An Efficient DSP Implementation of Real-time Stationary Vehicle Detection by Smart Camera at Outdoor Conditions

Antony Louis Piriyakumar Douglas, Mayank Prasad, Sunil Gowatham B, Archana Kalyansundar, Vishnu Swaminathan, Rita Chattopadhyay

Siemens Corporate Technology, Siemens Information Systems Limited, 84, Electronics City, Bangalore 560100, India.
Antony.piriyakumar@siemens.com

ABSTRACT

With the advent of third generation powerful smart cameras, the impact of digital video in the areas such as defense and civil surveillances, litigations, transportation systems, etc has been more evident. Coupled with the reduction in hardware costs and improvised image processing algorithms, nowadays it is affordable to design and operate such embedded systems with smart camera. Given the unabated rise in terrorism, number of road accidents and traffic jams, it is pertinent to use such systems to minimize the occurrences. One of the crucial problems, the Stationary Vehicle Detection (SVD) is solved in real time in the paper. Unlike the previous work, which demands ambient lighting conditions such as in tunnel to detect the accident vehicle, the paper solves the problem with natural lighting conditions using adaptive thresholds based on reflectance modeling. For real-time performances, the algorithm is implemented efficiently on a DSP, which drastically reduces the execution time. To the best of our knowledge, there is no algorithm that executes faster than our implementation. The ample results prove the efficiency of the system, which is tested with the traffic videos from Germany and USA. The algorithm can be applied to detect stationary vehicles that may be bomb-loaded or a security threat in real time also.

Keywords: Video processing and analysis, transportation systems, traffic monitoring, stationary vehicle detection, real-time accident and suspicious vehicle detection.

1. INTRODUCTION

With the advent of third generation of powerful smart cameras, lot of interest is evinced on using digital video in the applications such as border surveillance, building security, bomb-loaded vehicle detection, crucial evidences in the litigations, traffic monitoring and control transportation systems, due to their improved performances coupled with round the clock availability and intelligent detections of specified events. The reduction in hardware costs and improvised complex image processing algorithms paved way for designing such systems with smart camera to detect the specific events such as stationary vehicle detection. In the literature, there are ample papers that deal with video surveillances [1] and intelligent traffic monitoring [2] in particular. The fundamental problem of stationary vehicle detection has been solved under ambient lighting conditions only [3]. However, to detect the bomb-loaded stationary vehicle [4] or detect the accident vehicles [5] anywhere on the road instead of only in tunnels, the algorithm needs to be improvised.

The paper is organized as follows. Section 2 defines the stationary vehicle detection problem with embedded constraints. The improvised algorithm is presented in section 3. In section 4, the embedded hardware architecture of a smart camera, designed and developed in-house, is explained. Section 5 discusses the efficient implementation of the improvised stationary vehicle detection algorithm on a smart camera with DSP. In section 6, the obtained results are analyzed and section 7 concludes the paper.

2. PROBLEM DEFINITION OF SVD AND EMBEDDED SYSTEM CONSTRAINTS

2.1. Classical definition of SVD

Given a video input, whenever any vehicle is stationary over a stipulated time, the presence and location of the stationary vehicle is found as the output.

2.2. Mathematical formulation of SVD

Let the Image at time t be $I_t$. After $t+k$ time, the images are averaged as, $\mu_{od} = \text{Mean Observation Distribution} = \text{mean of } I_j \text{ for } j = t, \ldots, t+k$.

$\sigma_{od} = \text{Standard deviation of } I_j \text{ for } j = t, \ldots, t+k$.

