On-board Image Processing in Wireless Multimedia Sensor Networks: a Parking Space Monitoring Solution for Intelligent Transportation Systems

Claudio Salvadori\textsuperscript{1,2}, Matteo Petracca\textsuperscript{3,1}, Marco Ghibaudi\textsuperscript{1,2}, and Paolo Pagano\textsuperscript{3,1}

\textsuperscript{1} Scuola Superiore Sant’Anna, Italy
\textsuperscript{2} CNIT – Scuola Superiore Sant’Anna Research Unit, Italy
\textsuperscript{3} CNIT – National Laboratory of Photonic Networks, Italy

May 16, 2012
Abstract

Wireless Sensor Network (WSN) has been adopted during the years for a large set of applications, such as environmental monitoring, industrial automation, process control, and, more recently, for video surveillance and multimedia streaming applications.

Surveillance applications – like traffic monitoring, vehicle parking control, and intrusion detection – involve monitoring of the environment in order to detect and interpret the activities of relevant objects and their behavior. These applications demand real-time images from the scene in order to collect relevant information. Wireless Multimedia Sensor Network (WMSN) provides a low-cost and flexible solution for distributed video surveillance based on low-power autonomous smart camera nodes.

Image processing in the WMSNs scenario is very challenging due the necessity to store and process huge amount of visual data on energy and resource constrained sensor nodes. To deal with this problem, many researchers have mainly proposed distributed in-network processing and low-complexity computer vision techniques to be executed on-board.

In this chapter we focus on the development of on-board image processing techniques for detecting the occupancy status of parking spaces. The developed techniques are presented as an effective solution for vehicle parking lot monitoring applications in the domain of Intelligent Transportation Systems (ITS). Performance results show as the developed image processing techniques are able to reach 99.92% sensitivity and 95.59% specificity in detecting the parking spaces occupancy status. Moreover, in the chapter a full implementation of the developed algorithms in embedded devices is presented and their overall performance evaluated in terms of execution time and memory occupancy.
0.1 Introduction

Wireless Sensor Networks (WSNs) have been experiencing a rapid growth in recent years due to the joint efforts of both academia and industry in developing this technology. If from one hand the academia is going ahead in developing new innovative solutions looking at enabling sensor network pervasiveness, on the other hand the industry has started to push on standardization activities and real implementations concerning reliable WSN-based systems [1, 2]. WSNs are nowadays envisioned to be adopted in a wide range of applications as an effective solution able to replace old wired and wireless systems which are more expensive and hard to setup because of their necessity of power and connection cables. A reduced set of WSNs applications include climatic monitoring [3, 4], structural monitoring of buildings [5, 6], human tracking [7], military surveillance [8], and, more recently, multimedia related applications [9, 10, 11].

The Wireless Multimedia Sensor Networks (WMSNs) development has been mainly fostered by a new generation of low-power and very performant microcontrollers, able to speed-up the processing capabilities of a single wireless node, as well as the development of new micro-cameras and microphones imported from the mobile phones industry. Along with classical multimedia streaming applications in which voice and images can be sent through the network, pervasive WMSNs, consisting in large deployments of camera equipped devices, may support new vision-based services. By collecting and analyzing images from the scene, anomalous and potentially dangerous events can be detected [12], advanced applications based on human activities [13] can be enabled and intelligent services, such as WMSNs-based Intelligent Transportation Systems (ITS) [14], can be provided.

A successful design and development of vision-based applications in WMSNs cannot be achieved without adopting feasible solutions of the involved computer vision techniques. In such a context state-of-the-art computer vision algorithms cannot be directly applied [15] due to reduced capabilities, in terms of memory availability, computational power and CMOS resolution, of the camera nodes. In fact, since WMSNs usually require a large number of sensors, possibly deployed over a large area, the unit cost of each device should be as small as possible to make the technology affordable. As a consequence of the strong limitations
In hardware capabilities, low-complexity computer vision algorithms must be adopted, while reaching a right trade-off between algorithms performance and resource constraints.

