Image Segmentation and Pattern Matching Based FPGA/ASIC Implementation Architecture of Real-Time Object Tracking

K. Yamaoka, T. Morimoto, H. Adachi, T. Koide, and H. J. Mattausch
Research Center for Nanodevices and Systems, Hiroshima University,
1-4-2 Kagamiyama, Higashi-Hiroshima, 739-8527, Japan
Phone:+81-82-424-6265, Fax:+81-82-424-3499
Email:yamaoka, morimoto, adachi, koide, hjm@sxsys.hiroshima-u.ac.jp

Abstract— A novel algorithm for object tracking in video pictures, based on image segmentation and pattern matching, as well as its FPGA/ASIC implementation architecture are presented. With image segmentation, we can detect all objects in the images no matter whether they are moving or not. Using image segmentation results of successive frames, we exploit pattern matching in a simple object feature space for tracking of objects. The proposed algorithm can be applied to multiple moving and still objects even in the case of a moving camera. The FPGA/ASIC implementation architecture is verified to enable real-time tracking of up to 220 objects, when realized with modern FPGA hardware [1].

I. Introduction

For intelligent information processing of visual data, the moving object tracking in video pictures has attracted recently a great deal of interest [2]. Scene surveillance or object recognition are typical examples, where object tracking is an indispensable technology. Many moving object tracking algorithm and architectures have already been proposed. Most of them are based on difference evaluation between the current image and a previous image or a background image [3, 4]. However, algorithms based on the difference of images have problems with following practical cases. (1) Still objects included in the tracking task exist. (2) Multiple moving objects are present in the same frame. (3) The camera is moving. (4) Occlusion of objects occurs. Our novel algorithm for object tracking [5], based on image segmentation and pattern matching (Fig. 1), aims at solving above problems. In this algorithm we extract all objects from an input image by image segmentation. Next we extract simple object features and use these features to form pattern representing the objects. Then we compare the features of extracted objects in the current frame and those of extracted objects in the preceding frame by pattern matching. The most similar objects (i.e. the objects which have the smallest distance) between successive frames are judged to be corresponding objects. In spite of the motion condition of objects, this algorithm is effective because each object's motion vector is determined and used as one of its features. Additionally, we can increase the number of extracted object features from segmentation results, so that detection and tracking accuracy can be improved to a suitable level. However, image segmentation processing needs a great deal of calculations, so that realizing real-time object tracking with the proposed algorithm by software implementation is difficult. That is why, we have developed an FPGA/ASIC architecture for realizing real-time object tracking. For the segmentation part we exploit a previously developed digital image segmentation architecture [6]. Furthermore, pipeline processing of segmented objects is applied in order to achieve tracking of more objects in real time.

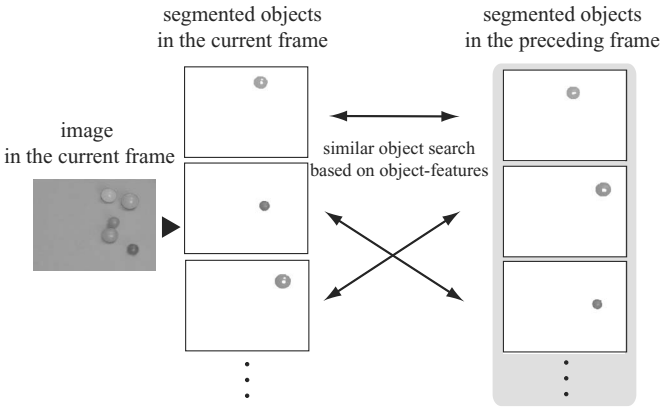


Fig. 1. Object-tracking based on image segmentation and similar object feature matching.

II. Object Tracking Algorithm

A. Concept

In the proposed object tracking algorithm, we employ image segmentation of each frame and extraction a number of features for all segmented objects. Then pattern matching with the objects of the previous frame is carried out. A coarse flow chart of the proposed algorithm is shown in Fig. 2. The detailed processing consists of the following steps.

Step 1: With the image segmentation algorithm, we extract all objects in the input image.

