Feature extraction of objects in moving images and implementation of the purification algorithm on the CNN Universal Machine

Emel Arslan, Zeynep Orman, Sabri Arik

***Abstract*—** This paper presents an algorithm that detects certain objects in a moving image and extracts their features. We named this algorithm as the purification algorithm because it is also used for purifying the remains of detected objects and for saving each purified object as a separate image file. The algorithm is implemented on the Bi-i Cellular Vision System which is a Cellular Neural Network(CNN) Universal Machine. The CNN Universal Machine is known as the analogical array computer and it contains two processors which can work interactively with each other. These processors are the ACE16k that is the hardware implementation of CNNs and the Digital Signal Processor(DSP). The purification algorithm is implemented with two different applications. In the first application, all phases of the algorithm are implemented just on the DSP. In the second application, the morphological operations of the algorithm are performed on the ACE16k and all other operations are performed on the DSP. Therefore, in the latter one, the application is run in coordination with both the ACE16k processor and the DSP. The obtained results are evaluated in terms of the run-time of the purification algorithm to show the comparison of these applications. Experimental results show that the performance of the proposed algorithm is good.

***Keywords*—** ACE16k, Bi-i Cellular Vision System, Cellular Neural Networks, CNN Universal Machine, Digital Signal Processor, Image processing.

I. Introduction

IMAGE processing is one of the most important research topics in recent years. It is widely used in areas such as military, security, health, biology, astronomy, archeology and industry [1-13]. For an image to be processed, it should be presented in a format that a computer can understand, this means it should be converted into its related digital form. In the digital form, each of its pixel is expressed by means of the corresponding element of a matrix.

Manuscript received April 29, 2011.

This work was supported by Scientific Research Projects Coordination Unit of Istanbul University. Project number: 14586.

Emel Arslan is with the Research and Application Center for Computer Sciences, Istanbul University, Istanbul, Beyazıt, 34452, Turkey (phone: +90-212-4400093; fax: +90-212-4400094; e-mail: earslan@istanbul.edu.tr).

Zeynep Orman is with the Computer Engineering Department, Istanbul University, Istanbul, Avcilar, 34320, (e-mail: ormanz@istanbul.edu.tr).

Sabri Arik is with the Computer Engineering Department, Istanbul University, Istanbul, Avcilar, 34320, (e-mail:ariks@istanbul.edu.tr)

Algorithms that are developed for digital image processing require fast systems due to their processing load. These computers cannot satisfy the need for speed, when we especially consider the implementation of real-time moving image processing algorithms that require at least 15-25 frames to be processed in seconds.

Cellular Neural Network (CNN) theory that was proposed by Chua and Yang in 1988, is an analog, nonlinear and real-time processing neural network model [14]. CNNs also have advanced features for image processing applications. In 1993, Roska and Chua have presented the CNN Universal Machine [15, 16]. This analogical array computer has cellular processors (ACE4k, ACE16k, etc.) which are the hardware implementation of CNNs and it is very suitable for image processing applications with its advanced computing capabilities. Bi-i Cellular Vision System is a CNN Universal Machine that can process high-speed and real time transactions and can be defined as a compact, independent and intelligent camera. This system has high-resolution sensors and two different processors named as CNN (ACE16k) and Digital Signal Processor (DSP) that can communicate with each other [17].

In this study, we will first discuss how to detect certain objects in a colored image and propose a new purification algorithm that is used to purify the remains of each object. This algorithm as a whole combines the preprocessing and the purification phases to remove false minutiae such as holes, spikes as well as the remains of the object. These remains are actually the parts of other objects that can stay within the boundary of the frame. Then, we will present an implementation of this algorithm on the Bi-i Cellular Vision System and evaluate the results that are obtained.

The remainder of this letter is organized as follows. Section II introduces fundamental concepts about CNN architecture, CNN Universal Machine, ACE16k processor, Bi-i Vision System and Bi-i programming, respectively. In Section III, an algorithm that detects certain objects in moving images and purifies each object from its remains is proposed. In Section IV, a real implementation of the purification algorithm is provided to compare the obtained results and finally, Section V evaluates the results and concludes the paper.

II. CNN ARCHITECTURE AND BI-I CELLULAR VISION SYSTEM

This section provides fundamental concepts about CNN architecture and Bi-i Cellular Vision System.

A. Architecture of the Cellular Neural Networks

Cellular Neural Networks (CNNs) derived from Hopfield Neural network is introduced in [16]. The two most fundamental components of the CNN paradigm are the use of analog processing cells with continuous signal values, and local interaction within a finite radius.

