METHODS OF OBJECT TRACKING

Gabriel DANCIU, Iuliu SZEKELY

Electronics and Computers Department
Transilvania University of Brasov, Eroilor 29, 500036, Brasov, Romania
Email: danciu@vega.unitbv.ro

Abstract: This paper presents two methods of object detection/tracking in a video. The first method will use the motion tracking technique combined with density detection which will be described. The second one is based on the SURF algorithm.

Keywords: key points, motion detection, camshift, SURF.

INTRODUCTION

The issues encountered when trying to detect objects in an image are separated into two categories:

- Different light conditions and different focus on the objects, aperture etc.

- Different geometrical stages of the object.

Aside these two issues, the applied mathematical models do not always work. After a considerable time spent on classifying objects into patterns we discover that there will be cases unpredicted which will influence the detection. The two methods presented will try to solve this problem.

FIRST METHOD

The first method is based on the *motion tracking algorithm*. The idea is to extract the contours of the objects as they are moving. After collecting all the points we will group them into clusters of points. One major cluster is approximated as an object.

1.1. Motion detection

Motion templates were invented in the MIT Media Lab by Bobick and Davis (Bobick, *et al.*, 1996) and were further developed jointly with one of their co-inventors (Davis, *et al.*, 1999).

This more recent work forms the basis for the implementation in OpenCV. The idea is to capture a silhouette of the object. This can be done either by capturing the difference between frames, or to learn the background and isolate it from foreground objects.

Other technique is using thermal images, and the hot portions will be foreground objects.

The function used in this approach is cvUpdateMotionHistory, which considers all images arrays of single-channel images.

The mhi parameter is a floating-point image that represents the motion template (motion history image).

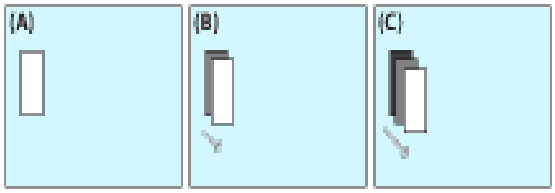

Fig.1. Motion template diagram: (A) Segmented object at current time stamp (white) (B) next time step (C) object moves further

Using other two functions based on the gradient of the mhi image we can compute the orientation of the moving blob.

This paper was recommended for publication by Rustem Popa

1.2. Distribution approximation

The method used is *mean-shift*. The mean-shift algorithm is a robust method of finding local extreme in the density distribution of a data set. In this case the data set will be the points that will be captured by the function described previously.

The mean-shift algorithm runs as follows.

1. Choose a search window:

-its initial location, its type (uniform, polynomial, exponential, or Gaussian);

-its shape (symmetric or skewed, possibly rotated, rounded or rectangular);

-its size (extent at which it rolls off or is cut off).

2. Compute the window's (possibly weighted) center of mass.

3. Center the window at the center of mass.

4. Return to step 2 until the window stops moving (it always will) (Bradsky, *et al.*, 2008).

This algorithm it is related to the discipline of *kernel density estimation*, where by "kernel" we refer to a function that has mostly local focus (e.g. a Gaussian distribution).

If we consider a rectangular kernel then computing mean shift vector means calculating the center of the mass:

$$(1)\ x_c = \frac{M_{10}}{M_{00}},\ y_c = \frac{M_{01}}{M_{00}}$$

Where

$$(2)\ M_{00} = \sum_x \sum_y I(x,y)$$

and

$$(3)\ M_{10} = \sum_x \sum_y xI(x,y)$$

$$(4)\ M_{01} = \sum_x \sum_y yI(x,y)$$

The mean-shift vector in this case tells us to re-center the mean-shift window over the calculated center of mass within that window. This movement will, of course, change what is in the window and so we iterate this re-centering process (Fig. 2).

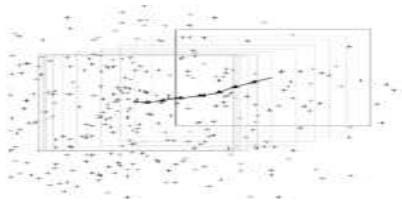

Fig.2. An initial window is placed over a two-dimensional array of data points and is successively re-centered over the mode (or local peak) of its data distribution until convergence

The algorithm we used is *camshift*. It differs from the mean-shift in that the search window adjusts itself in size. If you have well-segmented distributions (say face features that stay compact), then this algorithm will automatically adjust itself for the size of object, as it moves closer to and further from the camera.

The method has two steps:

a. We compute the moving objects using motion detection and place them in an image.

b. On the obtained image we apply the cam-shift algorithm to find the centers of different distributions.

The images shown in Fig. 3 and 4 represent the results for different tests.