Step 2: Then we extract coordinates of 4 object-pixels which are indicated in Fig. 3(a). P_{xmax} and P_{xmin} have the maximum and minimum x-component, while P_{ymax} and P_{ymin} have the maximum and minimum y-component, respectively.

Step 3: Afterwards we calculate characteristic features of

0-7803-9451-8/06/$20.00

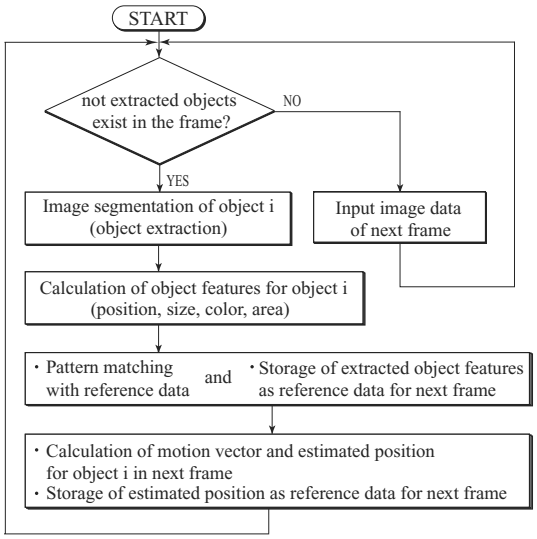
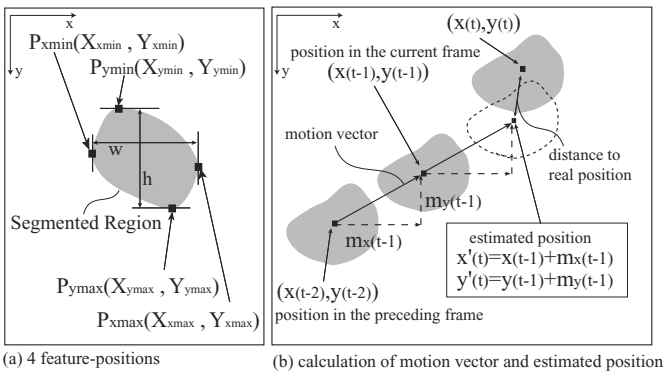


Fig. 2. Flowchart of the proposed algorithm based on image segmenation and pattern matching.



(a) 4 feature-positions (b) calculation of motion vector and estimated position

Fig. 3. Definition of four feature-positions, motion vector and estimated position.

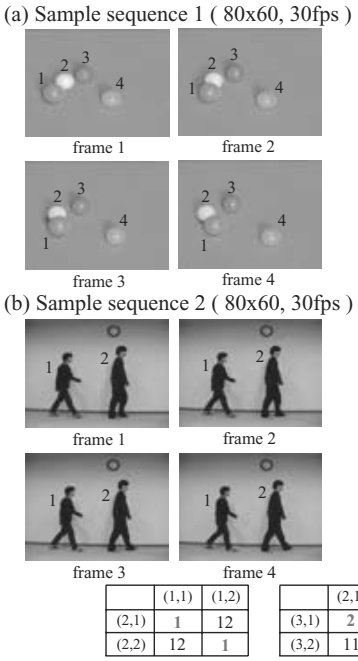
the segmented object, that is, object position (x, y), object size (width, height), color information (R, G, B), and object area, respectively. Object position (x, y), width w and height h are calculated according to below equations.

$$w = X_{xmax} - X_{xmin}, \qquad h = Y_{ymax} - Y_{ymin},$$
$$x = \frac{X_{xmax} + X_{xmin}}{2}, \qquad y = \frac{Y_{ymax} + Y_{ymin}}{2}.$$

The object area is determined by counting the number of its constituting pixels. As object color information, average RGB data of the 4 pixels, P_{xmax}, P_{xmin}, P_{ymax} and P_{ymin}, are used.

Step 4: The minimum distance search in the feature space is performed between each object in the current frame and all objects in the preceding frame. Then we identify each object in the current frame with the object in the preceding frame which has the minimum distance or in other words which is the most similar object.