The structural design of CNN is formed with basic circuits called cells. Each cell containing a linear capacitor, a non-linear voltage controlled current source, and a few resistive linear circuit elements is connected to its neighboring cells; therefore direct interactions take place only among adjacent cells. Mathematical expression of the standard CNN model is described by the set of linear differential equations given by (1) which are associated with the cells in the circuit. The activation function of the CNN cell can be expressed by the set of nonlinear equation (2).

$$C\frac{d}{dt}x_{ij}(t) = -\frac{1}{R}x_{ij}(t) + \sum_{C(k,l)\in S_r(i,j)} A(i,j;k,l)y_{kl}(t) + \sum_{C(k,l)\in S_r(i,j)} B(i,j;k,l)u_{kl}(t) + z_{ij} \quad (1)$$

$$y_{ij} = f(x_{ij}) = \frac{1}{2}\left(|x_{ij}+1| - |x_{ij}-1|\right) \quad (2)$$

where,

$x_{ij} \in R$; State variable of cell *C(i,j)*,

$y_{kl} \in R$; Outputs of cells,

$u_{kl} \in R$; Inputs of cells,

$z_{ij} \in R$; Threshold,

A(i, j ; k, l) ; Feedback operator,

B(i, j ; k, l) ; Control operator.

y_{ij} ; Output equation.

Without loss of generality, linear resistor (R) and linear capacitor (C) values can be set to 1. The block diagram of a cell C(i, j) is shown in the Fig. 1.

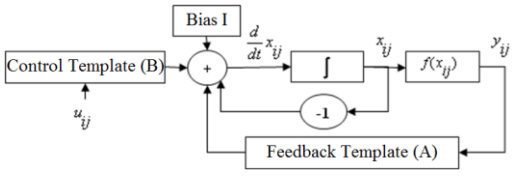

Fig. 1 The block diagram of a cell C(i, j)

In this way CNNs have provided an ideal framework for programmable analog array computing. It means that the CNN can be used as a programmable device where the instructions are represented by the templates which define the connections between a cell and its neighboring cells. In general, the CNN templates consists of the feedback template (B), control template (A) and bias value. Basically, three different images can describe a CNN layer, that is, the input U, the state X and the output Y.

Each cell of a CNN is represented by a square and shown in Fig. 2. In this CNN architecture, each cell is linked only to its neighbors.

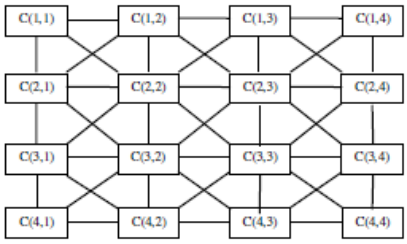

Fig. 2 A 4x4 cell two-dimensional CNN

Let us assume a CNN with MxN cells are arranged in M rows and N columns and the cell in row i and column j is denoted as C(i,j)[16]. r-neighborhood of a C(i,j) cell is defined with the following definition (3) provided that r is a positive value [18].

$$N_r(i,j) = \left\{ C(k,l) \,\middle|\, \max_{1\le k\le M, 1\le l\le N} \{|k-i|,|l-j|\} \le r \right\} \quad (3)$$

B. CNN Universal Machine

The hardware implementation of CNN is easier compared to the Artificial Neural Networks as there is only connection between the neighbor cells and the cell structure. Analogical Cellular Engines (ACE4k, ACE16k etc. [19, 20]) are based on CNN Universal Machine architecture. CNN Universal Machine(CNN-UM) architecture has been called by Roska and Chua as analogical computation since it can both perform the analog array operations and the logical operations together [17].

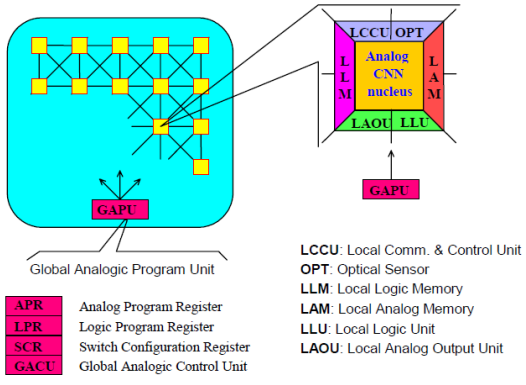

Fig. 3 The architecture of the CNN Universal Machine

Fig. 3 denotes the CNN-UM architecture which is based on the dynamic computing of a simple CNN. This figure shows the elements in the complex CNN Nucleus and the functional blocks of the Global Analogic Programming Unit[21].

C. ACE16k Processor

ACE16k, is a CNN based processor of CNN Universal Machine which can perform analog operations. ACE16k which is used to perform various image processing operations contain low resolution (128 x 128) CMOS gray level image sensor and analog processor arrays. This processor array is much faster (30000 frames per second) than the conventional processors in image processing applications since it can processes the whole image in parallel.

D. Bi-i Cellular Vision System

The Bi-i Cellular Vision System which contains two different processors, a CNN based ACE16k and a DSP that can be defined as a compact, standalone and intelligent camera capable of real time operations at very high speed [2, 21]. The images are stored in local memories with the help of two different sensors as a (1280x1024) color CMOS sensor, and a (128x128) ACE16K sensor [22, 23].

