CAMSHIFT OBJECT TRACKING ALGORITHM IMPLEMENTATION ON DM6437 EVM

Anton Varfolomieiev[1], *Oleksandr Antonyuk, Oleksandr Lysenko*[2]

National Technical University of Ukraine "Kyiv Polytechnic Institute"
Faculty of Electronics, Department of Design of Electronic Digital Equipment
37, Peremogy Prospect, 03056, Kyiv, Ukraine
phone: (003 044) 4549538, 4549363; E-mail: [1] 7is7@ukr.net, [2] o.lysenko@kpi.ua

ABSTRACT

An approach to creation of an automatic tracking system on the evaluation module, type DM6437 EVM, is considered. As a core of the system the camshift algorithm was chosen. The model of the algorithm was implemented in Matlab Simulink and ported to the evaluation module. The work contains description of some peculiarities related to porting of the algorithm's model to DM6437 EVM board. The pictures included show behavior of the system working with actual video sequences. Comparison of the optimized and non-optimized versions of the program from the viewpoint of operation speed is also performed.

1. INTRODUCTION

Automatic object tracking is a digital image processing method, which consists in finding and tracking various moving objects through successive frames of the video sequence [1]. This function of computer vision is widely used in computer-human interfaces, robotics, medical applications, surveillance, etc. [1, 2]. Of particular interest is realization of automatic tracking systems on embedded platforms. The work described below represents one of such implementations.

Creation of automatic tracking systems on embedded platforms includes two problems: selection of appropriate platform and of the corresponding tracking algorithm able to provide acceptable reliability and speed.

For our applied problem we used a digital signal processor (DSP) oriented to video image processing. Particularly, we took the DSP, type TMS320DM6437, used as a basis for the debugging means called Spectrum Digital DM6437 EVM – in accordance with the task to be solved.

Most of recently developed tracking algorithms use the following principles: correlation methods, optical flow, background subtraction, particle filtering, methods based on probability density evaluation, etc. The correlation and optical flow methods are distinguished with their high computational complexity making them hardly suitable for real-time applications [3]. The background subtraction has low robustness in the presence of noise, and cannot work when the observation camera is moving. The particle filtering is a trade-off of reliability and operation speed (its implementation on DSP platform was described in [4]). In our design, we use an approach based on the probability density estimation algorithm known as camshift. This simple technique has low computational cost and can perform in real-time mode on DSP. Camshift has sufficient reliability, is able to track non-rigid objects when the camera is moving, shows low sensitivity to noise and occlusions.

The considered below DSP implementation may be particularly useful as the part of surveillance system, which allows holding the object of interest within the camera coverage. It also can be used as additional perceptual user interface in multimedia devices based on some OMAP platforms, permitting calculations to be performed on embedded DSP core.

The next sections of this paper describe the camshift algorithm, its model implementation on Matlab Simulink and peculiarities related to the model porting to DM6437 EVM board. Experimental results contain the examples of the tracking performed by the algorithm on DM6437 EVM.

2. CAMSHIFT ALGORITHM

The camshift algorithm was developed for effective face-and-head tracking in perceptual user interfaces. Its main part represents a robust non-parametric technique for climbing density gradients permitting to find the peak of probability density. In the literature, this approach is called the mean-shift algorithm [5].

The original camshift algorithm uses one-dimensional histogram as a captured object model. The histogram consists of the hue (H) channel in HSV colour space.

The object search is being conducted through finding the probability distribution maximum obtained from a so-called histogram back-projection procedure. In order to reduce the amount of calculations, the colour probability distribution is scanned not over the whole image. Instead, we restrict ourselves with calculating the distribution in a smaller image region surrounding the current search window.

The camshift algorithm reduces to the following sequence of steps [5]:

1) Set the calculation region of the probability distribution equal to the whole frame.
2) Choose the initial location of the two-dimensional mean shift search window.

3) Calculate the colour probability distribution in the 2D region centred on the search window location in an area slightly larger than the mean shift window size.

4) Perform the search of the maximum density probability using the mean shift parameter for convergence or for setting the number of iterations. Store the zero moment (area or size) and middle position.

