SPATIAL OBJECT DETECTION IN JPEG BITSTREAMS

Lei Zhou and Charles D. Creusere

New Mexico State University
Klipsch School of Electrical and Computer Engineering
Dept. 3-O, Thomas & Brown 325,Las Cruces, NM 88003 USA
Phone: (505) 646-3919, FAX: (505) 646-1435,email: ccreuser@nmsu.edu
Web: www.ece.nmsu.edu/~ccreuser

ABSTRACT

Because of the large amount of raw data involved, images are generally compressed before storage or transmission. If one were interested in detecting and localizing spatial objects of interest, it would be considerably more efficient to do so in the compressed domain because less data would need to be processed and the computations required to decode the image would be avoided. In this paper, we study detection and localization in the bitstream of JPEG-compressed imagery and compare these new results with our earlier work in which detection was performed in the DCT domain.

1. INTRODUCTION

The CCITT and ISO have defined a number of continuous tone image storage formats, including in particular the original JPEG standard. In our work here, we use the lossy JPEG baseline coding algorithm [3], although the results should be similar for most of the other lossy variations of the original JPEG standard. In earlier work, we have shown that object detection and localization can be performed both efficiently and effectively in the DCT domain using multiple templates to compensate for the shift-variation inherent in the transformation [1]. In this paper, we use multiple templates in the same way as in [1], but we now design and apply them in directly on the compressed bitstream generated by the JPEG baseline encoder. While this approach is faster than both spatial domain matching and DCT-domain matching, there is a performance penalty associated with it. We quantify that penalty later in this paper.

2. THE JPEG BASELINE SYSTEM

In the baseline system, an image is encoded as shown in Figure 1. Specifically, the compression is performed in several sequential steps [2, 4].

- Each pixel of an image is level shifted by subtracting 128.
- The image is then subdivided into 8x8 pixel blocks, and the 2-D discrete cosine transform is computed for each block.
- The data is then quantized. Most values of the data matrix will be zeros after quantization.
- The 8x8 data matrix is reordered according to the zigzag pattern (starting from the upper left-hand corner) to form a 1-D sequence of quantized coefficients.
- The data sequence is then encoded to create a bitstream using the run-length Huffman coding. There are three tables used for coding, JPEG Coefficient Coding Categories, JPEG Default DC Code, and JPEG Default AC Code. The nonzero AC coefficients are coded according to the coefficient's value and number of preceding zeros. The DC coefficients are coded according to the difference value relative to the DC coefficient of the previous image block. The method used to code the DC coefficients will influence the result of detection, which we will discuss later.

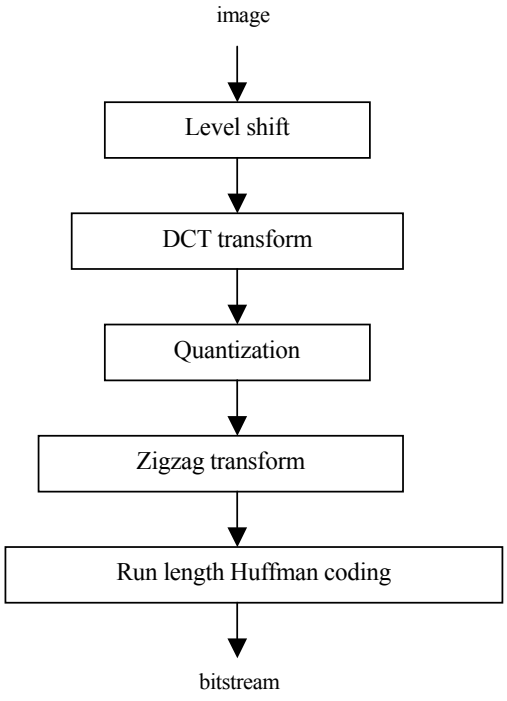

Figure 1: JPEG compression system.

