Optical-logic-array processor using shadowgrams. II. Optical parallel digital image processing

J. Tanida and Y. Ichioka

Department of Applied Physics, Osaka University, Yamadaoka 2-1, Suita, Osaka 565, Japan

Received May 18, 1984; accepted February 8, 1985

The optical-logic-array processor (OLAP), using a lensless shadow-casting system, has been applied to execute optical digital image processing for binary objects. The OLAP is an optical digital processor based on techniques of image coding and optical correlation that can optically implement parallel logic gates. To demonstrate further potential of the OLAP, several processing methods for gray-level objects are presented. Experimental results indicate applicability of the OLAP for parallel optical digital image processing for both binary and gray-level objects.

1. INTRODUCTION

Spatial-filtering techniques capable of processing two-dimensional (2-D) image data in parallel demonstrate the great potential of light in information processing.[1] The skillful adaptation based on the physical characteristics of light offers this potentiality. However, spatial-filtering techniques requires delicate adjustment of the optical setup because of this skillfulness. As a result, they lack flexibility.

On the other hand, the development of electronic digital computers, which can perform precise and flexible operations at high speed, has been accelerated these several years. Along with the recent tremendous progress in science and technology, processing for a large amount of data by electronic digital computers has been increasingly required in a wide variety of applications. Remote sensing and digital image processing in geography and meteorology are examples of such applications.

However, the present computers do not have ability enough to process massive data required in these applications. Thus a higher grade of digital computers than the present ones is required.[2] Much effort has been devoted to develop large-scale digital computers or parallel computers capable of processing massive data at high speed. However, the architectures of such computers must become complicated. Therefore, in practice, development of large-scale computers or parallel computers with desirable performance still seems to be difficult. Hence computer and optical researchers have again focused on the capability of high-speed, parallel information processing by light as one of the most important techniques in the development of parallel computers.

Recently, a new field of optical information processing classified as optical computing has grown up. Although optical computing includes various research fields, investigation for developing new computing systems appears especially noteworthy. The capability of optical information processing is expected to be utilized most efficiently for this purpose. Many fruitful results have been already obtained in the research fields of optical computing, such as optical parallel logic gates using spatial-light modulators,[3,4] interconnection devices of parallel gates,[5] and a computational technique using parallel symbolic substitution.[6]

In the previous paper, we presented a new, optical parallel digital processor called OLAP (optical-logic-array processor), using an image-coding technique and a simple incoherent optical-correlation technique.[7] The OLAP can implement any combinational logic operation for two binary objects in parallel.

In this paper, we present some applications of the OLAP to parallel digital image processing. First, we summarize the mechanism of the OLAP and explain image-processing methods for binary discrete objects. Then we describe three methods of optical parallel digital processing for gray-level discrete objects that we called the bit-decomposition method, the digital-to-analog- (D/A-) conversion method, and the gray-level coding method.

2. OPTICAL-LOGIC-ARRAY PROCESSOR

Figure 1 is the schematic diagram of the OLAP presented in the previous paper.[7] For convenience, we call this optical system the basic type of OLAP. In Fig. 1, the case of executing the XOR operation for two binary objects with 4×4 pixels is depicted. Referring to this figure, we explain the method of implementing desired parallel logic operations.

First, two input objects are converted into coded versions. All pixels in individual objects are replaced by the code patterns shown in Fig. 2 according to a pixel value of 1 or 0. Each code pattern is composed of opaque and transparent rectangular portions with a half-amount of cell size. Here we call the small square area around a pixel a cell. Note that coding methods are different for the pixels a_{ij} and b_{ij} in the directions of stripe patterns as shown in Fig. 2. After both objects are coded, they are superimposed so that corresponding cells in both objects are precisely overlaid.

The superimposed coded object is set in the input plane of the optical system in Fig. 1 and illuminated by divergent light beams radiated from incoherent point sources. Then multiple shadowgrams of the superimposed coded object are projected onto the screen, overlapping one another. The optical system is adjusted so that shadowgrams projected by neighboring point sources are interlaced, shifting each other by an amount that is half of the projected cell size along the vertical and the horizontal directions.

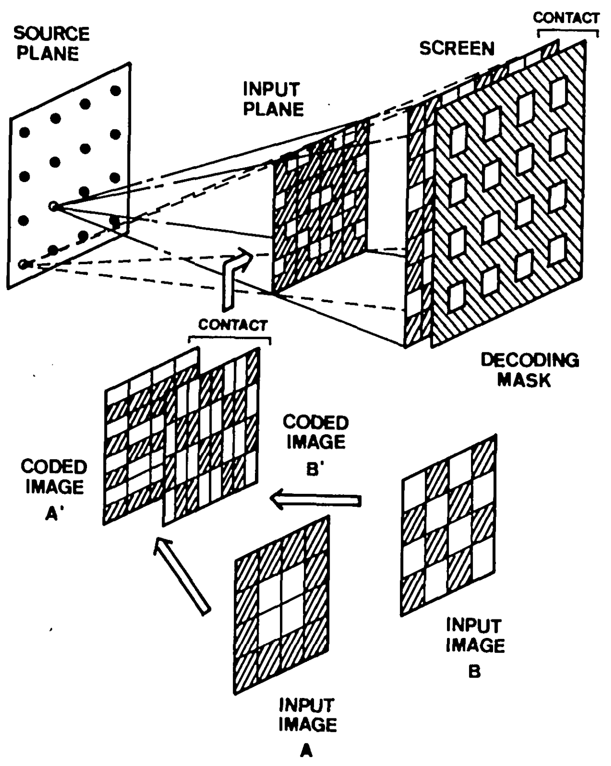


Fig. 1. Schematic diagram of the OLAP.