Step 5: Afterwards, we calculate the motion vector $(m_x(t-1), m_y(t-1))$ from the difference in position between the object in the current frame and matching object in the preceding frame (Fig. 3(b)). By adding the motion vector $(m_x(t-1), m_y(t-1))$ to the current position $(x(t-1), y(t-1))$ of the object, we determine an estimate for the object's position $(x'(t), y'(t))$ in the next frame



(a) The Manhattan distance between succesive frames for the sample sequence 1.

	(1,1)	(1,2)	(1,3)	(1,4)
(2,1)	1	19	10	10
(2,2)	18	4	19	21
(2,3)	12	18	1	11
(2,4)	13	23	12	2

	(2,1)	(2,2)	(2,3)	(2,4)
(3,1)	2	17	10	13
(3,2)	19	6	19	26
(3,3)	10	19	2	15
(3,4)	10	25	14	3

	(3,1)	(3,2)	(3,3)	(3,4)
(4,1)	2	17	12	10
(4,2)	19	2	23	27
(4,3)	10	21	2	14
(4,4)	11	24	13	1

(b) The Manhattan distance between succesive frames for the sample sequence 2.

	(1,1)	(1,2)
(2,1)	1	12
(2,2)	12	1

	(2,1)	(2,2)
(3,1)	2	13
(3,2)	11	2

	(3,1)	(3,2)
(4,1)	2	10
(4,2)	12	0

Fig. 4. Successive image sequence for algorithm simulation.

(Fig. 3(b)). This estimate position is exploited instead of the extracted position $(x(t-1), y(t-1))$ for pattern matching after a start-up phase from the 3rd frame onwards.

Step 6: By carrying out this matching procedure with all segments obtained for the current frame, we can identify all objects one by one and can maintain tracking of all objects between frames.

B. Verification

For verifying the effectiveness of the proposed object tracking algorithm, we tested sample picture sequences with 80×60 pixels per frame, consisting of four successive frames (30fps) as shown in Fig. 4. The sample sequence 1 is an example, which includes multiple moving objects and also the occlusion effect among objects. The sample sequence 2 is an example, which includes non-rigid objects. Note that the object labels are explicitly shown in the pictures. Tables (a) and (b) in Fig. 4 show Manhattan distances between the objects of successive frames. We employ the simple Manhattan distance as the distance measure, because matching quality is found to be approximately the same as with the more complicated Euclidean distance [5]. The distances express the similarity of objects, namely as the Manhattan distance approaches to 0 the similarity increases. In this table, the notation (t,i) stands for the objects i in the t-th frame. For making possible distance contributions of each object feature equal, we have normalized their numerical values. From the table(a) in Fig. 4, we can confirm correct matching between objects in successive frames and thus confirm the validity of the proposed algorithm. Furthermore, we also have confirmed the effectiveness of the proposed algorithm for many other difficult cases such as rapid direction of movement changes by object collision, rotating complex objects or non-rigid objects like walking humans (sample

sequence 2 in Fig. 4). Since the object feature difference is also very small between successive frames for non-rigid objects, the proposed pattern matching based method can correctly track the same objects. By doing the verification based on the algorithm's software implementation, we could confirm that unless very small and very complex objects are tracked, sufficient tracking reliability is achieved with an image resolution of 80×60 pixels. For QVGA or VGA size images, we can shrink the original image to 80×60 pixel size image. Therefore, we design the FPGA/ASIC implementation architecture aiming at video pictures reduced to 80×60 pixels in size.

III. Proposed Architecture for FPGA/ASIC Implementation

This section presents the FPGA/ASIC architecture for implementing the proposed algorithm in more detail.

A. Architecture Overview

Fig. 5 shows the overall block diagram of the developed FPGA/ASIC implementation architecture. This architecture roughly consists of 4 blocks. The first block is the image segmentation cell-network in which all objects of the frame are extracted. The second block is the feature extraction block in which object features for each segmented object are calculated using the image segmentation results. The third block is the pattern matching block in which the most similar object is searched among the reference data from the previous frame. The fourth block is the estimated position calculation block in which the estimated position of each object in the next frame is calculated.

