Moving Object Tracking in Video

Yiwei Wang and John F. Doherty
Department of Electrical Engineering
The Pennsylvania State University
University Park, PA16802, USA
{yxw131,jfdoherty}@psu.edu

Robert E. Van Dyck
National Institute of Standards and Technology
Gaithersburg, MD20899, USA
vandyck@antd.nist.gov

Abstract

The advance of technology makes video acquisition devices better and less costly, thereby increasing the number of applications that can effectively utilize digital video. Compared to still images, video sequences provide more information about how objects and scenarios change over time. However, video needs more space for storage and wider bandwidth for transmission. Hence is raised the topic of video compression. The MPEG 4 compression standard suggests the usage of object planes. If the object planes are segmented correctly and the motion parameters are derived for each object plane accordingly, a better compression ratio can be expected. Therefore, to take full advantage of the MPEG 4 standard, algorithms for tracking objects are needed. It is also obvious that there is great interest in moving object tracking algorithms in the fields of reconnaissance, robot technology, etc. So, we propose an algorithm to track moving objects in video sequences.

The algorithm first separates the moving objects from the background in each frame. Then, four sets of variables are computed based on the positions, the sizes, the grayscale distributions and the presence of textures of the objects. A rule-based method is developed to track the objects between frames, based on the values of the variables. Preliminary experimental results show that the algorithm performs well. The tests also show that the algorithm obtains success in indicating new tracks (object starts moving), ceased tracks (object stops moving) and possible collisions (objects move together).

1. Introduction

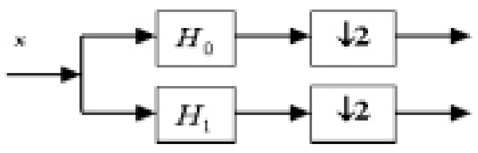
Because of the advance in technology, there are more affordable digital video acquisition devices in the market. This means more applications for digital video. Having witnessed the success of web camera applications and the appearance of high definition digital video cameras, we believe that digital video will soon become a part of everyday life. Unlike still images, video sequences provide more information about how objects and scenarios change over time, but at the cost of increased space for storage and wider bandwidth for transmission. Therefore, the topic of video compression have drawn more and more attentions during recent years. The MPEG 4 standard suggests the usage of object planes in video compression. By segmenting the object planes and determine the motion parameter for each one correctly, good compression results can be achieved. Hence, algorithms of tracking objects are needed. Also, in the context of reconnaissance, robot technology, etc., there are great interests in moving object tracking algorithms. Therefore in this paper, we present an algorithm of moving object tracking.

Many existing algorithms [1]-[4] segment each video frame to determine the objects; this action can be computationally expensive, and it is not necessary if the goal is to determine the moving objects. Alternatively, we proposed an algorithm [5] that derives the objects based on the motion between frames. While initial results were promising, the tracking algorithm in [5] is limited and not able to handle some complex situations such as new tracks (object starts moving), ceased tracks (object stops moving) and possible collisions (objects move together). Therefore, we present a rule-based method to deal with these situations. The rest of the paper is organized as follows: Section 2 will review some necessary background on the wavelet transform and camera motion models. We will review the algorithm that we developed in [5] in Section 3 and introduce the tracking algorithm in Section 4. The experimental results are shown in Section 5, and the conclusions are given in Section 6.

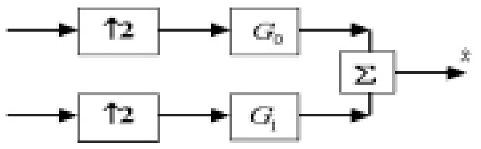
2. Background

2.1. Wavelet Transform and Filter Banks

Due to the extensive study done on it, the wavelet transform is now a very powerful tool in signal analysis and relevant fields. Unlike some traditional transforms, the Fourier transform for example, the wavelet transform can achieve both spatial and frequency localization. In the area of discrete signal analysis, the wavelet transform is closely related to filter banks. A two-channel filter bank is shown in Fig. 1 [6].