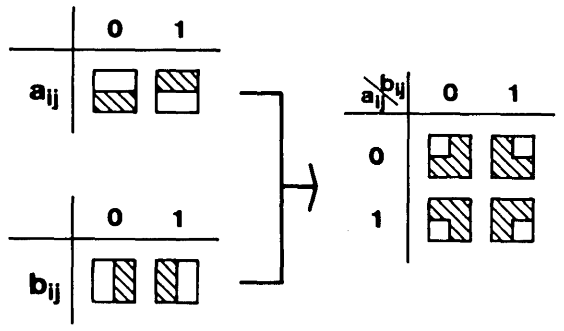


Fig. 2. Code patterns for input binary variables.

In the optical setup of Fig. 1, there exist regions where the four shadowgrams of individual cells in the superimposed coded object are overlapping. These overlapping regions are observed or detected through a decoding mask with square windows set on the screen. Selecting the proper combination of switching modes of point sources, we can obtain results of optical parallel logical operations for the two binary objects as patterns constructed by assembly of bright and dark parts. The OLAP can execute a complete set of logical operations for two binary signals in parallel by changing the combination of the switching modes of four point sources arranged in square array.

Table 1 represents operations implemented by the OLAP, in which the switching configurations of light sources and the corresponding operations are tabulated. Hereafter, we call the combination of switching modes of the point sources a source pattern. In this table, a source pattern is expressed by using a matrix form whose elements indicate positions and intensity levels of individual point sources. If more than four point sources are used, parallel shift operations can be easily implemented. Examples of source patterns for these cases are shown in the rightmost column of Table 1.

The most significant advantage of the OLAP is that it can optically achieve logical and arithmetic processing for all pixels over the input objects in parallel. That is, this optical processing system operates in a single-instruction-stream multidata-flow (SIMD) architecture.

3. BINARY IMAGE PROCESSING

The OLAP can execute bipolar-valued processing by devising the source patterns. Bipolar-valued processing makes it possible to extend the application of incoherent optical processing. By using this feature, we present two kinds of processing for discrete objects.

A. Optical Parallel Digital Filtering

Processing executed by the OLAP is inherently parallel 2-D digital filtering.[8] A source pattern of the OLAP corresponds

Table 1. Matrix Expressions of Source Patterns for Logical Operations

Source Pattern	Function	Source Pattern	Function	Source Pattern	Function
$\begin{pmatrix}0&0\\0&0\end{pmatrix}$	FALSE	$\begin{pmatrix}1&0\\0&0\end{pmatrix}$	$\overline{A+B}$	$\begin{pmatrix}0&0&0&0\\0&0&0&0\\0&0&0&0\\0&0&1&1\end{pmatrix}$	SHIFT-X A
$\begin{pmatrix}0&0\\0&1\end{pmatrix}$	$A\times B$	$\begin{pmatrix}1&0\\0&1\end{pmatrix}$	$\overline{A\oplus B}$		
$\begin{pmatrix}0&0\\1&0\end{pmatrix}$	$A\times\bar{B}$	$\begin{pmatrix}1&0\\1&0\end{pmatrix}$	$\bar{B}$	$\begin{pmatrix}0&0&0&0\\1&1&0&0\\0&0&0&0\\0&0&0&0\end{pmatrix}$	SHIFT-Y A
$\begin{pmatrix}0&0\\1&1\end{pmatrix}$	A	$\begin{pmatrix}1&0\\1&1\end{pmatrix}$	$A+\bar{B}$		
$\begin{pmatrix}0&1\\0&0\end{pmatrix}$	$\bar{A}\times B$	$\begin{pmatrix}1&1\\0&0\end{pmatrix}$	$\bar{A}$	$\begin{pmatrix}0&0&0&0\\0&0&0&0\\0&0&0&1\\0&0&0&1\end{pmatrix}$	SHIFT-X B
$\begin{pmatrix}0&1\\0&1\end{pmatrix}$	B	$\begin{pmatrix}1&1\\0&1\end{pmatrix}$	$\bar{A}+B$		
$\begin{pmatrix}0&1\\1&0\end{pmatrix}$	$A\oplus B$	$\begin{pmatrix}1&1\\1&0\end{pmatrix}$	$\overline{A\times B}$	$\begin{pmatrix}0&1&0&0\\0&1&0&0\\0&0&0&0\\0&0&0&0\end{pmatrix}$	SHIFT-Y B
$\begin{pmatrix}0&1\\1&1\end{pmatrix}$	$A+B$	$\begin{pmatrix}1&1\\1&1\end{pmatrix}$	TRUE		

	PROCESSING KERNEL	SOURCE PATTERN
(a)	$\begin{matrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{matrix}$	$\begin{pmatrix} 0&0&0&0&0&0 \\ 0&0&0&0&0&0 \\ 0&0&1&1&0&0 \\ 0&0&0&0&0&0 \\ 0&0&0&0&0&0 \\ 0&0&0&0&0&0 \end{pmatrix}$
(b)	$\begin{matrix} 0 & 0 & 0 \\ 0 & 1 & -1 \\ 0 & 0 & 0 \end{matrix}$	$\begin{pmatrix} 0&0&0&0&0&0 \\ 0&0&0&0&0&0 \\ 0&0&0&0&1&1 \\ 0&0&1&1&0&0 \\ 0&0&0&0&0&0 \\ 0&0&0&0&0&0 \end{pmatrix}$
(c)	$\begin{matrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{matrix}$	$\begin{pmatrix} 0&0&1&1&0&0 \\ 0&0&0&0&0&0 \\ 1&1&0&0&1&1 \\ 0&0&4&4&0&0 \\ 0&0&1&1&0&0 \\ 0&0&0&0&0&0 \end{pmatrix}$
(d)	$\begin{matrix} 0 & 0 & 0 \\ 0 & 1 & -1 \\ 0 & 0 & 0 \end{matrix}$ masked by B	$\begin{pmatrix} 0&0&0&0&0&0 \\ 0&0&0&0&0&0 \\ 0&0&0&0&0&1 \\ 0&0&0&1&0&0 \\ 0&0&0&0&0&0 \\ 0&0&0&0&0&0 \end{pmatrix}$

