Realization of Traffic Video Surveillance on DM3730 Chip

Xin Zhang(✉) and Hang Dong

College of Electronics and Information Engineering,
Tongji University, Dianxin Building, Jiading District, Shanghai, China
{mic_zhangxin, dh}@tongji.edu.cn

Abstract. A general method for traffic video surveillance task involves foreground detecting and moving objects' tracking. The Gaussian mixture model is generally used in detecting foreground and the Kalman filter is used in multi-objects tracking. This paper has implemented a multi-objects tracking system using DM3730 development board as the hardware platform, which is powerful at image processing and analysis. This paper will adopt an Open Computer Vision library (OpenCV) to efficiently implement the overall system. The OpenCV library with a large amount of optimized algorithms in computer vision and machine learning will facilitate the realization of the system. The testing results demonstrate the effectiveness of the system through tracking of vehicles.

Keywords: Multi-objects tracking · DM3730 · OpenCV

1 Introduction

With the urgent demand of constructing a smart city and intelligent transportation, the techniques of traffic video surveillance have been significantly developed and widely used. Especially, we require a sound and excellent digital video surveillance system in the areas where traffic profile is heavy and complicated [1].

An effective approach was proposed for real-time tracking in [2]. The Gaussian mixture model (GMM) was used to distinguish the moving objects. Then the Kalman filter was used to keep track of moving objects in image sequences [2, 3]. This paper has realized the tracking scheme in hardware and can follow and keep track of the moving objects in traffic video correctly.

Initially, a sequence of images is obtained using V4L2 to drive camera to capture images [4]. V4L2 is the application programming interface (API) for video capturing in Linux system which offers unified interfaces for application. Then, this paper has applied GMM in detecting foreground of image sequences [2] and Kalman filter in keeping track of each moving object [6].

The rest of the paper will be organized as follows: In Sect. 2, the hardware platform based on embedded Linux system, where the overall scheme was implemented, will be discussed firstly. In Sect. 3, the flowchart of the scheme proposed in [2] will be explicitly explained. In Sect. 4, the testing results of the realized system will be shown. In Sect. 5, we will conclude the important procedures of the system.

X. Gu et al. (Eds.): MLICOM 2017, Part II, LNICST 227, pp. 402–409, 2018.
https://doi.org/10.1007/978-3-319-73447-7_44

2 Hardware Platform Based on Embedded Linux System

2.1 Hardware Platform

In order to solve the real-time tracking problem for multi-objects, we need a set of processing and controlling system with high speed performance, low power consumption, high speed data I/O, large storage capability and high reliability. The programmable ARM + DSP engine allows multiple signal processing tasks and its video processor is suitable for video processing tasks [4]. Therefore, this paper will use the DM3730 circuit as the hardware platform to implement the tracking scheme.

The Application Processor module (AP module) of DM3730 is a powerful circuit. AP module integrates an ARM Cortex A8 core, a powerful TI C64x + DSP core, a POWERVR SGX graphic accelerator, and a TI TPS65950. The AP Module functional block diagram is shown in Fig. 1 [4].