The block diagram of the Bi-i V2 Cellular Vision System is given in Fig. 4. As seen from the figure, this system has a color CMOS sensor array (IBIS 5-C) and two high-end digital signal processors (TX C6415 and TX C6701). This system runs an embedded Linux on the communication processor and has complex external interfaces like USB, FireWire and a general digital I/O in addition to the Ethernet and RS232[21].

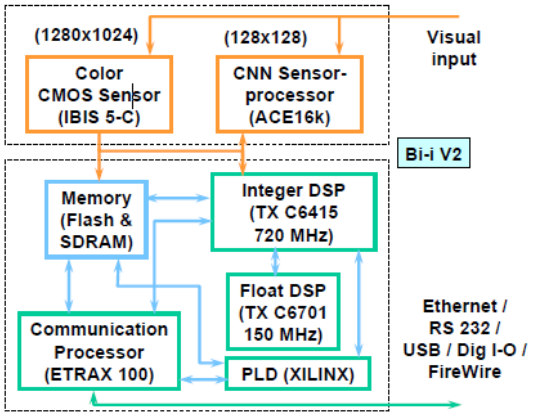

Fig. 4 The block diagram of the Bi-i V2 Cellular Vision System

E. Bi-i Programming

CNN Universal Machine has two different programming methods. One of them is AMC (Analogical Macro Code) language which is a conventional Bi-i programming method. The codes written in AMC language are converted to binary basis and run on Bi-i. Another method is the Bi-i (Software Development Kit - SDK) which is used to develop more complex applications. Bi-i SDK, consists of the C++ programming library which is a group used to develop applications. These libraries can also used for the Digital Signal Processor (DSP) with the development unit Code Composer Studio and they contain many functions to control the whole ACE16k circuit [2].

III. AN ALGORITHM THAT DETECTS OBJECTS IN MOVING IMAGES AND PURIFIES THE REMAINS OF EACH OBJECT

An algorithm that detects objects in moving images and extracts their features is developed by using the Bi-i Cellular Vision System. We named this algorithm as the purification algorithm because it is also used for purifying the remains of certain objects that are detected in moving images and for saving each purified object as a separate image file. These remains can be the parts of other objects that stay within the frame of a certain object and/or be the background of the object itself. This kind of remains problem inevitably occurs in image processing applications because each pixel of an object is expressed as an element of a matrix. Therefore, within the frame of a certain object, there can be some remains that belong to other objects of the moving image.

The algorithm that we developed provides a solution to this problem for especially colored and moving images. A block diagram of the purification algorithm is shown in Fig. 5.

As clearly seen from Fig. 5, this algorithm gets a colored moving image as an input. For each frame of the moving image, two phases are applied generally which are the preprocessing phase and the purification phase, respectively. In the preprocessing phase, some processes are performed on the DSP while some others are run on the analog processor. The first operation with the preprocessing phase is to convert the colored frame into a gray level image on the DSP. The obtained gray level frame is then transferred to analog processor (ACE16k) to apply low pass filtering. Thus, sharp color transitions and spikes are partly softened to reduce the noise on the image. The filtered image is again transferred on the DSP to apply the thresholding process and the gray level image is converted to a binary image. After this process, in the obtained binary image, the background is represented with the white color and the objects are represented with the black color because of the luminosity of the gray level image. However, the Bi-i V2 that is used to develop our implementation evaluates white colors as the objects and black parts as the background of the image while it determines the objects and it processes the image in this manner. Therefore, there is a need to apply a negation process to the binary image so that the Bi-i V2 can process it. After the image is converted to the appropriate format, it is given as an input to the Feature Extraction function that is implemented on the DSP. As a result of this process, the objects in the image and their features are extracted.

