Proceedings of the 12th
KACC, October 1997

IMPLEMENTATION OF THE MULTI-TARGET TRACKER FOR MIROSOT

°Chu-sik In*, Yong-hee Choi**, Ja-sung Lee**

* : Tech. Center, DAEWOO Motor Co., LTD, 199 Chongchon-dong, Pupyung-gu, Inchon 403-714, KOREA
Tel : +82-32-520-2264; E-mail : workics@hitel.kol.co.kr
** : Dept. of E&E Eng., Ajou Univ., San-5, Wonchon-dong, Paldal-gu, Suwon 441-749, KOREA
Tel : +82-331-219-2482; E-mail : jsung@madang.ajou.ac.kr

Abstract One of the most important design factor for the image tracker is the speed of the data processing which allows real-time operation of the system and provides reasonably accurate performance at the same time. Use of powerful DSP alone does not guarantee to meet such requirement.

In this paper, a simple efficient algorithm for real-time multi-target image tracking is suggested. The suggested method is based on a recursive centroiding technique and color table look-up. This method has been successfully implemented in a image processing system for Micro-Robot Soccer Tournament (MIROSOT). This tracker can track positions of a ball, 3 enemies, and 3 agents at the same time. The experimental results show that the processing time for each frame of image is less than 7ms, which is well within the 60Hz sampling interval for real-time operation.

Keywords Real-time, Multi-target, Tracker, Centroiding, Look-up table

1. INTRODUCTION

Tracking objects more rapidly and precisely has become an increasingly important problem for such applications as strategic target tracking and robot control systems which require real-time processing of huge amount of multiple target moving image data.

The image tracker is the system which provides the positions of the moving targets by processing data from image sensor. One of the most important factors for the image tracker is the processing speed. Image sensors like CCD camera generate huge amount of data in such a frame rate of 60Hz. Thus, the image tracker needs to have capability to process each frame of data in 1/60 seconds for continuous and smooth tracking[1].

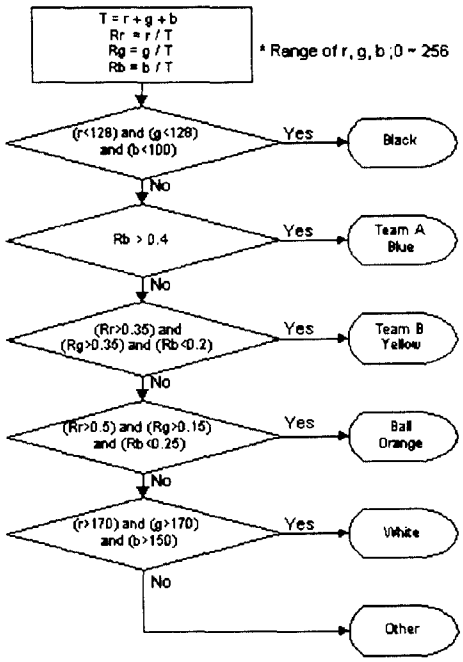
In conventional image trackers, reduced frame rate has been often employed due to the limit of the processing speed of the system. For slow systems, frame rate may be reduced to like 10Hz, or 5Hz. But in this case, the continuity of frames will also be reduced, and the tracking performance may become poor or inadequately slow to track fast moving targets[2][3]. One may reduce the size of the image data by making image resolution low. But, this will also reduce the tracking accuracy[2][3]. To overcome such problems, conventional image trackers often use high speed processor like DSP. Such devices are usually very expensive.

In this paper, an efficient and practical real-time multi-target tracker is suggested. This tracker can be operated on a regular PC, without the need for extra processor like DSP. The tracker is applied to an image processing system for micro-robot soccer tournament games(MIROSOT). The tracker tracks seven targets - a ball, three enemies, and 3 agents at a time. For high speed processing, a color look-up table is used for separating the objects from background, and a recursive centroiding technique is used to estimate the positions of the objects.

2. PRE-PROCESSING

In pre-processing, the target images are first separated from the background. According to the rules of MIROSOT, robots should have unique team color (blue or yellow) on their top, and orange colored golf ball is used for the target ball. The play ground is colored black and has white grid-lines. The most easy and efficient method to separate targets from the background is color classification.

Intensities of the light from the play ground are not regular. So, in order to get rid of the irregular effect of the light and to classify colors of target at the same time, we use the ratio of R, G, B of the image data. Fig. 1 shows the sequence of the pre-processing.



<Fig.1> Color classification

3. Tracking

Tree binary images $Fo(x, y)$, $Fb(x, y)$, $Fy(x, y)$ which correspond to the three kinds of objects (Ball, Enemy, Agent) are obtained from pre-processed color codes. To get positions of objects from each binary images, projection profiles are used[4].