(a) analysis part



(b) synthesis part

Figure 1. 2-channel wavelet decomposition and reconstruction structure

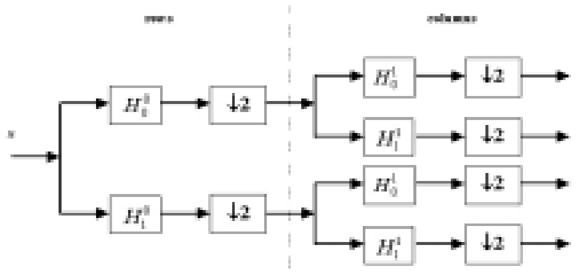
For digital images, the signal is two dimensional. Therefore, in image analysis and compression, two 1-D wavelet analyses are usually applied to the horizontal and vertical directions of the images separately; the structures in Fig. 1 are cascaded, as shown in Fig. 2. Using the decomposition structure in Fig. 2 repeatedly, a wavelet pyramid can be created for multi-resolution analysis. A wavelet pyramid of the image "Lena" is given in Fig. 3 as an example [7].

If we examine Fig. 3 carefully, we can see that in the high frequency bands, the coefficients have large amplitudes at the location of the edges. This property is sometimes used in edge detection.

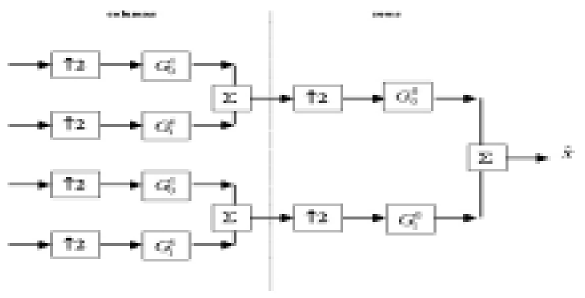
2.2. Camera Motion Estimation

In video sequences, the differences between consecutive frames are usually created by a combination of camera motion and the movement of objects. Since in this work our interest is tracking the moving objects, we need to remove the differences caused by camera motion as much as possible.

Camera motion has been studied for a long time, and there are many papers in the literature. The model that we use is the projective/bilinear model [8][9]. It can characterize almost all possible camera motions, such as translation, rotation, zooming, panning, tilting, etc. The projective model is described by eight parameters ($m_i, i = 1, 2, \ldots, 8$)



(a) analysis part



(b) synthesis part

Figure 2. image decomposition and reconstruction structure

through equation

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} m_1 & m_2 & m_3 \\ m_4 & m_5 & m_6 \\ m_7 & m_8 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

where (x, y) is the original coordinates. The transformed coordinates (x', y') can be obtained using equations

$$x' = \frac{m_1 x + m_2 y + m_3}{m_7 x + m_8 y + 1} = \frac{u}{w};$$

$$y' = \frac{m_4 x + m_5 y + m_6}{m_7 x + m_8 y + 1} = \frac{v}{w}.$$

The bilinear model, described by the equations below, is an approximation of the projective model,

$$x' = q_1 xy + q_2 x + q_3 y + q_4,$$

$$y' = q_5 xy + q_6 x + q_7 y + q_8;$$

which also uses eight parameters ($q_i, i = 1, 2, \ldots, 8$) to describe the camera motion.

The estimation is based on the 2-D optical flow assumption given by equation

$$u_f E_x + v_f E_y + E_t \approx 0,$$

where u_f and v_f are the velocities along the x and y directions, while E_x, E_y and E_t are the partial derivatives of the