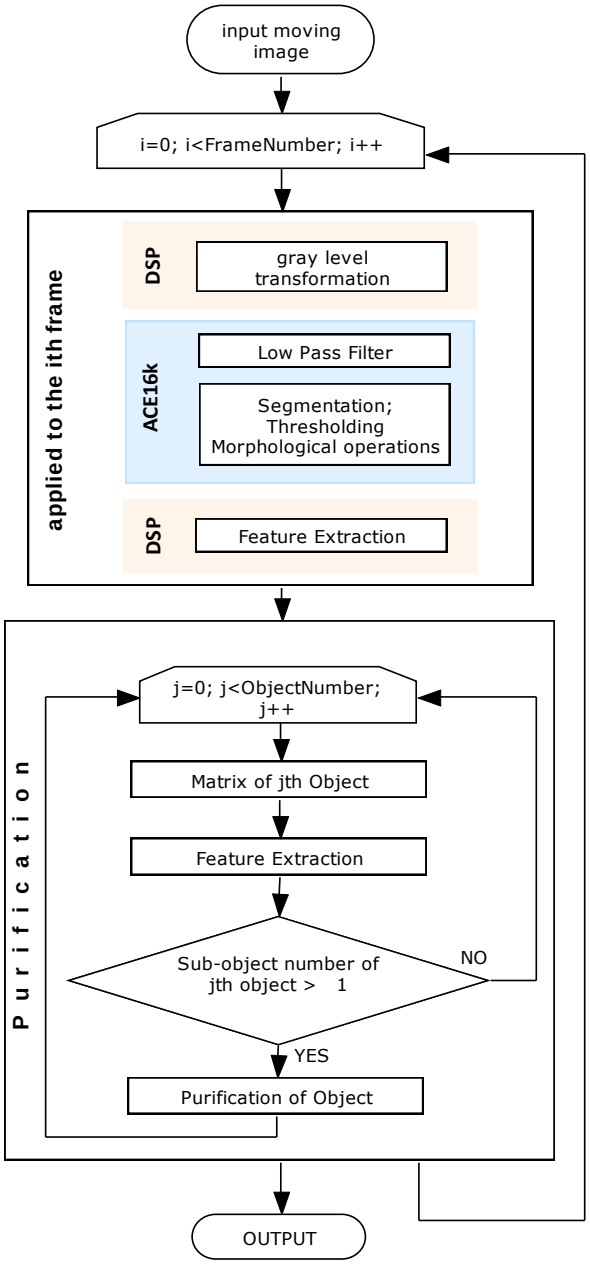

Fig. 5 A block diagram of the purification algorithm

After the feature extraction process, the purification phase is applied to the objects that are detected within the image. In this phase, each object is given to the Feature Extraction function as an input and as a separate image by using the bounding box features of the objects. The next step in this phase is to determine if there are any other objects or remains of other objects within the boundary of the frame. After the feature extraction process, if the number of objects determined is one in the bounding box, this means there is no overflow of other objects and the algorithm continues with the next object for feature extraction. On the other hand, if the number of objects determined is greater than one, this means there are some remains that belong to other objects within the boundary of the frame and the purification process is applied. This purification phase is applied to all objects in each frame. After this process is completed, the objects are all cleared from the remains in the moving image.

A. Preprocessing

Each frame of a moving object which is an input to the algorithm, is processed as a separate image. The first operation is to transform the frame to be processed into its relevant gray level image that runs on the DSP processor by using the RGB2ByteMatrix function. This function is located in Utils.h library under Instant Vision BaseData. The command line that is used for this transformation is given below:

```
RGB2ByteMatrix(sourceGRAY,sourceRGB,CH_RGB);
```

The moving image matrix that is transformed to a gray level image is then transferred to the ACE16k processor for morphological operations. Here, the aim is to detect objects in a moving image as close as possible. The morphological operations are Low Pass Filtering [2, 24], Thresholding, Negation, Point Removing, Hole Filling and Opening, respectively. For thresholding, we use the ConvLAMtoLLM function that is located in TACE.h library. For other operations we use the functions in TACE_IPL.h library. These are all ACE16k libraries that are under Instant Vision BaseData.

For morphological operations, the first step is to pass the moving image through a linear Low Pass Filter to remove the noise. This filter is used to get rid of some little details known as minutiae and fill the holes in lines before detecting the objects.

For filtering a moving image, we use the following command lines in our implementation:

```
ace << C_LAM1 << sourceGRAY;
ace.LowPassFilter (C_LAM2, C_LAM1,1, 0.05);
ace.ReadLAM(sourceGRAYFiltered, C_LAM2);
```

A gray level image that is sourceGray, is loaded to a local analogical memory (C_LAMI) that is on the ACE16k processor to be processed by using the first command line. The second command line applies low pass filtering to an image matrix that is on the C_LAM1 by using the LowPassFilter( ) function and passes the resulting image to another local analogical memory (C_LAM2). Finally, the last line assigns the resulting image to a variable named sourceGRAYFiltered.

By applying the low pass filtering, we now have a smoother image. Afterwards, this image is converted to a binary image by applying thresholding process. The command lines that are used for this conversion is given below:

```
ace.ConvLAMtoLLM(C_LLM2,C_LAM2,50);
ace.ReadLLM(sourceBit, C_LLM2);
```

In the first line, the image matrix that is on C_LAM2, is converted into a binary image and transferred to another local

analogical memory (C_LLM2) by using ConvLAMtoLLM( ) function. The second line assigns the resulting matrix to a variable named sourceBit.