The row and column projection profiles of 256X230 binary image can be written as follow.

$$H_x(x) = \sum_{y=0}^{229} F(x, y) \quad (1)$$

$$H_y(y) = \sum_{x=0}^{255} F(x, y) \quad (2)$$

If both $Hy_x(x)$ and $Hy_y(y)$ are greater than the specified threshold, and $F(x, y)$ is binary 1, then it is possible that the object is in position x, y. We can determine if the object is in position x,y by checking the number of connected pixels which is binary 1. If the number is greater than the threshold, the object is there. The position of the object (Px_n , Py_n) is then calculated by centroiding. Through these processes (thresholding, centroiding), the effect of the noise is reduced. And the use of projection profiles greatly reduces the amount of the data to be processed.

Nevertheless, amount of the pre-processed data is still too big for real-time processing, especially in the low transmission rate of PC I/O BUS environment. So we need further reduction of the data while not losing necessary information for tracking. The algorithm explained above is used in initial positioning before targets start moving. Once targets start moving, tracker estimate positions of targets through positions at previous state. The moving pattern of targets can be assumed as uniform acceleration motion as follow.

$$P_{t+1} = \frac{1}{2} \cdot \frac{d^2 P_t}{dt^2} + \frac{dP_t}{dt} + P_t \quad (3)$$

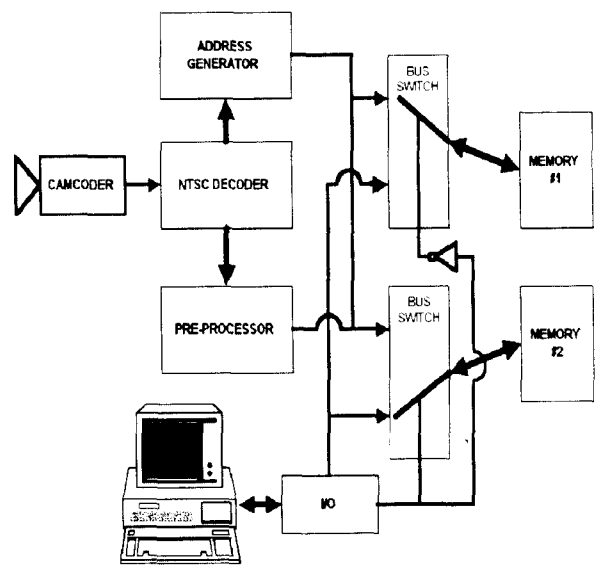
where P_{t+1} is estimated next position of the target, and P_t is present position. From this position, the recursive centroiding algorithm will search the object, and get the position of the object.

4. Implementation

4.1 Frame Grabber

The image sensor used in this paper is the camcoder, which generates analog video signal in NTSC format. To process this analog signal and get positions of the objects, the frame grabber is needed. The frame grabber consists of NTSC decoder part, pre-processing part, address generator part, frame memory part, and PC interface part.(shown in Fig. 2)[2]

This analog video signal is directly converted into color code of the corresponding objects by pre-processor part of the frame grabber. And color codes are stored in frame memory of the frame grabber. Two frame memories are prepared for even and odd frame. When one frame is processed, the other is filled with the color code of new image data.



<Fig.2> Frame grabber

4.2 Color Look-up table

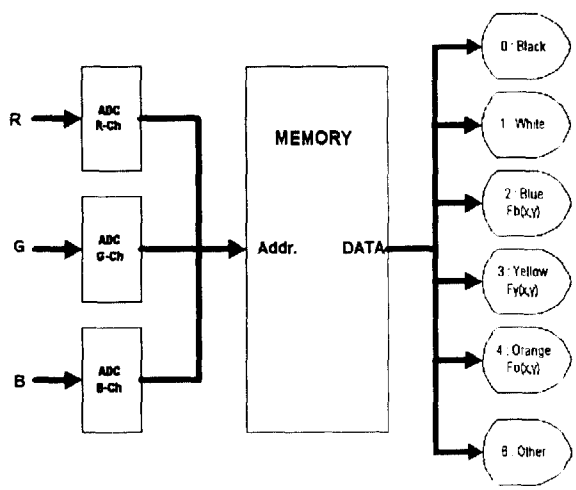
The sequence shown in Fig. 1 may be expressed in terms of a transfer function as follow.

$$Y(x, y) = G \cdot X(x, y) \quad (4)$$

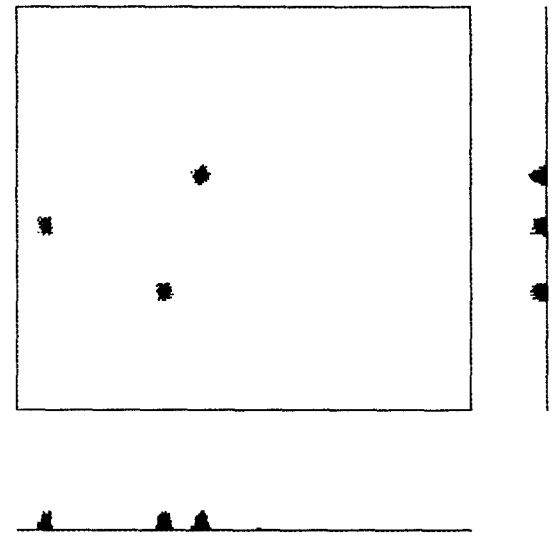
where is the color of the pixel on x, y location that is the class of the object, and $X(x, y)$ is R,G,B value.