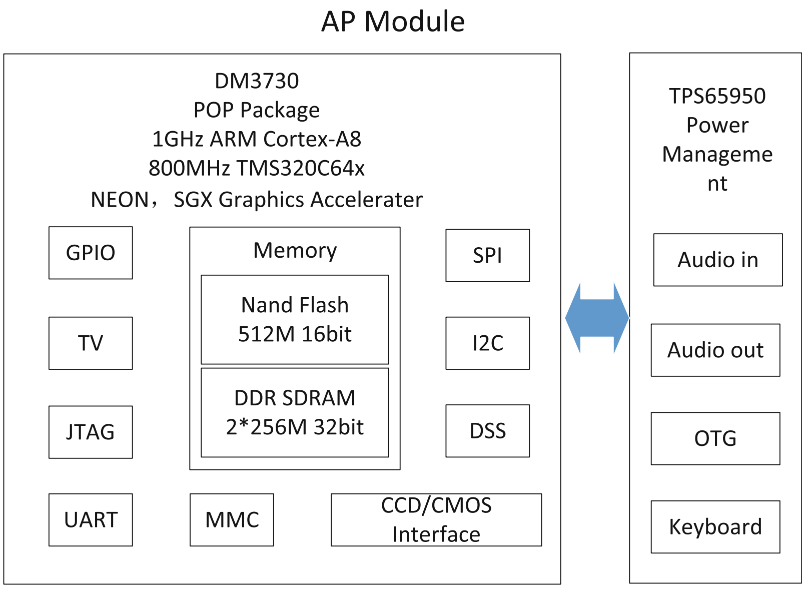

Fig. 1. AP module functional block diagram [4].

2.2 Cross-Compiling for OpenCV Library

OpenCV is an open source computer vision and machine learning software library with more than 2500 optimized algorithms [6]. Since the OpenCV can be run and transplanted on different platforms, this paper will use OpenCV library to implement the tracking scheme. Therefore, we need to cross-compile the OpenCV Library and transplant the compiled files to the kernel of DM3730 so that the optimized algorithms in computer vision can work on the hardware system.

3 The Implementation Scheme

The implementation scheme of the multi-objects detecting and tracking system involves following steps: video capturing, objects detecting, and multi-objects tracking. Initially, the system will capture image sequences using V4L2. Afterwards, GMM will be adopted to detect moving objects. Ultimately, the Kalman filter will be used to keep track of individual moving object. The complete diagram of the implementation scheme is shown in Fig. 2.

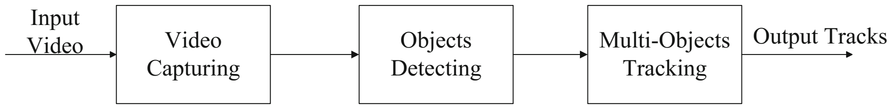

Fig. 2. Implementation scheme diagram.

3.1 Video Capturing Based on V4L2

As mentioned above, V4L2 is an API for video capturing in Linux system, offering explicit model and unified interface for developing camera drivers. Despite the differences among camera devices, the application program can use the same API function. The frequently used API functions of V4L2 are shown in Table 1.

Table 1. The frequently used API functions of V4L2.

Operating functions	Function description
open()	Open a V4L2 device
close()	Close a V4L2 device
ioctl()	Set parameters for device
mmap()	Map kernel space to user space
munmap()	Cancel device memory mapping
read()	Read data from V4L2 device
write()	Write data into V4L2 device

The flowchart of video capturing is shown in Fig. 3.

The principal steps of video capturing are as follows:

(1) Open the camera device and get file descriptor of device. The application can open the video device under blocking mode or non-blocking mode.
(2) Check the functions of the camera device such as video capturing and stream collecting.
(3) Set format of image sequences. The application can set image format manually as MJPEG format, BMP format or YUV format.
(4) Reserve buffer area and use mmap function to map memory to the user space.
(5) Start capturing video and wait for the driver to put images to the user space so as to capture images successively.
(6) Stop capturing video and shut the device.

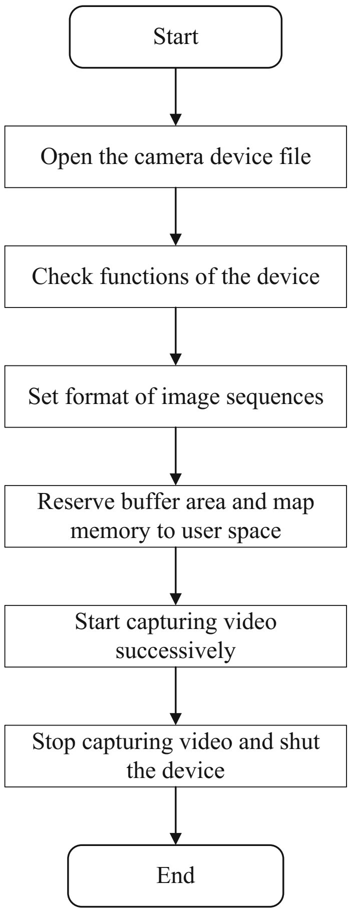
Start
Open the camera device file
Check functions of the device
Set format of image sequences
Reserve buffer area and map memory to user space
Start capturing video successively
Stop capturing video and shut the device
End

Fig. 3. The flowchart of video capturing.