To perform thresholding by using ByteMatrix2BitMatrix( ) function that runs on the DSP, we use the following command line:

```
ByteMatrix2BitMatrix(sourceBit, sourceGRAY,50);
```

After thresholding, the objects are represented as black and other parts are represented as white in the resulting image matrix. However, our algorithm detects groups of white pixels as objects and carries out its operations in this manner. To solve this problem, we need to negate the resulting image. This means, we should replace 0s with 1s and 1s with 0s in the image matrix by applying a negation process. This binary negation process is performed by the following command line:

```
ace.Not(sourceBitNegation,C_LAM2);
```

The same process can be run on the DSP processor with a Negation( ) function that is written in C++ programming language. This function is used with the following command line:

```
Negation(sourceBitOut, sourceBit);
```

After the negation process, the morphological operations that are given with the following command lines, are applied to the resulting image, respectively.

```
ace.SetIPLMode(IPL_MORPH);
ace.Calibrate();
ace.PointRemove();
ace.HoleFiller(1);
ace.Opening8(1);
ace.Dilate8(1);
```

All these morphological operations are run on the ACE16k processor. Before going on with other morphological operations, the SetIPLMode( ) function should be used to set up the ACE16k processor to carry out these procedures. Then, before each morphological operation, the Calibrate( ) function is used to calibrate the ACE16k processor for morphological operations. It is very important to use this function and repeat it in every 10-20 milliseconds if the processor works in an IPL mode constantly because the ACE16k processor can forget this calibration process**.** This process is not important for gray level images.

After the calibration process, the PointRemove( ) function that cleans the minor pixels from the image, the HoleFiller( ) function that fills the holes of the image, the Opening8( ) function that performs the opening process and finally the Dilate8( ) function is applied to the image, respectively [21].

The command lines that perform these morphological operations on the DSP processor are as follows:

```
PointRemove(sourceBitOut,sourceBitOut);
HoleFiller(sourceBitOut, sourceBitOut, 1);
Opening8(sourceBitOut, sourceBitOut, 1);
Dilate8(sourceBitOut, sourceBitOut, 1);
```

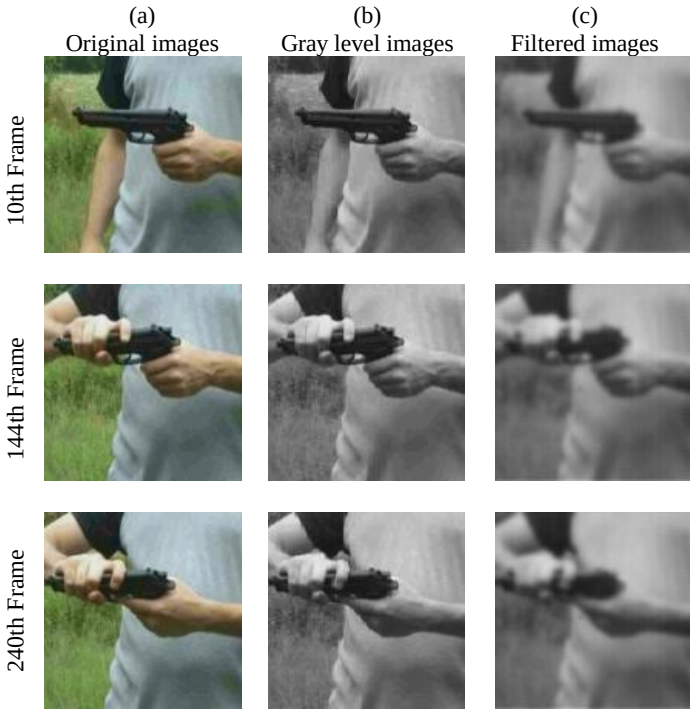

Fig. 6 Low pass filtering steps for different frames

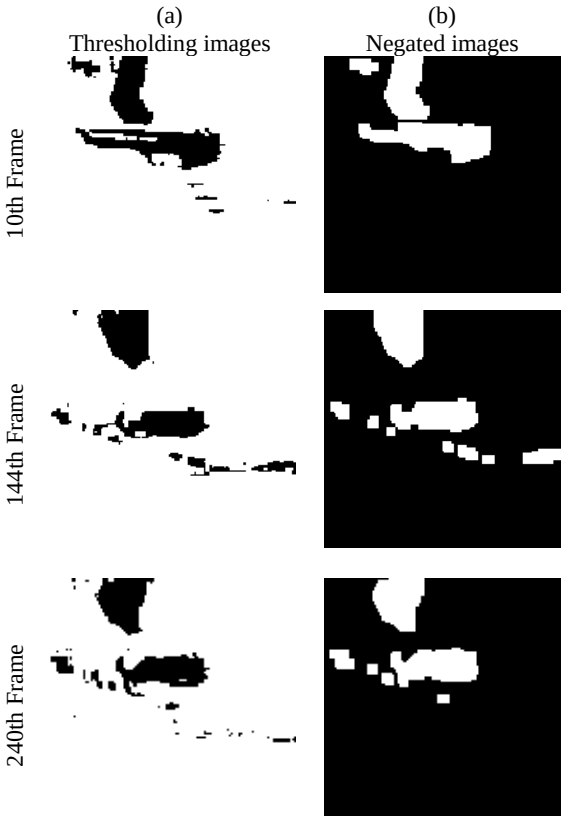

Fig. 7 Intermediate results for different frames

Sample output images that are obtained after the preprocessing phase, are shown in Fig. 6 and Fig. 7, respectively for different frames that belong to a specific moving image. Fig. 6 denotes the original images, the gray level images and the filtered images for frames 10, 144 and 240. For the same frames, Fig. 7 denotes the obtained images after applying the processes thresholding and negation.