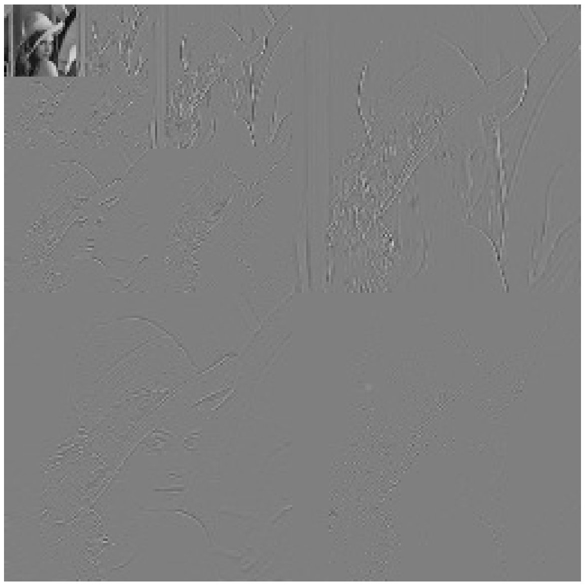


Figure 3. the wavelet pyramid for “Lenna”

gray-scale values with respect to x, y, and t. If we define $u_m = x' - x$ and $v_m = y' - y$, by using the bilinear model and minimizing

$$\epsilon = \sum_{(x,y)} (u_m E_x + v_m E_y + E_t)^2,$$

we can get the parameters for the bilinear model. Using the original and the transformed coordinates of four points, we can approximate the parameters of the projective model.

To lower the computational burden, the above procedure is usually performed on a multi-scale pyramid. Since we later plan to incorporate the motion registration algorithm into a wavelet-based compression algorithm, we use a wavelet pyramid instead of the commonly used Laplacian-Gaussian pyramid.

3. The object isolation algorithm

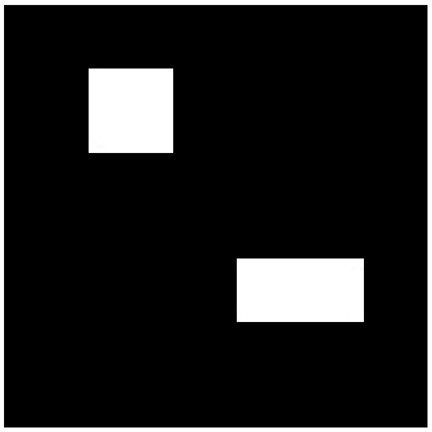
Although the focus of this paper is the tracking algorithm presented in the next section, we first give a short review of the object isolation algorithm introduced in [5].

The algorithm starts with the current frame and the reference frame. The wavelet transform is applied to each frame to create a wavelet pyramid and the filter banks and the number of levels of decomposition are chosen depending on the size of the frame and the content of the video. The camera motion model described in the previous section is applied to the two lowest resolution DC images. Assuming the moving objects only occupy a small portion of the whole image, the motion estimation result corresponds mostly to the background differences. In other words, the above result is very close to the real camera motion at that level. Two edge images are then created by adding the amplitudes of the AC bands of the lowest resolution. The motion parameters computed above are used to align the edge images and the differences are calculated. Because the differences at the location of the moving objects are larger, the lowest resolution images can be roughly segmented into possible object and background regions by thresholding the differences. Furthermore, we apply the motion estimation algorithm only to the possible background areas to achieve a more accurate camera motion estimation. After we finish processing the level of the lowest resolution, we repeat the above procedure on higher resolution levels. The DC bands of a higher resolution level can be created from the lower resolution level using synthesis filters. The corresponding motion parameters from the lower resolution level are used as initial values for a higher resolution level, and only the parts of images that have been classified as background areas in the lower resolution levels are considered. Basically, the algorithm will select more areas as possible object regions when the resolution gets higher, until it reaches the highest resolution level. A flowchart of the procedure is given in [5].