In this work we tackle with the problem of developing low-complexity computer vision algorithms targeted to WMSNs devices for enabling pervasive Intelligent Transportation Systems. To this end, we present a parking space monitoring algorithm able to detect the occupancy status of a parking space while filtering spurious transitions in the scene. The algorithm has been developed by adopting only basic computer vision techniques and its performance evaluated in terms of sensitivity, specificity, execution time, and memory occupancy by means of a real implementation in a WMSN device.
0.2 Smart Cameras for Wireless Multimedia Sensor Networks

The performance of vision-based algorithms targeted to WMSN devices mainly depends on the computational capabilities of the whole smart camera system. In this section of the chapter we provide an overview of the most popular embedded vision platforms in the WMSNs domain, as well as a description of their main hardware and image processing characteristics. A final comparison overview among the described platforms is reported in Table 1.

In the last years, several research initiatives produced prototypes of smart cameras able to perform an on-board image processing. Among the first developed devices must be cited the WiCa [16] camera, developed by NXP Semiconductors Research. The platform is equipped with NXP Xetal IC3D processor based on a SIMD architecture with 320 processing elements and can host up to two CMOS cameras at VGA resolution (640x480). The communication standard adopted to send data through a wireless network is the IEEE802.15.4. WiCa has been adopted for image-based local processing and collaborative reasoning applications.

The Cyclops [17] project is another research initiative aimed at developing an embedded vision platform for wireless multimedia sensor networks. The device is equipped with a low performance ATMega128 8 bit RISC microcontroller with 128 KB of FLASH program memory and only 4 KB of SRAM data memory. The CMOS sensor supports three image formats of 8 bit mono chrome, 24 bit RGB color, and 16 bit YCbCr color at CIF resolution (352x288). The board is not equipped with wireless transceivers, however wireless communications can be enabled in IEEE802.15.4 networks via MicaZ mote. In the Cyclops board, the camera module enables for a whole image processing pipeline for performing demosaicing, image size scaling, color correction, tone correction and color space conversion. The Cyclops effectiveness has been demonstrated in collaborative object tracking applications.

In the MeshEye [18] project, an energy efficient smart camera mote architecture based on the ARM7 processor was designed, mainly targeted to intelligent surveillance application. MeshEye mote has an interesting special vision system based on a stereo configuration of two low resolution low power cameras, coupled with a high resolution color camera. In particular,
the stereo vision system continuously determines position, range, and size of moving objects entering its fields of view. This information triggers the color camera to acquire the high resolution image subwindow containing the object of interest, which can then be efficiently processed. To communicate among peer devices MeshEye is equipped with an IEEE802.15.4 compliant transceiver.

Another interesting example of low cost embedded vision system is represented by the CMUcam3 [19], developed at the Carnegie Mellon University. More precisely, the CMUcam3 is the third generation of the CMUcam series, which has been specially designed to provide an open source, flexible, and easy development platform targeted to robotics and surveillance applications. The hardware platform is more powerful with respect to previous CMUcam boards and may be used to equip low cost embedded systems with vision capabilities. The hardware platform is constituted by a CMOS camera, an ARM7 processor, and a slot for MMC cards. Wireless transceivers are not provided on board and communications can be enabled through mote systems (e.g., IEEE802.15.4 based communications via FireFly mote).

More recently, the CITRIC [20] platform integrates in one device a camera sensor, a XScale PXA270 CPU (with frequency scalable up to 624 MHz), a 16 MB FLASH memory, and a 64 MB RAM. Such a device, once equipped with a standard wireless transceiver, is suitable for the development of WMSNs. The design of the CITRIC system allows to perform moderate image processing tasks in network nodes. In this way, there are less stringent issues regarding transmission bandwidth with respect to simple centralized solutions. CITRIC capabilities have been illustrated by three sample applications: image compression, object tracking by means of background subtraction, and self localization of the camera nodes in the network.