$f = \text{learning coefficient} = \text{Update factor (can be predefined)} = 0.1 \text{ to } 0.9$

$\mu_{bg} = \text{Mean Background} = (1-f) \mu_{bg} + f.\mu_{od}$. 
\[\sigma_{bg} = \text{Standard deviation} = (1-f) \sigma_{bg} + f \sigma_{od}.\]

\[D_i = \text{Difference Image of OD and BG}\]

\[D_i = g_i \text{ if } |\mu_{od} - \mu_{bg}| > \theta, \text{ Illumination Threshold (predefined)} \text{ else } g_i \text{ for } q \text{ frames (} g_1, g_2 \text{ are gray levels).} \]

\[C = \text{the set of connected components } c_1, c_2, \ldots, c_n \text{ in } D_i.\]

Let \(S_i\) be the size (the number of pixels) of the connected component \(c_i\). If \(S_i > \varphi\), Size Threshold (predefined), Superimpose \(c_i\) on the image \(I_t\) to locate the stationary vehicles.

### 2.3. Real-time and Embedded system constraints

One of paramount factor is the execution in real time. Depending upon the frame rate of the camera (30 fps or 15 fps), the SVD output also must be produced at the similar frame rates. It must be recalled that smart camera has limited memory and processing power. Under such conditions, it is challenge to execute complex image processing algorithms in real time which demand extensive redesigning as presented in the paper with various stages of optimizations.

### 3. IMPROVIZED ALGORITHM FOR SVD FOR EMBEDDED IMPLEMENTATION

The SVD algorithm [3] has 4 steps namely, calculation of the observation distribution (OD), adaptation of the background (BG) unlike complex learning [6], forming connected components (CC) and stationary vehicle detection. However, preprocessing is required as frames (images) from the smart camera are in Bayer Pattern [7] that has to be converted to RGB or monochrome Image, which is called as color conversion here. The smart camera we used captures frames of size 640 * 480 which is downsampled to 320 * 240. This color converted downsampled image is given as input to the SVD algorithm.

#### 3.1. Memory Efficient Calculation of Observation Distribution (OD)

On reading a new frame \((I_t)\) from the smart camera or video, along with the \(k\) previous frames (first \(k\) frames will have zero value), the mean \((\mu_{od})\) is computed. Obviously, the incremental mean reduces the execution time. It has also reduced the space required to store \(k\) frames in memory.

#### 3.2. Computationally Efficient Adaptation of Background Distribution (BG)

After every \(k\) frames, the background will be updated. To update, first the updation factor \((f)\) is calculated as follows, \(f = \alpha \div (1 + e^{(a) \sigma_{od} + (b) \sigma_{bg}})\). The updation factor \((f)\) can also be fixed. Normally \(\alpha = 0.1\) and \(a = 1.0\) as in [3]. As exponential function computation is not embedded systems friendly, \(f\) is fixed as 0.1.

The background updation is done as

\[
\mu_{bg} = \text{Mean Background} = (1-f) \mu_{bg} + f \mu_{od} .
\]

\[
\sigma_{bg} = \text{Standard Deviation} = (1-f) \sigma_{bg} + f \sigma_{od} .
\]

### 3.3. Fast computation of Connected Components

First the difference image \((D_i)\) of OD and BG is found. If the difference is more than the predefined Illumination Threshold \((\theta)\) for \(q\) frames (i.e., it remains stationary), gray value \(g_i\) is assigned else \(g_2\). Using the recursive connected components in eight directions, the connected components \(c_i\) are found. To make it one pass efficiently, once a pixel is connected to a component never the pixel participates in the check for other connected components.

### 3.4. Stationary Vehicle Detection

If the size of the connected component, \(S_i > \varphi\), Size Threshold (predefined), it is concluded that stationary vehicle is present at the location of \(c_i\).

#### 3.5. Adaptive Thresholds using Reflectance Model

One of the major contributions in the paper is the detection of stationary vehicle in non-ambient light conditions also. It is absolutely required for bomb-loaded stationary vehicle detection. This demands proper adaptation of Illumination Threshold \((\theta)\) and Size Threshold \((\varphi)\).

#### 3.5.1. Adaptive Illumination Threshold

As reflectance modeling is a major issue in changing illumination conditions like from dawn to dusk and from bright day to stark night, the illumination threshold \((\theta)\) is modified as follows,

\[
\text{The adaptation factor } \beta = \Sigma (\mu_{od} \text{ new } \Theta \mu_{od \text{ old }}) / \text{ Image size.}\]

Where \(\Theta\) is the pixel wise image division.