The image segmentation cell-network implements a region-growing algorithm and has the structure of a two-dimensional array of image segmentation cells corresponding to the pixels of an input image. By taking advantage of the cell-network, we can access the segmentation result of each cell in parallel in x-direction and in y-direction. This is done in the feature extraction block where the width of each segmented object is calculated from the cell-network data which is outputted in parallel into y-direction and where the height and the area of each object are calculated from the cell-network data which is outputted in parallel into x-direction. With this chosen approach it is not necessary to scan pixels belonging to a segmented region one by one, and thus high speed processing is achieved. Then the determined object-features are transmitted to pattern matching block so as to search the most similar object among the reference data in the previous frame. In the estimated position calculation block, the estimated position of the current input object in the next frame is calculated from the positions of the matched object and the input object. Then the estimated position is stored in the pattern matching block as one feature of the input object's reference pattern in the next frame.

Due to the sequential nature of the segmentation by region growing, we can apply pipeline processing, as shown in Fig. 6, to interleave the processing steps of image segmentation, feature extraction and pattern matching. As compared to the case of completing the frame's segmentation before advancing to feature extraction, higher processing speed of the complete algorithm is achieved, so that more objects can be tracked in real time. This is evaluated quantitatively in SectionIV.

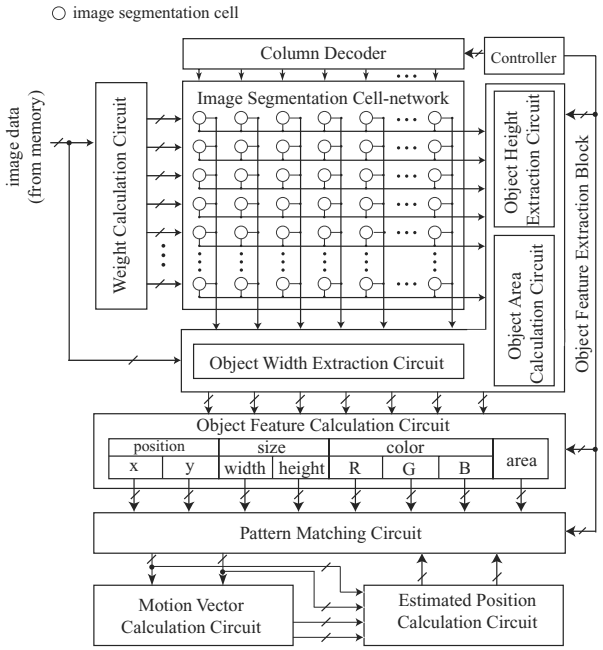


Fig. 5. Block diagram of proposed FPGA/ASIC implementation architecture.

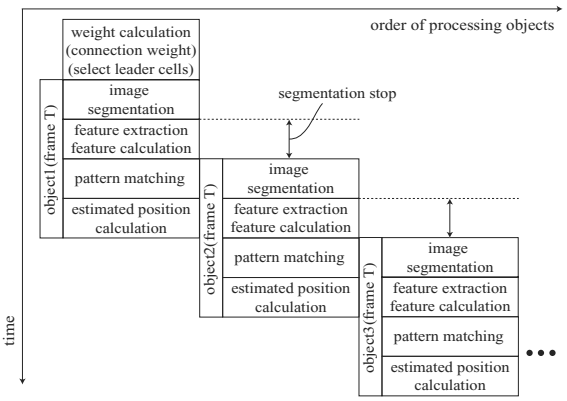


Fig. 6. Process flow of the FPGA/ASIC implementation architecture when using pipeline processing.

In following subsections, we explain the circuit structure and function of each of the 4 blocks.

B. Image Segmentation Cell-Network

The proposed implementation architecture uses a region-growing-type image segmentation cell-network [6, 7]. This cell-network realizes high-speed object segmentation from an input image. The cell-network is an array of image segmentation cells, where each cell corresponds a pixel of the input image. In a preprocessing step to the segmentation, connection weights between adjacent pixels are calculated from the pixel's RGB data in a weight calculation circuit. These calculated connection-weights are transmitted to the image segmentation cell-network,

