Research Journal of Applied Sciences, Engineering and Technology 7(18): 3894-3905, 2014
DOI:10.19026/rjaset.7.748
ISSN: 2040-7459; e-ISSN: 2040-746
© 2014 Maxwell Scientific Publication Corp.
Submitted: November 30, 2013 Accepted: December 24, 2013 Published: May 10, 2014

Research Article

Model Based Design of Video Tracking Based on MATLAB/Simulink and DSP

Chachou Mohamed Yacine, Zhiguo Zhou and ZhiWen Liu
School of Information and Electronics, Beijing Institute of Technology (BIT), Beijing, China

Abstract: The implementation of digital image processing on electronic boards is a current problem. In this study, we present a Model-Based Design of video tracking based on Matlab/Simulink and DSP. The implementation on DSP, of multi-objects detection and tracking algorithms of two kinds of applications inside and outside, is obtained by using automatic code generation that is code composer studio. The transmission and reception of data is realized by a network connection via Ethernet port between DSP and PC. This allows us, in the future, to extend the number of DSP working in parallel and their IP addresses would be generated by a DHCP server.

Keywords: Detection, DSP implementation, kalman filter, merge blobs, model-based design, tracking

INTRODUCTION

With recent advances of computer technology automated visual surveillance has become a popular area for research and development. Machines that see and understand their environment already exist and their development is accelerated by advances both in micro-electronics and in video analysis algorithms.

The video tracking system consists of the hardware system and the software development environment. The hardware system implements the function of image acquisition, storage, display and running of the detection and tracking program. Because, video processing and specially, video tracking, requires a large number of iteration, this leads to a large occupation of memory space. The use of a multi-cores processor (DSP or FPGA) helps to minimize the processing time and so work on real time. The software development environment is for the establishment of algorithm's model and the realization of the automatic code generation. Covering areas as diverse as automotive, aerospace, defense and Mechatronics, Model-Based Design is an efficient methodology to specify, design, simulate and validate real-time physical systems and associated algorithms (http://www. mathworks.com/model-based-design).

Moving video tracking is one of the most important methods in computer science. Recently, there are many approaches for motion detection in a continuous video stream. All of them are based on comparing of the current video frame with one from the previous frames or with something that we'll call background http://www. codeproject.com/Articles/10248/Motion-Detection-Algorithms). The challenge of detecting the movement lies in the ability to identify moving objects regardless of their size, their velocities and their contrasts with the background. For this, many algorithms have been proposed for object detection in video surveillance applications. They rely on different assumptions e.g., statistical models of the background (Wren *et al*., 1997; McKenna and Gong, 1999), minimization of Gaussian differences (Ohta, 2001) minimum and maximum values (Haritaoglu *et al*., 1998), adaptively (Seki *et al*., 2000) and (Koller *et al*., 1994), or a combination of frame differences and statistical background models (Collins *et al*., 1999).

Object video tracking, by definition, is to track objects over a sequence of images and in general, is a challenging problem. Different estimation algorithms are used in literature (Emilio and Andrea, 2011). One of famous algorithms is Kalman filter (Caius *et al*., 2010) and it can be applied to track groups.

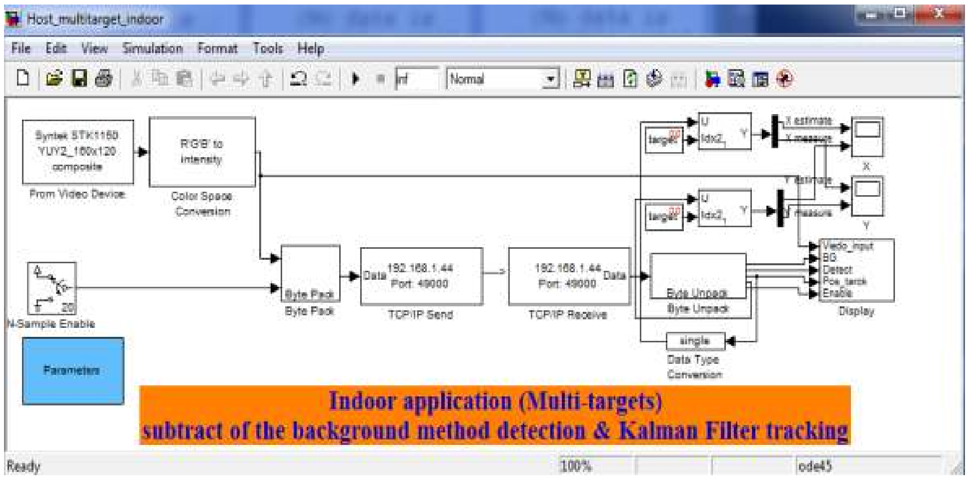
The objective of this study is to do a model-based design of multi-objects video detection and tracking of the indoor and the outdoor applications.

MODEL BASED DESIGN (MBD)

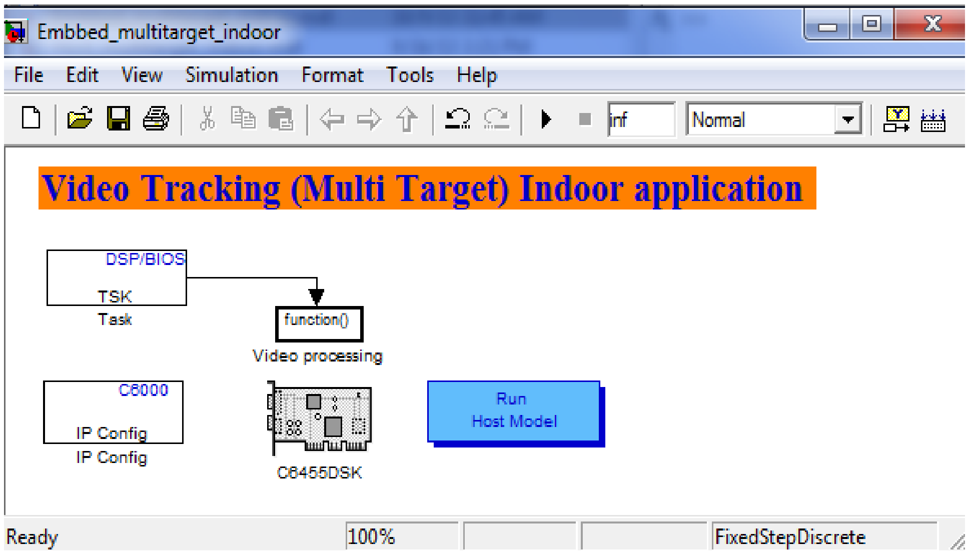
Software development based on the model, automatic code generation, production and testing of computer are more established in recent years. The automotive industry has adopted and successfully deployed these methods in many sets projects around the world. This allowed the reduction of development time and better quality (http://www.mathworks. com/model-based-design) due to the automatic code generation and verification and early validation through simulation. So, the MBD is a methodology applied in designing embedded software.

