An O(n) Algorithm for Edge Detection in Photos Compressed by JPEG Format

Erick Fredj and Yair Wiseman
Computer Engineering Department, The Jerusalem College of Technology
P.O.B 16031, Jerusalem 91160
Israel

ABSTRACT

This paper presents a direct method for edge detection in images compressed by JPEG format. The algorithm reduces the magnitude of the edge detection algorithm from the known O(nlog(n)) into O(n) in many cases. In parallel, the algorithm can reduce the magnitude into O(1).

KEY WORDS: edge detection, contour extraction, compression, JPEG

1. INTRODUCTION

One of the basic classical applications of imaging technology is to represent images as a set of pixels. Each pixel is defined by a numerical value, and is separately manipulated. There are many ways to store image data without degrading the original data. Some of these ways lose part of the data, but it usually isn't noticeable. For example the BMP[1] standard is a well known format for image storage without data compression, while the GIF[2] standard is a well known format for image storage which uses the LZW[3] algorithm for compressing the image's data.

Edge Detection [4,5] is the task of finding a set of points on the imaged objects' boundaries. The three main approaches for edge detection generally considered are contour extraction, region growing, and active contours. These existing low-level techniques do not provide a satisfactory solution for automatic edge detection.

Contour extraction used in many cases. e.g. finding objects in an X-Ray Photos[6]. Contour extraction tries to find edges of objects in an image. An edge in an image appears as significant local change in intensity. The gradient of a function measures the amount of local change; thus a gradient value above a certain threshold implies an edge. Considering the gradient values of an image, peaks in the gradient function imply zero crossings in the second derivative of images.

An ideal edge is a step edge but this rarely occurs in a sampled image because it is unlikely that the ended effect of the sampling process is such that it will cause pixel boundaries to line up with the edge. In practice a real scene will include deficiencies by the transfer characteristic of the sampling system such as a distortion due to lenses, defects in the image sensor array, and so on.

Region growing techniques require seed points, and a user defined similarity function. These techniques work by starting from regions that only consist of the seed points and checking for similarities in neighboring regions.

Active contour techniques work by receiving as input an initial contour and then changing the contour according to energy minimization, which defines the contour's new shape.

Edge detection is a wide area that includes both machine vision and image processing. It is used centrally in shape recognition or description. It is obviously a low-level vision operation. By this we mean that shape or geometry information are delineated by edges. If we are trying to extract shape structure of an image without using any previous knowledge, then detecting edges seems clearly a major step of the process.

Although classical edge detection algorithms have been used the pixel information, and the ability to manipulate numerically the digital image in an uncompressed file.

Until recently edge detection has been usually detected by a two stage process, in which the computation needed increased as O(nlog(n)) or by using difference operators, in which the number of operations grows as also O(nlog(n)) . Recently the authors propose a novel approach for directly detecting the edges that grows only as O(n) by working on the JPEG compressed data[7,8].

This article achieves two main goals:

- It reduces the complexity of the edge detection procedure from O(nlog(n)) known order, into O(n).
- It exploits the relevant hidden information of the JPEG data without decompressing it.

2. METHODS

The JPEG is a well known standardized image compression. JPEG is algorithm, which loses information, meaning that the decompressed picture isn't the same as the original one. By adjusting the compression parameters process, we can adjust the degree of loss. Why use JPEG? There are two principal reasons: to reduce your image files, and to store full color information.

Reducing images files is an enormous reason for transmitting files across networks and for archiving

libraries. JPEG is usually able to compress a 2 Mbyte fill down to 100 Kbytes, which produces a huge difference in disk space and transmission time. Unfortunately, even with a JPEG viewer, it takes longer time to decode and view a JPEG image than to view an image of a simpler format such as GIF, BMP, etc… .

The second principal advantage of JPEG is that it stores full color information: 24 bits/pixels or 16 million colors. In other hand, the GIF format widely used in the Internet network, can only store 8 bits/pixels or 256 colors.

Here is a brief overview of the baseline compression algorithm is:

The first step transforms the image color into a suitable color space. There are several methods to transform the image into a color space [9,10]. The most common methods are the split into RGB components[11] or the split into YUV components[12]. These components are intertwined together. The ration between these parts is usually not one by one. When YUV components are used, usually The Y component will grow to 4 times its weight. The reason for doing this is that the human eye is less sensitive to high frequency chrominance information than to a high frequency luminance which is represented by the Y component in the YUV format [13].