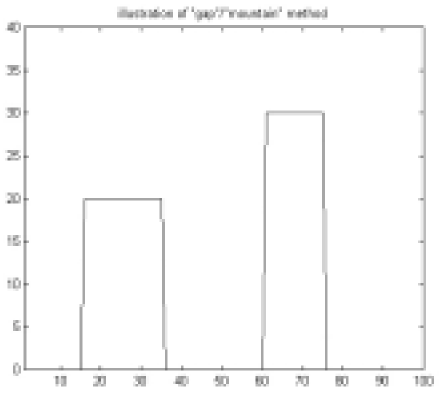
The above procedure will produce possible object blocks of different sizes. Some of the blocks are obtained due to moving objects, while others are just the result of noise. To group the blocks of the same moving object together and eliminate the noise effect, we developed a projection-based algorithm denoted the “gap”/“mountain” method. We start with the edge image at the highest resolution level and all pixels classified to the background are set to zero. A column vector is then created by adding the pixel values for each row. It is obvious that a zero element in that column vector means that there are no object pixels in the corresponding row. A “gap” is defined as consecutive zeros whose number is larger than a preset threshold and the group of elements between two “gaps” is defined as a “mountain”. The element with the largest value in each “mountain” is called a “peak”. If the width of a “mountain” is larger than a preset threshold and its “peak” is high enough, we conclude that there is at least one object in the “mountain”. For an original edge image of $p \times q$ pixels, the above procedure places all the possible objects in m smaller matrices; each matrix has q columns, while the number of rows is equal to the size of the “mountain”. Furthermore, if we apply the same idea to these matrices and add the values for each row instead of each column, the total size of the matrices that contain the objects will become even smaller. The same algorithm is applied iteratively until the total size of the matrices does not change any more. Each final matrix will be considered as an object location. In summary, by adding along columns and rows repeatedly, we are able to obtain several rectangles that each contains an isolated object.

To illustrate the above “gap”/“mountain” method more clearly, Fig. 4 is given as an example. A 100 by 100 binary image containing two objects is shown in Fig. 4(a). One object is 20 by 20 and the other one is 15 by 30. The grayscale value is zero for the background and is one for the objects. A column vector of 100 elements is created by adding the value of pixels in each row. We plot it in Fig. 4(b). By setting the thresholds for “gap” width and “mountain” width to be 10, we can see that there are three “gaps” and two “mountains”. Since the objects must be in “mountains”, instead of knowing the possible objects are in the initial 100 by 100 matrix, now we know that they are in two smaller matrices. The size for the first one is 20 by 100 and the size for the second one is 15 by 100. Then, we apply the same idea to these two smaller matrices one by one. Instead of adding the values for each row, we now add along columns and detect the “gaps” and “mountains” in the corresponding row vectors. The algorithm is applied iteratively until the sizes of matrices do not change anymore. In this example, we need three steps. The left, right, top, and bottom boundary locations for the first object are 21, 40, 16 and 35, respectively, and those for the second object are 56, 85, 61 and 75.

To achieve accurate results, both the preceding frame and the following frame are used as the reference frame. The mutual object areas detected in the current frame are then considered as the final results.



(a) the example image



(b) the plot of the column vector

Figure 4. an example of the “gap” / “mountain” method

4. The tracking algorithm

Once the object areas are determined in each frame, the tracking algorithm is needed to trace the objects from frame to frame. In this section, we will present a rule-based algorithm using the information of the object trajectories, sizes, grayscale distribution, and textures. Variables based on the information are first computed, and then the tracking results are decided based on variable values.

The variables for object trajectories are the object position coordinates. To decide the object position, we define the centroid of an object (c_x, c_y) as

$$c_x = (\sum_{(i,j)\in\mathbf{O}} p_{i,j} \cdot i)/(\sum_{(i,j)\in\mathbf{O}} p_{i,j}),$$

$$c_y = (\sum_{(i,j)\in\mathbf{O}} p_{i,j} \cdot j)/(\sum_{(i,j)\in\mathbf{O}} p_{i,j});$$

where $\mathbf{O}$ is the set of coordinates of an object area and $p_{i,j}$ is the value of the edge image at position (i, j). Each object is then corresponding to a point. Furthermore, we assume that the object trajectories are close to straight lines in a few adjacent frames and the object acceleration rate is a constant in these frames. The object location coordinates from the previous three frames are used to compute v and a in the equation

$$S = vt + \frac{1}{2}at^2,$$

where v is the initial speed and a is the acceleration rate. Then, the values of v and a are used to predict the locations in the current frame. By comparing the predicted positions and real positions, it is possible to achieve trajectory-based tracking.

