Research Journal of Applied Sciences 4 (4): 129-133, 2009
ISSN: 1815-932X

Translation Based Estimation Technique to Handle Occlusion While Using Mean-Shift in Tracking

[1]A.H.M. Kamal and [2]Montse Parada
[1]Department of CSE, Jatiya Kobi Kazi Nazrul Islam University, Bangladesh
[2]Department of Signal and Theory, Cataluniya University of Technical, Barcelona, Spain

Abstract: Detection of occlusion is a common problem in video surveillance field. When, moving objects overlap one another, an occlusion is created. It may happen very often in real life video. Many researchers have devoted their time to overcome that problem and have proposed many techniques. We propose here a blob translation based estimation technique to detect occlusion. We use the velocity, area and direction to do that.

Key words: Meanshift, Gaussian, occlusion, orientation, kalman filter, estimation

INTRODUCTION

Digital video processing is becoming widely used in many aspects of the nowadays life. The availability of high-computation-power systems allows processing of huge amount of raw data to achieve content based functionalities, such as search and manipulation of objects, semantic description of scenes, detection of unusual events and recognition of objects, etc.

A lot of contributions related to video surveillance belong to occlusion handling. Occlusion and tracking trajectories are much related work and tracking objects and making trajectories completely depends on the solution of occlusion problem. In tacking objects in live video foreground extraction is a common technique, whereas mixture of Gaussians used by Stauffer and Grimson (1999), KaewTrakulpong and Bowden (2001) and Landabaso *et al.* (2004), Baye's decision rule used by Li *et al.* (2003) and using threshold by Rosin (1998) are some very common algorithms to separate foreground. In tracking objects, uses of some generalized features by Landabaso *et al.* (2004) mean-shift procedure by Beleznai *et al.* (2005) and Collines (2003) and kalman filter by Stauffer and Grimson (1999) are very well-known methods. However, occlusion is a major problem in making trajectory. But, researchers are contributing regularly to improve the tracking performance by handling occlusion. Beleznai *et al.* (2002) resolved occlusion events based on the smoothness constraint of kinematic and color features. A heuristic is used by Landabaso *et al.* (2004) to do the same research. Another interesting technique, data association algorithm: a probabilistic approach is used by Beleznai *et al.* (2005) to detect and handle occlusion.

In this study, we have not described the mean-shift algorithm and some other definitions related to mean-shift as our target, here, is to propose a technique for occlusion detection and to handle it. For the same reason, we have avoided here the explanation of the way of processing and simplifying foreground image. We have just collected those foreground images applying mixture of Gaussians algorithm of (2) and then applied some preprocessing to simplify foreground images.

BASIC ATTRIBUTES AND THEIR USES IN THE METHOD

In this algorithm, we calculated change of velocity, direction and estimated velocity and then translated blobs using those estimated values to track and handle occlusion.

Measuring velocity and sampling orientation: The velocity is measured from centre's displacement of a tracked blob in two consecutive frames. Consider that the centre of a blob in frame t is C (x_1, y_1) and at frame t +1 is C (x_2, y_2). Then the velocity of that blob is:

$$\text{Velocity} = \text{root}((x_1 - x_2)^2 + (y_1 - y_2)^2)$$

CamShift algorithm also estimates the orientation of tracked blob. But, mean-shift did not estimate the real size of blob. So, calculating orientation from these parameters that will be used in translation is more desired. But, objects can move from here to there in any direction. So, next we calculate the direction of a blob. To do that we quantize the directions into eight sample directions. We measure the change of x-ordinate as xchange = $x_1 - x_2$. Similar way, we find the ychange = $y_1 - y_2$.

Corresponding Author: A.H.M. Kamal, Department of CSE, Jatiya Kobi Kazi Nazrul Islam University, Bangladesh

Then, we use the following equations to measure the direction:

$$\begin{cases} slop = \dfrac{ychange}{xchange} \\ angle = arcton(abs(slop)) \\ \theta = angle^{\circ}180/pi \end{cases}$$

The absolute value of slop is used to consider all calculation in first quadrant. So, the value of θ will be limited to 0-90. But, the sign value of xchnage and ychange will help us to define the exact direction. If the sign value of both xchange and ychange are positive, it is easy to sample the direction in first quadrant. If θ<28°, direction will be sampled to 33 according to the Fig. 1. If θ>65 then direction will be 22 otherwise direction will be 23. This way based on the value of theta and the sign value of xchnage and ychange, direction is sampled. Following Fig. 2 will give an idea of doing sample.