<table>
<thead>
<tr>
<th>Platform</th>
<th>Sensor</th>
<th>CPU</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>WiCa [16]</td>
<td>2 Color CMOS 640x480</td>
<td>Xetal IC3D</td>
<td>Local processing, collaborative reasoning</td>
</tr>
<tr>
<td>Cyclops [17]</td>
<td>Color CMOS 352x288</td>
<td>ATmega 128</td>
<td>Collaborative object tracking</td>
</tr>
<tr>
<td>Mesheye [18]</td>
<td>Color CMOS 640x480</td>
<td>ARM7</td>
<td>Distributed surveillance</td>
</tr>
<tr>
<td>CMUcam3 [19]</td>
<td>Color CMOS 352x288</td>
<td>ARM7</td>
<td>Local image analysis</td>
</tr>
<tr>
<td>CITRIC [20]</td>
<td>Color CMOS 1280x1024</td>
<td>XScale PXA270</td>
<td>Compression, tracking, localization</td>
</tr>
<tr>
<td>Vision Mesh [21]</td>
<td>Color CMOS 640x480</td>
<td>ARM9</td>
<td>Image-based water analysis</td>
</tr>
</tbody>
</table>

Table 1: Platforms characteristics comparison
Finally we cite the Vision Mesh [21] platform. The device integrates an Atmel 9261 ARM9 CPU, 128 MB NandFlash, 64MB SDRAM, and a CMOS camera at VGA resolution (640x480). The high computational capabilities of the embedded CPU permit to compress acquired images in the JPEG format as well as to perform advanced computer-vision technique targeted to water conservancy engineering applications. In-network processing of the acquired visual information may be enabled by means of IEEE802.15.4-based communications.

All reported smart camera devices represent an effective solution for enabling vision-based applications in WMSNs. The general trend in developing such devices is to increase computational capabilities without taking into account power consumption issues, the lowest experienced power consumption among the presented smart cameras devices is bigger than 650 mW [22], still a prohibitive figure for autonomous set-up’s in pervasive contexts.
0.3 Image processing for parking space occupancy detection

Wireless multimedia sensor networks are considered a key technology in enabling pervasive Intelligent Transportation Systems [23]. The use of low-cost smart cameras to detect parking spaces occupancy levels provides an effective and cheaper alternative to state-of-the-art magnetic field sensors installed under the asphalt [24]. In this section of the chapter we present a low-complexity computer vision solution aiming at detecting the occupancy status of a single parking space. The algorithm can be easily instantiated in multiple instances in real smart camera devices dedicated to monitor a set of parking spaces, until reaching the full coverage of the parking lot.

0.3.1 Background subtraction approach in WMSNs

Classical computer-vision approaches for monitoring applications usually consist on background subtraction based algorithms [25]. As a function of the required constraints in terms of frame rate and image size, as well as of the adopted technique to model the background, (e.g., mixture of gaussians, kernel density estimation, etc.) a background subtraction approach can respond to a variety of performance and complexity levels.

A background subtraction based approach, with a frame differencing enforcement, is at the basis of the presented algorithm in which a low-complexity objective has been followed. As already discussed in Section 0.1, state-of-the-art computer vision algorithms cannot be directly applied in a WMSNs scenario due to several smart camera constraints: memory size, computational power, CMOS resolution, and energy consumption. The reduced amount of memory and CMOS capabilities strong impact on image frame resolutions and color depths: feasible resolution values are 160x120, 320x240, and 640x480 pixels, usually in gray scale. The energy consumption constraint is directly related to the maximum allowable frame rate, feasible values are lower than 2 fps (state-of-the-art computer vision algorithms usually work at 25, 30 fps), due to the necessity of increasing the device idle time in which turning off all the peripherals. Regarding the limited computational capabilities these require the development of simple background modeling techniques, strongly related to the developed application. In
our approach a custom background modeling technique is defined with the aim to react to permanent changes in the scene (e.g., luminosity variation) and filter spurious transitions (e.g., once-off variations) while guaranteeing a real-time response.

0.3.2 Parking space status analysis

In order to effectively model the background of the monitored parking space scene, a behavioral analysis regarding the possible status of a parking space must be performed. Considering a parking space identified by a Region Of Interest (ROI), as depicted in Figure 1, three main possible states are possible: full, partially full/empty, and empty. While the full and empty states do not require further investigations, the third one must be better detailed. The car who is parking or leaving the monitored space usually requires several video frames before to complete all the manoeuvrings, thus giving the possibility to slowly move from full to empty or vice versa. In a better explicative way it is possible to call the partially full/empty state as *transition* state. Although the transition state can model all the car manoeuvrings, it can be also used to model possible errors due to car and people passing through the ROI and causing false empty to full transitions.