3.2 Objects Detecting Using GMM

The objects detecting module has been implemented using GMM to model each individual pixel. The parameters of background Gaussian distributions will be updated to improve the adaptability of the background changes. The flowchart of objects detecting algorithm is shown in Fig. 4.

The objects detecting module will model individual pixel of each image with K Gaussian distributions [2]. Then the difference between the pixel and the mean value will be checked if it is within 2.5 times of standard deviation in order to determine whether the pixel belongs to background or not. Afterwards, the weights of all Gaussian distributions will be updated respectively. If none of the Gaussian distributions match with the pixel value, the distribution with least probability will be substituted by current pixel value.

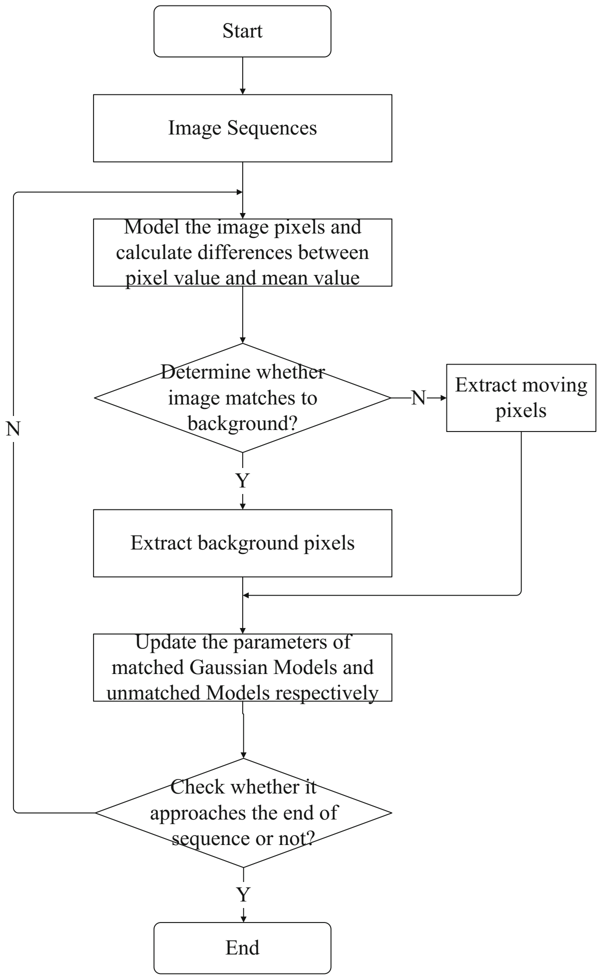
Start
Image Sequences
Model the image pixels and calculate differences between pixel value and mean value
Determine whether image matches to background?
N
Extract moving pixels
Y
Extract background pixels
Update the parameters of matched Gaussian Models and unmatched Models respectively
Check whether it approaches the end of sequence or not?
N
Y
End

Fig. 4. The flowchart of objects detecting.

3.3 Multi-objects Tracking Using Kalman Filter

After successfully extracting foreground and updating background using GMM, we need to keep track of moving objects in image sequences. Once moving objects detected, a Kalman Filter will be assigned to keep track of moving objects. Multi-objects tracking scheme will be implemented using online multi-objects tracking method for maintaining sets of Kalman filters [2]. The flowchart of the implementation scheme is shown in Fig. 5.

The location of each moving object will be recorded and updated in a vector only if the maximum SIFT feature is greater than threshold value [5]. The moving object will be compared with five predicted locations. If none of the locations matches with the threshold value, it means that the moving object has disappeared.