Assuming the frame rate is adequate, the sizes of the objects should not change dramatically between adjacent frames. The dispersion variable is used for tracking the objects based on size. The dispersion of an object, $disp$, is defined as

$$disp = (\sum_{(i,j)\in\mathbf{O}} \sqrt{(i - c_x)^2 + (j - c_y)^2} \cdot p_{i,j})/(\sum_{(i,j)\in\mathbf{O}} p_{i,j});$$

where (c_x, c_y) is the object centroid, while $\mathbf{O}$ and $p_{i,j}$ have the same definition as in the equations that defined the centroid. When the dispersions are computed in each frame, we can track the objects by comparing them.

The grayscale distribution of an object usually does not change too much, given that the lighting condition stays relatively constant between consecutive frames. In other words, the span of grayscale values for the same object is similar from frame to frame. The variables that we use, based on grayscale distribution, are the mean of the whole grayscale range gr_m, the mean of the 10% pixels of largest grayscale value gr_h, and the mean of the 10% pixels of smallest grayscale value gr_l. These three variables will indicate the grayscale span of the object. Hence, grayscale-based tracking can be achieved by matching the variables of the objects in different frames.

The last variable is based on object texture. The surfaces of the objects are usually not homogeneous. If we consider the grayscale variations on the object, they are usually different from object to object. These differences are reflected in the wavelet transform coefficients. A variable, denoted tx, that can roughly indicate the texture property of an object is the mean of the 10% of the pixels with the largest values in the constructed "edge" image. Generally speaking, large values indicate more textures on the object.

Because there are extremes that violate the assumptions that we have made, it is obvious that none of the above four sets of variables will be accurately tracking the objects all the time. Therefore, we cannot just depend on one set of the variables and need to integrate four sets together.

In the current frame, each object is associated with four sets of variables. Each existing track in the previous frame will also produce four sets of variables. Therefore, the tracking problem becomes finding the best matches between the objects and the existing tracks. A natural way to do this is to compute the differences between the variable values and then threshold the differences. If there are m objects in the current frame and n existing tracks, there will be a total of $m \times n$ sets of differences to evaluate. Considering the situations of new tracks, ceased tracks and track collisions, we have to make sure that the variables of an object are similar to those of an existing track when we extend the track to that object. Since we have four sets of variables, the first rule that we use is that at least three sets of differences must be less than the threshold. Otherwise, we will not consider that object as a possible extension for the track.

After we evaluate $m \times n$ sets of differences, we will have a matrix of size $m \times n$. The elements of the matrix indicate how many sets of variable differences between the specific object and the certain track are less than the threshold. As indicated in the previous paragragh, an element greater than 2 corresponds to a possible track extension. However, it is obvious that there usually will not be one and only one eligible element in each row and each column of the matrix. Therefore, we develop the following strategy.

First, we start from the simple cases. If an element is the only one eligible in its row and its column, there is no ambiguity. We simply extend the corresponding track to the corresponding object and simplify the matrix by eliminating the row and the column that the element is in. Second, suppose $e_{i,j}$ is the only eligible element in row i, if all other eligible elements in column j are not the only element in their corresponding rows, we will extend track j to object i and eliminate the ith row and jth column to simplify the matrix. A similar procedure is performed when $e_{i,j}$ is the only eligible element in column j and all other eligible elements in row i are not the only one in their corresponding columns. After using the above two rules repeatedly, the eligible elements left in the matrix correspond to complicated situations that cannot be solved using simple thresholding. We then adopt a weighted sum as the cost function, which is described by the equation

$$\begin{aligned} dif \; &= w_{tr}(|c_x^f - c_x^t| + |c_y^f - c_y^t|) \\ &+ w_{disp}(|disp^f - disp^t|) \\ &+ w_{gr}(|gr_l^f - gr_l^t| + |gr_m^f - gr_m^t| + |gr_h^f - gr_h^t|) \\ &+ w_{tx}(|tx^f - tx^t|), \end{aligned}$$

where dif is the cost function, $w_{tr}, w_{disp}, w_{gr}, w_{tx}$ are the weights, and superscripts f and t indicate whether the variable is computed from the current frame or the track. For each eligible element in the matrix, a weighted sum is computed. Then for each column, the row producing the smallest dif value is selected, which is equivalent to finding the best matching object of an existing track.