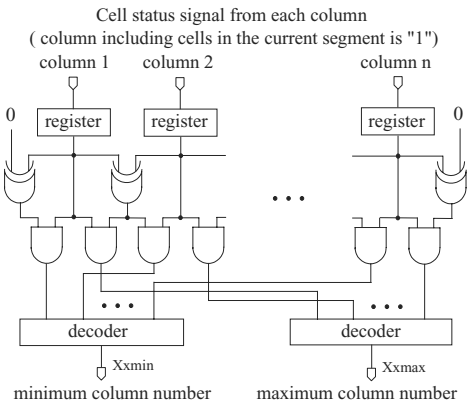


Fig. 7. Width extraction circuit for the current segment.

and image segmentation is done based on these connection weights. In the image segmentation cell-network, all segments are grown sequentially by a region-growing process starting from leader cells, which are also determined by the connection weights. After segmentation of one object, if a pixel belongs to the currently segmented object, a flag register of the corresponding cell has been set to 1 and a label number of the object has been stored in an internal register.

C. Feature Extraction and Calculation Block

C.1 Basic Feature Extraction Circuits

The feature extraction circuit is shown in Figs. 7 and 8. The four feature-positions shown in Fig. 3(a) are extracted in these circuits. The circuit shown in Fig. 7 extracts the maximum and the minimum x-coordinates, X_{xmin} and X_{xmax}, for calculating width of the segmented object. As shown in Fig. 5, status signals are transmitted from each column of the image segmentation cell-network to this circuit in parallel. The signal from each column becomes 1 if the column contains the pixel belonging to current segment. Otherwise it becomes 0. These input signals are stored in registers corresponding to each column in Fig. 7. The boundaries between 1 and 0 are detected with the shown combinational circuit consisting of EXOR and AND gates, and are transformed to the column index in the two decoders. The max-min detection unit (y-component) shown in Fig. 8 extracts the maximum and the minimum y-coordinates, Y_{ymin} and Y_{ymax}, for calculating height of the current segment and calculates the area at the same time. To extract Y_{ymin} and Y_{ymax}, the same circuit shown in Fig. 7 is used. After an image segmentation of an object finished, cell status signals (0 or 1) are inputted to registers which connected from rows of the cell-network in column parallel and Y_{ymax} and Y_{ymin} of each column are sequentially outputted from decoders. Then by inputting cell status signals of X_{xmax}-th and X_{xmin}-th column, Y_{xmax} and Y_{xmin} are calculated. To extract remaining coordinates, X_{ymin} and X_{ymax}, min and max position detection units (x-component) depicted in Fig. 8 are used. Input signals to these units are the output signal from each decoder and the current column number. When the output from each decoder becomes minimum or maximum, the current column number is

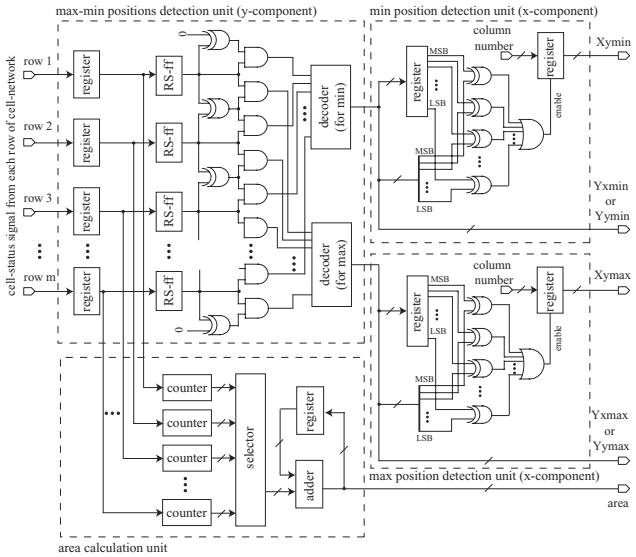


Fig. 8. Second basic feature extraction circuit which extracts height and area of the segmented region.