If either xchange or ychange are negative the direction will change to other quadrant. Following sample code is used in our programming to measure the sample direction of i-th blob.

```
if (θ≤28) { if(xchange<0)  direction[i]=33; else
direction[i]=11;}
else if(θ>28 && θ<65){
if(xchange<0 && ychange<0)        direction[i]=23;
        else if(xchange<0 && ychange>0)
direction[i]=34;
        else if(xchange>0 && ychange<0)
direction[i]=12;
        else if (xchange>0 && ychange>0)
direction[i]=41;}
else{    if(ychange<0)    direction[i]=22;   else
direction[i]=44;}
```

This direction value is used to estimate, the possible translation of ith blob in the next frame. The area of a blob is calculated from its width and height. We always translate the centre of a blob based on direction and last five velocities. Translation of the centre of a current blob is performed based on the estimated velocity for the next and unseen frame. This translation will help us to find a best but, possible movement of a blob in coming frame. We also, may need to estimate the area of the blob around that translated centre. As like as velocity estimation, the area can also be estimated from an estimation of width and height of a blob in current frame. But, in the following description, we will see that we are not working with the area estimation. Rather, we will estimate the velocity and

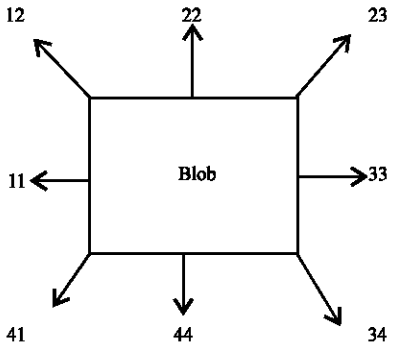

Fig. 1: Quantized direction

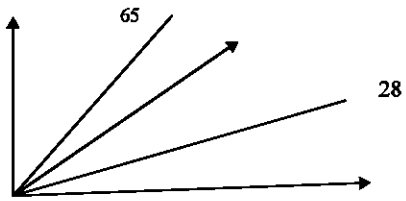

Fig. 2: θ value for quantizing direction

move the current searching window from frame to frame in the estimated direction remaining the area unchanged. The reason is that we save histogram of a blob that we had just at one frame before the occlusion and we reuse that histogram and searching window to track as soon as we guess a de-occlusion. For example, if ith and jth blob make an occlusion at frame t, we saved the histogram and the searching window of t-1 frame. Then, for the coming frame we just translated the searching window. We make a separate calculation of histogram for the occluded objects. Again if we realize a de-occlusion for that object at (t + n)th frame, we again apply the saved histogram and the translated window in CamShift algorithm to re-track the ith and jth onjects. So, we need to apply the same searching window that we used just before occlusion. That is why just before occlusion we saved histogram and searching window and next we just translated that window from frame to frame as long as we guessed a de-occlusion. When, we guessed a de-occlusion we again applied the saved window and histogram at the translated position to re-track the objects.

Velocity estimation: We know the direction of the blobs up to current frame. We also know the area, width, height and velocity of all the blobs which are found till current frame. Now, we have to estimate the velocity of the blob to find a possible movement in the next and coming frame. To translate the center point we first need to estimate the velocity (Fig. 3 and 4).

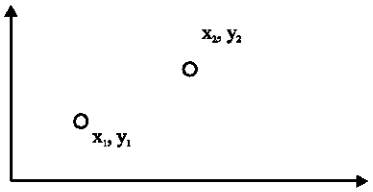

Fig. 3: Center (x_1, y_1) at frame t - 1 and (x_2, y_2) at frame t

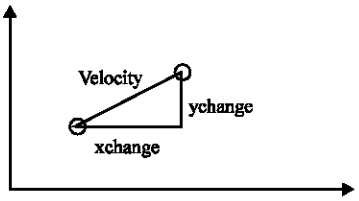

Fig. 4: Triangle with the two center points

The velocity is estimated using the following equation:

$$\text{Estimated velocity}(i) = \text{Current velocity}(j) + \frac{1}{5}\sum_{j=1}^{5} \text{Change of velocity}(i)(j)$$

Where, i is the blob ID and change of velocity (i) (j) contains last five change of velocities of each blob i. The change of velocity of a blob is measured from its velocity in frame t minus the velocity in frame t-1 (velocity (t)-velocity (t-1)). In the estimated velocity equation, we use only last five changes of velocities of a blob to estimate its velocity in the next frame because the velocity does not increase or decrease with the same rate when object is too far from and too near to the camera (Fig. 5).