In our study, a Model-Based Design is developed by using MATLAB/SIMULINK to design the indoor

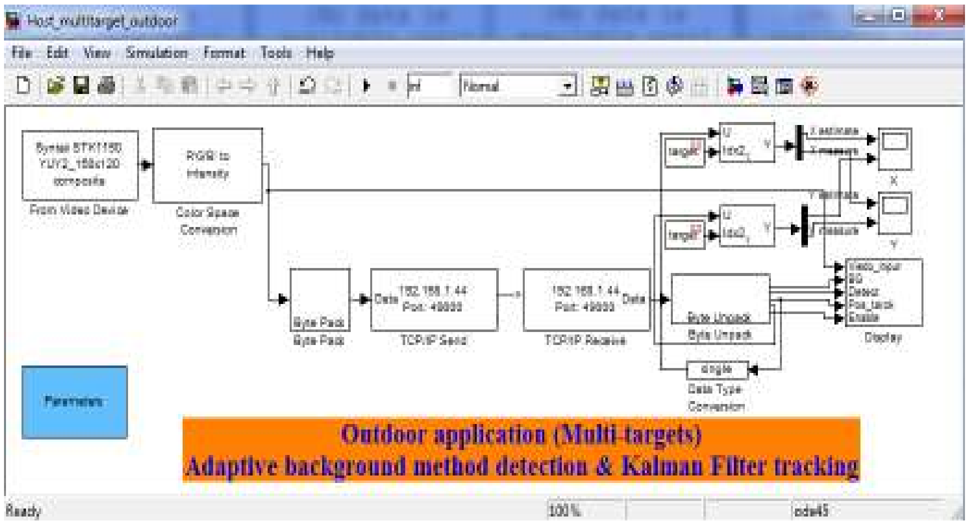
Corresponding Author: Chachou Mohamed Yacine, School of Information and Electronics, Beijing Institute of Technology (BIT), Beijing, China



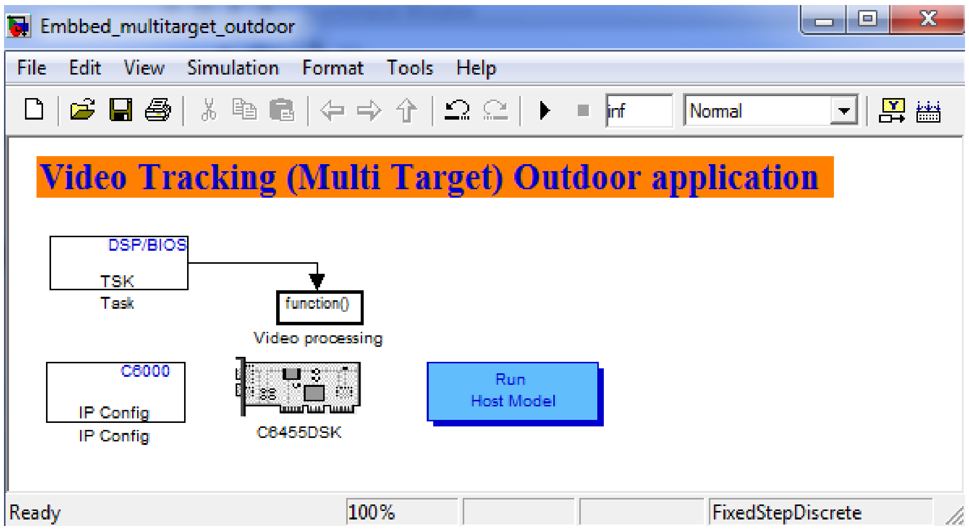
(a) Main software block diagram, indoor


Embbed_multitarget_indoor
File Edit View Simulation Format Tools Help
Video Tracking (Multi Target) Indoor application
DSP/BIOS
TSK
Task
function()
Video processing
C6000
IP Config
IP Config
C6455DSK
Run Host Model
Ready
100%
FixedStepDiscrete

(b) Slave block diagram, indoor

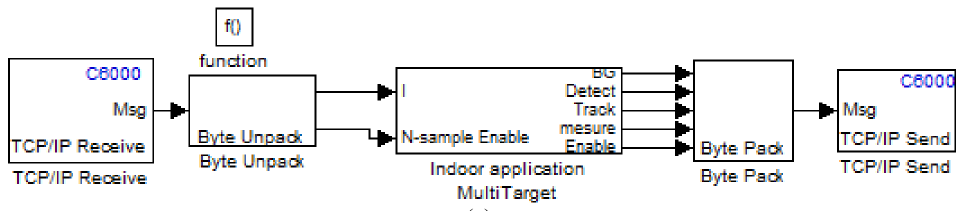


(c) Main software block diagram, outdoor

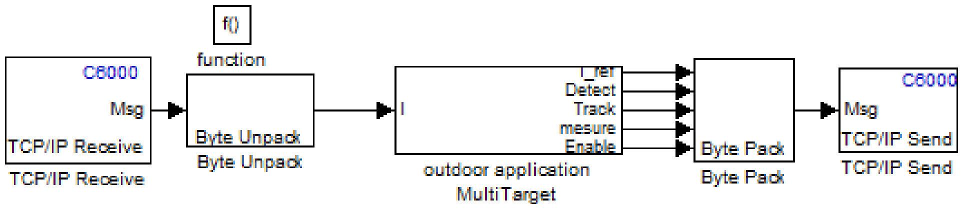


(d) Slave block diagram, outdoor

Fig. 1: Model-based design video tracking, indoor and outdoor applications

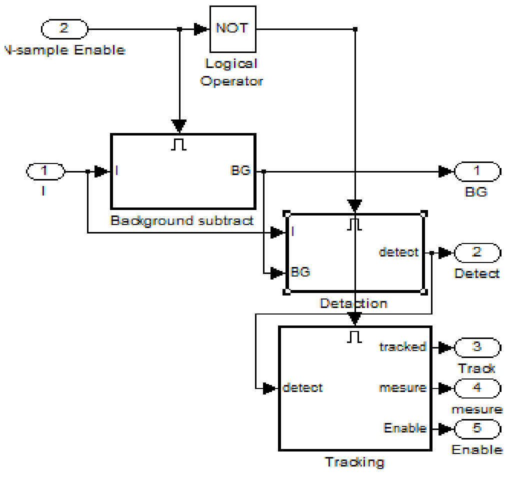


(a)

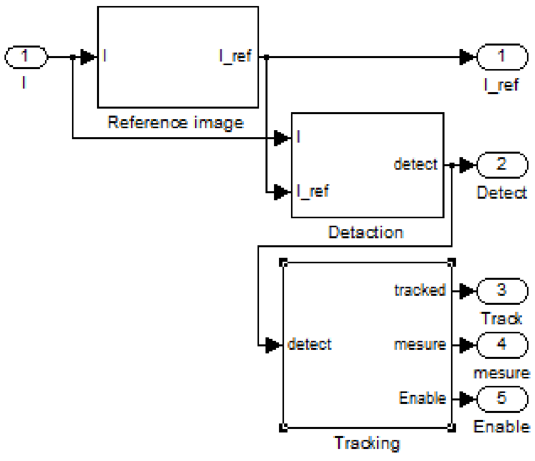


(b)

Fig. 2: Indoor and outdoor applications video processing bloc



(a)



(b)

Fig. 3: Indoor and outdoor detection, tracking algorithms

and outdoor applications and the implementation on DSP is automatically done, using CCS (Code Composer Studio) as a software-to-hardware conversion scheme. Because we made a TCP/IP connection between PC (data reading and data displaying) and DSP TMS320C6455 (data processing), two software architectures are designed, the main software block diagram system and the software architecture of the video processing system.

