# Efficient Object Tracking Algorithm using Image Segmentation and Pattern Matching 

Osamu Kiriyama, Takashi Morimoto, Hidekazu Adachi, Zhao-Min Zhu, Tetsushi Koide, and Hans Jürgen Mattausch<br>Research Center for Nanodevices and Systems, Hiroshima University, 1-4-2 Kagamiyama, Higashi-Hiroshima, 739-8527, Japan<br>Phone: +81-82-424-6265 Fax:+81-82-424-3499 e-mail: \{kiriyama, koide, hjm\}@sxsys.hiroshima-u.ac.jp

## 1. Introduction

The moving object tracking in video pictures has attracted a great deal of interest in computer vision. For object recognition, navigation systems and surveillance systems, object tracking is an indispensable first-step.
The conventional approach to object tracking is based on the difference between the current image and the background image. However, algorithms based on the difference image cannot simultaneously detect still objects. Furthermore, they cannot be applied to the case of a moving camera. Algorithms including the camera motion information have been proposed previously, but, they still contain problems in separating the information from the background.
In this paper, we propose a novel algorithm for object tracking in video pictures. Our algorithm is based on image segmentation and pattern matching. With the image segmentation algorithm, we can extract all objects in images. The proposed algorithm for tracking uses pattern matching between successive frames. As a consequence, the algorithm can simultaneously track multiple moving and still objects in video pictures and can even be applied in the case of a moving camera.

## 2. Proposed Concept for Moving Object Tracking

An accurate image segmentation algorithm is necessary for our object tracking algorithm. A candidate for such an algorithm has been proposed in Ref. [2]. The image segmentation algorithm allows digital VLSI implementation and has already verified by designed and fabricated image segmentation testchip [3].
Using image segmentation results, we extract the following eight features of segmented objects:

1) Area: By counting the number of pixels included in segment $i$ of the $t$-th frame, we calculate the area of the object $a_{i}(t)$.
2) Width and Height: We extract the positions of the pixel $P_{x \max }$, $P_{x \min }, P_{y \max }$, and $P_{y \text { min }}$ and then calculate the width $w_{i}(t)$ and $h_{i}(t)$ as follows (see Fig. 1)

$$
w_{i}(t)=X_{\max , x}-X_{\min , x}, h_{i}(t)=Y_{\max , y}-Y_{\min , y} .
$$

3) Positions: We define the positions of each object ( $x_{i}(t)$ and $\left.y_{i}(t)\right)$ in the frame as

$$
x_{i}(t)=\left(X_{\max , x}+X_{\min , x}\right) / 2, y_{i}(t)=\left(Y_{\max , y}+Y_{\min , y}\right) / 2 .
$$

4) Color: Using the image data at $P_{x \max }, P_{x \min }, P_{y \max }$, and $P_{y m i n}$, we define the color features of each object for the $R$ (Red) component

$$
R_{i}(t)=\left[R\left(P_{x \max }\right)+R\left(P_{x \min }\right)+R\left(P_{y \max }\right)+R\left(P_{y \min }\right)\right] / 4
$$

and vice versa.
We now go into details of our algorithm. The proposed algorithm for object tracking exploits pattern matching with the features above. We make use of the minimum distance search between $(t, i)$ and all objects in the preceding frame $(t-1, j)$ (the notation $(t, i)$ stands for the object $i$ in the $t$-th frame). The object $(t, i)$ is identified with the object in the $(t-1)$-th frame which has the minimum distance from $(t, i)$. Repeating this matching procedure for all segments in the current frame, we can identify all objects one by one and can keep track of the objects between frames.

A few comments on refinements of the proposed algorithm are in order.
(1) In calculation of the distance between $(t, i)$ and $(t-1, j)$ in position space, it is more appropriate to take account of motion determination and use estimated positions $x_{j}{ }^{\prime}(t-1)$ and $y_{j}{ }^{\prime}(t-1)$ instead of raw positions $x_{j}(t-1)$ and $y_{j}(t-1)$ (see Fig. 2). These replacement are available and used from the third frame onwards.
(2) We have not specified the distance measure used for matching yet. In the simulation experiments we could confirm that besides the Euclidean distance the simpler Manhattan distance is already sufficient for object tracking purposes.
(3) In order to treat all object features with equal weight, it is necessary to normalize the features. One possible way is dividing them by their maximum values. Dividing by $2^{n}$, where the integer n is determined for each feature so that approximately equal weights result, is another possibility. The later has the advantage that the division can be realized by a shifting operation in a hardware realization.