Translation: Now, we have the estimated values of velocity and direction. We can now do the translation of the blob from these estimated values. We also know, the theta value and the sample direction of a blob. We have calculated these values in the study.

Using sampled direction: In this method, we use the theta value and the direction to find a translated point. We can do it using following equation:

$$x_{translation}(i) = \text{Estimated velocity}(i) * \cos(\theta)$$

$$y_{translation}(i) = \text{Estimated velocity}(i) * \sin(\theta)$$

$$\text{Estimated center}_x(i) = \text{Current center}(i) \otimes x_{translation}(i)$$

$$\text{Estimated center}_y(i) = \text{Current center}(i) \Theta y_{translation}(i)$$

In the above equation the operators Θ and $\otimes$ are used as sign value. It may be plus (+) or minus (-). The sign value depends on the quantized value of direction. For example if the direction (i) is 12 then $\otimes$ is negative and Θ is positive. This way the cenetr translation is performed.

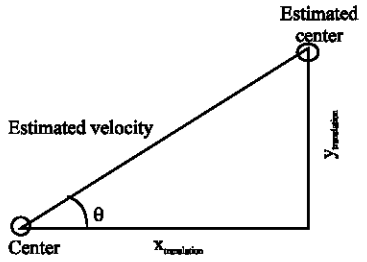

Fig. 5: Estimating center

TRACKING AND MANAGING OCCLUSION AND ANALYSIS RESULT IN FRAMES

The mean-shift algorithm will try to find a match of blobs in frame t-1 with the blobs in frame t. This matching algorithm leads the tracking of blobs from frame to frame. But, if the matched area maps to a larger blob in foreground frame t, an occlusion may exist. Besides if an occlusion occurs, the shape, area and the center point's movement rate (velocity) of a blob will be changed irregularly. Following is a discussion of tracking occlusion and handling it.

Above mentioned estimation procedure works silently without making any affect in tracking by regular CamShift procedure until an occlusion is guessed. All times, it calculates the velocity, change of velocity, estimated velocity, width, height and orientation in a sample direction in parallel with CamShift tracking. It does these as a sleeping agent. It completes the process in the following 4 steps.

- Before a guess of an occlusion it calculates velocity, change of velocity, orientation, area and estimated velocity
- When, an occlusion is detected, it saves the histogram and searching window that CamShift used at that point to track, starts the translation procedure to work silently, until a de-occlusion is detected, calculate the searching window and the histogram for that occluded object to track in next frames
- Until, a guess of de-occlusion CamShift track the occluded object along with other objects and translation procedure performs a translation of the saved window
- If a de-occlusion is guessed and detected, CamShift applies the saved histogram of step 2 and the translated window of step 3 to track the occluding objects again

When, an occlusion occurs, the algorithm will reveal that the foreground object is too bigger than the tracked area as the foreground object is in occluded shape and compose of many objects.

The left frame in Fig. 6 is not the exact foreground frame. It is generated from extracted foreground image and then simplified to help our tracking procedure. This is formed by drawing a rectangle of the same size of objects in foreground in an empty frame and then filling it. This is used in CamShift algorithm as foreground. Just at occlusion the CamShift algorithm estimated the location of car and it is shown in the left frame. The rectangle inside into the left-bottom big blob in the left frame is the estimated window by CamShift. But when, the algorithm tried to collect exact area, covered by that blob, from foreground, it reveals that it is a big blob. The area of the blob increased abnormally. So, it detected the situation as an occlusion and saved the searching window at previous frame. It found and save the occluding objects and their IDs. Then it assigned a new ID for that occluded blob.

This way, it will discover that the change of area is abnormal and a sudden change. If the area changes abnormally (increase >15%) it marks it as an occlusion and does the following operations at that step.

- Save the ID of the blob to which an occlusion is guessed
- Save the window and its position (the searching window that was applied in CamShift at that point but estimated at previous frame by CamShift)
- Save the histogram that was applied at that point to generate probability image
- Increase the counter value, e.g., probability occlusion, for that foreground object by one to measure the probability of being occlusion

Continue the same process from steps a-to-d for other blobs, if an occlusion is guessed. Next for final decision the following steps should be checked

- Now check whether the value of probability occlusion is >1. If it is more than one, an occlusion is assured and save the objects ID (e.g., p_1, p_2, p_3, … …), which participated in that occlusion
- Set that foreground object's area as a searching window for CamShift algorithm to track this occlusion portion at next frames and mark these as occlusion. Calculate the histogram for that filtered occluded object and use it to generate probability image in next frames. Provide an ID: Occluded (i) to that object. Start the CamShift procedure from next frame for that occluded object to be tracked
- Start the translation procedure to translate the saved windows