The next step groups the pixel values into 8X8 blocks. Then, transform each block through a Forward Discrete Cosine Transform (FDCT)[14]. The DCT gives a frequency map, with 8X8 or 64 elements. The reason for doing this is that you can now erase high-frequency information without affecting low frequency information. The DCT coefficients are composed of a single Direct Current (DC) coefficient number, which represents the average intensity level value in each block or a blocky approximation of the image and the remaining 63 are named Alternating Current (AC) coefficients. The 63 AC coefficients are ordered into a zig-zag sequence which sorts them into a one dimensional array of increasing spatial.

The second step is the *quantization*. In each block, we divide each of the 64 frequency components by a separate "*quantization coefficient*", and round the results to smaller integers. Furthermore, we note that even in the case of the quantization coefficient is 1, we lose some piece of information, because the DCT coefficients are typically real numbers.

The last step, encodes the reduced coefficients using either Huffman[15] or Arithmetic coding[16]. Here we apply only the Huffman coding. Usually a strong correlation appears between DC coefficients of adjacent 8X8 blocks. Therefore, the baseline JPEG model uses two different Huffman trees to encode the data, one for the DC coefficients' length and the other for the AC coefficients.

The JPEG space is defined by a space with three types of coefficients members field, the DC coefficient's field, the encoded AC coefficients field, and the End Of Block (EOB) field to symbolize the ending point.

Finally, all the compression parameters are written in the headers so that the decompressor module can reverse the process. JPEG's philosophy is summarized in Figure 1.

The decompression process performs an inverse process. First, it decompresses Huffman codes. Second, it performs the inverse process of the Quantization process, which was done by compression process. In this stage it raises the small numbers to larger numbers, which are close to the original numbers, of the DCT coefficient. The last stage is doing an Inverse Discrete Cosine Transform (IDCT) on the data received from the previous stage.

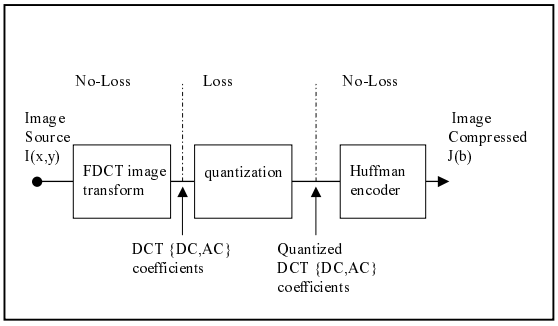


Figure 1: JPEG Model for lossy image compression

3. JPEG Edge Detection (JED) Algorithm

In this section you will find the JED algorithm described. A JED algorithm is a method of implementing discontinuity in a JPEG space that is produced by a discontinuity in the reflected light over its surface, or illumination discontinuities on the degraded image. The JED algorithm is a kind of "high pass filter". The general idea of JED is based on a search in the JPEG space. Let b be a block of 8X8 pixels. Here we have an image I(x,y) stored into the JPEG space where it is represented as J(b) by multiplying by a filter H(b). The product, $H(b)\cdot J(b)$ in the JPEG space reduced certainly the complexity of the transformation.

In the JPEG space, DCT wastes most of its data in order to represent the changes in the image, so smooth image which has no drastic changes in it, will be compressed in a small space. Actually, this fact is true for each block of 8X8. A smooth block will be compressed into a smaller space. When JPEG loses its data, most of the AC coefficients of a smooth block are rounded into zeros and it's compressed into a very small space. If the goal is edge detection, we will seek for blocks containing the edges. This makes the JED algorithm efficiency, and easy of hardware[17].

A method for extracting embedded binary data from JPEG images without decoding the image, was suggested in the past[18,19]. However these papers don't define the extracted object according to its edges. The JED algorithm will extract the objects according to their edges. A brute force algorithm of JED would first be passed through the quantized coefficient of the JPEG space. The major problem with the JED algorithm is the bottleneck.