Research supported by Sandia National Laboratories, Contract SURP25363

For example, if the original 8x8-image block is as below:

$$\begin{bmatrix} 52 & 55 & 61 & 66 & 70 & 61 & 64 & 73 \\ 63 & 59 & 66 & 90 & 109 & 85 & 69 & 72 \\ 62 & 59 & 68 & 113 & 144 & 104 & 66 & 73 \\ 63 & 58 & 71 & 122 & 154 & 106 & 70 & 69 \\ 67 & 61 & 68 & 104 & 126 & 88 & 68 & 70 \\ 79 & 65 & 60 & 70 & 77 & 68 & 58 & 75 \\ 85 & 71 & 64 & 59 & 55 & 61 & 65 & 83 \\ 87 & 79 & 69 & 68 & 65 & 76 & 78 & 94 \end{bmatrix}$$

and the DC coefficient of image block to its immediate left is –17, then the completely coded bitstream is 1010110 0100 001 0100 0101 100001 0110 100011 001 100011 001 001 100101 11100110 110110 0110 11110100 000 1010

From this string of bits, we wish to detect the presence of a spatial object of interest in the image. We will contrast this with the case in which object detection is performed after the quantized DCT coefficients have been decoded but before the inverse DCT has been applied [1].

3. DETECTION AND LOCALIZATION

Template matching is a natural method of object detection. In template matching, the goal is to find the region in one image that matches a specific template. A distance function or metric is applied to measure the similarity between the template and image at different positions. The position with the smallest distance gives the location of the object. Without distortions like rotation or expansion, template matching works effectively [1]. There are also other methods such as statistical approach and syntactic approach [5]. In this paper, we use only the template-matching method.

Because of shift variation in the blocked DCT, multiple templates can usually improve the results of detection. Previously, we developed two methods of building these additional templates [1]. One method is called Full Size Average Fill (FSAF). In this case, the first template is the entire target image and the others are shifted version of it for which areas uncovered by motion are filled in using predicted information. The second method is called Reduced Windowing (RW). In RW, only a portion of the larger target image is used for matching, so we never need to worry about uncovered areas in the shifted templates. We use both these methods in the experiments that follow.

4. DETECTION IN THE BASELINE SYSTEM

Since the baseline JPEG system organizes its bitstream in the same way as the spatial regions in the original image are organized, we can theoretically perform template matching between the bitstream of the image and that of the given template. The object detection is performed in the following way:

- Each image and template is separated into 8x8 blocks, and the data of each block is encoded into the bitstream.
- The bitstream of each block is then assembled into a byte sequence. For example, the bitstream such as 0110 1110 110 would be assembled as 0x6E, 0x06. Each block of the image is encoded as a byte sequences represented by Y(i, j, k) where i and j is the index for the block number and k indexes byte sequence of each block. The templates are encoded as X(i, j, k) in a similar way.

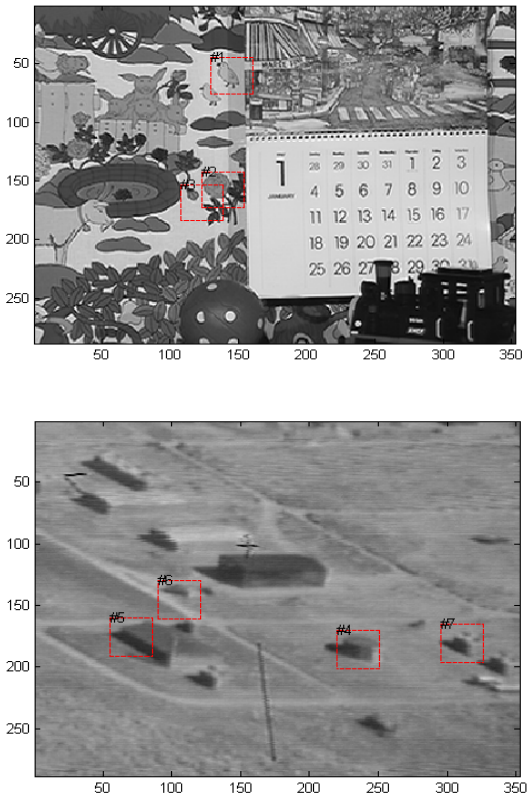

Figure 2: Two frames from the test sequences, Film #1 and Film #2, with the templates marked.