Fig. 3. Processing kernels and corresponding source patterns for digital filtering. (a) Inversion of object A; (b) first derivative of object A in the horizontal direction; (c) Laplacian of object A; and (d) first derivative of object A in the horizontal direction only for pixels of which the values of corresponding pixels in object B are 1.

to a processing kernel of the 2-D digital filtering, which is correlated with all pixels over an entire input object at a time. It should be mentioned that one element value in a processing kernel can be expressed by a switching mode of a pair of point sources within an array consisting of four point sources. If a pair of point sources on the upper half is in the on state and those on the lower half are in the off state, the source pattern describes a positive value; if those switching states are reversed, it indicates a negative value, in which the intensity of the point sources describes the absolute value.

Figures 3(a)–3(c) show the correspondence between processing kernels of digital filters and source patterns. For Fig. 3(a), inversion of an input object is obtained. Using such a source pattern capable of representing negative value, we attempted to process the first derivative in the horizontal direction and the Laplacian of the input object. Figures 3(b) and 3(c) show source patterns used for both types of processing. In the similar manner, any digital filter with a bipolar value can be realized.

However, to obtain the correct output values by digital filtering, bias component must be subtracted from the processed result. The bias component is inherently superposed upon the result processed by an incoherent optical system. Fortunately, this bias component can be easily suppressed by using an electronic method. Hence this problem is not so serious.

Since the OLAP can treat two input objects at a time, we can utilize one object as a control signal of a filtering operation for the other. As a result, localized processing can be realized by SIMD architecture, in which one input object is used to mask a current operation for the other. Such a masking operation is indispensable for parallel processing. Figure 3(d) shows a source pattern for implementing the first derivative of the object A, in which the object B is utilized as a control signal. That is, the operation of first derivative for the object A is executed only for the pixels with a value of 1 in the object B.

To verify the above processing, we made experiments for the input objects with 64 × 64 pixels. Two input objects are converted into a superimposed coded transparency with an 18 mm × 18 mm area. An array of light-emitting diodes (LED's) radiating yellow light is used as point light sources. The intensity of individual LED's is modulated by changing the duty ratio of the driving pulses of the LED's. This switching method is used for other experiments in this paper.

Figure 4 exhibits results of digital filtering when the source patterns of Figs. 3(b) and 3(c) are used. Figure 4(a) is an input object, Fig. 4(b) is the result of first derivative in the horizontal direction, and Fig. 4(c) is that of the Laplacian. Figure 5 shows the result of the localized first derivative by the source pattern of Fig. 3(d). Figures 5(a)–5(c) are an input object, a masking pattern, and the processed result, respectively.

B. Template Matching

To demonstrate the usefulness of the OLAP in practical applications, template matching[8] applying the logical operation is attempted. Template matching is the process of detecting a specific pattern within objects, in which a replica of a pattern of interest (called a template) is compared with all kinds of patterns in the object. This processing can be achieved by correlating the input object with the template. Figure 6 illustrates the procedure of template matching. Figures 6(a) and 6(b) are a discrete object and a template, respectively. Figure 6(c) is the output signal obtained by cross correlation of Figs. 6(a) and 6(b), in which the locations where patterns coincide with the template are indicated by the strongest

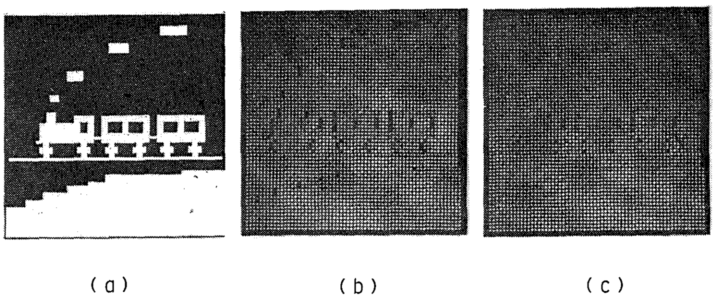


Fig. 4. Experimental results of digital filtering. (a) Input object, (b) result of first derivative in the horizontal direction, and (c) that of the Laplacian.

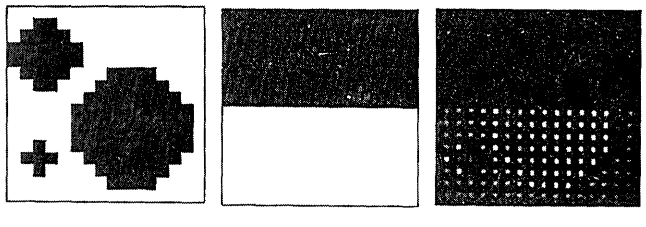


Fig. 5. Experimental results of localized first derivative. (a) Input object, (b) masking pattern, and (c) output result.