This is when significant information is lost by "compression" of the original image during the JPEG coding. This problem can be solved by using the lossless format which explain in the JPEG standard[8]. The principle of edge detection using JED can be summarized as follows:

1. for each decoding b
2. set sb=size_of_block(b)
3. if (sb>t)
4. then b is an edge and paints it according to its sb.
5. if is not ended (EOB) return to step 1

Where

- b is a block number.
- sb is the block length.
- t is a threshold.

If we have no idea what the threshold value 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 in the inflection point. When most of the coefficients in a block are zero or very low magnitude: JPEG will give a very short bits sequence for such a block. Zero sequences are treated very efficiently by JPEG compression. When there is a drastic change in a block of 8X8, the value of many frequency coefficients will be high. Such a sequence will be compressed into many more bits: this is clearly an edge.
JED will work well with JPEG2000 standard[20] and can be adapted easily to JPEG's architectures, which implement the JPEG standard algorithm in a dedicate hardware as was mention above.

In [21] a method for parallel decoding of a JPEG picture is presented, exploiting the tendency of Huffman codes to resynchronize quickly in most cases. This paper can be considered as a continuation of it. If there are processors as many as need, the method of [21] can be the first stage for decoding the image. Then JED can do the edge detection in parallel with no dependence between the processes. Hence when there are infinity processors, the complexity of parallel JED will be O(1).

4. Results and Discussions

1. N × N FDCT and IDCT size influence

In this subsection we analyze how influence the Discrete Cosine Transform or (DCT) on the JED algorithm. In the JPEG format the source image samples are grouped into 8X8 blocks, shifted from unsigned integer to signed integer, and input to Forward DCT. The following equations are summarized the mathematical definition of the N × N (FDCT) and N × N (IDCT).

(1)
$$F(u,v)=C(u)C(v)\left[\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x,y)*\cos\frac{(2x+1)u\pi}{2N}\cos\frac{(2y+1)v\pi}{2N}\right]$$
$$F(u,v)=C(u)C(v)\left[\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x,y)*\cos\frac{(2x+1)}{2N}\cos\frac{(2y+1)}{2N}\right]$$

(2)
$$f(x,y)=\left[\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} C(u)C(v)F(u,v)*\cos\frac{(2x+1)u\pi}{2N}\cos\frac{(2y+1)v\pi}{2N}\right]$$

Where:
$$C(u),\ C(v)=\begin{cases}\sqrt{\frac{1}{N}} & for \quad u=v=0\\ \sqrt{\frac{2}{N}} & otherwise\end{cases}$$

The DCT-based compression can be viewing the FDCT as harmonic analyzer and the IDCT as harmonic synthesizer. In the case of JPEG format each 8X8 block of the source image samples is effectively a 64-point discrete signal which is a function of the 2D dimensional space *x*, and *y*. The FDCT takes such a signal as its input and decomposes it into 64 orthogonal basis signals. The output of the FDCT is the set of unique 64 basis signal amplitudes, which can be regarded as the relative amount of the 2D spatial frequency contained in the 64-point input signal. The JED algorithm is affected by limiting the number of DCT coefficients. This principle can be illustrated in a Figure 2. We can see from the figure that:
1. We can control the quality of the edge detection using the JED algorithm by selecting the number of the DCT coefficients.
2. The best image reconstruction is obtained when we select all 64 DCT coefficients.