- The distance between image and a template is calculated as:

$$Dis = \sum_{i,j} \sum_{k} \frac{(Y - X)^2}{(Y^2 + X^2) * k^{0.5}} \qquad (1)$$

The portion of the bitstream image with the smallest distance contains the object of interest.

- The key problem in performing the object detection in the image bitstream is to define the distance between two bitstream; we do not, however, have any theory to guide us in this. Equation (1) is the normalized mean square error distance except for the weighting coefficient $k^{0.5}$. Because low frequencies are more important in natural images, this weighting coefficient is used to emphasis the importance of low frequency data. Other distance measures such as:

$$\sum_{i,j} \sum_{k} (Y - X)^2 \ , \sum_{i,j} \sum_{k} \frac{(Y - X)^2}{(Y^2 + X^2)}, and \frac{\sum_{i,j} \sum_{k} (Y - X)^2}{\sum_{i,j} \sum_{k} (Y^2 + X^2)}$$

can also detect the target sometimes, but we have found that (1) works best.

Table 1: Percentages of detection of different DC coefficient processing method.

Test method	*#5*	*#4*	*#3*
Method 1	85.8	57.1	100
Method 2	68.6	42.9	97.1
Method 3	74.3	54.3	100
Method 4	68.6	37.1	94.3

Table 2: Percentages of detection for objects in bitstream and DCT domain [1].

	objects	templates sort	% of correctly detection
detection in bit-stream domain	#3	RW	98
	#2	RW	44
	#1	RW	50
	#4	FSAF	42
	#5	FSAF	80
detection in DCT domain	#1	FSAF	63
	#4	RW	100
size:32x32, 9 templates			

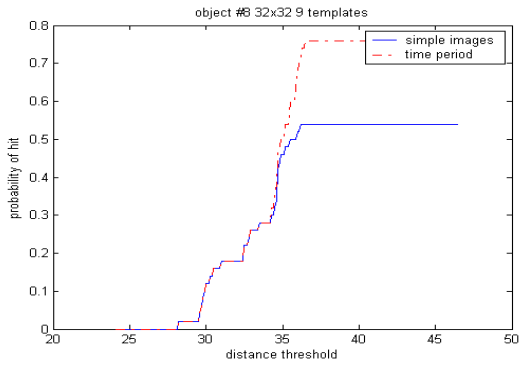

Figure 3: Hit probability curves of object #5 for different criterion.

Detection in the DCT domain is more direct [1]. We calculate the Euclidian distance between the DCT coefficients of each template corresponding to the given object of interest and the coefficients corresponding to an area of image. The area with the smallest distance to any of the templates in the set is judged to be the object area.

5. DETECTION RESULT

Targets are obtained from a single frame in each video sequence; sample frames and the target objects are shown in Figure 2.

Before studying the results in detail, we must first discuss the problem of encoding of the DC coefficient in a block. In the baseline system, the difference value between the current DC coefficient and that of the previous encoded block is coded. When the templates are compared to different areas of the images, the previous blocks of the first column will change. To calculate the distance between templates and images, we have considered several methods:

Table 3: Percentages of detection of deferent size of objects in bitstream and DCT domain [1].

	object #	template sort	32x32	24x24	16x16
Bit stream domain	#3	RW	98	64	4
	#2	RW	44	22	4
	#7	NSAF	48.6	37.1	8.6
DCT do-main	#1	RW	100	61	22
9 templates					

Table 4: Percentages of detection for different templates in bitstream and DCT domain [1].

	object #	Size	templates number		
			17	9	5
bit stream domain	#4	32x32	57.1	42	34.3
	#1	32x32	82	50	40
	#3	24x24	46	64	62
	#6	32x32	48.6	42.9	42.5
DCT do-main	#3	24x24	71	74	53
	#6	24x24	100	92	89

- The DC coefficients are decoded each time.
- The DC coefficients problem is not considered when calculating the MSE.
- The values of the first byte of the first column are not summed
- The values of the entire first column are not summed.