The main software block diagram system is run online, Fig. 1a and c. The main function of this module is transmitting video data from PC to DSP by TCP/IP network and receiving the video targets detection and tracking from DSP to PC also by the same network connection, in order to display these informations.

The whole software architecture of the video processing system is based on DSP/BIOS real-time operation system, Fig. 1b and d. It is built offline in order to generate the automatic code that will be lead on the DSP. The video processing bloc, Fig. 2, can be divided into three modules: network reception module, video processing module and network sending module. The video processing module contains both of detection and tracking algorithms, Fig. 3. Depending of the application (indoor or outdoor), the architecture can be effectively chooses.

DETECTION AND TRACKING

Detection: Motion detection is to separate on each image in a video sequence, the moving zones of the static zones. There are different detection methods all are represented as shown in the Fig. 4:

In our research, we use differential method, especially reference image, as the way of motion detection. Depending of the application, the reference image can be static background (indoor application) or adaptive background (outdoor application).

Indoor application: Because we have chosen a stationary background method to build the referential image, we first send N-samples enable (N equal to 20) to the DSP (Fig. 1a and Fig. 2a). In this case, unlike to the case of outdoor environment, the reference image can be taken when the scene is empty. From Fig. 3a, we see that, in the indoor application the detection and tracking blocs, in the beginning, are disabled. So we use the first few frames of the video to estimate the background image. After that, the background estimation bloc turns off and the detection and tracking blocs become enabled. In this way, by buffering the first few frames, we get the referential image. The detection uses the absolute difference in pixel values between the input image and the background image to determine which pixels correspond to the moving objects in the video; this allows the separation of the pixels that represent objects from that represent background. Mathematically, the detection is given as following:

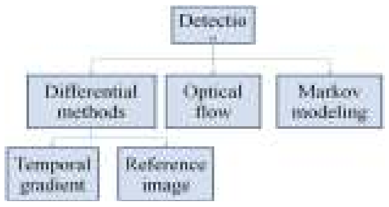


Fig. 4: Motion detection methods

$$D_t = |I_t - I_{ref}| \tag{1}$$

where, D_t is the absolute difference image in pixel values at time t between the input image I_t at the same time and the stationary background I_{ref}. This operation does not often give, to it alone, good results on a real image where the intensity changes are rarely sharp and abrupt. A thresholding operation is indeed necessary to eliminate the noise. So after using the autothreshold bloc and erosion bloc as filters we get our detected objects.

Outdoor application: In contrast to the indoor application, in this case we should choose the adaptive background method to build the referential image, because we have no idea about the scene, that's why, we just need to sent alone, the input video to the DSP processor (Fig. 1c and Fig. 2b). And from Fig. 3b, we see that all reference image, detection and tracking blocs, runs at the same time. So during the all execution of the application, the background is estimated, to be adaptive. The referential image is given by the equation:

$$I_{ref}(p, t+1) = (1-\alpha)I(p, t+1) + \alpha I_{ref}(p, t) \tag{2}$$

where, $I_{ref}(p, t+1)$ and $I(p, t+1)$ is the intensity values of pixel p at the time t+1 in the referential image and the current image, α is the forgetting factor ($\alpha \in]0,1]$). Similar to the indoor application, the difference image is given by the same equation of D_t Eq. (1) and the detection is done also in the same way as the indoor process.

Tracking: The process of estimating over time the location of one or more objects using a camera is referred to as video tracking. Generally, the different types of tracking algorithms, [http://www.mcn.ece.ufl.edu/public/taoran/website/mysite/object% 20 tracking.htm], are shown in the Fig. 5.

In this study, we chose the probabilistic method to do the tracking, because all detected objects are represented by points. Figure 6 shows us that, first in both cases, indoor and outdoor, the same strategy is used to estimate the positions of the detected objects. Second, before doing the merge blobs belonging to the same target and the Kalman filter to the tracking, we start by the blob analysis block to calculate the centroid of the blobs and output the number of blobs found.