## 3. Simulations

This section presents simulated results of the object tracking algorithm. In Fig. 3, a sequence of four sample frames with QVGA size can be seen. Note that we explicitly show object indices in the pictures. The extracted features of the objects in frames are listed in Table I. In this table, we have normalized the area feature by division with $2^{8}$ and the other features by division with $2^{4}$. Furthermore, the decimal parts of the numbers have been omitted.
The tracking quality is evaluated with the Euclidean and the Manhattan distances. In Tables II and III, distances between successive frames are listed. One can see that all objects correctly match with their counterparts in preceding frame no matter whether the Euclidean distance is used or the Manhattan distance is. These results of simulation experiments verify the proposed algorithm's efficiency.
We have also confirmed that the algorithm works very well for more complicated video pictures including rotating objects and occlusion of objects. Furthermore, if mistracking occurred at some frame by reason of occlusion, newly appearing or disappearing objects, the proposed algorithm could recover correct tracking after a couple of frames. This stability characteristic of the algorithm results from the fact that the object matching is performed in feature space between all objects in successive frames.

## 4. Conclusions

We have proposed an object tracking algorithm for video pictures, based on image segmentation and pattern matching of the segmented objects between frames in a simple feature space. Simulation results for a frame sequence with moving balls verify the suitability of the algorithm for reliable moving object tracking.
The relative simplicity of this tracking algorithm promises that an FPGA implementation is possible and already sufficient for real-time applications with a few moving objects. As noted earlier, it is sufficient for the tracking to use the simple Manhat-
tan distance. Thus, VLSI implementation of the algorithm is possible by using our developed architectures for image segmentation [3] and a fully-parallel associative memory for highspeed minimum Manhattan distance search [4], both of which have been already realized as VLSI circuits.

## Acknowledgment

The work was supported in part by Grant-in-Aid for Encouragement of Young Scientists (No. 16700184) from the Ministry of Education, Culture, Sport, Science and Technology, Japanese Government.

TABLE I: Extracted features of the objects. The symbol $(t, i)$ denotes the object $i$ in the $t$-th frame. In order to treat all features with equal weight, we have normalized area by division with $2^{8}$ and the other features by division with $2^{4}$. Furthermore, the decimal parts of the numbers have been omitted.


Fig. 1: Explanation of the proposed feature extraction from the image segmentation result. Notice that $X_{\text {max, }, x}, X_{\text {max, }, ~}, X_{\text {min, },}$, and $X_{\text {min }, y}$ $\left(Y_{\max , x}, Y_{\max , y}, Y_{\min , x}\right.$, and $\left.Y_{\min , y}\right)$ are the $x$ - and $y$-coodinates of the rightmost and leftmost (uppermost and lowermost) boundary of segment $i$.


Fig. 2: Estimation of the positions in the next frame. Here, $(t, i)$ denotes the object $i$ in the $t$-th frame and vice versa and $m_{x, j}(t-1)$ and $m_{y ; j}(t-1)$ denote the motion vector of $x$ and $y$-directions of the object $j$. These estimations are available from the third frame onwards and included in pattern matching.

| object | $a$ | $w$ | $h$ | $x$ | $y$ | $m_{x}$ | $m_{y}$ | $R$ | $G$ | $B$ |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| $(1,1)$ | 3 | 2 | 2 | 9 | 9 | - | - | 13 | 4 | 6 |
| $(1,2)$ | 3 | 2 | 1 | 11 | 3 | - | - | 15 | 6 | 4 |
| $(1,3)$ | 4 | 2 | 2 | 16 | 4 | - | - | 15 | 4 | 4 |
| $(1,4)$ | 2 | 1 | 1 | 15 | 6 | - | - | 7 | 6 | 15 |
| $(1,5)$ | 2 | 1 | 1 | 18 | 11 | - | - | 0 | 5 | 11 |


| object | $a$ | $w$ | $h$ | $x$ | $y$ | $m_{x}$ | $m_{y}$ | $R$ | $G$ | $B$ |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| $(3,1)$ | 3 | 2 | 2 | 11 | 9 | 1 | 0 | 13 | 4 | 6 |
| $(3,2)$ | 3 | 2 | 1 | 12 | 3 | 0 | 0 | 15 | 6 | 4 |
| $(3,3)$ | 4 | 2 | 2 | 15 | 3 | 0 | 0 | 15 | 4 | 5 |
| $(3,4)$ | 2 | 1 | 1 | 13 | 7 | -1 | 0 | 6 | 6 | 14 |
| $(3,5)$ | 2 | 1 | 1 | 16 | 11 | -1 | 0 | 0 | 5 | 12 |


| $(4,1)$ | 3 | 2 | 2 | 12 | 9 | 1 | 0 | 13 | 4 | 6 |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| $(4,2)$ | 3 | 2 | 2 | 12 | 2 | 0 | -1 | 15 | 6 | 4 |
| $(4,3)$ | 4 | 2 | 2 | 15 | 3 | 0 | 0 | 15 | 4 | 5 |
| $(4,4)$ | 2 | 1 | 1 | 13 | 7 | 0 | 0 | 6 | 5 | 14 |
| $(4,5)$ | 2 | 1 | 1 | 16 | 11 | 0 | 0 | 0 | 5 | 11 |



Fig. 3: Sample video pictures " 5 moving balls", including a collision between the balls. We have explicitly shown object indices in the pictures.