5) For the next video frame, place the search window in the middle position fixed in step 4, and set the window size in conformity to the last moment. Go to step 3.

As mentioned above, on step 2 the probability distribution is calculated by the back-projection procedure. This low-level operation puts the pixel values in the image into correspondence with the values of target's histogram bins.

For discrete 2D image probability distributions, the mean location (the centroid) within the search window (steps 3 and 4) can be determined as follows:

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

$$M_{10} = \sum_x \sum_y xI(x,y),\ M_{01} = \sum_x \sum_y yI(x,y), \qquad (1)$$

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

where $I(x, y)$ is the value of discrete probability distribution at a point (x, y) on the back-projected image; values x and y change within a search window; M_{00} – zero moment; M_{01} and M_{10} – first moments; (x_c, y_c) – mean location of a search window (centroid). Calculation of mean location is performed iteratively until a minimum shift of one pixel in either of x_c and y_c values will be achieved. The maximum number of iterations is usually taken to be 10-20 [6].

The scale and shape of the distribution (object's scale and shape) are calculated using the equations [5, 6]

$$M_{20} = \sum_x \sum_y x^2 I(x,y),\ M_{02} = \sum_x \sum_y y^2 I(x,y),$$

$$M_{11} = \sum_x \sum_y xyI(x,y),$$

The first two eigenvalues (major length and width) of the probability distribution (corresponding to our object) captured by camshift can be calculated in closed form through central moments:

$$\mu_{20} = \frac{M_{20}}{M_{00}} - x_c^2,\ \mu_{02} = \frac{M_{02}}{M_{00}} - y_c^2,$$

$$\mu_{11} = 2\left(\frac{M_{11}}{M_{00}} - x_c y_c\right),$$

The minor and major semi-axes w and l of the distribution centroid can be determined from the following equations:

$$w = \sqrt{\frac{(\mu_{20} + \mu_{02}) - \sqrt{\mu_{11}^2 + (\mu_{20} - \mu_{02})^2}}{2}},$$

$$l = \sqrt{\frac{(\mu_{20} + \mu_{02}) + \sqrt{\mu_{11}^2 + (\mu_{20} - \mu_{02})^2}}{2}}.$$

The object orientation (major axis inclination) can be found by the formula:

$$\theta = \frac{1}{2}\arctan\left(\frac{\mu_{11}}{\mu_{20} - \mu_{02}}\right).$$

If the target histogram contains a significant number of features that belong to a background image or some adjacent objects, target position and scale can hardly be determined accurately. To cope with this problem, a simple technique was employed: we decreased the weights of target's histogram bins belonging simultaneously to the background histogram. In other words, we used the ratio between target's histogram bins and the respective background histogram bins (the background corresponds to images outside the initial search window) – the so-called histogram weighing. Having simplified denotations adopted in [7], the histogram weighing can be written as:

$$q_r = \begin{cases} q_o / q_b,\ q_b \neq 0 \\ q_o,\ \text{otherwise} \end{cases}, \qquad (2)$$

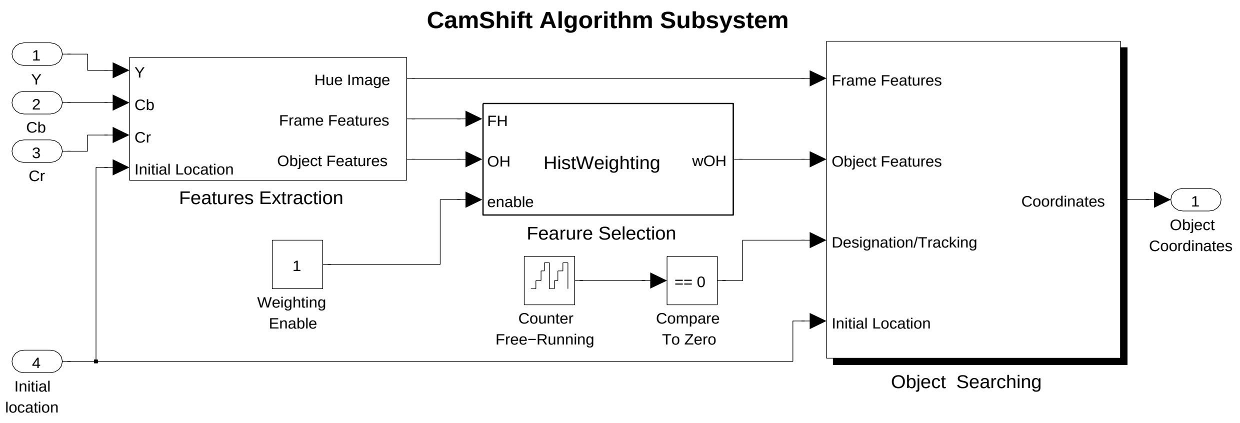
where q_r is the weighed histogram (ratio histogram); q_o is the target histogram; and q_b is the background histogram.