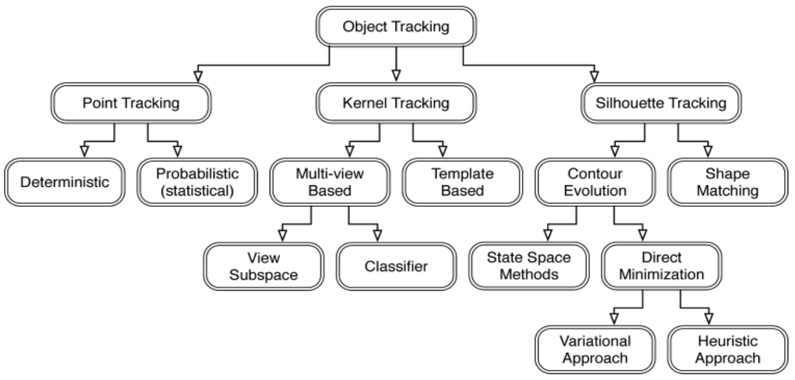


Fig. 5: Types of tracking algorithm

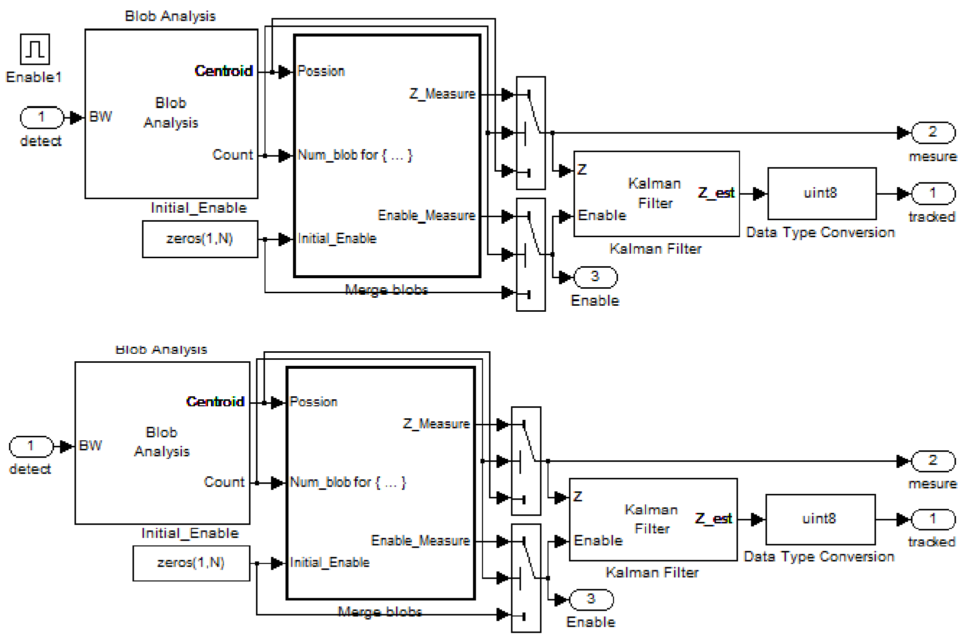


Fig. 6: Indoor and outdoor tracking blocs

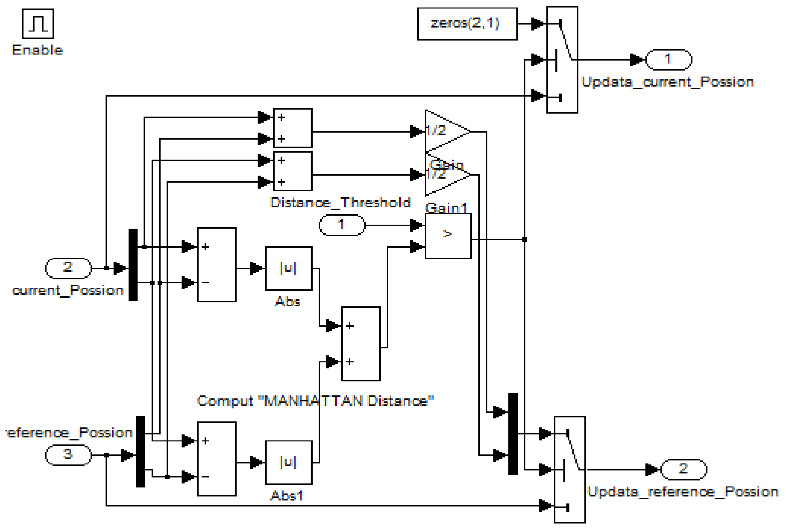


Fig. 7: Merge the centroids

Merge blobs: The aim from this step is to selects a blob for determining its distance to other blobs and to computes, the distance between the selected blob and all remaining blobs. Check the distance between the current and reference target that means, compute the pixel distance between the current and reference positions. In our case, we chose the MANHATTAN distance as a pixel distance. It's given by the equation:

$$d = |x_c - x_r| + |y_c - y_r| \tag{3}$$

where, d is the MANHATTAN distance between the current position represented by pixel $c(x_c, y_c)$ and the reference position represented by pixel $r(x_r, y_r)$. So if the distance is within threshold then we update the reference target by merged blobs and reset the current ones. Otherwise, we keep the values of the blobs. Figure 7 shows us how to merge the centroids if their distances are less than the threshold by computing the MANHATTAN distance and updating the current and reference positions. The threshold is fixed by the user before running the application in the parameters bloc (Fig. 1a and c).

Kalman filter: The Kalman filter block uses the locations of the centroids detected in the previous frames to estimate the locations of these in the current frame. Based on Caius *et al*. (2010), from the basic kinematic equations with constant acceleration and value of the sampling period set to 1, we have:

$$\begin{bmatrix} s_{k,x} \\ s_{k,y} \\ v_{k,x} \\ v_{k,y} \\ a_{k,x} \\ a_{k,y} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 & 0.5 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0.5 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_{k-1,x} \\ s_{k-1,y} \\ v_{k-1,x} \\ v_{k-1,y} \\ a_{k-1,x} \\ a_{k-1,y} \end{bmatrix} \tag{4}$$

Here, the subscripts x and y refer to the direction of the object's position (s), velocity (v) and the constant acceleration (a) in the two-dimensional plane. The above equation can be writing in that way, is because the process model is characterized by the assumption that the present state, x_k, can be related to the past state, x_{k-1}, as follows:

$$x_k = \Phi_k x_{k-1} + W_k \tag{5}$$

where, W_k is a discrete, white, zero-mean process noise with known covariance matrix, Q_k and Φ_k is the state transition matrix which determines the relationship between the present state and the previous one. In our case, Φ_k is deduced directly from Eq. (4) and the process noise covariance is taken like this:

$$Q_k = 0.05 \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{6}$$

The measurement equation is defined as:

$$z_k = H_k x_k + V_k \tag{7}$$

where,

- z_k = The measurement vector
- V_k = Discrete, white, zero-mean process noise with known covariance matrix R_k Eq. (9)
- H_k = The matrix that describes the relationship between the measurement vector z_k and the state vector x_k, called measurement matrix:

$$H_k = \begin{bmatrix} 1 & 0 & 1 & 0 & 0.5 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0.5 \end{bmatrix} \tag{8}$$

Measurement noise covariance:

$$R_k = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{9}$$

The main role of the Kalman filtering bloc is to assign a tracking filter to each of the measurements entering the system from the detection bloc. We know that, the Kalman filter attempts to improve the prior state estimate using the incoming measurement which has been corrupted by noise. This improvement can be achieved by linearly blending the prior state estimate, $\hat{x}_{k-1}$, with the noisy measurement, z_k, in:

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H_k \hat{x}_k^-) \tag{10}$$

With $\hat{x}_k^-$ means the a-priori estimate, K_k is the blending factor. The minimum mean squared error of the estimate is obtained when the blending factor assumes the value of the Kalman gain:

$$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + R_k)^{-1} \tag{11}$$

where, P_k is the state covariance matrix, generally it's a diagonal matrix. The state covariance matrix is determined from the a-priori state covariance matrix as follows:

$$P_k = (I - K_k H_k) P_k^- \tag{12}$$

After that, the Kalman filter makes projections for the next value of k. These projections will be used as the a-priori estimates during processing of the next frame of data. So the projection equations for the state estimation and for the state covariance matrix are as following:

$$\begin{cases} \hat{x}_{k+1}^- = \Phi_k \hat{x}_k \\ P_{k+1}^- = \Phi_k P_k \Phi_k^T + Q_k \end{cases} \tag{13}$$

IMPLEMENTATION AND RESULTS

Here we present the results obtained from our model-based design of video tracking of the indoor and outdoor applications. We note that, the user can select

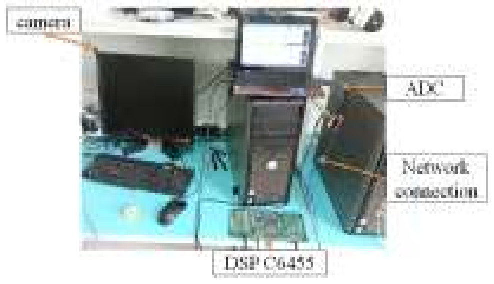


Fig. 8: Experimental equipments

previously, in the parameters bloc (Fig. 1a and c), the number of tracked objects and which target result will be displayed. The Fig. 8 shows the experimental equipments.

We used an ADC to convert the analog signal provided from the camera to the digital signal.

Indoor application: As we said, in the indoor application we used the first few frames to take the background. So from Fig. 9, we see that the background viewer is contained by a fixed image (referential image) in both cases one and multi objects video tracking. Also, we found that the detection and the tracking are made well. It is to emphasize that, the number of targets are previously set at 5 and we can see it from the tracking viewer. Because we chose the second object, noted by B, to display its result, we see (Fig. 10a and b) that if B isn't available then its coordinates (x, y) will be equal to $(0,0)$. But when it is provided (Fig. 10c and d) we get some variation in the coordinates.

We know that, in case of the indoor application, the stationary background causes a problem of noise. This noise is usually the result of a change of intensity. The Fig. 11 illustrates the phenomenon.