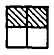
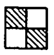
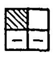
```
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 1 1 1 1 1 1 1 0 0
0 0 1 1 0 0 0 1 1 0 0
0 0 1 0 0 1 0 0 1 0 0          0 1 0
0 0 1 0 1 1 1 0 1 0 0          1 1 1
0 0 1 0 0 1 0 0 1 0 0          0 1 0
0 0 1 1 0 0 0 1 1 0 0
0 0 1 1 1 1 1 1 1 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0

        ( a )                  ( b )

0 0 0 0 0 0 0 0 0 0 0    0 0 0 0 0 0 0 0 0 0 0
0 0 1 1 1 1 1 1 1 0 0    0 0 0 0 0 0 0 0 0 0 0
0 1 3 4 3 3 3 4 3 1 0    0 0 0 0 0 0 0 0 0 0 0
0 1 4 3 2 2 2 3 4 1 0    0 0 0 0 0 0 0 0 0 0 0
0 1 3 2 2 2 2 2 3 1 0    0 0 0 0 0 0 0 0 0 0 0
0 1 3 2 2 5 2 2 3 1 0    0 0 0 0 0 1 0 0 0 0 0
0 1 3 2 2 2 2 2 3 1 0    0 0 0 0 0 0 0 0 0 0 0
0 1 4 3 2 2 2 3 4 1 0    0 0 0 0 0 0 0 0 0 0 0
0 1 3 4 3 3 3 4 3 1 0    0 0 0 0 0 0 0 0 0 0 0
0 0 1 1 1 1 1 1 1 0 0    0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0    0 0 0 0 0 0 0 0 0 0 0

        ( c )                    ( d )
```

Fig. 6. Template-matching processing. (a) Input object, (b) template, (c) output by correlation, and (d) output by logical operation.

TEMPLATE			
SOURCE PATTERN	$\begin{pmatrix} 0&0&0&0 \\ 1&1&1&1 \\ 1&1&1&1 \\ 0&0&0&0 \end{pmatrix}$	$\begin{pmatrix} 0&0&1&1 \\ 1&1&0&0 \\ 1&1&0&0 \\ 0&0&1&1 \end{pmatrix}$	$\begin{pmatrix} 0&0&1&1 \\ 1&1&0&0 \\ 0&0&0&0 \\ 0&0&0&0 \end{pmatrix}$

▨ : 0 □ : 1 ⊟ : DON'T CARE

Fig. 7. Examples of the templates and the corresponding source patterns.

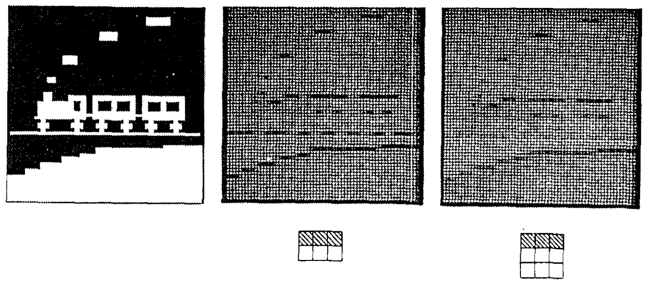


Fig. 8. Experimental results of template matching. (a) Input object, (b) and (c) results of logical template matching executed by the OLAP. Dark areas in the results indicate locations where the templates shown below the pictures match the patterns in the input object.

peaks within the output signal accompanied with a halo. The halo reduces the sensitivity of detection of the peak signal in the correlation image and sometimes causes a detection error because of noise. Thus improvement of detectability is not expected so far as a conventional correlation technique is employed.

On the other hand, template matching can be also achieved by logical AND operations for all patterns in the object and the template. In this case, the locations where the patterns coinciding with the template exist are detected as pointwise signals. This situation is depicted in Fig. 6(d). As is shown in the figure, this method would promise high sensitivity.

The OLAP can implement this logical template matching in parallel where specific source patterns defining templates are used. Each element of a template is defined by a switching mode of a pair of point sources in an array consisting of four point sources. If the sources in the upper half are in the on state and those in the lower half are in the off state, the element value is defined as 1, and, if the states are reversed, it is defined as 0. When none of the sources is in the on state, the element is out of consideration. Using multiple arrays consisting of four point sources, any template can be defined. Figure 7 shows examples of the templates and the corresponding source patterns. For defining larger sizes of templates, the number of point sources must be increased. A masking operation described in Subsection 3.A is also implemented by using this technique of logical template matching.

Figure 8 shows the results of logical template-matching operations executed by the OLAP. Figure 8(a) is an input object, and Figs. 8(b) and 8(c) are the output images processed with the templates shown below the pictures. The output images contain several dark points that point out the locations of local patterns in the input object matched with the template.

4. PROCESSING OF GRAY-LEVEL OBJECTS

In this section, we present three methods of processing gray-level objects by the OLAP. One is a method using the concept of Boolean algebra. We called it the *bit-decomposition method.* Although Boolean algebra is indispensable for an electronic digital computer, it is not necessarily adaptable for an optical computer. Sometimes it is reasonable to utilize the analog nature of the light signal in the optical computer. The two other methods make good use of the analog nature of the light signal. These methods are the *D/A-conversion method* and the *gray-level coding method.*

A. Bit-Decomposition Method

The bit-decomposition method is similar to the image-processing method of gray-level objects in an electronic digital computer, except for parallelism. When a gray-level object is processed by an electronic computer, the object is decomposed into binary bit planes, and the binary objects in individual bit planes are separately processed. This procedure can also be followed by the OLAP.

We consider the case treating gray-level objects expressed by 4-bit signals as an example. Figure 9(a) shows the way in which a gray-level object is decomposed. The object to be processed is decomposed into four binary objects in 2^0, 2^1, 2^2, and 2^3 bit planes. Since individual bit planes consist of binary signals, they can be processed by the OLAP.