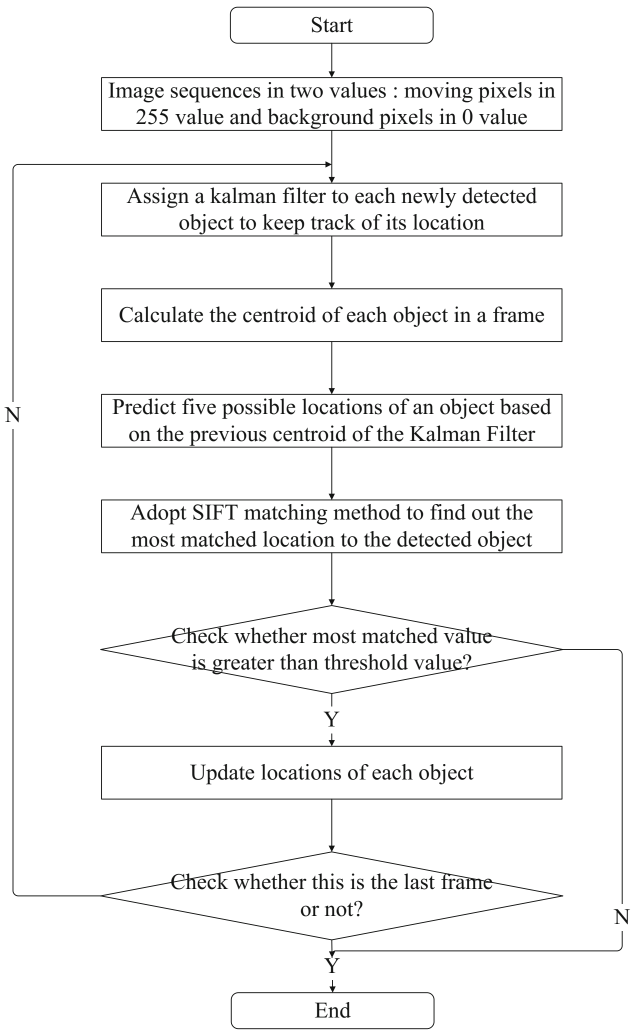
Start
Image sequences in two values : moving pixels in 255 value and background pixels in 0 value
Assign a kalman filter to each newly detected object to keep track of its location
Calculate the centroid of each object in a frame
Predict five possible locations of an object based on the previous centroid of the Kalman Filter
Adopt SIFT matching method to find out the most matched location to the detected object
Check whether most matched value is greater than threshold value?
Y
Update locations of each object
Check whether this is the last frame or not?
N
N
Y
End

Fig. 5. The flowchart of multi-objects tracking.

4 Testing Results

After applying GMM for image sequences in Sect. 3, moving pixels have been detected from video clips. In this section, we will display the hardware platform in Fig. 6 and some testing results in real scenes such as crossroads in Figs. 7 and 8.

There are four tracks of vehicles in the video shown in Fig. 7 and two tracks of pedestrians in Fig. 8.

These four pictures show the tracks existing in video clips. Figure 7(a) shows that a vehicle is driving vertically first and then turning left. Figure 7(b) shows that a vehicle

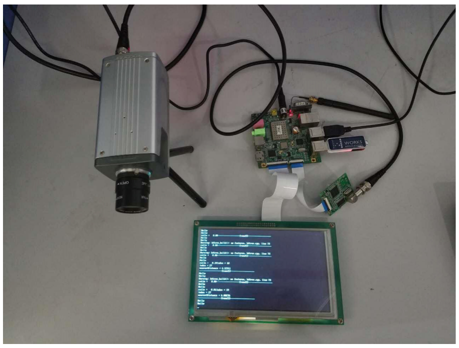

Fig. 6. The hardware platform.