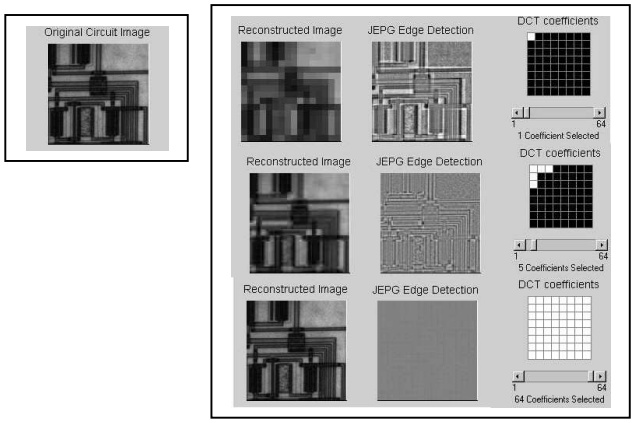


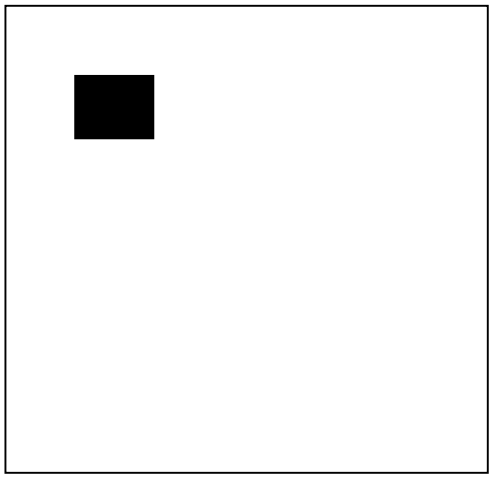
Figure 2

Figure 2 demonstrates how the number of selected DCT coefficients influences JED algorithm. The left image shows the original image. The upper-right image shows the Reconstructed Image and the result of JED for various selected DCT coefficients.

2. Illustrative graphic model.

In this section, we investigate JPEG's capability for edge detection. Figure 1 shows a sample image of how JPEG can be used for edge detection. The original image in Figure 2 is compressed in the grayscale baseline JPEG format with 75% quality.

Original image



JPEG format of uniform area

0	0	1	0	1	0

JPEG format of the upper left corner of the black square

-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

Figure 3

Figure 3 demonstrates how JPEG is used for contour extraction. The left image shows the original image. The upper-right image shows the JPEG format in the white or black area. The lower-right image shows the JPEG format in the upper left corner of the black square.

The JPEG file reports the difference of magnitude between the DC's coefficients of a previous block relative to the current block. In the case of a white or black area there are no changes in the coefficients' magnitudes. This type of block is encoded as six bits by the JPEG standard. The output of the JPEG file is shown in Figure 3.

The "00" reflects that there are no differences between the values of the previous and the current block's DC coefficients, and "1010" symbolizes the end of the block.

If there is a difference between the intensity of the DC coefficients of the previous and the current block's, the size of the encoding block will be slightly bigger.

For example, a block which 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 edges of the shape from the rest of the image. Figure 2 shows a sample of the block, which contains the upper left corner of the black square. In order to compress these values in JPEG standard, 243 bits are needed. The difference between 6 to 243 is obviously significant.

By using three parameters the length of the block, its magnitude and the number of consecutive blocks the threshold can extract the contour with a range of scalar values. The extra parameters allow more control over the resulting mechanism. Noise in the image can decrease the effectiveness of the simple threshold.

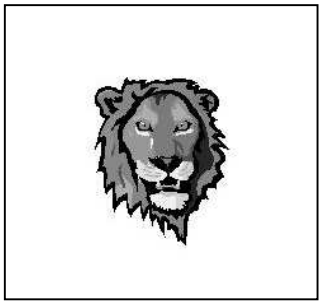
This method can be useful for extracting large objects from an image. Since blocks of 8X8 pixels represent the border, a small object can be derelict. If a small object is the target that should be found, another method, which will be described later, must be used. However, when a large object needs to be found, the method described here is significantly better.

Let n be the number of pixels in an image. The current state of the art algorithm order for edge detection is O(nlog(n)), while the order here is O(n). Every bit is read only once, just to determine the location of the End Of Block (EOB). When EOB is detected, the bytes count is checked to determine if the block contains an object's contour. In fact only the compressed file which is smaller than original file, is checked. But the size of a compressed file is linear in the size of the uncompressed file. Therefore the order of this algorithm is O(n).

3. An Animal Picture.

The same concept used in the above section can be employed in a realistic picture see Figure 4. If we observe the JPEG space of the upper left part of the lion's ear.

(4.a) Original image



(4.c) Edge detection using JED

(4.b) The table shows the JPEG space of the upper left part of the lion's ear.

