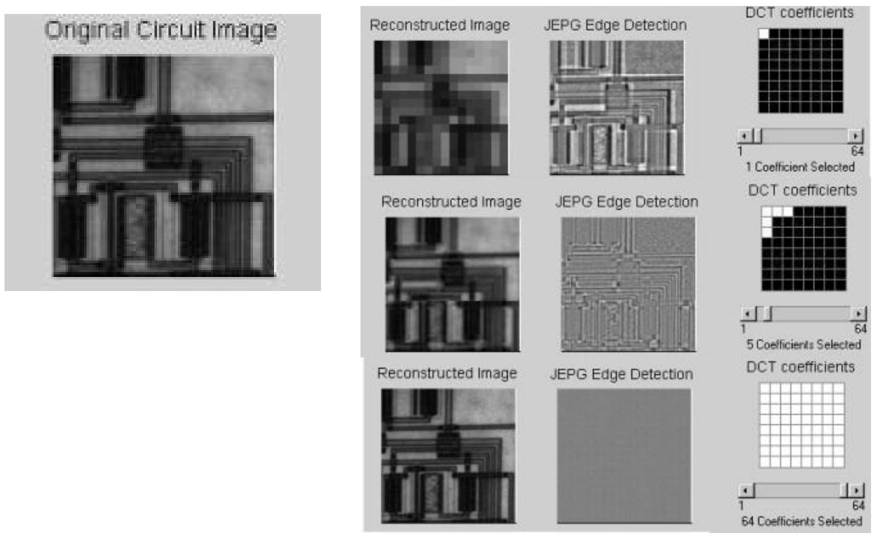


Figure 3. The picture of 1600 × 1200 pixels demonstrates how the number of selected DCT coefficients influences the extraction algorithm. The left image shows the original image. The right image shows the reconstructed image (top left) and the result of the procedure for various selected DCT coefficients.

can extract the contour with a range of scalar values. These extra parameters allow more control over the resulting mechanism.

Noise in the image can decrease the effectiveness of the simple threshold. This method can be useful for extracting large objects from an image. Since blocks of 8×8 pixels represent the border, an object of less than 8×8 pixels can be lost and for such objects, we must use another method. However, when a large object is to be found, the method described here works well. If noise occurs in the image, it can increase the size of the block so the algorithm recognizes it as a border of the object. If the resolution is very high, the square with the noise will be very small, and that noise will be noticeable in almost the same way as in the original image, with just a few more pixels. (The noise will be square regardless of the original shape of the noise.)

Let n be the number of pixels in an image. The current state-of-the-art algorithm for edge detection in a JPEG image is $O(n \log(n))$, since decoding DCT is $O(nlog(n))$[Jain 89]. Our algorithm, however, is $O(n)$. Every bit is read only once to determine the location of the EOB. When the EOB is detected, the byte count is checked to determine whether the block contains an object's contour. In fact, only the compressed file (which is smaller than the original file) is checked. (The size of a compressed file is linear in the size of the uncompressed file.) Therefore, this algorithm is $O(n)$.

We demonstrate the algorithm on a high-resolution image. Figure 2(a) shows a 1600×1200 image courtesy of the Hadassah CT center in Jerusalem. Figure 2(b) shows the result of the JPEG contour extraction algorithm.

We can control the quality of the edge dectection by selecting the number of DCT coefficients, as illustrated in Figure 3. The best contour extraction is obtained when we select all 64 DCT coefficients.

References

[Huffman 52] D. Huffman. "A Method for the Construction of Minimum Redundancy Codes." In *Proceedings of the IRE 40*, pp. 1098–1101, Kansas City: IRE, 1952.

[Hwang, Yang 95] J. J. Hwang and G. H. Yang. "Optimal Variable Quantization for JPEG Extensions." In *IEEE International Conference on Consumer Electronics*, pp. 266–267, Singapore: IEEE Computer Society Press, 1995.

[Jain 89] A. K. Jain. *Fundamentals of Digital Image Processing*, pp. 150–154. Englewood Cliffs, NJ: Prentice-Hall, 1989.

[Klein, Shapira 2000] S. T. Klein and D. Shapira. "A New Compression Method for Compressed Matching." In *Proceedings Data Compression Conference, DCC-2000*, pp. 400–409, Snowbird, Utah: IEEE Computer Society Press, 2000.

[Navarro, Raffinot 99] G. Navarro and M. Raffinot. “A General Practical Approach to Pattern Matching over Ziv-Lempel Compressed Text.” In Proceedings 10th Symposium On Combinatorial Pattern Matching, LNCS 1645, pp. 14–36, Berlin: Springer Verlag, 1999.

[Rao, Yip 90] K. R. Rao and P. Yip. *Discrete Cosine Transform Algorithms, Advantages, Applications*. London: Academic Press Inc., 1990.

[Wallace 91] G. K. Wallace. “The JPEG Still Picture Compression Standard. *CACM* 34: 3–44 (1991).

[Witten et. al. 87] I. H. Witten, R. M. Neal, and J. G. Cleary. “Arithmetic Coding for Data Compression.” *CACM* 30: 52–54 (1987).

[Wittenberg 93] U. Wittenberg. “Application of the JPEG Standard in a Medical Environment.” SPIE Video Communication and PACS for Medical Applications.” pp. 121–129, Berlin: SPIE-The International Society for Optical Engineering, 1993

[ISO/IEC 93] *Information Technology Digital Compression and Coding of Continuous-Tone Still Images Requirements and Guidelines International Standard*. Jersey City, NJ: ISO/IEC 10918-1, 1993.

Web Information: http://www.acm.org/jgt/papers/WisemanFredj01

Yair Wiseman, Math & Computer Science Department, Bar Ilan University, Ramat Gan 52900, Israel (wiseman@cs.biu.ac.il)

Erick Fredj, Computer Science Department, The Jerusalem College of Technology, P.O.B 16031, Jerusalem 91160, Israel (fredj@mail.jct.ac.il)

Received November 15, 2000; accepted in revised form October 22, 2001.