Table 1 shows the test result. Method 1 generates the best results, but it requires a more complex decoding procedure. Method 3 provides the next best results, and is much more efficient. We therefore have used method 3 in the following experiment.

We can see from Table 2 that the percentage of correctly detected target in the DCT domain is much higher than that in bitstream domain; however, it is more efficient to detect objects in bitstream domain. We have to decode the bitstream for the detection in the DCT domain and the number of bytes that must be processed in the DCT domain is about 6 times higher than in the bitstream domain.

In many situations, we do not need to detect the target every frame - one correct detection over a small period of time is good enough. We assume now that if one of the images is correctly detected during a period of 83.3ms (five images), the task of detection is achieved successfully. From Fig 3, we can see that our algorithm works quite well under this assumption.

We see from Table 3 that the size of a target object also plays an important role in bitstream domain detection. The percentage of correct recognition decreases quickly with the decreasing object size. We have found that our approach is not effective if the object size is less than 24x24. Note that detection in DCT domain still works when the target is small.

Table 4 shows that detection with more templates usually increases the percentage of correct detection but not always. This happens in both bitstream domain and DCT domain

Both the FSAF and RW method can be used to build the templates. The experiments show that the percentages of correct detection using the RW templates is higher than that using the FSAF templates, if the final template size is the same.

6. COMPUTATIONAL COMPLEXITY

In our tests, the image size is 288x352 pixels. We assume that the object size is SXxSY. Most image data is stored and transferred as bitstreams, and thus we have to decode the bitstream prior to DCT-domain object detection. The run-length Huffman decoding used in JPEG is a time-consuming table searching process and de-quantization requires SXxSY multiplications. Even if we do not consider the cost of decoding, calculation in bitstream domain is more efficient than that in DCT domain, because less data must be compared in the template matching process.

The number of individual template comparisons is the same for both the DCT and bitstream domains, therefore we need only compare the computational complexities when one template is compared with a selected area of one image frame.

For the detection in DCT domain, each evaluation requires SXxSY additions and SXxSY multiplications [1].

For the detection in bitstream domain, we must calculate (1):i.e.,:

$$Dis = \sum_{i,j} \sum_{k} \frac{(Y - X)^2}{(Y^2 + X^2) * k^{0.5}}$$

where (i, j) is the index for the block number, and k is the index of the byte sequence in the bitstream. We can calculate $k^{0.5}$ and X^2 in advance. Experimentally, we have found that the maximum value of k varies between 5 and 12, and its average value is 9. Each evaluation in bitstream domain thus requires approximately 4x9xSXxSY/64 multiplications and 2x9xSXxSY/64 additions. Consequently, the number of multiplications needed for the detection in the bitstream domain is about 40% lower than that in the DCT domain, and the number of additions is about 70% lower.

7. CONCLUSION

The above results show that we have developed an algorithm that works well for the objects detection in bitstream domain of the baseline JPEG compression system. This algorithm is effective for target objects larger than 24x24 pixels, and is more computational efficient than detection in the DCT domain. Future work will involve the object detection in the bitstream domain of other entropy-coding algorithm.

REFERENCES

[1] C. D. Creusere and G. Dahman, "Object Detection and Localization in Compressed Video," in IEEE Int. Conf. Signals, Systems and Computers, Pacific Grove, CA USA, Nov. 2001, pp. 93–97, vol.1.

[2] D. S. Taubman and M. W. Marcellin, *JPEG 2000 Image compression fundamentals and practices*. Boston, MA: Kluwer Academic Publishers, 2002.

[3] G. K. Wallace, "Overview of the JPEG (ISO/CCITT) still image compression standard," Image Proc. Algorithms and Techniques, Vol. 1244, pp.220–33,1990.

[4] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*. New Jersey: Addison-Wesley Publishing Company, 1993.

[5] R. O. Duda, P. E.Hart, and D. G. Stork, *Pattern Classification*. New York: JOHN EILEY & SONS, INC, 2001.