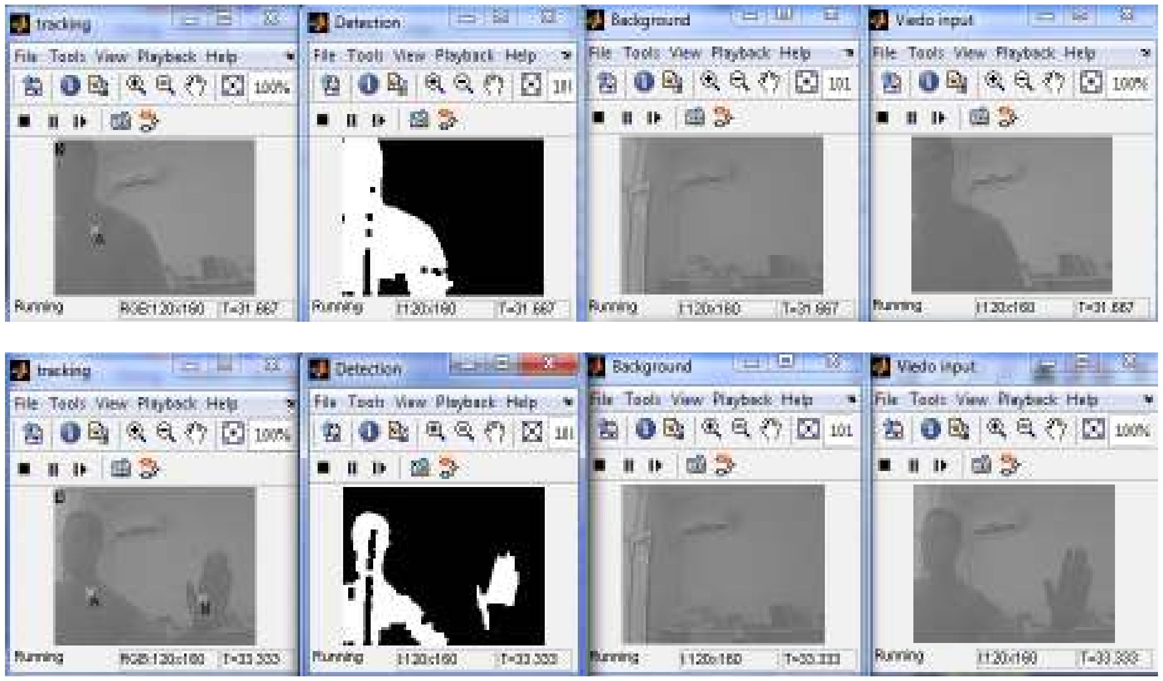
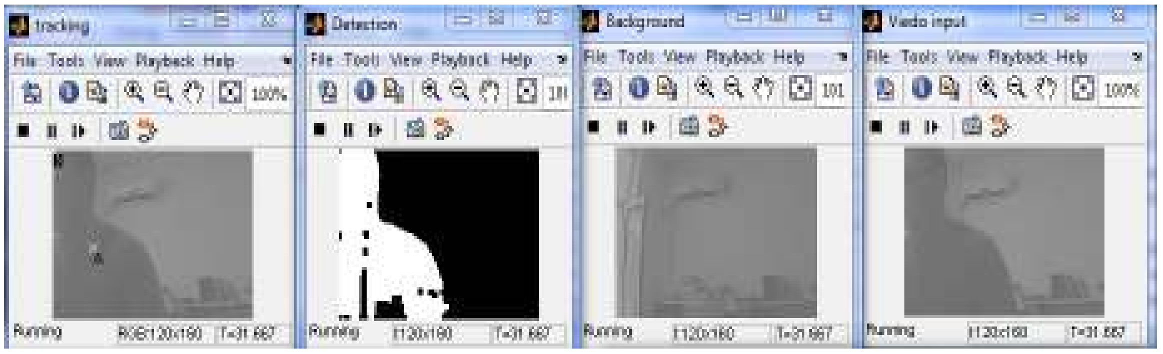
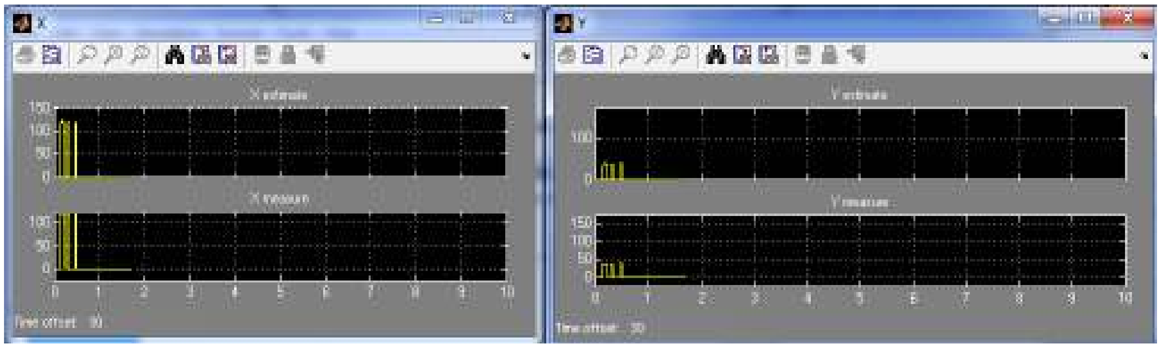


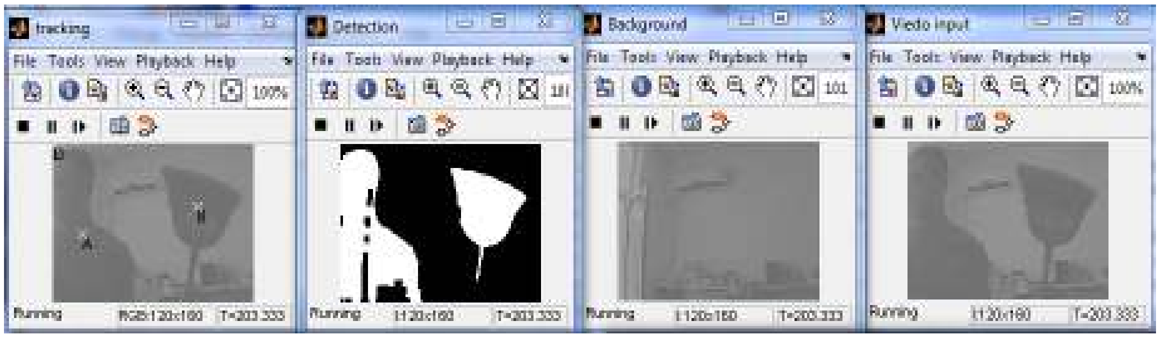
Fig. 9: One and multi objects video tracking indoor application