Fig. 6: Occlusion detection

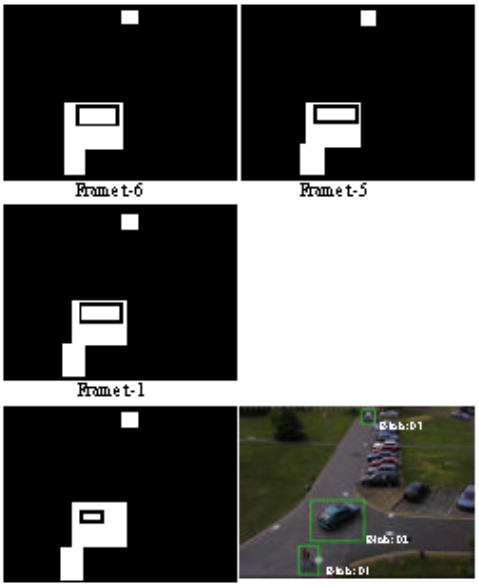

Fig. 7: De-Occlusion detection

This way, we handled the occlusion detection phase. In a reverse way, when an object with ID: occluded (i) reduces its size abnormally it is guessed a de-occlusion. But, it was saved in early step, which blobs were participated to an occlusion. So, when an object, marked as occlusion, reduces its size abnormally, the saved histogram of the corresponding objects (p_1, p_2, p_3, … …) is applied with their translated window in CamShift algorithm to track these objects after the separation from occlusion and the corresponding ID is reassigned.

In frame t-5 one can see the estimated window maps to a foreground object whose has no sudden change in area with respect to same object in frame t-6 (Fig. 7). But when at frame t the estimated window locates only the blob of car, the estimated window for car is printed at frame t, the algorithm finds that it is a much smaller area than last tracked blob in frame t-1. So, it guesses an occlusion. Then, it applies the CamShift with the translated windows for all occluding blobs of that occluded object. This way, it re-tracks the occluding blobs and reassigns IDs.

CONCLUSION

The technique research well in the experiment with the sample video PETS 2001. But, the algorithm can be failed in one situation if the blob changes the direction in reverse way and if just at that point an occlusion occurs. We always measure the estimated values based on previous result. So if a blob moves in direction = 23 (Fig. 1: Quantized direction) at frame t-1 and change the direction = (12 or 11 or 41) at frame t, the estimation and translation for blob in frame t-1 will guide us to a wrong direction. It is only an exception case when the algorithm can fail to detect occlusion.

REFERENCES

Beleznai, C., T. Schlogl, B. Wachmann, H. Bischof and W. Kropatsch 2002. Tracking multiple objects in complex scenes. Aus. Comput. Soc., 160: 175-182.

Beleznai, C., B. Fruhstuck, H. Bischof and W. Kropatsch, 2005. Model-based occlusion handling for tracking in crowded scenes. Aus. Comput. Soc., 192: 227-234.

Collines, R.T., 2003. Mean-shift blob tracking through scale space. IEEE Conf. Computer Vision and Pattern Recognition, 2 (3): 471-474.

KaewTrakulpong, P. and R. Bowden, 2001. An improved adaptive background mixture model for real-time tracking with shadow detection. In Proc. 2nd European Workshop on Advanced Video Based Surveillance Systems, Vol. 5308, AVBS01. Kingston, UK.

Landabaso, J.L., L.Q. Xu and M. Pardàs, 2004. Robust tracking and object classification towards automated video surveillance; international conference on image analysis and recognition. ICIAR, No. 2, pp: 463-470. DOI: 10.1007/b100438.

Li, L., W. Huang, I.Y.H. Gu and Q. Tian, 2003. Foreground object detection from videos containing complex background. Proceedings of the 11th ACM International Conference on Multimedia, Berkeley, CA, USA, pp: 2-10. ISBN: 1-58113-722-2. http://doi.acm.org/10.1145/957013.957017.

Rosin, P.L., 1998. Thresholding for change detection; Proceedings of the 6th International Conference on Computer Vision (ICCV). Publisher IEEE Comput. Soc. Washington, DC, USA., pp: 274. ISBN: 81-7319-221-9.

Stauffer, C. and W.E.L. Grimson, 1999. Adaptive background mixture models for real-time tracking. Proceedings IEEE Conf. Comput. Vision and Pattern Recognition, June 23-25, pp: 246-252. http://people.csail.mit.edu/stauffer/Home/stauffer_publications.