Sometimes, after the process described above, there will be tracks and objects left with no matches. These situations often correspond to new tracks, ceased tracks, and track collision. We first consider the possibility of track collision by examining the variable values. If the mean values of the predicted positions of several existing tracks are "close" to one of the objects'(compatible with the dispersion values), we then evaluate the dispersion values. If the object dispersion value is larger than the largest dispersion value of the tracks and smaller than the summation of the dispersion values of all those tracks, we will mark the object and the tracks as possible track collision. The rest of unmatched objects and tracks are then labeled as new tracks and ceased tracks.

An example is given here to make the above procedure clearer. If after the thresholding, the matrix is

$$\begin{bmatrix} 4 & 2 & 0 & 1 & 1 \\ 1 & 3 & 2 & 4 & 1 \\ 2 & 0 & 4 & 1 & 3 \\ 0 & 1 & 1 & 3 & 2 \\ 1 & 2 & 3 & 0 & 3 \\ 0 & 1 & 0 & 0 & 2 \end{bmatrix},$$

where each element indicates how many sets of variables are smaller than the corresponding thresholds. From the matrix, we can see that there are five existing tracks and six objects in the current frame. $e_{1,1}$, which is 4, is the only eligible element in the first row and the first column. Therefore, we extend the first track to the first object. The matrix is then simplified to

$$\begin{bmatrix} - & - & - & - & - \\ - & 3 & 2 & 4 & 1 \\ - & 0 & 4 & 1 & 3 \\ - & 1 & 1 & 3 & 2 \\ - & 2 & 3 & 0 & 3 \\ - & 1 & 0 & 0 & 2 \end{bmatrix}.$$

In the new matrix, $e_{4,4}$, which is 3, is the only eligible element in row 4. Although $e_{4,4}$ is not the only element in column 4, the other element is in row 2, a row that has more than one eligible elements. Hence, using rule number two, we extend the fourth track to the fourth object and simplify the matrix as

$$\begin{bmatrix} - & - & - & - & - \\ - & 3 & 2 & - & 1 \\ - & 0 & 4 & - & 3 \\ - & - & - & - & - \\ - & 2 & 3 & - & 3 \\ - & 1 & 0 & - & 2 \end{bmatrix}.$$

After using the first rule again, the matrix becomes

$$\begin{bmatrix} - & - & - & - & - \\ - & - & - & - & - \\ - & - & 4 & - & 3 \\ - & - & - & - & - \\ - & - & 3 & - & 3 \\ - & - & 0 & - & 2 \end{bmatrix}.$$