The new updated \(\theta = (alb) \times \beta\), where \(a\) is the minimum required gray level difference and \(b\) is the adequate maximum gray level difference. In our implementation \(a = 60\) and \(b = 130\). This ensures that any illumination change does not affect the detection process.

#### 3.5.2. Adaptive Size Threshold

During stark night conditions, almost nothing is visible excepting 2 bright headlights of a vehicle. Obviously, size of the vehicle and size of the headlight vary considerably. So, it is important to change the size threshold \((\varphi)\) also.

As per our experiments conducted, it is logical to presume that the illumination threshold \((\theta)\) will be considerably less compared to the value at the daytime. Using this, the size threshold \((\varphi)\) is updated approximately to the size of the headlight.
4. EMBEDDED HARDWARE ARCHITECTURE OF A SMART CAMERA

The smart camera designed and developed in-house has the Pentium 2 processor with 1GHz and memory of 240 MB. The imaging sensor is CMOS lens with the frame size of 640 * 480. The maximum frame rate is 30 frames per second. The internal communication from the camera is through the IEEE 1394 firewire board. It also has USB, Internet port, RS 232 serial port and one trigger port. Fig.1. shows the hardware architecture of the smart camera in Fig. 2.

The smart camera also has a TI DSP TMS320C6416 with L2 memory of 1 MB, which is programmable, and L3 memory as external memory. The processor communication is through DMA access.

5. DSP IMPLEMENTATION OF STATIONARY VEHICLE DETECTION ON THE SMART CAMERA

The crucial contribution is the efficient implementation of the SVD algorithm in real time on the smart camera with DSP. Initially, to check the correctness and performance of the SVD algorithm, it is implemented on a PC with Intel celeron processor 1.3 GHz, 512 MB main memory and 30 GB hard disk. The experiments were conducted with the video files of traffic flow in USA and Germany.

As the real-time performance is the paramount factor in most applications, the implementation on DSP (TI TMS320C6416) with the smart camera processor is carried out in six stages that are explained in the following subsections.

5.1. Stage 1 – No Optimization
Stage 1 is basically without any optimization of the algorithm. Chiefly, it is meant for comparison and evaluation of various optimization techniques. The modules profiled are 1) Reading a frame 2) Colour Conversion 3) Down sampling 4) Initiate Data Transfer to DSP 5) Trigger the SVD algorithm on DSP to complete 6) Transfer the output data back to CPU.

5.2. Stage 2 – Using L2 Memory
Instead of reading the image data from the external memory, depending upon the available L2 memory space, some of the image data is operated from the L2 memory. This reduced the SVD execution time per frame from 384.5 ms (no optimization) to 148.3 ms.

5.3. Stage 3 – Using Parallel Threads
Parallel threads are implemented such that one thread is executing the SVD on DSP and the other thread is executing rest of modules on CPU. This reduced the total execution time per frame to 118.68 ms.

5.4. Stage 4 – Using Double Buffering
In the double buffering, while one buffer is getting filled up by new frames, data from other filled up buffer is fed as input to SVD algorithm. In addition to the double buffering, instead of down sampling and colour conversion, only the green values of the Bayer pattern are considered. The total execution time is 90.67 ms in this case.

5.5. Stage 5 – Incremental Mean Calculation
Instead of the direct calculation of the mean of k frames, current mean is computed using the incremental mean method. This sharply reduced the SVD part into half resulting the total execution time to be 76.0 ms.

5.6. Stage 6 – Fixed Point Arithmetic
In all the previous stages, SVD algorithm is implemented using floating point arithmetic on a fixed point DSP that made the execution really slow. The implementation of fixed point arithmetic of SVD algorithm resulted in total execution time of 35.7 ms by drastically reducing the effective waiting time of CPU.