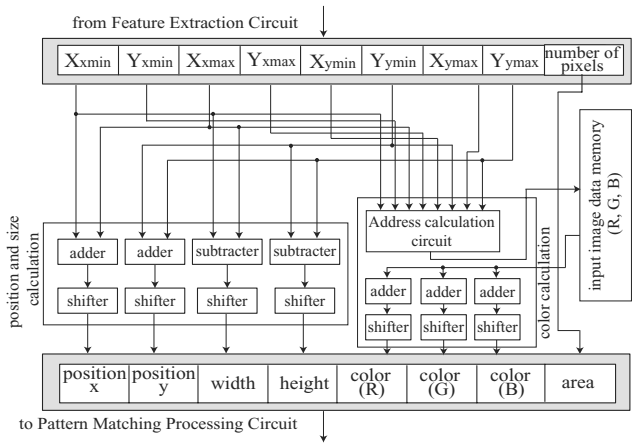


Fig. 9. Dependent feature calculation circuit.

stored to each register as X_{ymin} or X_{ymax}. Concurrently, area of the current segment is calculated by counting the number of the cells in the current segment of each row, and all counter's values are finally summed up.

C.2 Dependent Feature Calculation Circuit

As shown in Fig. 9, the dependent feature calculation circuit consists of registers, adders, subtracters, and shifters. This circuit calculates the eight features, position(x, y), size(width, height), color(R, G, B) and area, by using 4 feature-positions detected with the basic feature extraction circuit and outputs these object features to pattern matching circuit. Color information is calculated by reading four boundary pixels data shown in Fig. 3(a) and taking the average of luminances (R, G, B) of four pixels.

D. Pattern Matching Processing Circuit

The structure of the pattern matching circuit is shown in Fig. 10. Two memories, memory-A and memory-B are used for pattern matching. One of the memories stores the object-features in the current frame, the other stores the object-features in the preceding frame as reference data.

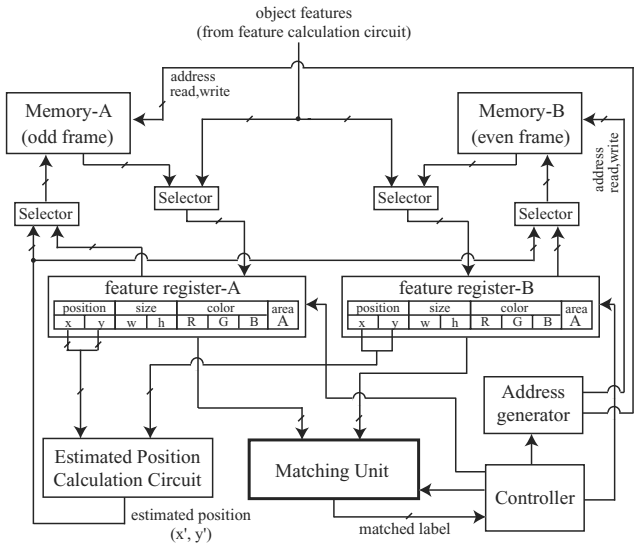


Fig. 10. Block diagram of the pattern matching circuit and estimated position calculation circuit.

For odd frames, an input data is written in memory-A and the reference data is read from memory-B. For even frames, an input data is written in memory-B and the reference data is read from memory-A. In this way, the functionarity of each memory is switched when processing frame is changed.

The pattern matching process is done in the matching unit. All combinations of the current object and all reference objects in the preceding frame are checked in the matching unit one by one. For that purpose, a feature-register-A and a feature-register-B are temporarily used as the buffer to storing the current object and a reference object. Then the matching unit compares the input data and the reference data which are set in both feature-registers.

The block diagram of the matching unit is shown in Fig. 11. This unit consists of a Manhattan distance calculation circuit which calculates the distance (i.e. similarity) between the input data and all reference data, and a Manhattan distance comparison circuit, which searches the most similar combination (i.e. the smallest distance). Each difference is calculated sequentially per object feature pair and is normarized by shifters. Afterward all distances components are added serially and the calculated Manhattan distance is transmitted to the Manhattan distance comparison circuit, and is stored in a comparison register (A or B) of the circuit with the label of the reference object. Then, this newly calculated distance is compared with up to now smallest distance between the input data and other reference data. And smaller distance is selected to the candidate of the most similar object. The selected distance is kept intact in its register and the other is deleted from its respective register. Repeating these processes for all reference data of the previous frame in memory, the most similar reference object is selected and the matched object-label is outputted from the matching unit.