More detailed description of the camshift and mean-shift algorithms for object tracking is contained in [5-7].

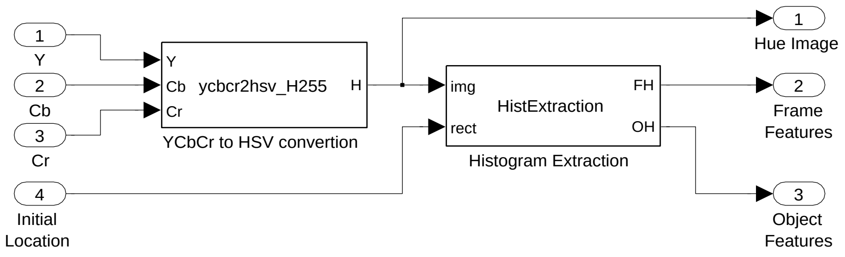
3. SOFTWARE IMPLEMENTATION

To simplify realization of the above algorithm on DM6437 EVM, we used Matlab Simulink environment. Such an approach allows for debugging the algorithm on a general-purpose computer, and then porting it to the desired platform. The Simulink model, which realizes camshift algorithm, is presented in Fig. 1-a. Its input parameters represent the frame in YCbCr colour space format and initial position of the object to be tracked. The object's representation (its histogram) extracted from the designated location of the first frame in the video sequence.

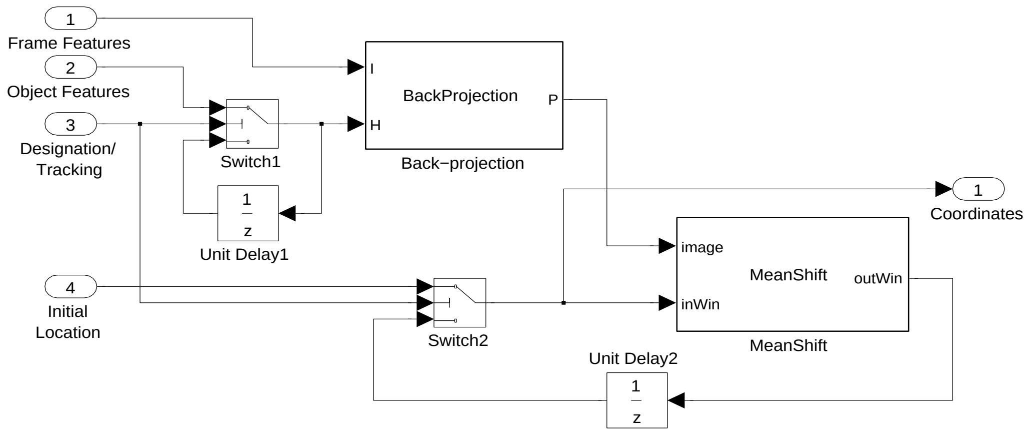
The *Features Extraction* unit, shown in Fig. 1-*b*, is responsible for selection of proper features from the input frame. Since Simulink does not contain the procedure of direct colour conversion from YCbCr to HSV space, it is realized as an embedded Matlab function. The colour space conversion is performed through the intermediate RGB colour system. The function's output is only the Hue component. After that, *HistExtraction* procedure calculates two histograms: the first one is the object histogram (*OH*) while the second – histogram of the image outside the object's rectangle (*FH*). The signals *Hue*, *OH* and *FH* will