B. Extracting the Objects and their Features

The binary image that is preprocessed depending on the input moving image is now available for object detection and is an input parameter to the CalcFeatures( ) function. This function is used as follows:

CalcFeatures(ObjNum, Features, sourceBitOut, FEAT_ALL);

After this process, the objects in a moving image are detected and their features are extracted. These features can be given as follows:

- Bounding Box: It draws a minimal rectangle around the object by using its upper-left and lower-right coordinates.
- Extremes: These are the extreme points on the bounding box of the object. These points are: Upper-Left, Upper-Right, Right-Upper, Right-Lower, Lower-Right, Lower-Left, Left-Lower and Left-Upper.
- Eccentricity: It is the proportion between the focus distance and the length of the longest axis of the current ellipse of the object. The eccentricity of a circle is 0 whereas the eccentricity of a line is 1.
- Diameter: It is the diameter of the circle which has the same area with the object.
- Orientation: It is the angle of the largest axis value of the current ellipse as the object to the positive direction of the horizontal axis.
- Extent: It is the proportion between the area of the object and the area of the bounding box.
- Center: It denotes the coordinates of the geometric coordinates of the object according to the upper left corner of the image.

In this implementation, to determine the handgun object in each frame, the features that are extremes, bounding box, center, eccentricity, diameter and orientation are taken into account primarily.

Some of the statistics like minimum (Min), maximum (Max) and average values obtained as a result of these features are presented in Table I.

Feature	Min.	Max.	Average
Area	229	1546	733.93
Eccentricity	0.50	0.99	0.96
Diameter	17.08	44.37	30.31
Orientation	0.01	179.99	92.88
Extent	0.32	0.87	0.66
Center x value	17.42	108.23	60.98
Center y value	12.58	76.70	48.36

Table I. Statistical values of certain features of the objects

All of these statistical information have been computed by processing the whole moving image (1394 frames). Since the object does not have exactly the same shape and the same position in every frame of the moving image, the average values should be used to describe the certain object to facilitate the identification process.

After the objects and their features are extracted as the last step of the preprocessing phase, the resulting image is then saved to be purified from its remains.

C. Purifying Certain Objects from its remains

Each pixel of an image is expressed by means of the corresponding element of a matrix, so even no remains of other objects are detected, a part of the background will still be within the boundary of the certain object that is detected in the image. That is why we should purify the detected objects from the remains of other objects and/or from the background of the frame. The input moving images that we are working on are colored images and this makes the purification process easier because we can use the color codes of the objects. In our implementation, we will consider the moving image given in Fig. 6 and try to purify the most distinct object in that certain image - which is the handgun.

Fig. 8. Original and output images after the purification process

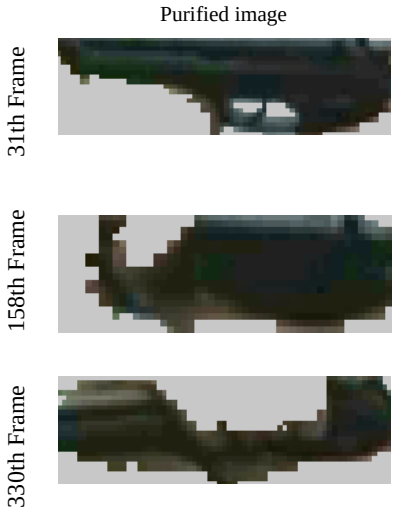

Fig. 9 Purified images after the purification process

For this implementation, we first obtain a matrix that represents the object by using the extreme points. This object is then saved as a separate image file with the help of the related matrix and purified from its remains that belong to other objects and from the background image by using the color codes. This function that performs this process is coded in C++ programming language.

Because the image to be processed is colored, we can get the Red-Green-Blue(RGB) values of each pixel of the handgun object and with these values, we call the ForMask( ) function. This function gets three input parameters. These are the matrix that represents the colored image, the [x,y] coordinate values of the pixel that is to be processed and the upper and lower values of the image that is to be purified. ForMask( ) function examines each element of the input matrix in accordance with the constraint generated by the color codes. After the completion of this process, the extracted object is all freed from the colors that do not belong to it. Finally, the last step of this purification process is to resave the detected object with a pre-determined plain background color. Some example results for different frames from our implementation are shown in Fig 8 and Fig 9.