![Figure 1: Parking spaces identified by their own ROIs.](image)

The full parking and leaving process described above can be modeled by means of the three state Markov chain depicted in Figure 2. In fact, the probability to be in a state at time $i + 1$ depends only by the state at time $i$. This observation can be expressed in mathematical terms as:
\begin{equation}
P\{s_{i+1} = x_{i+1}|s_0 = x_0, s_1 = x_1, \ldots, s_i = x_i\} = P\{s_{i+1} = x_{i+1}|s_i = x_i\},
\end{equation}

where \(s_i\) is the status at time \(i\) and \(x_{i+1}, x_i, \ldots, x_1, x_0 \in \{\text{empty, transition, full}\}\).

Regarding the transition probabilities of the Markov chain, these represent the usage trends of the parking space and can be experimentally evaluated in time windows. The effective transition frequencies can be obtained by means of a ground-truth human analysis, and will be in turn used to measure the performance of a given detection algorithm. Better algorithms give transition values closer to the human ground-truth.

\begin{figure}[h]
\centering
\includegraphics[width=0.8\textwidth]{markov_chain.png}
\caption{Markov chain based parking space model.}
\end{figure}

### 0.3.3 Background modeling

In background subtraction based algorithms the reference background of the scene requires to be updated at run-time to react to luminosity variations while filtering once-off changes. In order to create a background modeling technique feasible to be implemented in a camera network device, simple computer vision techniques have been implemented and applied following the Markov chain behavioral model discussed in Section 0.3.2. A background modeling technique specifically designed for the final application can guarantee good performance while keeping low the computational complexity.

In the proposed parking space monitoring application the developed background model-
ing technique aims at compensating the luminosity variations and once-off changes effects in a predictable way starting from the system state knowledge, thus guaranteeing the background model consistency with respect to the real system state. The background luminosity variation compensation is performed by adopting an exponential forgetting algorithm [26]: each background pixel is updated according to the following equation:

\[ B_{i,j}(t_n) = (1 - \alpha)B_{i,j}(t_{n-1}) + \alpha I_{i,j}(t_n), \]  

where \( B(t_{n-1}) \) is the old background, \( I(t_n) \) is the last acquired frame and \( \alpha \) is the learning rate \( (\alpha \in (0, 1)) \). The reported background update process for luminosity variation is performed only in the stable states of the system, empty and full, while it is avoided in the transition state, to fully control the once-off changes filtering procedure. Hence, when a change in the scene is detected, ROI partially occluded due to manoeuvrings or passing cars, and system move from empty/full to transition the exponential forgetting is not applied until the system moves into a stable state. When a transition in one of the two stable states is considered completed the last acquired image is set as background and the exponential forgetting is enabled again. The background update policy as a function of the system state has been depicted in Figures 3a, 3b, and 3c where the current state is identified by a colored area, green for the states in which the exponential forgetting is performed and blue otherwise, while a transition from a previous state is identified by a red arrow.

0.3.4 Status change detection

In the three states based Markov chain adopted to describe the parking space behavior, and used to define the background modeling logic, a transition from one state to the other is achieved when a change in the scene is detected. In the proposed work the change detection is based on a joint background subtraction (BS) and frame differencing (FD) approach.

In all possible states of the system, both BS and FD are performed. The background subtraction procedure is performed subtracting the last acquired frame to the background
image and counting the difference image pixels \( n_{BS} \) bigger than a \( TH_D \) threshold. In case \( n_{BS} \) is bigger than a \( TH_{BS} \) threshold, a possible change in the system status is detected. The frame differencing is based on the same logic, the last acquired frame is subtracted to
the previous image frame and the number of the difference image pixels ($n_{FD}$) is evaluated against a $TH_{FD}$ threshold.

When the system is in one of the two stable states (i.e., empty and full), the condition $n_{BS} < TH_{BS}$ confirms that the system is in a stable state and the background can be updated by Equation (2). In this case a FD can be used to cross-check the results retrieved by BS: the condition $n_{FD} < TH_{FD}$ confirms a lack of dynamics in the scene. If $n_{BS} > TH_{BS}$ and $n_{FD} > TH_{FD}$ a change in the system status is detected and the state of the system moves from empty/full to transition. The background subtraction output can be seen as a system trigger, enforced by FD, moving from a stable state to the transition one.