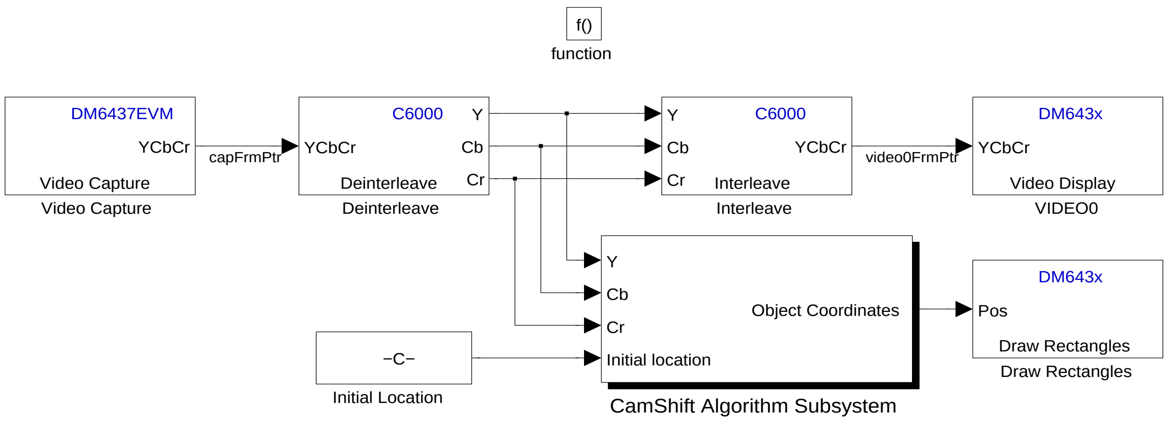
a)



b)



c)



d)

Figure 1 – Simulink model of camshift algorithm: algorithm subsystem (a); features extraction subsystem (b); object searching subsystem (c); DSP/BIOS function which realizes camshift object tracking on DM6437EVM (d).

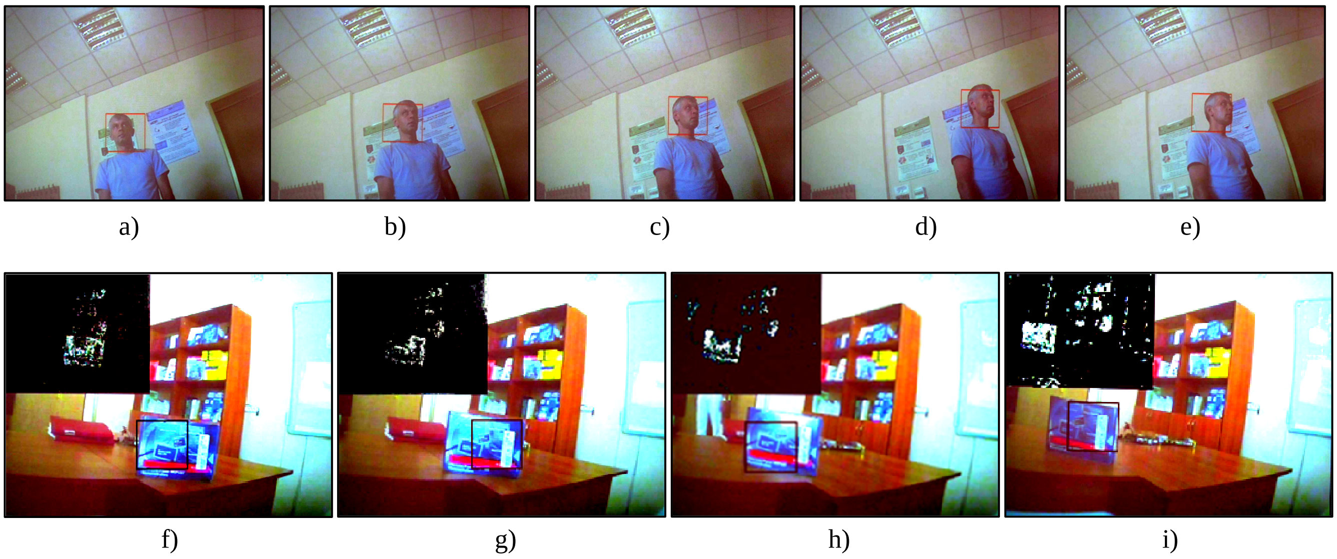


Figure 2 – Example of object tracking by camshift on DM6437 EVM (all the images were captured from output of the evaluation module): face tracking (*a-e*)*; tracking of the object with non-uniform colour distribution* (*f-i*).

be transferred to other units.