The main issue with that approach is that is too slow. The camshift algorithm takes a long time to complete which affects the whole processing time. In Table 1 we have the times in comparison to a classic motion detection algorithm:

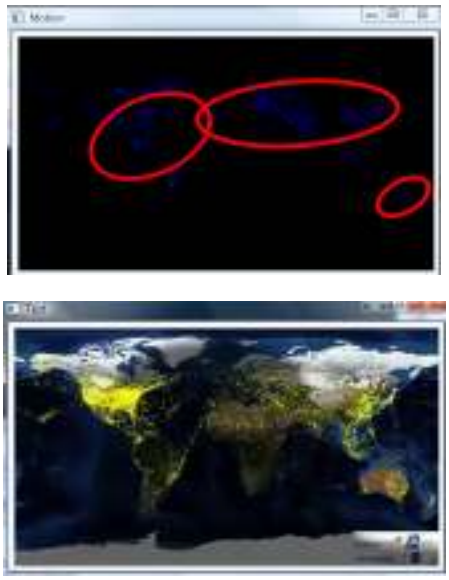

Fig.3. Example: Tracking world air traffic

Table 1 Comparison of times for Camshift and Classic motion detection algortihm

Number of splits per image	1	2	4	6	8
Time per frame(seconds) with Camshift	0.026	0.065	0.211	0.45	0.79
Time per frame(seconds) without Camshift	0.009	0.009	0.009	0.009	0.009

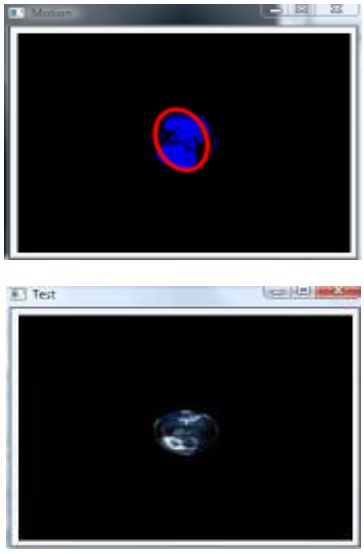

Fig.4. Example: Tracking a water drop

The detection is however much better when adding the camshift algorithm.

SECOND METHOD USING SURF

1.3. Integral images

The second method implies information using keypoints detected with the SURF algorithm, as is described below.

The entry of an integral image $I_{\Sigma}(x)$ at a location x = (x; y) represents the sum of all pixels in the input image ***I*** within a rectangular region formed by the origin and **x**.

$$(5)\quad I_{\Sigma}(x)=\sum_{i=0}^{x}\sum_{j=0}^{y}I(x,y)$$

Once the integral image has been computed, it takes three additions to calculate the sum of the intensities over any upright, rectangular area (Bay, *et al.*, 2008) (Fig. 5).

1.4. Hessian Matrix Based Interest Points

Finding features is based on the computation of the Hessian matrix:

Given a point ***x*** = in an image *I* at a scale of

$$(6)\quad H(x,\sigma)=\begin{vmatrix} L_{xx}(x,\sigma) & L_{xy} \\ L_{xy}(x,\sigma) & L_{yy} \end{vmatrix}$$

where L_{xx} is the convolution of the Gaussian second derivative $\partial^2/\partial x$ with the image ***I*** in point ***x***. The same applies for L_{xy} and L_{yy}.

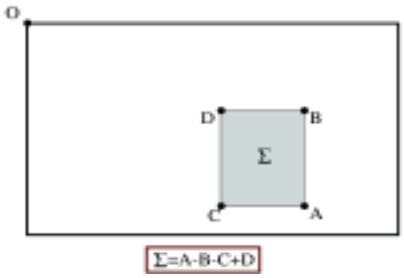

Fig.5. Computing integral images

In order to localize interest points in the image and over scales, a non-maximum suppression in a 3x3x3 neighborhood is applied. The maxima of the determinant of the Hessian matrix are then interpolated in scale and image space with the method proposed by Brown (Brown, *et al.*, 2002).

1.5. Method description

If we select at a certain moment or load from an auxiliary image, the objects we like to track, they will be processed in this way. The key points are extracted from the loaded object and searched in the whole indicated frame. Then, if found, we will crop that part of the image where it was found, forming a new region of interest. This will be the new starting image and we will compare it with the next frame and so on.

As shown in the following pictures (Fig. 6 and 7) the object will be found even if it will suffer photometric and geometric transformations.

Fig.6. Start Image

Fig.7. Transformed Image

CONCLUSIONS

The object tracking works well and will maintain the correct object coordinates even if we apply zoom and other transformations. The condition is that the captured object should not have a predominant color. In this case the number of key points will be low making the tracking impossible.

REFERENCES

Bay, H., Ess, A., Tuytelaars, T. and Van Gool, L. (2008): *Speeded-Up Robust Features (SURF)* , ETH Zurich, BIWI, vol. 1, pp. 1–14.

Bishop, C.M. (2007) *Pattern Recognition and Machine Learning*, New York, Springer-Verlag.

Bobick, A. and J. Davis, J.(1996) *Real-time recognition of activity using temporal templates*, IEEE Workshop on Applications of Computer Vision, pp. 39–42.

Bradsky, G. and Kaehler, A. (2008) *Learning OpenCV* , O'Reily Media Inc., USA, First Edition, ISBN: 978-0-596-516130

Brown, M. and D. Lowe, D. (2002): *Invariant features from interest point groups*. In *British Machine Vision Conference, BMVC 2002,* Cardiff, Wales, pp. 656-665.

Davis, J. and G. Bradski, G. (1999): *Real-time motion template gradients using Intel CVLib*, IEEE ICCV'99 FRAME-RATE WORKSHOP.