Once the system is in the transition state, the FD is used as main metric to move to a stable one. When $n_{FD} > TH_{FD}$, moving objects are still detected (e.g., car manoeuvrings, people going down from the car, etc.) and no possible changes to stable states are considered. When $n_{FD} < TH_{FD}$ is found for a number of frame bigger than a given $TH_{N}$ threshold, the system moves in a stable state decided according to $n_{BS}$. If $n_{BS} < TH_{BS}$ the new state is the same hold before the transition, otherwise a status change has occurred. The FD output can be seen as a system trigger able to move from the transition state to the stable ones: to this end the $n_{FD}$ variable used to detect the event stability must be kept as low as possible to avoid inefficiencies.

The frame differencing enforcement adopted in the change detection logic avoid the use of computational expensive recognition techniques, even if it imposes the use of frame rates relatively high with respect to a car parking dynamic (e.g., 1 or 2 fps instead of one image every minute). The use of frame rates lower than 1 or 2 fps could result in a wrong synchronization between state and background thus giving a wrong output regarding the parking space occupancy status. This situation could happen when a black car exits from the parking space and a white car enters in the same. In case of an excessively high sample time the change will be interpreted as a change in the system state, from full to empty (BS above threshold and FD lower threshold in the next frame) even if the parking spaces is still full. How is depicted in Figure 4 an acquisition time equal to 1 fps is enough to understand the car parking dynamics.
0.3. IMAGE PROCESSING FOR PARKING SPACE OCCUPANCY DETECTION

0.3.5 The confidence index

In this section we describe the logic for deciding whether the parking space is empty or full. The adopted metric is a confidence index (CI) since it describes the probability of a parking space to be full. The CI is evaluated as a function of the time and it is retrieved by the parking space occupancy algorithm. The index is evaluated at the end of each change detection evaluation and then quantized on a 8 bit value in order to reduce the packet payload in a wireless communication. In an application scenario in which several parking spaces are monitored, several CI must be sent together with other possible acquired data (e.g., temperature, light, CO2 level), the use of a tiny amount of bytes for each status notification allows to reduce the transceiver usage, thus saving energy.

Due to the necessity of describing the parking space status according to the three states of the Markov chain the confidence index values range has been divided in three main parts, depicted in Figure 5, and each of them mapped in a possible state of the system. The range

![Figure 5: Confidence index values range and system states](image)
Figure 6: Effect of different types of transitions on CI values
0.3. IMAGE PROCESSING FOR PARKING SPACE OCCUPANCY DETECTION

from the empty state goes from 0 to $T_e$, while the full from $T_f$ to 255, where $T_e$ and $T_f$ are close to 0 and 255 respectively.

Moving from empty to full the CI increases as a broken line from 0 to 255 (as shown in Figure 6), following two different behaviors in the transition state. The change detection procedure splits the transition zone in two parts: a transition unstable zone and a transition stable zone. The transition unstable zone is close to the previous stable state and represents the period of time dedicated to enter or leave the parking space. The transition stable zone, instead, represents the period of time between the end of the car manoeuvrings and the transition to the stable state.

0.3.6 Parking space occupancy detection algorithm pseudocode

In this section we report the algorithm pseudocode while explaining its components in respect of the logic described in the previous sections. The pseudocode reported in Algorithm 1 is applied to a single ROI covering a parking space in the scene.

The first step in the algorithm is an initialization procedure in which the last acquired ROI is used as background and previous frame. At this stage the state of the system must be known and imposed to the algorithm. This initialization step corresponds to a situation in which the device is installed and manually configured for the utilization. While the algorithm is running, for each acquired ROI the background subtraction and frame differencing procedures are performed. When the previous state is a stable state, and the conditions $n_{BS} > TH_{BS}$ and $n_{FD} > TH_{FD}$ occur, the state changes from stable to transition. In all the other cases the state does not change and the background is updated by the exponential forgetting algorithm reported in Equation (2).