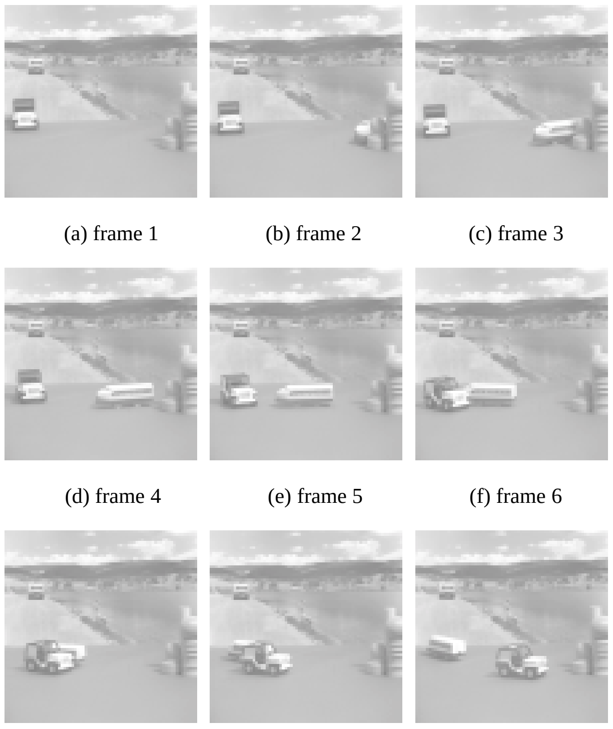
Now, assuming $dif_{i,j}$ is the weighted sum for $e_{i,j}$, if $dif_{3,3} > dif_{5,3}$ and $dif_{3,5} < dif_{5,5}$, then the third track is extended to the fifth object and the fifth track is extended to the third object. Now the matrix is

$$\begin{bmatrix} - & - & - & - & - \\ - & - & - & - & - \\ - & - & - & - & - \\ - & - & - & - & - \\ - & - & - & - & - \\ - & - & - & - & - \end{bmatrix}.$$

Since all the tracks have been extended, the sixth object is then considered the beginning of a new track. The final results are $(1,1),(2,2),(3,5),(4,4),(5,3),(6,6^*)$, where (i,j) means object i is corresponding to track j and $*$ indicates new track.

5. Experimental results

The above algorithm is tested on the "toy vehicle" sequence. The frame size is 512 by 512. Nine consecutive frames are shown in Fig. 5.



(a) frame 1 (b) frame 2 (c) frame 3

(d) frame 4 (e) frame 5 (f) frame 6

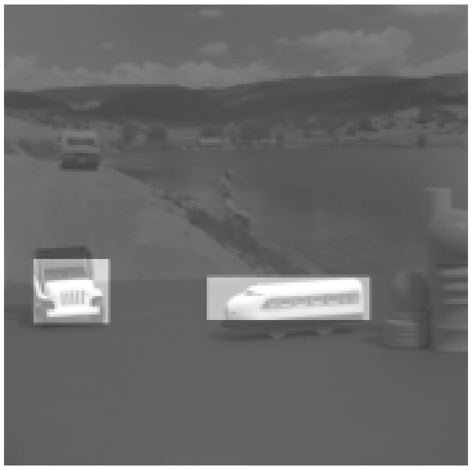
(g) frame 7 (h) frame 8 (i) frame 9

Figure 5. frames in the toy vehicle sequence

Fig. 6. illustrates the object isolation algorithm described in section 3. The algorithm is applied to frame 4 and frame 5. We choose Daubechies' 5/7 biorthogonal filters and use four levels of wavelet decomposition. We highlight the objects in Fig. 6. by adding special effects to the images. It is obvious that two moving objects are detected in each frame.

The algorithm is performed on every frame to isolate the objects. Then, the tracking algorithm presented in section 4 is used. The tracking results are given in Table 1. The same object in different frames is labeled with the same object ID number to indicate a track. Centroid values are also given to provide the location information of the object.

Evaluating Table 1, we can see that the algorithm successfully tracks two toy vehicles as they approach each other and then separate. Two comments are worth mentioning about Table 1. First, in the starting three frames, the characteristics of the toy vehicle emerging from the right change considerably because of occlusion. So, the algorithm does not pick up the track. However, as all of the object begins to appear in the image, the tracking results are correct in frame 4 and frame 5. Second, in frame 6, neither of the existing tracks matches the new object. However, as the mean of the predicted positions from the tracks is (140.7, 338.2), which is close to the centroid of the ob-