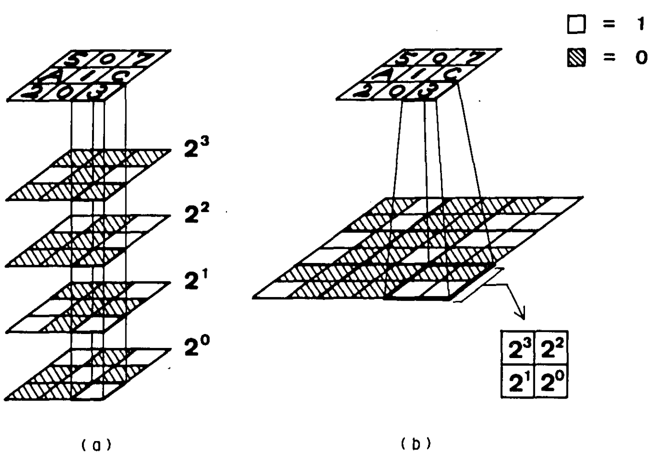


Fig. 9. Decomposition methods of 4-bit signals by using (a) conventional bit planes and (b) an interlaced bit plane.

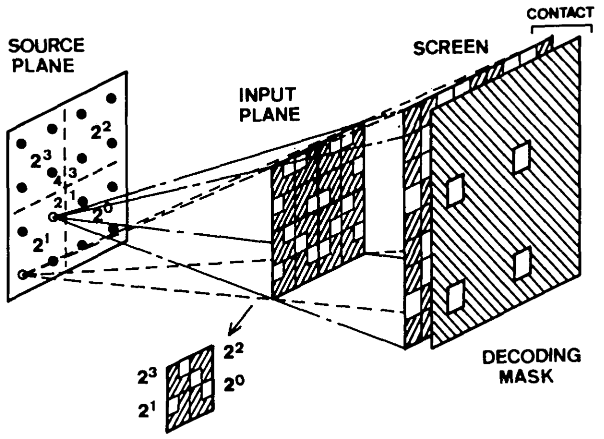


Fig. 10. Multi-input type of OLAP. Gray-level image processing by means of the bit-decomposition method and the D/A-conversion method is executed by this type of OLAP.

However, in order to process these binary signals in different bit planes, we must consider a method of data communication between different bit planes. To do so, we devise the method of rearranging the decomposed-bit elements in one plane. Figure 9(b) illustrates the manner of data rearrangement. Exchanging or transferring of information among different bit data can be realized for the rearranged data merely by using a simple shift operation. As is shown in the previous paper, the OLAP has the capability of implementing parallel shift operations.[7] Using these operations, the entire data of a bit position can be easily communicated to those of other bit positions in parallel.

Figure 10 shows a shadow-casting system that can process 4-bit gray-level objects by means of the bit-decomposition method. The difference between this system and the basic type of OLAP is that of the window pitch in the decoding mask. Four blocks of LED arrays are used for reading out signals of individual bit positions. That is, source array 1 serves to read out the 2^0-bit signals, array 2 the 2^1-bit signals, etc. In this system, a logical gate for a cell operates like that for multiple inputs, and then we call this system a multi-input type of OLAP. Table 2 shows the relations between source patterns and readout signals.

Although the arrangement of bit signals shown in Fig. 9(b) causes spatial extension of input objects, this does not restrict the limitation of the method. The reason is as follows: Individual decomposed and rearranged objects must be divided into several subobjects for processing because of restrictions resulting from the space–bandwidth product of the OLAP. Thus individual subobjects are processed by several OLAP's. If the decomposed-bit signals describing a pixel value are arranged so as to adjoin one another, operations among corresponding pixels in the two objects are implemented in a specific OLAP. In addition, communication between different pixel data can be attained by using the optical-interconnection technique.[5] Therefore processing for the whole object can easily be implemented by dividing it into several subobjects.

Figure 11 shows the experimental results obtained by using the bit-decomposition method. Figures 11(a) and 11(b) are an input object and its decomposed version, respectively. Figures 11(c)–11(f) are experimental results of reading out signals of individual bit planes, which are obtained merely by changing the source pattern.

After an operation of bit decomposition, individual data are processed by the same way as by electronic digital computers

Table 2. Matrix Expressions of Source Patterns Used in Bit-Decomposition Method

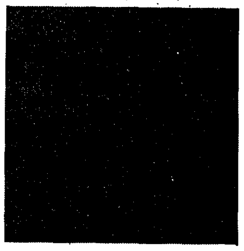
Source Pattern	Function	Source Pattern	Function	Source Pattern	Function
$\begin{pmatrix}0&0&0&0\\0&0&0&0\\0&0&0&0\\0&0&1&1\end{pmatrix}$	A_0	$\begin{pmatrix}0&0&0&0\\0&0&0&0\\0&0&0&1\\0&0&0&1\end{pmatrix}$	B_0	$\begin{pmatrix}0&0&0&0\\0&0&0&0\\0&0&0&0\\0&0&0&1\end{pmatrix}$	$A_0 \times B_0$
$\begin{pmatrix}0&0&0&0\\0&0&0&0\\0&0&0&0\\1&1&0&0\end{pmatrix}$	A_1	$\begin{pmatrix}0&0&0&0\\0&0&0&0\\0&1&0&0\\0&1&0&0\end{pmatrix}$	B_1	$\begin{pmatrix}0&0&0&0\\0&0&0&0\\0&0&0&1\\0&0&1&1\end{pmatrix}$	$A_0 + B_0$
$\begin{pmatrix}0&0&0&0\\0&0&1&1\\0&0&0&0\\0&0&0&0\end{pmatrix}$	A_2	$\begin{pmatrix}0&0&0&1\\0&0&0&1\\0&0&0&0\\0&0&0&0\end{pmatrix}$	B_2	$\begin{pmatrix}0&0&0&0\\0&0&0&0\\0&0&0&0\\0&1&0&0\end{pmatrix}$	$A_1 \times B_1$
$\begin{pmatrix}0&0&0&0\\1&1&0&0\\0&0&0&0\\0&0&0&0\end{pmatrix}$	A_3	$\begin{pmatrix}0&1&0&0\\0&1&0&0\\0&0&0&0\\0&0&0&0\end{pmatrix}$	B_3	$\begin{pmatrix}0&0&0&0\\0&0&0&0\\0&1&0&0\\1&1&0&0\end{pmatrix}$	$A_1 + B_1$

```
0 0 0 0 0 0 0 0
0 2 2 2 2 2 2 0
0 2 4 4 4 4 2 0
0 2 4 8 8 4 2 0
0 2 4 8 8 4 2 0
0 2 4 4 4 4 2 0
0 2 2 2 2 2 2 0
0 0 0 0 0 0 0 0
```