In Fig.8 (a), some sample original images that are taken from the input moving image to be processed are given for frames 31, 158 and 330. Fig.8 (b) shows the separate image files that contain the detected handgun object by using the bounding box values of the object for the same frames.

Finally, Fig.9 shows the resulting images in which the detected handgun object is purified from its remains that are overflow within the frame. As a result of this process, the handgun object is all freed from its remains and also from its background image to be obtained just itself.

IV. Experimental Results

In this algorithm, the filtering and the segmentation processes that are applied to the image in the preprocessing phase, can be run on the ACE16k processor. For this implementation, we write two different program codes that one of them is just run on the DSP and the other one is run both on the ACE16k and the DSP interactively. The results obtained from the implementation are evaluated as run-time of the purification algorithm and shown in Table II.

When we compare these results, one can easily notice that processing the algorithm on both processors is 37122 μs faster than processing it only on the DSP. Although, the ACE16k processor is just used for the morphological operations of the preprocessing phase, we obtained a significant improvement for the total run time of the purification algorithm. Experimental results also show that the performance of the proposed algorithm is good because the total serial run-times of the algorithm are in admissible ranges on both processors.

Process	**DSP**	**DSP+ACE16k**
Filtering and Segmentation	65312 μs	28190 μs (ACE16k)
Feature Extraction	327703 μs	327703 μs (DSP)
Purification of object	42654 μs	42654 μs (DSP)
TOTAL	435669 μs	398547 μs

Table II. Serial Run-Time of the Purification Algorithm

V. Conclusion

In this paper, we have studied on the detection of certain objects in moving images and the extraction of their features by using the Bi-i Cellular Vision System. We have also implemented an algorithm that purifies the remains of each object that we have detected by using the extracted features. The filtering and the segmentation processes of the algorithm are implemented on the ACE16k processor. The other processes like gray level transformation, feature extraction and purification of objects are implemented on the Digital Signal Processor (DSP). After the implementation of this algorithm, each purified object that is detected in a moving image is saved as a separate image file with a predetermined plain background color.

The results given in Table II have shown that when the algorithm is implemented both on the DSP and the ACE16k processors, the run-time is faster than when it is just implemented on the DSP processor.

This algorithm can also be applied to other various moving images. For example, removal of objects that could constitute advertising or that are thought to negatively affect children's psychological development from visual broadcasts are very common applications. In all these cases, the objects that are wanted to be removed from the broadcasts can be determined as the objects to be purified in our implementation.

For a future work, the purification algorithm that we proposed can be improved to develop applications that can totally remove certain objects or replace objects with other objects in a moving image.

REFERENCES

[1] C. Gonzales and R.E. Woods, "Digital Image Processing", Prentice Hall, New Jersey, 2002

[2] T. Acharya and A.K. Ray, "Image Processing: Principles and Applications", Wiley and Sons, 2005

[3] Cs. Rekeczky, B. Roska, E. Nemeth, and F. Werblin, "The Network Behind Spatiotemporal Patterns: Building Low-complexity Retinal Models in CNN Based on Morphology, Pharmacology and Physiology", International Journal of Circuit Theory and Applications, Vol. 29, pp. 197-239, March-April 2001.

[4] F. S. Werblin, T. Roska, and L. O. Chua, "The Analogic CNN Universal Machine as a Bionic Eye", International Journal of Circuit Theory and Applications, Vol. 23, pp. 541-569, 1995.

[5] ARENA, P., FORTUNA, L., FRASCA, M., PATANÈ, L. and POLLINO, M., An autonomous mini- hexapod robot controlled through a CNN-based CPG VLSI chip, Proc. of CNNA, 2006, Istanbul.

[6] WARCHOL, W. , WARCHOL, J. B., FILIPIAK, K., KARAS, Z. and JAROSZYK, F. , 1996, "Analysis of spermatozoa movement using a video imaging technique" , Histochemistry and Cell Biology, 106(5).

[7] STOFFELS, A., ROSKA, T. And CHUA, L. O., 1997, Object-oriented image analysis for very-low-bitrate video-coding systems using the CNN Universal Machine, Int. J. Circuit Theory Applic., 25, 235-258.