6. RESULTS AND ANALYSIS

Table I gives the comparison of the execution times of SVD. It is clearly shown that the DSP implementation has drastically reduced the execution time from 7000 ms [3] to 35.7 ms. Processing various video files from different countries, the average detection is found to be above 95 %. There were some objects which were stationary which are not vehicle have been wrongly identified. The number of misses is very less excepting the cases in which the color of the background and vehicle is nearly the same. The very high percentage detection proves the efficiency and correctness of our approach.

Figure 3 shows how efficiently the DSP implementation of SVD algorithm is optimized that resulted in the reduction of the execution time from 385.5 ms to 35.7 ms as compared to the execution times mentioned in [3]. It may be recalled that same DSP is used. It is obvious from the optimization that the performance is in real time. In our case, it is approximately 28 frames per second as compared to 2.4 frames per second in [3].
Table II compares the actual reduction in execution times of the major modules with and without optimization. It is evident that major reduction is in the SVD part of the whole algorithm, which is executed on the DSP that drastically reduced execution time from 322.0 ms to meager 0.193 ms. Figure 4 presents the stationary vehicle during a cloudy day, Figure 5 during bright day and Figure 6 at stark night. In Figs.4-6, top left is actual image, bottom left is the OD, bottom right is BG and top right is SVD. The results confirm the correctness of the algorithm despite illumination changes executing in real time. Thus, it becomes a classic example of an extent to which image processing can be tuned and its real-time performance on the embedded systems for critical applications.

Table I

Comparison of Execution Times of SVD

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Image Size</th>
<th>Platform</th>
<th>No. of Frames</th>
<th>Execution time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>[3]</td>
<td>720*586</td>
<td>PC</td>
<td>20</td>
<td>7000</td>
</tr>
<tr>
<td>Our-PC</td>
<td>320*240</td>
<td>PC</td>
<td>100</td>
<td>168.12</td>
</tr>
<tr>
<td>Our-DSP</td>
<td>320*240</td>
<td>DSP</td>
<td>20</td>
<td>35.7</td>
</tr>
</tbody>
</table>

Table II

Effect of Optimization on SVD

<table>
<thead>
<tr>
<th>Major Module (# module- not Optimized)</th>
<th>Without Optimization</th>
<th>With Optimization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read a frame #</td>
<td>7.2</td>
<td>7.2</td>
</tr>
<tr>
<td>Colour Conversion (CC)</td>
<td>24.3</td>
<td></td>
</tr>
<tr>
<td>Downsampling</td>
<td>7.0</td>
<td></td>
</tr>
<tr>
<td>Data Transfer to DSP #</td>
<td>13.0</td>
<td></td>
</tr>
<tr>
<td>CPU wait time for SVD execution on DSP</td>
<td>322.0</td>
<td>0.193</td>
</tr>
<tr>
<td>Data Transfer from DSP #</td>
<td>11.0</td>
<td>11.0</td>
</tr>
<tr>
<td>Total Execution time</td>
<td>384.5</td>
<td>35.7</td>
</tr>
</tbody>
</table>

7. CONCLUSION

The stationary vehicle detection problem can be applied to a wide spectrum from traffic monitoring, accident detection to security and surveillance. The crucial problem is the performance in real time under non-ambient light conditions on smart camera. The previous works can not be applied as such to the non-ambient light conditions. In this paper, not only the non-ambient light conditions are solved, but also an efficient DSP implementation on a smart camera is presented. The adaptive thresholds using reflectance models are used to alleviate the problems of non-ambient light conditions. Double buffering, parallel threads, memory management and partitions of the tasks between CPU and DSP resulted in efficient DSP implementation on a smart camera. This reduced the execution time from 416 ms [3] to 35.7 ms. The positive detection rate is near 95% including bright day and stark night. It is expected that stationary vehicle (bomb-loaded /suspicious) within the protected area can be detected using the proposed method in the paper.

8. REFERENCES


[4] [http://security.lifesafety.ca/canada/physical_security/Vehicle_Bombs.htm](http://security.lifesafety.ca/canada/physical_security/Vehicle_Bombs.htm)