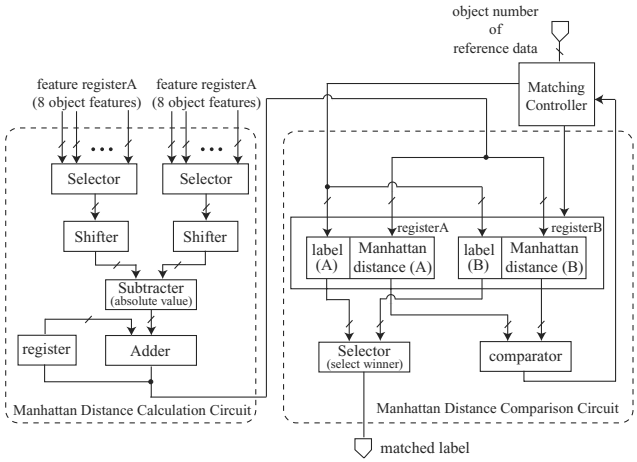


Fig. 11. Details of the matching unit.

E. Estimated Position Calculation Circuit

The estimated position calculation circuit consists of adder and subtracter and a motion vector of input object is calculated with distance between the matched object position in the preceding frame and the input object position in the current frame (Fig. 10). The estimated position of the input object for the next frame is calculated by adding the motion vector to the current position of the input object, and is stored to the reference memory to apply to the pattern matching of the next frame.

IV. Performance Estimation of Proposed Architecture

Here we estimate the processing speed of the proposed FPGA/ASIC architecture and judge the possibility of real time processing (30fps). The process flow of the proposed architecture is shown in Fig. 6 and we assumed that the achievable clock frequency with the FPGA is 20MHz in the worst case. First, when an image is inputted, the connection weights are calculated and leader cells are determined in the weight calculation circuit. These data are then transfered to image segmentation cell-network. With the assumed worst-case operating frequency of 20MHz, it takes 480μsec to finish calculating and transfering the connection-weight and leader cell data. Next, image segmentation is done, which requires about 200μsec for 80×60 image size at 20MHz. After image segmentation is finished, feature extraction processing is started. In the basic feature extraction circuits, calculating the area of segmented objects needs scanning of all columns belonging to the segmented object, so that area calculation becomes the critical path. In the worst-case it takes 80 clock cycles for counting the number of pixels belonging to the segmented region. Additionally, it takes 60 clock cycles to calculate the sum of the counted pixels of all columns. The delay of the dependent feature calculation circuit is only 5 clock cycles and thus quite small. Therefore it takes all together 145 clock cycles or 7.25μsec at 20MHz to extract the features of one object in the worst case.

After finishing the extraction of object features, pattern matching is done. With the described implementa-

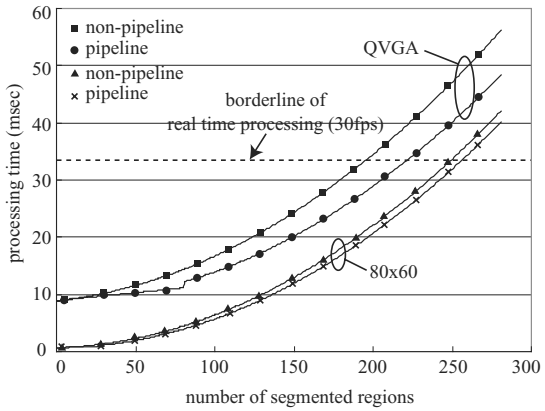


Fig. 12. Relationship between the number of segmented regions and the processing time

tion circuit it takes 10 clock cycles or 0.5μsec at 20MHz to finishing the processing for one reference pattern. In the case of N objects existing in the current frame and correspondingly N reference pattern from the previous frame, it takes therefore $0.5\times N^2\mu$sec to execute pattern matching for all objects. The final processing in the estimated position calculation circuit needs only a few clock cycles and can be neglected.

Fig. 12 shows the graph which plots the number of segmented regions vs. the processing time for two kinds of image size, QVGA and 80×60 pixels. The process of pattern matching with the sequential implementation of Figs. 10 and 11 requires much time. But as shown in Fig. 6 we can shorten the total amount of time by executing image segmentation, feature extraction and pattern matching in a processing pipeline. In this way, about 220 objects for QVGA image size and about 255 objects for 80×60 pixel image size can be handled in a real-time tracking application under the condition that the operating frequency is 20MHz. If the number of objects is less than about 50, then we can slow down the clock frequency to reduce the power dissipation. Futhermore Fig. 12 shows, that as the image size becomes larger, the effect of pipeline processing on processing time becomes more significant.