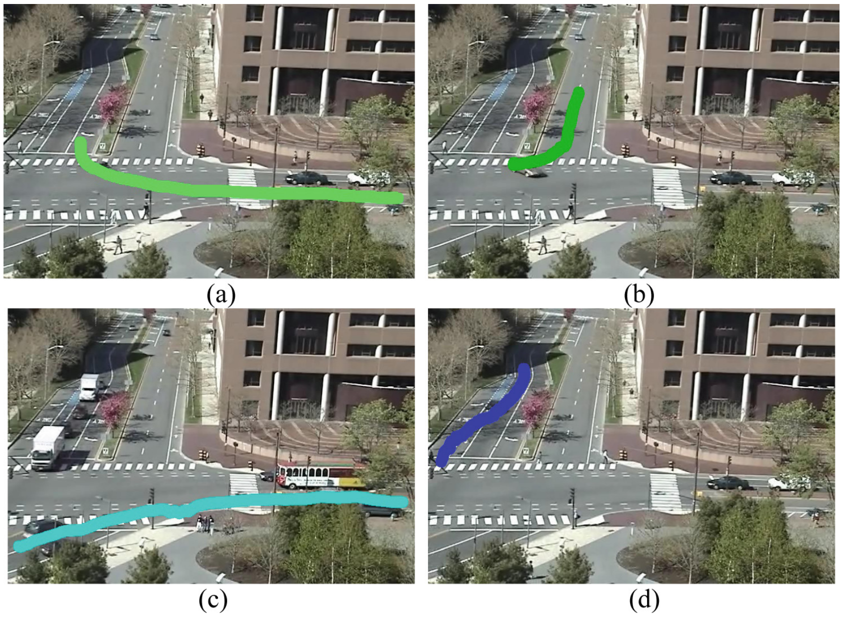

Fig. 7. Tracks of individual vehicle.

is making a turnaround. Figure 7(c) shows that a vehicle is turning right. Figure 7(d) shows that a vehicle changes its lane to the rightmost one preparing to turn right.

The two pictures in Fig. 8 show the tracks of pedestrians in video clips. Figure 8(a) shows that a person is walking across the zebra crossing and Fig. 8(b) shows that a person is walking along the road and turns right.

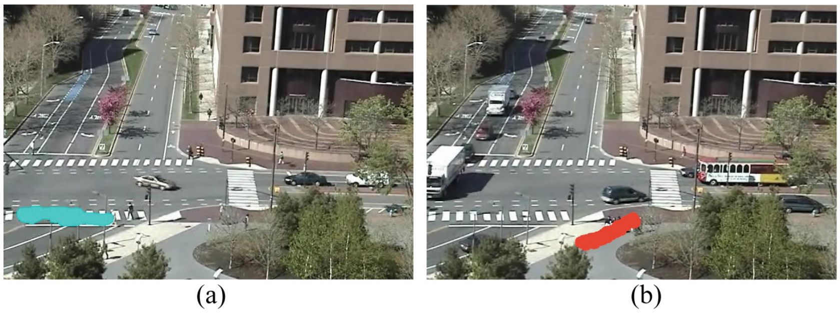

Fig. 8. Motions of individual pedestrian.

5 Conclusions

We have used DM3730 as hardware platform and OpenCV to implement the real-time tracking method proposed in [2]. The procedures of using V4L2 to drive camera capture image sequence, GMM to extract foreground, and Kalman Filter to keep track of moving objects have been explicitly explained. The multi-objects tracking results are presented using the realized system with DM3730 hardware platform.

References

1. Wang, X., Ma, X., Grimson, W.E.L.: Unsupervised activity perception in crowded and complicated scenes using hierarchical bayesian models. TPAMI **31**(3), 539–555 (2009)
2. Stauffer, C., Grimson, W.E.L.: Adaptive background mixture models for real-time tracking. In: Proceedings of IEEE CVPR, pp. 246–252, June 1999
3. Faragher, R.: Understanding the basis of the Kalman filter via a simple and intuitive derivation. IEEE Sig. Process. Mag. **29**(5), 128–132 (2012)
4. Varfolomieiev, A., Lysenko, O.: An improved algorithm of median flow for visual object tracking and its implementation on TI OMAP. In: Proceedings of EDERC, pp. 261–265, September 2012
5. Sakai, Y., Oda, T., Ikeda, M., Barolli, L.: An object tracking system based on SIFT and SURF feature extraction methods. In: INWC, pp. 561–565, September 2015
6. Li, D., Liang, B., Zhang, W.: Real-time moving vehicle detection, tracking, and counting system implemented with OpenCV. In: Proceedings of IEEE ICIST, pp. 631–634, April 2014