(a) frame 4

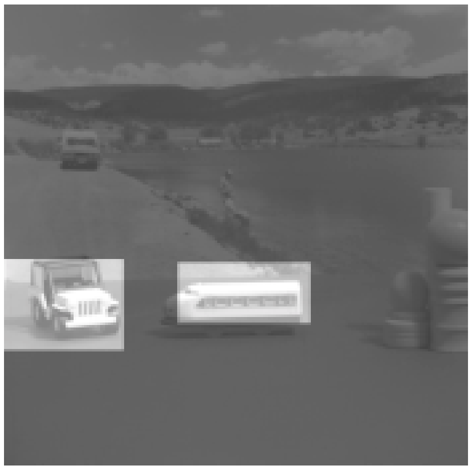

(b) frame 5

Figure 6. detected object areas in frame 4 and frame 5

ject (135.5, 332.1), we evaluate the dispersions of the object and the tracks. The dispersion for the new object is 66.0 while those for the two tracks are 35.2 and 42.5. Because $42.5 < 66.0 < 35.2 + 42.5 = 77.7$, we can claim that there is possible track collision in frame 6.

6. Conclusion and comments

In this paper, we first reviewed an algorithm to isolate the moving objects in video sequences and then presented a rule-based tracking algorithm. The preliminary experimental results demonstrate the effectiveness of the algorithm even in some complicated situations, such as new track, ceased track, track collision, etc.

The goal of the algorithm is to identify and track the moving object quickly. However, the projective model for camera motion is somewhat computational expensive. It cannot deal with occlusion very well either. Therefore, we will investigate ways to adopt other faster and more robust camera motion estimation methods, such as feature-based algorithms.

Table 1. Tracking results

Frame No.	Obj. ID	Centroid	Obj. ID	Centroid
1	1	(53.2, 295.4)	-	-
2	1	(51.8, 303.4)	-	-
3	1	(53.2, 310.4)	2	(377.1, 336.9)
4	1	(66.5, 317.6)	3	(325.4, 338.6)
5	1	(79.7, 326.5)	3	(257.0, 338.9)
6	4	(135.5, 332.1)	-	-
7	4	(135.1, 334.8)	-	-
8	4	(144.2, 336.6)	-	-
9	5	(96.9, 313.5)	6	(283.3, 353.5)

References

[1] D. Wang, Unsupervised Video segmentation Based On Watersheds And Temporal Tracking, *IEEE Trans. Circuits Syst. Video Technol.*, 8(5):539-546, September 1998.

[2] G. L. Foresti, Object Recognition And Tracking For Remote Video Surveillance, *IEEE Trans. Circuits Syst. Video Technol.*, 9(7):1045-1062, October 1999.

[3] P. Salembier, F. Marqués, M. Pardàs, J. R. Morros, I. Corset, S. Jeannin, L. Bouchard, F. Meyer, B. Marcotegui, Segmentation-Based Video Coding System Allowing The Manipulation Of Objects, *IEEE Trans. Circuits Syst. Video Technol.*, 7(1):60-74, February 1997.

[4] A. J. Lipton, H. Fujiyoshi, R. S. Patil, Moving Target Classification And Tracking From Real-time Video, *Applications of Computer Vision, 1998. WACV '98. Proceedings., Fourth IEEE Workshop on*, pp. 8-14, 1998.

[5] Y. Wang, R.E. Van Dyck, J. F. Doherty, Tracking Moving Objects in Video Sequences, *Proc. Conference on Information Sciences and Systems*, Princeton, NJ, March 2000.

[6] M. Vetterli, J. Kovacevic, *Wavelets And Subband Coding*, Prentice-Hall, INC., Upper Saddle River, NJ, 1995.

[7] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies, Image Coding Using Wavelet Transform *IEEE Trans. on Image Processing*, 1(2):205-220, April 1992.

[8] S. Mann, R. W. Picard, Video Orbits of The Projective Group: A Simple approach To Featureless Estimation Of Parameters, *IEEE Trans. on Image Processing*, 6(9):1281-95, September 1997.

[9] S. Lertrattanapanich, N. K. Bose, Latest Results On High-resolution Reconstruction From Video Sequence, *Technical Report Of IEICE. DSP99-140*, pp. 59-65, December 1999.