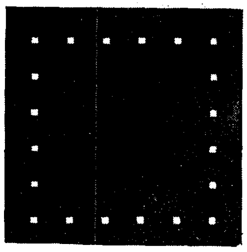
(a)

```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 1 0 1 0 1 0 1 0 1 0 1 0 0 0
0 0 0 0 0 1 0 1 0 1 0 1 0 0 0 0
0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 1 1 0 1 0 0 1 0 0 0 0
0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 1 1 0 1 0 0 1 0 0 0 0
0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 1 0 1 0 1 0 1 0 0 0 0
0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 1 0 1 0 1 0 1 0 1 0 1 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

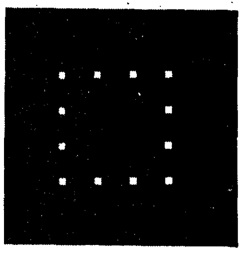
(b)



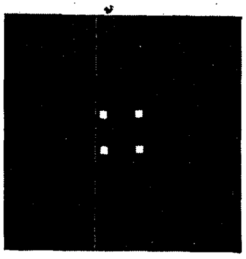
(c) 2^0 bit



(d) 2^1 bit



(e) 2^2 bit



(f) 2^3 bit

Fig. 11. Experimental result obtained by the bit-decomposition method. (a) Input object, (b) decomposed object, (c)–(f) results of reading out individual bit signals.

but in parallel. For example, decomposed data are read out, fetched to the arithmetic logic unit, and stored in the memory. Of course, the processing units used for such operations should be constructed by using optical spatial-light modulators or parallel electronic devices. To implement the desired operations for high-resolution objects, we have to construct parallel logic circuits by connecting OLAP's.

The salient feature of the bit-decomposition method is the ability to employ the theoretical basis and techniques accumulated in the field of electronic computers. However, it appears that construction of an all-optical version of the system might be difficult because of the complexity of the system architecture.

Hence, for digital processing by the OLAP, we would adopt another technique based on the concept of array logic[9] in recent digital electronics. The mechanism of the OLAP can correspond well to that of array logic. Use of this correspondence promises to simplify data controlling and reduce the number of optical devices composing an optical-processing system. A novel optical parallel digital-computing system based on this concept is reported in a subsequent paper.[10]

B. Digital-to-Analog-Conversion Method

The D/A-conversion method is a modification of the bit-decomposition method. Although output signals have been expressed by only two states in the bit-decomposition method, the D/A-conversion method can handle multilevel signals by utilizing the technique of optical parallel D/A conversion.[7] Processing by this method is realized by using the optical setup in Fig. 10. The optical parallel D/A conversion is executed by reading out all bit signals simultaneously with illumination of the LED array, in which the radiance of component LED's is modulated by bit-weight factors. Examples of possible operations and the corresponding source patterns are tabulated in Table 3. The examples are shown for gray-level objects with 4-bit signals.

Figure 12 shows two gray-level objects with 4-bit signals. Examples of the experimental results for these input objects are shown in Fig. 13. Figure 13(a) is the readout image A, and Fig. 13(b) is the result of A + B, where + means sum. Both results are displayed as D/A-converted images.

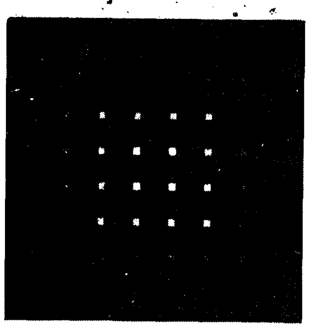
Table 3. Matrix Expressions of Source Patterns Used in D/A-Conversion Method

Source Pattern	Function	Source Pattern	Function
$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 8 & 8 & 4 & 4 \\ 0 & 0 & 0 & 0 \\ 2 & 2 & 1 & 1 \end{pmatrix}$	A	$\begin{pmatrix} 0 & 8 & 0 & 4 \\ 0 & 8 & 0 & 4 \\ 0 & 2 & 0 & 1 \\ 0 & 2 & 0 & 1 \end{pmatrix}$	B
$\begin{pmatrix} 0 & 8 & 0 & 4 \\ 8 & 16 & 4 & 8 \\ 0 & 2 & 0 & 1 \\ 2 & 4 & 1 & 2 \end{pmatrix}$	A + B	$\begin{pmatrix} 8 & 0 & 4 & 0 \\ 16 & 8 & 8 & 4 \\ 2 & 0 & 1 & 0 \\ 4 & 2 & 2 & 1 \end{pmatrix}$	A − B + 15

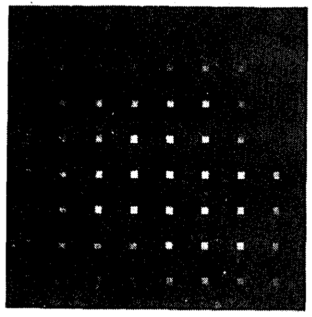
```
0 0 0 0 0 0 0 0      0 0 0 0 1 1 1 1
0 2 2 2 2 2 2 0      0 0 0 0 1 1 1 1
0 2 4 4 4 4 2 0      0 0 0 0 1 1 1 1
0 2 4 8 8 4 2 0      0 0 0 0 1 1 1 1
0 2 4 8 8 4 2 0      2 2 2 2 4 4 4 4
0 2 4 4 4 4 2 0      2 2 2 2 4 4 4 4
0 2 2 2 2 2 2 0      2 2 2 2 4 4 4 4
0 0 0 0 0 0 0 0      2 2 2 2 4 4 4 4