-4	7	-8	5	-3	1	-1	0
7	-10	8	-5	3	-1	1	0
-6	8	-7	4	-2	1	-1	0
5	-6	4	-3	1	-1	1	0
-4	4	-2	1	-1	0	0	0
2	-2	1	-1	1	0	0	0
-1	1	-1	0	0	0	0	0
0	0	0	0	0	0	0	0

Figure 4: (a) Lion picture. (b) The lower table shows the JPEG block space in the upper left part of the lion's ear. (c) Edge detection using JED

5. CONCLUSION

The JED algorithm adopts a new concept by working directly on the compressed space. The JED deals with object extraction in images compressed by JPEG format. The algorithm gives complexity of O(n) in many cases, which is better than the known state of the art algorithms, which give an O(nlog(n)) order. Such an implementation can be very useful in a wide range of machine learning. In conclusion the JED is an elegant and efficient algorithm, and easy to hardware.

REFERENCES

[1] Luse M. The BMP File Format *Dr. Dobb's Journal, Vol 9, Issue 10,* 1994, 18-22.

[2] Willard L., Lempel A., Ziv J. and Cohn M. Apparatus and method for compressing data signals and restoring the compressed data signals *US patent - US4464650* 1984.

[3] Welch T. A. A Technique for High-Performance Data Compression *IEEE Computer Society* 17 1984, 8-19.

[4] Canny J. A Computational Approach to Edge Detection *IEEE Trans. Pattern Analysis and Machine Intelligence, PAMI-8* 1986, 679-698.

[5] Nalwa V. S. and Binford T. O. On Detecting Edges *IEEE Trans. Pattern Analysis and Machine Intelligence, PAMI-6* 1986, pp. 699-714.

[6] Wiseman Y. and Fredj E. Contour Extraction in X-Ray photos Compressed by JPEG Format, *conditionally accepted to ACM - Journal of Graphics Tools*, 2000.

[7] Wallace G. K. The JPEG Still Picture Compression Standard *Communication of the ACM 34,* 1991, 30-44.

[8] Information Technology Digital Compression and Coding of Continuous-Tone Still Images Requirements and Guidelines *International Standard ISO/IEC 10918-1* 1993.

[9] Hearn D. and Baker M. P. Computer Graphic *Prentice Hall, Englewood Cliffs, NJ,* 1986, 295-307.

[10] Jain A. K. *Fundamental of digital Image processing* (Prentice Hall Information and Sytem Sciences Series, 1986, 553-557).

[11] Hunt R. W. G. *The reproduction of colour* (Fountain Press England, 1995, 507-511).

[12] Laplante P. A. and Stoyenko A. D. *Real time imaging, theory, techniques and applications* (IEEE Press Inc. NY, 1996, 122-124).

[13] Awcock G. J. *Applied image processing* (McGraw-Hill Book Company, 1996, 260-261).

[14] Rao K. R. and Yip P. *Discrete cosine transform algorithms, advantages, applications* (Academic Press Inc., London 1990).

[15] Huffman D. A method for the Construction of Minimum Redundancy Codes *Proc. of the IRE 40,* 1952, 1098-1101.

[16] Witten I. H., Neal R. M. and Cleary J. G. Arithmetic Coding for Data Compression, *Communication of the ACM 30,* 1987, 520-540.

[17]Okada S., Matsuda Y., Yamada T., and Kiyozaki K., System on a Chip for Digital Still Camera *Digest of Technical Paper of IEEE 1999 International Conference on Consumer Electronics,* 1999, 86-87,.

[18] Kobayashi H., Noguchi Y. and Kiya H. A Method of Embedding Binary Data into JPEG Bitstreams *IEICE Trans. Vol. J83-D-II, no. 6,* 1999, 1469-1476.

[19] Noguchi Y., Kobayashi H. and Kiya H. A Method of Extracting Embedded Binary Data from JPEG Bitstreams Using Standard JPEG Decoder *IEICE Trans. fund. Vol. E83-A, no. 8,* 2000, 1582-1588.

[20] Marcellin M. W., Gormish M. J., Bilgin A. and Boliek M. P. An Overview of JPEG-2000 *Proc. Data Compression Conference DCC-2000, Snowbird, Utah* 2000, 1582-1588.

[21] Klein S. T. and Wiseman Y. Parallel Huffman Decoding, *Proc. Data Compression Conference DCC-2000, Snowbird Utah,* 2000, 383-392.