We have coded the proposed FPGA/ASIC architecture in Verilog-HDL and carried out logic synthesis with a standard cell library in 0.35μm CMOS technology. The image segmentation cell-network in the case of 80×60 pixel images is estimated 124mm^2, and therefore consumes the largest area. All other elements of the proposed object-tracking circuit can be implemented on an area of only 0.94mm^2 when memories are excluded. This gives a total of about 125mm^2 for an object tracking ASIC (80×60 pixel images) in 0.35μm CMOS with external memories. The area can be reduced to about 20mm^2 with a state-of-the-art 100nm CMOS technology, so that memories can also be integrated on the ASIC.

V. Conclusions

We have proposed an object tracking architecture for video pictures, based on image segmentation and pattern matching of the segmented objects between frames in a simple feature space. The suitability of the proposed algorithm was verified by simulation. It could be comfirmed that the algorithm overcomes all insufficiencies of the conventional approach, which is based on the difference between the current image and a previous image or a background image. In particular the cases of a moving camera or object occlusion, where objects are hard to track, are no problem for our proposed algorithm. Then we have proposed an FPGA/ASIC implementation architecture, realizing its algorithm for real time object tracking. The detailed circuitry of each processing block of this architecture was described. Furthermore, we estimated the processing speed of the proposed architecture for FPGA implementation with 80×60 pixel and QVGA size images. We comfirmed that even this FPGA implementation can track multiple moving objects in the same frame in real time. The estimated area of proposed architecture for 80×60 pixel with an ASIC in 0.35μm CMOS technology is about 125mm^2. By applying state-of-the-art CMOS technology at the 100nm node, the ASIC area can be reduced to about 20mm^2, so that larger image sizes for the tracking algorithm become possible. It will be also possible to implement the proposed architecture with state-of-the-art FPGA and we have already proposed a compact image segmentation architecture for FPGA implementation in Ref.[8].

Acknowledgements

The authors are grateful to O. Kiriyama, Y. Harada, from Hiroshima University, Japan, for fruitful advices and discussions. Part of this work was supported by the 21st Century COE program "Nanoelectronics for Tera-bit Information Processing", a Grant-in- Aid for Young Scientists (B) (No.16700184), Ministry of Education, Culture, Sports, Science and Technology, Japanese Government and a Grant-in-Aid for JSPS Fellows, 1650741, 2005.

References

[1] StratixII, Altera Corporation, 2005,
URL: http://www.altera.com/products/devices/stratix2/.

[2] See, for example, W. G. Kropatsch and H. Bischof, "Digital Image Analysis," Springer, 2001.

[3] S. W. Seol et al., "An automatic detection and tracking system of moving objects using double differential based motion estimation," *Proc. of International Technical Conference on Circuits/Systems, Computers and Communications*, pp. 260–263, 2003.

[4] H. Kimura and T. Shibata, "Simple-architecture motion-detection analog V-chip based on quasi-two-dimensional processing," *Extended Abstracts of the 2002 International Conference on Solid State Devices and Materials*, pp. 240-241, 2002.

[5] T. Morimoto et al., "Object tracking in video pictures based on image segmentation and pattern matching," *Proc. of the IEEE Int. Symp. on Circ. and Syst.(ISCAS2005)*, pp.3215-3218, 2005.

[6] T. Morimoto et al., "Efficient video-picture segmentation algorithm for cell-network-based digital CMOS implementation," *IEICE Transactions on Information and Systems*, Vol. E87-D, No. 2, pp. 500-503, 2004.

[7] T. Morimoto et al., "Pixel-parallel digital CMOS implementation of image segmentation by region growing," *IEE Proceedings Circuits, Devices & Systems*, in press.

[8] H. Adachi et al., "Image-scan architecture for efficient FPGA/ASIC implementation of video-segmentation by region growing," *Proc. of The International SoC Design Conference(ISOCC2005)*, pp. 301-304, 2005.