    INPUT A              INPUT B
```

Fig. 12. Gray-level objects with 4-bit signals.



(a)



(b)

Fig. 13. Experimental results obtained by the D/A-conversion method: (a) Result of reading out object A and (b) that of reading out A + B.

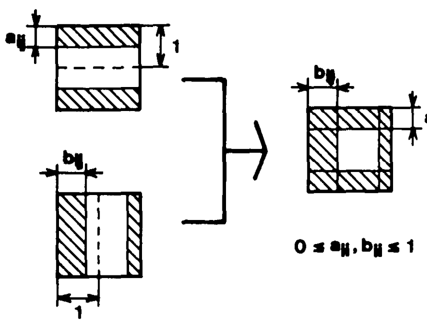


Fig. 14. Code patterns used in the gray-level coding method.

This D/A-conversion method is suitable for displaying output images. If we construct a computing system based on this D/A-conversion method, a new analog-to-digital converting technique is required. That is, gray-level images must be decomposed into multiple binary ones with the format adaptable for processing by the multi-input type of OLAP for subsequent processing. If this can be done, the problem of treating carry-bit signals will be solved and optical parallel digital computing will be easily executed.

C. Gray-Level Coding Method

It is also possible to process gray-level objects directly by using the basic type of OLAP, if objects are encoded by modified code patterns. When binary objects are processed by the basic type of OLAP, the four kinds of input code patterns shown in Fig. 2 are used. We call this coding method the *binary coding method.* Here, we introduce another coding method, shown in Fig. 14, in which pixel values expressed by gray level are converted into the amount of shift of the transparent area in the cell. We call this coding method the *gray-level coding method.* The binary coding method is regarded as a special case of the gray-level coding method.

Processing by means of the gray-level coding method is suitable for arithmetic operations rather than for logical ones. Table 4 indicates arithmetic operations for gray-level objects and corresponding source patterns using four point sources.

Figure 15 shows examples of output signals to be obtained by using the gray-level coding method. The numbers in the

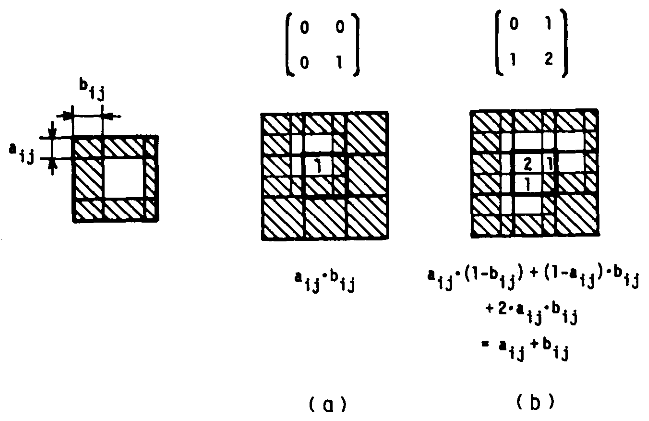

$\begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$
$\begin{pmatrix} 0 & 1 \\ 1 & 2 \end{pmatrix}$
b_{ij}
a_{ij}
1
2
1
1
$a_{ij} \cdot b_{ij}$
$a_{ij} \cdot (1-b_{ij}) + (1-a_{ij}) \cdot b_{ij}$
$+ 2 \cdot a_{ij} \cdot b_{ij}$
$= a_{ij} + b_{ij}$
(a)
(b)

Fig. 15. Format of output signal to be obtained by the gray-level coding method. (a) Output of A × B and (b) that of A + B, where × and + mean product and sum. The numbers in the boxed area show illuminance of the stated rectangular areas. The top shows source patterns.

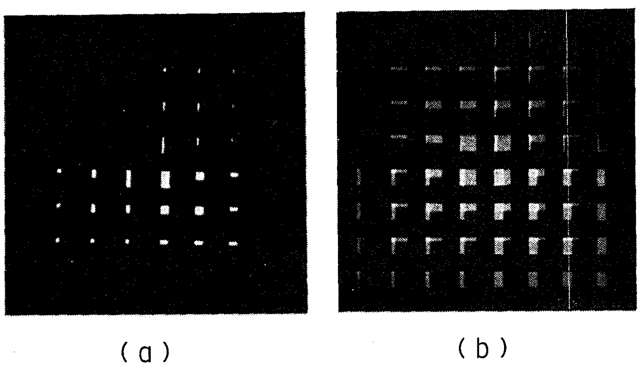


Fig. 16. Experimental results obtained by the gray-level coding method. (a) Results for A × B and (b) that for A + B.

Table 4. Matrix Expressions of Source Patterns Used in Gray-Level Coding Method

Source Pattern	Function	Source Pattern	Function	Source Pattern	Function
$\begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$	0	$\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$	(1 − A)(1 − B)	$\begin{pmatrix} 0 & 1 \\ 1 & 2 \end{pmatrix}$	A + B
$\begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$	AB	$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$	1 − (A + B) + 2AB	$\begin{pmatrix} 1 & 0 \\ 2 & 1 \end{pmatrix}$	A − B + 1
$\begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix}$	A(1 − B)	$\begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix}$	1 − B	$\begin{pmatrix} 1 & 2 \\ 0 & 1 \end{pmatrix}$	B − A + 1
$\begin{pmatrix} 0 & 0 \\ 1 & 1 \end{pmatrix}$	A	$\begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$	1 − B + AB	$\begin{pmatrix} 2 & 1 \\ 1 & 0 \end{pmatrix}$	2 − A − B
$\begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$	(1 − A)B	$\begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix}$	1 − A	$\begin{pmatrix} 0 & b \\ a & a+b \end{pmatrix}$	aA + bB
$\begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix}$	B	$\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$	1 − A + AB	$\begin{pmatrix} b & 0 \\ a+b & a \end{pmatrix}$	aA − bB + b
$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$	A + B − 2AB	$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$	1 − AB	$\begin{pmatrix} a & a+b \\ 0 & b \end{pmatrix}$	bB − aA + a
$\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}$	A + B − AB	$\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$	1	$\begin{pmatrix} a+b & a \\ b & 0 \end{pmatrix}$	(a + b) − aA − bB

rectangular areas in the boxes show illuminance. Figures 15(a) and 15(b) indicate the outputs of processing A × B and A + B, respectively, where × and + mean product and sum. Matrices at the top of these figures indicate corresponding source patterns. In both cases, the total light intensity reached in the boxed area is proportional to the pixel value to be obtained, which is the sum of multiplications of individual rectangular areas and their illuminance.