The *Features Selection* unit is responsible for histogram-weighting procedure according to Eq. (2). In fact, this procedure actually makes simple element-by-element division of the *OH* signal by *FH* in the event when the *enable* signal is nonzero (meaning that the histogram weighting is off).

The weighed histogram and Hue signal from the *Features Extraction* unit come to the input of the *Object Searching* unit (Fig. 1-c)). The latter includes two procedures: *BackProjection* and *MeanShift*. The first one, with the aid of weighed histogram and Hue image, forms the probability distribution function (PDF), while the second, based on this PDF and formulas (1), calculates coordinates of the object position. At the same time, in conformity with camshift algorithm, we take account of the object position obtained at the previous frame, by introducing *Unit Delay 2*. At this stage we do not perform the target scaling.

After debugging the algorithm, it can be ported to the DM6437 EVM evaluation module. To do this, Simulink includes Target Support Package. An example of model compilation, suitable for application in DM6437 EVM, is contained in Matlab help. To make it more clear, Fig. 1-*d* illustrates the *DSP/BIOS Task* function.

The main peculiar feature of the algorithm realization on this particular board is the signal format. The video decoder generates the signal in sparse 4:2:2 form. To avoid problems with further processing, transformation into 4:4:4 format is highly desirable. The procedure feasible by adding Chroma Resampling unit included in Simulink. This unit has to be placed in front of camshift Algorithm Subsystem. Video and Image Processing Blockset. One can be plugged before camshift Algorithm Subsystem. By some considerations, we did not put Chroma Resampling unit in our project we performed transformation manually, directly in *ycbcr2hsv_H255* function. In other words, we carried out subsumpling of lines and columns of the initial frame resulting in reducing it by half, particularly, from 720x576 to 360x288 pixels in PAL standard. It permits to reduce the amount further of computations performed by the at least by a factor of four.

Since TMS320DM6437 is an integer-type processor, execution of the algorithm can be accelerated by using the integer and fixed point arithmetic where possible. Moreover, the C code generated by Matlab is far from optimal. This stems from numerous checks for integer overflow, and usage of floating point operations not supported by the processor hardware. For this reason, the algorithm performance can be appreciably improved by C code optimization. In addition, calculation of histograms may be carried out on a special-purpose peripheral module [8]. This issue may become an item for further investigations.

In order to make the system well-controlled, i.e. to render the user an opportunity for turning the tracking on and off, or set the position and size of objects at any time instant, it is desirable to modify the C code and to introduce the control means into GEL files in Code Composer Studio environment.

4. EXPERIMENTAL RESULTS

In our experiments, a PAL video camera was used. To avoid performance slips in the process of algorithm execution, automatic white balance in the video camera was switched off. The camera was adjusted so that image brightness was neither too dim nor saturated.

Figure 2 gives examples of tracking performed by the designed system. Sequence (a-e) corresponds to tracking of a human head. As we can see, the designed system performs robust tracking in case of head rotation. Sequence (f-i) shows tracking of an object with non-uniform colour distribution. In the left-top corner of each frame in sequence (f-i) we can see images produced in the course of back-projection operation. The brightness of the minor frames permits to estimate probabilities for this frame.

The time needed for frame processing by the optimized and non-optimized code is shown in Fig. 3. Time measurements were performed with the aid of *CLK_gethtime*() procedure available in DSP/BIOS.