[8] D. Ginhac, J.Dubois, M. Paindavoine, and B. Heyrman, "An SIMD Programmable Vision Chip with High-Speed Focal Plane Image Processing", Hindawi Publishing Corporation EURASIP Journal on Embedded Systems, Vol. 2008, Article ID 961315, 13 pages, 2008

[9] N.LI, D. XU, B. LI "A Novel Background Updating Algorithm Based On the Logical Relationship", Proceedings of the 7th WSEAS International Conference on Signal, Speech and Image Processing, Beijing, China, September 15-17, 2007

[10] P. Kumsawat, K. Attakitmongcol, A. Srikaew, "An Optimal Robust Digital Image Watermarking Based on Genetic Algorithms in Multiwavelet Domain", WSEAS Transactions on Signal Processing, Issue 1, Volume 5, January 2009

[11] H. Furuya, S. Eda, T. Shimamura, "Image Restoration via Wiener Filtering in the Frequency Domain", WSEAS Transactions on Signal Processing, Issue 2, Volume 5, February 2009

[12] A. V. Baterina, C. Oppus, "Image Edge Detection Using Ant Colony Optimization", ", WSEAS Transactions on Signal Processing, Issue 2, Volume 6, April 2010

[13] A. Morar, F.Moldoveanu, A.Moldoveanu, V.Asavei, A. Egner, "Medical Image Processing in Hip Arthroplasty", WSEAS Transactions on Signal Processing, Issue 4, Volume 6, October 2010

[14] L. O. Chua and L. Yang, " Cellular neural networks: Theory and applications", IEEE Trans. on CAS, vol. 35 no. 10, pp.1257–1290, 1988

[15] T. Roska and L. O. Chua, "The CNN universal machine: an analogic array computer", IEEE Trans. on CAS-I, vol.. 40 no.3, pp. 163–173, 1993

[16] T. Roska and A. Rodriguez-Vazquez, "Towards visual microprocessors", Proceedings of the IEEE, vol. 90 no.7, pp. 1244–1257, 2002

[17] A. Zarandy and C. Rekeczky, "Bi-i: a standalone ultra high speed cellular vision system.", IEEE Circuit and Systems Magazine, vol. 5, no.2, pp. 36–45, 2005

[18] L. O. Chua and T. Roska, "Cellular neural networks and visual computing Foundation and applications", Cambridge University Press, 2004

[19] AnaLogic Computers Ltd http://www.analogic-computers.com /Support

[20] Eutecus Inc: http://www.eutecus.com/, Berkeley 2005.

[21] A.R. Vazquez, G. L. Cembrano, L. Carranza, E.R.Moreno, R.C. Galan, F.J. Garrido , R.D. Castro and S. E. Meana, "ACE16k: the third generation of mixed-signal SIMDCNN ACE chips toward VSoCs", IEEE Trans. CAS-I, vol. 51,no.5, pp. 851– 863, 2004

[22] S. Espejo, R. Carmona, R. Domingúez-Castro, and A. Rodrigúez-Vázquez, "CNN Universal Chip in CMOS Technology", International Journal of Circuit Theory and Applications, Vol. 24, pp. 93-111, 1996.

[23] G. Liñán, R. Domínguez-Castro, S. Espejo, A. Rodríguez-Vázquez, "ACE16k: A Programmable Focal Plane Vision Processor with 128 x 128 Resolution", ECCTD '01 - European Conference on Circuit Theory and Design, pp. 345-348, August 28-31, Espoo, Finland, 2001.

[24] Cs. Rekeczky, T. Roska, and A. Ushida, "CNN-based Difference-controlled Adaptive Nonlinear Image Filters", International Journal of Circuit Theory and Applications, Vol. 26, pp. 375-423, July-August 1998.

Emel Arslan received the B.Sc. and M.Sc. degrees from Trakya University, Edirne, Turkey, and Ph.D. degree from Istanbul University, Istanbul, Turkey, in 2001, 2004 and 2011, respectively. She is currently working as a Computer Engineer in the Research and Application Center for Computer Sciences, Istanbul University. Her research interests are artificial neural networks, natural language processing, image processing applications and intelligent systems.

Zeynep Orman received the B.Sc., M.Sc. and Ph.D. degrees from Istanbul University, Istanbul, Turkey, in 2001, 2003 and 2007, respectively. She has studied as a postdoctoral research fellow in the Department of Information Systems and Computing, Brunel University, London, UK in 2009. She is currently working as an Assistant Professor in the Department of Computer Engineering, Istanbul University. Her research interests are artificial neural networks, nonlinear systems, image processing applications and intelligent systems.

Sabri Arik received the Dipl.Ing. degree from Istanbul Technical University, Istanbul, Turkey, the Ph.D. degree from the London South Bank University, London, UK, and the Habilitation degree from Istanbul University, Istanbul, Turkey. He is now with the Department of Computer Engineering, Istanbul University as a Professor. His major research interests include cellular neural networks, nonlinear systems and matrix theory. He has authored and coauthored some 50 publications. Dr. Arik is a member of the IEEE Circuits and Systems Society Technical Committee of Cellular Neural Networks and Array Computing. He was the recipient of the Outstanding Young Scientist Award in 2002 from the Turkish Academy of Sciences, Junior Science Award in 2005 from the Scientific and Technological Research Council of Turkey and the Frank May Prize (Best Paper Award) in 1996 from the London South Bank University.