TABLE II: The Euclidean distances between successive frames. Distances shown in red correspond to the minimum distances. Obviously, all objects in the current frame match with their counterparts in the preceding frame.

| object | $(1,1)$ | $(1,2)$ | $(1,3)$ | $(1,4)$ | $(1,5)$ |
| :---: | :---: | :---: | :---: | :---: | :---: |
| $(2,1)$ | $\mathbf{1}$ | 7 | 8 | 12 | 16 |
| $(2,2)$ | 7 | $\mathbf{1}$ | 4 | 14 | 19 |
| $(2,3)$ | 9 | 4 | $\mathbf{1}$ | 14 | 18 |
| $(2,4)$ | 12 | 14 | 14 | $\mathbf{2}$ | 8 |
| $(2,5)$ | 16 | 19 | 18 | 9 | $\mathbf{1}$ |


| object | $(2,1)$ | $(2,2)$ | $(2,3)$ | $(2,4)$ | $(2,5)$ |
| :---: | :---: | :---: | :---: | :---: | :---: |
| $(3,1)$ | $\mathbf{0}$ | 7 | 8 | 11 | 15 |
| $(3,2)$ | 7 | $\mathbf{1}$ | 3 | 14 | 19 |
| $(3,3)$ | 7 | 3 | $\mathbf{1}$ | 14 | 18 |
| $(3,4)$ | 11 | 14 | 14 | $\mathbf{1}$ | 8 |
| $(3,5)$ | 15 | 19 | 19 | 7 | $\mathbf{0}$ |


| object | $(3,1)$ | $(3,2)$ | $(3,3)$ | $(3,4)$ | $(3,5)$ |
| :---: | :---: | :---: | :---: | :---: | :---: |
| $(4,1)$ | 0 | 7 | 7 | 11 | 14 |
| $(4,2)$ | 7 | $\mathbf{1}$ | 4 | 14 | 19 |
| $(4,3)$ | 7 | 4 | 0 | 14 | 18 |
| $(4,4)$ | 11 | 14 | 13 | $\mathbf{1}$ | 7 |
| $(4,5)$ | 14 | 18 | 18 | 8 | $\mathbf{1}$ |

TABLE III: The Manhattan distances between successive frames. Distances shown in red correspond to the minimum distances. Notice that all objects in the current frame match with their counterparts and the use of the simpler Manhattan distance is sufficient for the algorithm.

| object | $(1,1)$ | $(1,2)$ | $(1,3)$ | $(1,4)$ | $(1,5)$ |
| :---: | :---: | :---: | :---: | :---: | :---: |
| $(2,1)$ | $\mathbf{1}$ | 14 | 16 | 28 | 32 |
| $(2,2)$ | 16 | $\mathbf{1}$ | 9 | 27 | 39 |
| $(2,3)$ | 17 | 8 | $\mathbf{2}$ | 28 | 38 |
| $(2,4)$ | 27 | 28 | 30 | 4 | 18 |
| $(2,5)$ | 33 | 40 | 36 | 18 | $\mathbf{2}$ |


| object | $(2,1)$ | $(2,2)$ | $(2,3)$ | $(2,4)$ | $(2,5)$ |
| :---: | :---: | :---: | :---: | :---: | :---: |
| $(3,1)$ | 0 | 15 | 15 | 23 | 30 |
| $(3,2)$ | 14 | 1 | 7 | 27 | 38 |
| $(3,3)$ | 14 | 7 | 3 | 31 | 36 |
| $(3,4)$ | 24 | 25 | 31 | 1 | 16 |
| $(3,5)$ | 30 | 37 | 39 | 15 | 0 |
| object | $(3,1)$ | $(3,2)$ | $(3,3)$ | $(3,4)$ | $(3,5)$ |
| $(4,1)$ | 0 | 13 | 13 | 22 | 28 |
| $(4,2)$ | 14 | 1 | 9 | 26 | 38 |
| $(4,3)$ | 13 | 8 | 0 | 31 | 35 |
| $(4,4)$ | 22 | 27 | 29 | 2 | 14 |
| $(4,5)$ | 28 | 37 | 35 | 18 | 2 |

O. Kiriyama, T. Morimoto, H. Adachi, Z. Zhu, T. Koide, and H. J. Mattausch System Design and Architecture Research Division, Research Center for Nanodevices and Systems