Figure 16 exhibits experimental results of gray-level image processing by using the gray-level coding method. Two gray-level objects from Fig. 12 are used as input objects. Figures 16(a) and 16(b) are the results of processing A × B and A + B, respectively.

Unfortunately, this processing requires a complicated coding procedure and a delicate optical setup as compared with those in the binary coding method. However, it is considered that this method is a novel processing method making good use of the analog nature of light. Thus processing by this method is expected to be one of fundamental techniques for a new optical computing system.

Key techniques of handling gray-level objects by the OLAP described in this section are those of converting gray-level objects into coded versions with the format adaptable for the OLAP. Two kinds of image processing, described in Section 3, can be implemented with the methods presented in this section. Thus parallel digital gray-level image processing, such as digital filtering and pattern matching, can be executed by the OLAP.

5. CONCLUDING REMARKS

The most difficult problem for the execution of optical parallel computing by the OLAP is how to encode input objects efficiently. In order to construct an efficient, parallel computing system, a technique of dynamic coding must be developed. For efficient coding, two methods have been considered.[11,12] One is the method using color coding and the other is that using a holographic technique. However, further investigation is needed for dynamic coding.

Since the OLAP operates on the basis of the principle of geometrical optics, the space–bandwidth product of the system is necessarily restricted, owing to an effect of diffraction. The resolution limit was estimated in the previous paper.[7]

Another comment is presented relating to the amount of information processed by the OLAP. The OLAP processes 2-D sampled signals but not continuous ones. Although the discrete objects have fewer independent data points than those of continuous ones within the unit area, they have enough redundancy to be free from optical noise introduced during conventional optical processing. Therefore a decrease of the number of processing data points in the unit area caused by the coding process does not diminish the applicability of the OLAP.

The use of spatial-light modulators is indispensable for the construction of actual optical parallel computing systems, which will be used mainly to acquire input objects to the optical system. Thus the development of spatial-light modulators, which have a much faster response time and much higher resolving power than those now available, is needed.

In this paper, we have clarified the applicability of the OLAP by focusing on optical parallel digital image processing. The OLAP has great flexibility for executing parallel, logical, and arithmetic processing, so that it would be useful not only for other applications in image processing but also for constructing optical parallel computing systems with simple architectures.

REFERENCES

1. J. W. Goodman, *Introduction to Fourier Optics* (McGraw-Hill, New York, 1968).
2. L. S. Haynes, R. L. Lau, D. P. Siewiorek, and D. W. Mizell, "A survey of highly parallel computing," Computer **15**, 9–24 (1982).
3. M. T. Fatehi, K. C. Wasmundt, and S. A. Collins, Jr., "Optical logic gates using liquid crystal light valve: implementation and application example," Appl. Opt. **20**, 2250–2256 (1981).
4. S. H. Lee, "Nonlinear optical processing," in *Optical Information Processing Fundamentals*, S. H. Lee, ed. (Springer-Verlag, Berlin, 1981), pp. 261–303.
5. P. Chavel, R. Forchheimer, B. K. Jenkins, A. A. Sawchuk, and T. C. Strand, "Architectures for a sequential optical logic processor," in *Proceedings of the Tenth International Optical Computing Conference* (MIT U. Press, Cambridge, Mass., 1983), pp. 6–12.
6. A. Huang, "Parallel algorithms for optical digital computers," in *Proceedings of the Tenth International Optical Computing Conference* (MIT U. Press, Cambridge, Mass., 1983), pp. 13–17.
7. J. Tanida and Y. Ichioka, "Optical logic array processor using shadowgrams," J. Opt. Soc. Am. **73**, 800–809 (1983).
8. W. K. Pratt, *Digital Image Processing* (Wiley, New York, 1977).
9. H. Fleisher and L. I. Maissel, "An introduction to array logic," IBM J. Res. Develop. **19**, 98–109 (1975).
10. J. Tanida and Y. Ichioka, "Optical-logic-array processor using shadowgrams. III. Parallel neighborhood operations and an architecture of an optical digital-computing system," J. Opt. Soc. Am. A **2**, 1245–1253 (1985).
11. Y. Ichioka and J. Tanida, "Optical parallel logic gates using a shadow-casting system for optical digital computing," Proc. IEEE **72**, 787–801 (1984).
12. J. Tanida and Y. Ichioka, "Image encoding by a computer generated holographic filter," Proc. Soc. Photo-Opt. Instrum. Eng. **437**, 119–124 (1983).