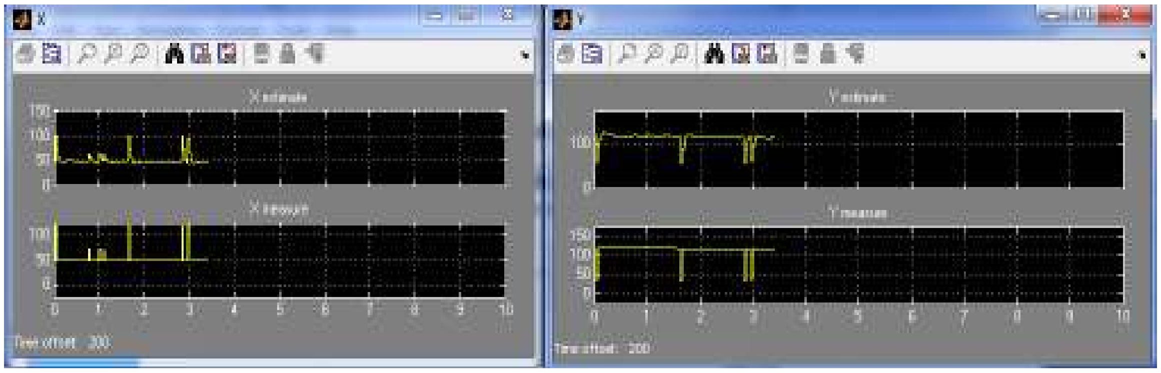
(a) First target



(b) Position estimation



(c) First and second target



(d) Position estimation

Fig. 10: Second target position tracking

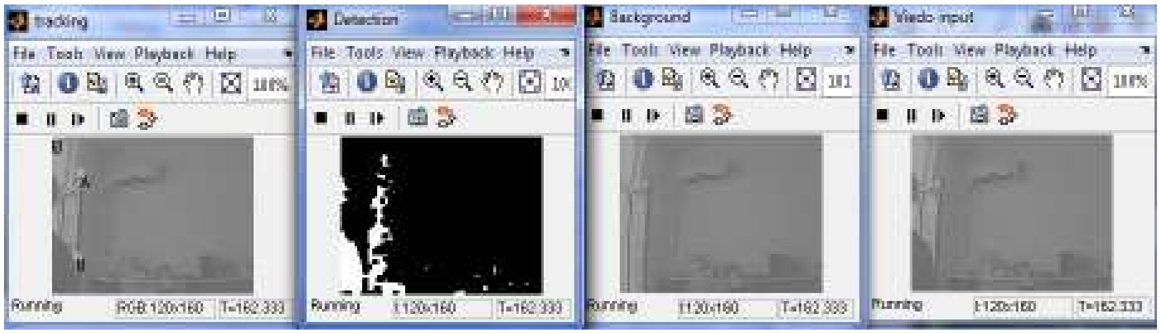
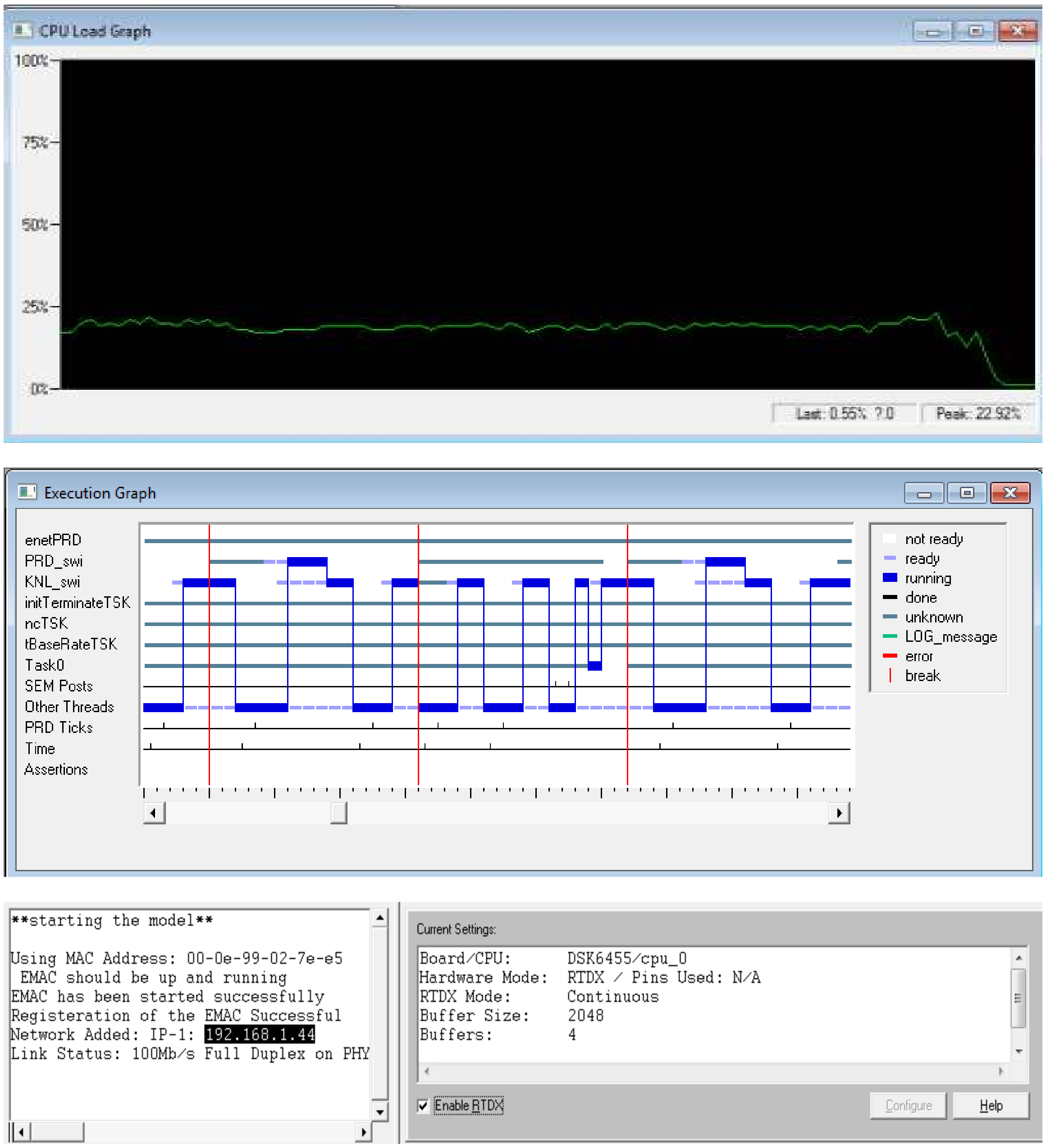


Fig. 11: Video tracking noise indoor application



```
**starting the model**

Using MAC Address: 00-0e-99-02-7e-e5
 EMAC should be up and running
EMAC has been started successfully
Registeration of the EMAC Successful
Network Added: IP-1: 192.168.1.44
Link Status: 100Mb/s Full Duplex on PHY
```

```
Board/CPU:       DSK6455/cpu_0
Hardware Mode:   RTDX / Pins Used: N/A
RTDX Mode:       Continuous
Buffer Size:     2048
Buffers:         4
```

Fig. 12: Results of DSP/BIOS tools of the code composer studio, indoor application

From the DSP/BIOS Tools we profile the DSP software performance. We find, from CPU load graph Fig. 12, that the maximum load is fewer than 25%, so there is still available capacity. We can also see, the time spent executing each task from Execution Graph. And starting the model viewer shows us the information about MAC and IP addresses.

Outdoor application: Unlike to the indoor application, in the outdoor application the background is adaptive. We can confirm that from the background viewer in Fig. 13. We see, from the viewers in that the fixed object (human) is taken as an object of background and the target in move, noted by 1, is detected and tracked. As we also see, in both cases, one and multi objects video tracking, the obtained results are satisfactory. To take an idea about the adaptive background, Figure 14 shows us that, when the motion is fast, the referential image is empty without any additional object. But in case of slow motion, we see in the background viewer, of Fig. 14b, that something is added to the empty background and this becomes the referential image.

Unfortunately, in the outdoor application we also have a noise problem, but in this case the noise phenomenon is provided from both changing of the intensity and updating of the background. The Fig. 15 presents that.