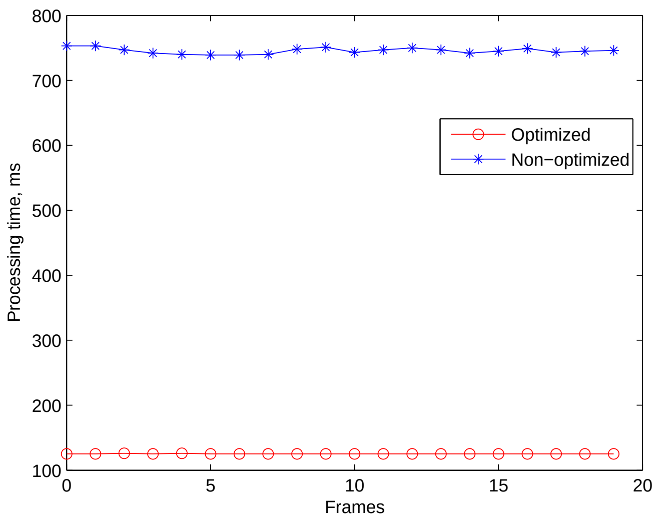


Figure 3 – Frame processing time attainable by the optimized and non-optimized programs.

As we can see, the average time of frame processing by the non-optimized algorithm is 750 ms while for the optimized one it is about 125 ms. Thus, manual optimization makes it possible to raise the speed of the code generated in Matlab at least by a factor of 6.

It turned out in the course of tests, that the most time-consuming was the procedure of conversion of colour systems from YCbCr to HSV. Its optimization makes it possible to raise to maximum degree the promptness of the designed system. The procedure of mean-shift, on the contrary, almost has no effect on the operation speed.

It should be noted that the algorithm "looses" the object if its shift during the interval between frames exceeds the size of the search window. Because of peculiarities of the Hue colour component, the system may exhibit instability when tracking objects of certain shades, grey in particular.

5. CONCLUSION

Realization of the camshift tracking algorithm on the Spectrum Digital DM6437 EVM evaluation module is suggested and tested. The designed system is able to track easily various colour objects, even in the case of changing their size and shape. The method holds its stability when the camera is moving, and shows low sensitivity to noise and occlusions.

The use of Matlab Simulink makes it possible to simplify the design and debugging of the tracking system, but the program generated in the C language by the medium turns out to be far from optimal. Its optimization "by hand" permits to increase the performance of the system six times.

After thorough trials of the algorithm, new ways to its improvement became clear. Particularly, scale adaptation would give much benefit. Also, the further C code optimization may give a considerable gain in speed. All this forms a basis for future investigation.

REFERENCES

[1] C. Yang, R. Duraiswami, A. Elgammal and L. Davis, "Real-Time Kernel-Based Tracking in Joint Feature-Spatial Spaces," Technical Report CS-TR-4567, UMIACS, College Park, 2003.

[2] F. Porikli, "Achieving Real-Time Object Detection and Tracking Under Extreme Conditions," in *Journal of Real-Time Image Processing*, Springer, vol. 1, no. 1, pp. 33–40, June 2006.

[3] E. Roichman, Y. Solomon, Y. Moshe, "Real-Time Pedestrian Detection and Tracking" in *Proc. of the 3rd European DSP Education and Research Symposium (EDERS 2008)*, Tel-Aviv, pp. 281-288, June 2008.

[4] H. Fu, Z. Cao and X. Cao, "Embedded omni-vision navigator based on multi-object tracking," in *Machine Vision and Applications*, http://www.springerlink.com/content/f58l7674x024x722.

[5] G. Bradski "Computer Vision Face Tracking For Use in a Perceptual User Interface," in *Intel Technology Journal*, pp. 1–15, Q2. 1998.

[6] J. Allen, R. Xu, J. Jin, "Object Tracking Using Camshift Algorithm and Multiple Quantized," in *VIP '05: Proceedings of the Pan-Sydney area workshop on Visual information processing*, pp. 3–7, 2004.

[7] D. Comaniciu, V. Ramesh, P. Meer, "Kernel-Based Object Tracking" in *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 25, no. 5, pp. 564–575, 2003.

[8] Texas Instruments, "TMS320DM6437 Digital Media Processor (Rev. D)," http://focus.ti.com.cn/cn/lit/ds/symlink/tms320dm6437.pdf