The transfer function in this case represents the look-up table. The look-up table is made in advance with varying R, G, B values from 0 to 255 and is implemented in hardware for faster processing. The PC can recognize the class of the object directly from this function. We could realize this procedure by adding just one memory chip. Another advantage of this method is that frame memories can be reduced by $\frac{1}{3}$. The conventional frame grabber should have 3 memory chips to store all of digitized values of R, G, B in each. But, in this method, only one is needed. This memory stores the color codes only. It means that amount of data to be processed after pre-processing is reduced by $\frac{1}{3}$. Fig. 3 shows the block diagram of the procedures. More advanced algorithm for classifying the color of objects can be applied without changing the hardware.

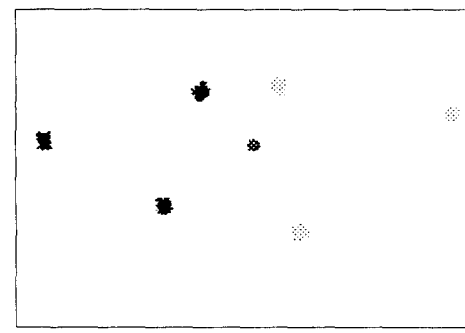
Fig. 4 shows that is the binary image of team B(Yellow) generated by pre-processing, and projection profiles $Hy_x(x)$, $Hy_y(y)$. And, Fig. 5 show the all of color classified objects.



<Fig.3> Pre-processor part.



<Fig.4> $Fy(x, y)$ and $Hy_x(x)$, $Hy_y(y)$



<Fig.5> Color classified Image

4.3 Software

Software is coded in C-language. For the initial Positioning, projection profile is used as mentioned before. It can be simply realized as follows

<List.1> Program for Projection profiles of Yellow image

```
for (x=0;x<256;x++)    Hx(x) = 0;
for (y=0;y<229;y++)    Hy(y) = 0;
for (y=0;y<229;y++) {
    for (x=0;x<256;x++) {
        Hx(x) += Fy(x,y);
        Hy(y) += Fy(x,y);
    }
}
```

And, initial positioning is coded as in List.2, where check_obj(x,y) is the recursive function that searches the object from position x,y by checking connected points, and calculate the position of the object, if the object is there. It returns the number of connected pixels. So, we can find out if there is an object on

position x, y by comparing returned value with some threshold, Kn.

After this, the function check_obj(x,y) is applied to the point that is estimated by eq. (3) for actual tracking.

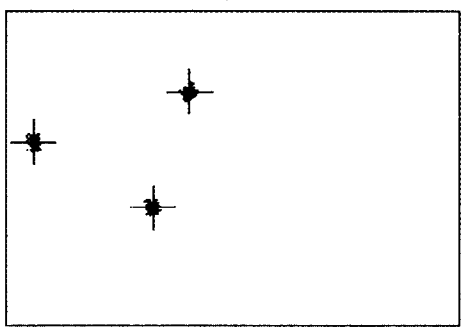
<List.2> Program for initial positioning

```
for (y=0;y<229;y++) {
    if ( Hy(y) > K ) {
        for (x=0;x<256;x++) {
          if ((Hx(x) > K) && (Fy(x,y) == 1)) {
                Qn = Qx = Qy = 0;
                if ( check_obj(x,y) > Kn ) {
                    PX(n) = Qx/Qn;
                    PY(n) = Qy/Qn;
                    n++;
                }
            }
        }
    }
}
```

5. Experimental results

All the components - hardware and software -are based on PC. We executed these on a AMD 5k86 133MHz PC. It takes about 7mS to track the 7 objects including 3 enemies, 3 agents, and a ball.

Fig. 6 shows the tracking screen of team B(Yellow).



<Fig.6> Tracking screen of Team B(Yellow)

6. Conclution

We have developed a low cost real-time multi-target tracker for MIROSOT based on simple and efficient procedures using color look-up table, projection profiles, and recursive algorithm for centroiding. It has been successfully implemented in the tracking robot for MIROSOT. It does not require extra processor like DSP, and yet has capability to track multiple targets correctly in real-time. We can further reduce the processing time by using PCI I/O card, because the most large portion of processing time occurs in data transfer. More advanced estimation scheme for object motion and color classification may be required to reduce the possibility of mistracking in all circumstances.

[REFERENCES]

[1] B.S Lee(Ed.), *All of th Image processing circuit technique*, Sewoon press, Korea, 1992, pp. 9-18

[2] C.S In, "Optical Image Processing and Precision Position Estimation for Moving Object Tracking", M.S thesis, Ajou Univ., February 1995

[3] C.S In, J.S Lee, S.K Hong, Y.G Koh, "Precision Position Estimation for Tracking the Moving Object", *KIEE Conference proceeding*, Seoul, Korea, Nov. 1994, pp. 335-337

[4] J.K Hong, "A Method of Feature Extraction from Facial Images Using Projection Profiles", *4th Workshop on Image processing and recognition*. Chungwon, Korea, Jan. 1992, pp. 177-183