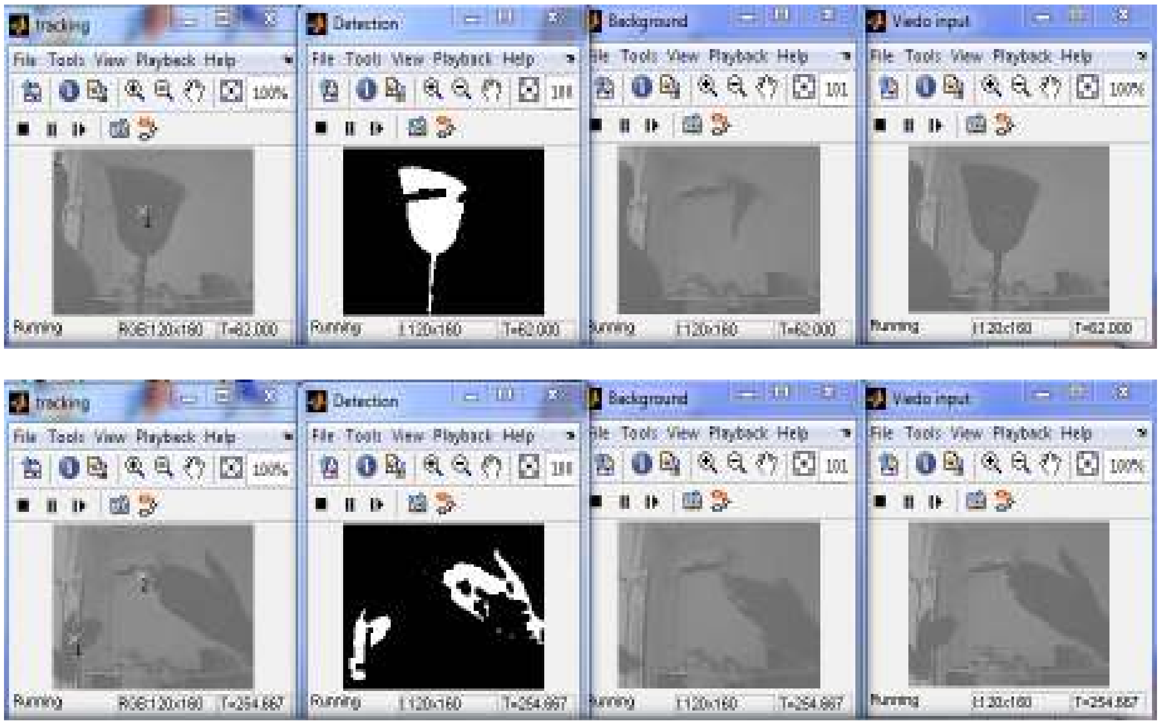
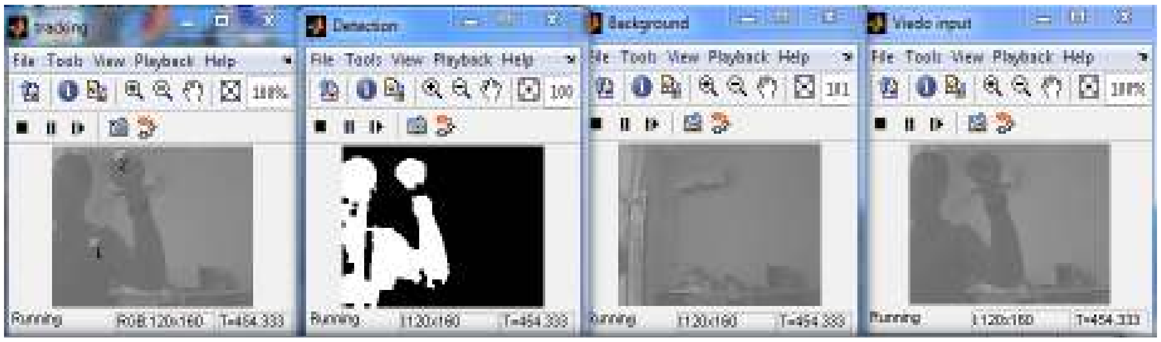
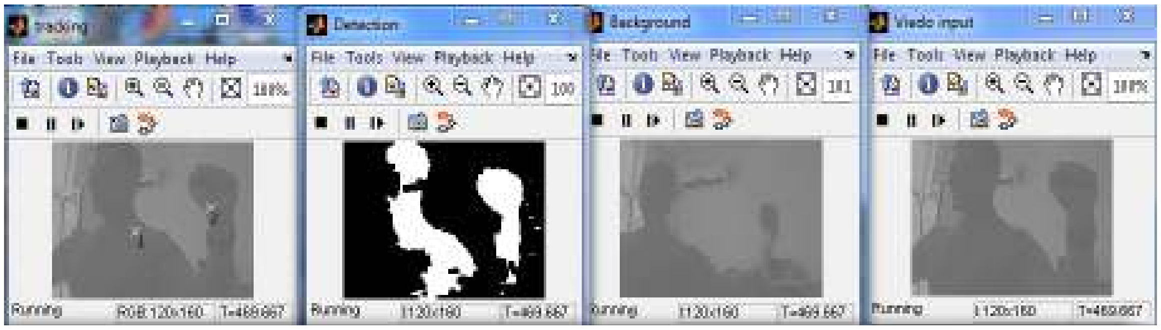


Fig. 13: One and multi objects video tracking outdoor application



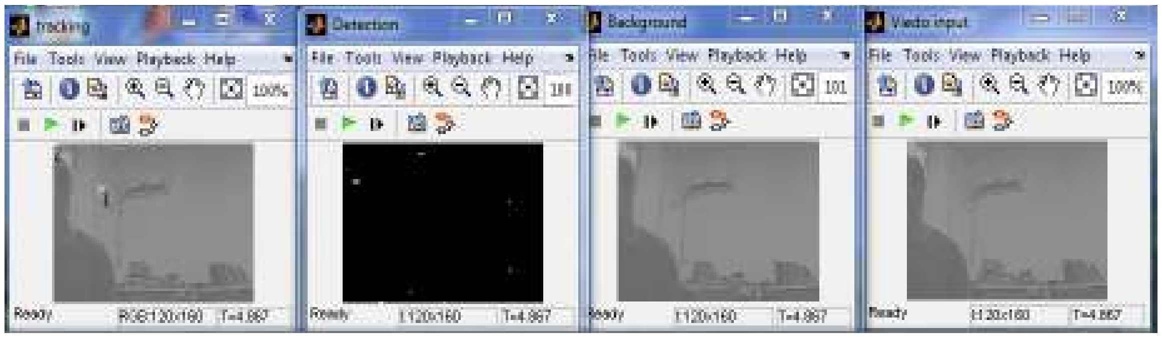
(a): Fast motion



(b): Slow motion

Fig. 14: Adaptive background outdoor application

We also provide, in this case, the DSP software performance under DSP/BIOS. From CPU load graph Fig. 16, we found that a 99% of the processor capacity is being used. In outdoor application, the use of



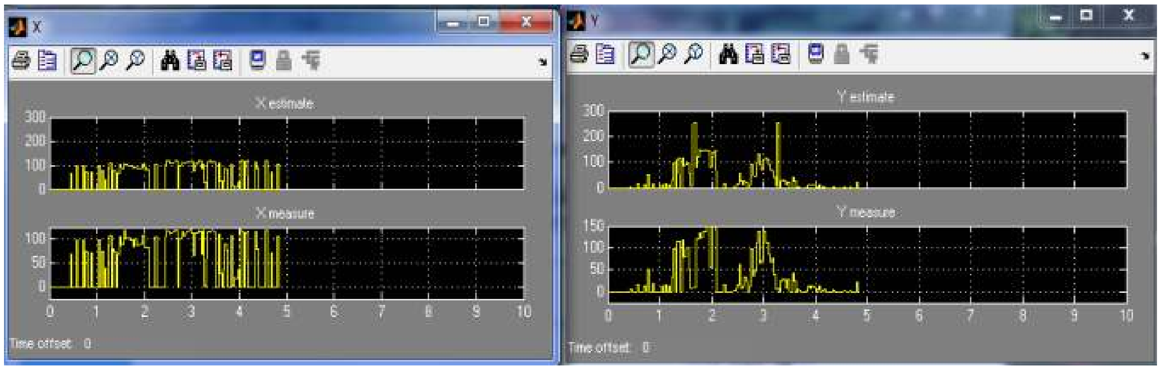
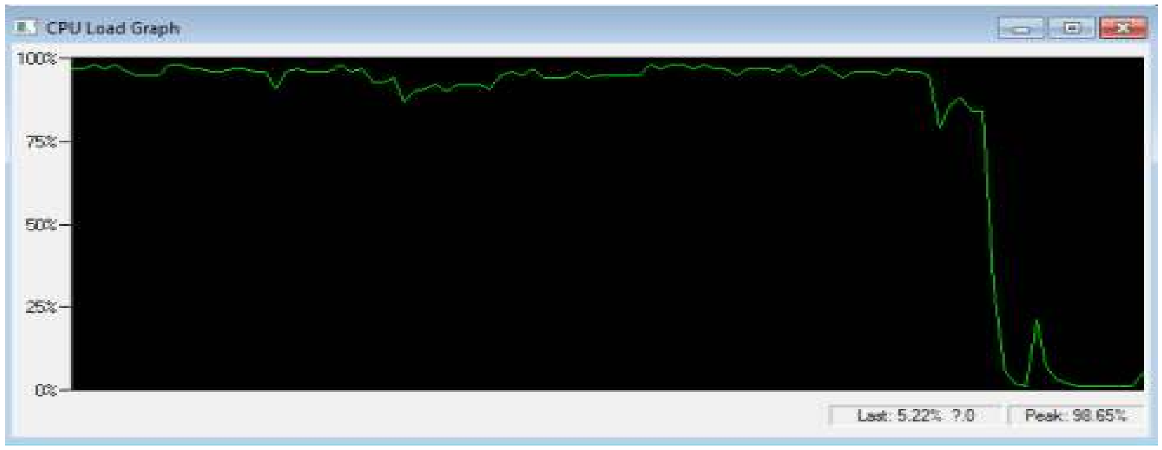


Fig. 15: Video tracking noise outdoor application



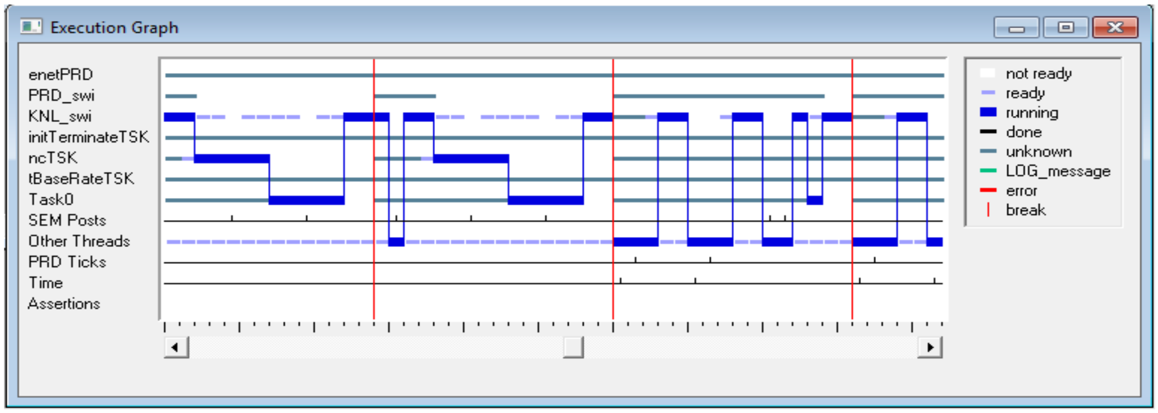


Fig. 16: Results of DSP/BIOS tools of the code composer studio, outdoor application

adaptive background cost more, in capacity performance, than indoor application, this because an additional processing set.

CONCLUSION

In this study a model-based design of multi-objects video detection and tracking of two kinds of applications, indoor and outdoor, was proposed. The use of TCP/IP connection between PC and DSP TMS320C6455, was helpful to send the video to the DSP, in which was implemented our detection and tracking algorithms, for a processing. The Kalman filter was able to correctly process target and to correctly assign a filter to the processed object. After reviewing the results we deduced that our model-based design of video tracking performed quite well showing a moderate consistency in tracking. We also confirmed that the processor capacity consumed in outdoor application was more than the indoor application because of additional processing set of adaptive background. Even the obtained results are acceptable, we seen that in both cases, indoor and outdoor environment, we got a noise problem. To remedy, we can do a normalization of the intensity or outright change the detection strategy.

REFERENCES

Caius, S., C. Cristina and M. Florin, 2010. Kalman filter based tracking in an video surveillance system. Proceeding of the 10th International Conference on Development and Application Systems. Suceava, Romania, pp: 54-58.

Collins, R., A. Lipton and T. Kanade, 1999. A system for video surveillance and monitoring. Proceeding of the American Nuclear Society (ANS), 8th International Topical Meeting on Robotic and Remote Systems. Pittsburgh, PA, pp: 25-29.

Emilio, M. and C. Andrea, 2011. Video Tracking: Theory and Practice. John Wily and Sons Ltd., UK.

Haritaoglu, I., D. Harwood and L.S. Davis, 1998. W4: Who? when? where? What: A real time system for detecting and tracking people. Proceeding of the IEEE International Conference on Automatic Face and Gesture Recognition, pp: 222-227.

Koller, D., J. Weber, T. Huang, J. Malik, G. Ogasawara, B. Rao and S. Russel, 1994. Towards robust automatic traffic scene analysis in real-time. Proceedings of the 12th IAPR International Conference on Pattern Recognition, Conference A: Computer Vision & amp; Image Processing, 1: 126-131.

McKenna, L.S.J. and S. Gong, 1999. Tracking colour objects using adaptive mixture models. Image Vision Comput., 17: 225-231.

Ohta, N., 2001. A statistical approach to background suppression for surveillance systems. Proceeding of the Conference on Computer Vision, pp: 481-486.

Seki, M., H. Fujiwara and K. Sumi, 2000. A robust background subtraction method for changing background. Proceedings of IEEE Workshop on Applications of Computer Vision, pp: 207-213.

Wren, C.R., A. Azarbayejani, T. Darrell and A.P. Pentland, 1997. Pfinder: Real-time tracking of the human body. IEEE T. Pattern Anal., 19(7): 780-785.