When the state is equal to transition, the condition $n_{BS} < TH_{BS}$ means that a spurious event has happened and the state is changed with the last stable one, instead $n_{BS} > TH_{BS}$ confirms a possible status change. In this last case FD is used to evaluate whether the system enters the transition stable zone: when this happens the new state is set and the background updated.
Algorithm 1 Parking space monitoring algorithm pseudocode

```plaintext
/*Initialization step*/
roi = get_ROI();
prev=roi;
bgnd=roi;
state = init_state;
p_state = init_state;
cont = 0;
while 1 do
    /*Get a new image and perform BS and FD*/
    roi = get_ROI();
    n_bs = n_diff_over_th(roi, bgnd);
    n_fd = n_diff_over_th(roi, prev);
    if (state = FULL) or (state = EMPTY) then
        /*Full/empty state analysis*/
        if (n_bs > TH_BS) and (n_fd > TH_FD) then
            p_state = state;
            state = TRANSITION;
        else
            p_state = state;
            state = state;
            bgnd = update_bgnd(roi, bgnd);
        end if
    else if (state = TRANSITION) then
        /*Transition state analysis*/
        if (n_bs > TH_BS) then
            /*Real transition*/
            if (n_fd > TH_FD) then
                /*Transition unstable zone*/
                cont = 1;
            else
                /*Transition stable zone*/
                if (cont > TH_N) then
                    cont = 0;
                    if (p_state = EMPTY) then
                        p_state = TRANSITION;
                        state = FULL;
                        bgnd = roi_copy(roi);
                    else
                        p_state = TRANSITION;
                        state = EMPTY;
                        bgnd = roi_copy(roi);
                    end if
                end if
            end if
        else
            /*Spurious event*/
            state = p_state;
            p_state = TRANSITION;
            cont = 0;
        end if
    end if
    ci = compute_ci(state);
    prev = roi_copy(roi);
end while
```
0.4 Algorithm thresholds tuning and performance evaluation

The parking space status detection algorithm described in Section 0.3 permits to decide whether a parking space is full or empty as a function of several thresholds used for both background subtraction and frame differencing algorithms. In this section we first discuss how to tune the thresholds to reduce possible incorrect decisions, then we show the algorithm performance in terms of sensitivity, specificity, execution time, and memory occupancy by means of a real implementation in a WMSN device.

0.4.1 Algorithm thresholds tuning

The effectiveness of the proposed algorithm can be seen as its ability in reflecting the real behavioral trend of monitored parking spaces. In terms of performance, the algorithm detection capabilities can be measured with respect to real ground-truth values evaluated by means of a human-based analysis, considering that better algorithm performance means detection outputs much more similar to the reference ground-truth. As a consequence, an effective algorithm thresholds tuning process must therefore select the best thresholds to reach detection performance consistent to the human ground-truth. To this end, starting from real images belonging to the IPERDS [27] dataset collected within the IPERMOB [28] project, we first evaluated the real ground-truth of tuning image sequences with a human-based frame by frame process, then we tuned all the algorithm thresholds to make it able to follow the real trend.

Figure 7: Parking spaces considered in the tuning trace.
As previously introduced, the image dataset adopted in the tuning process is the IPERDS, which is basically a collection of gray scale images acquired with a resolution of 160x120 pixels at 1 fps and related to traffic and parking spaces conditions. All the images composing the dataset have been collected by using a real WMSN device equipped with a low-cost camera, hence they have all the necessary characteristics to prototype video streaming and computer vision algorithms targeted to low-end devices. Among all the IPERDS traces related to parking spaces monitoring we selected one characterized by heavy shadows effects. The selected trace, in fact, can be considered the most challenging for the developed algorithm, because false change transitions can be detected in case of shadows in the selected ROI, causing in turn a wrong synchronization between the real status and the algorithm output.

The real ground-truth of the IPERDS trace adopted to tune algorithm thresholds has been evaluated by a human operator with a frame by frame analysis. In order to have a human output in the same range of the algorithm (CI output range) the empty status has been notified with the value 0, the transition with 127, and the full with 255. Two main rules have been imposed in evaluating the ground-truth: only parking cars can trigger status transitions, thus filtering moving people, and a transition ends when people inside the car

![Figure 8: Ground-truth confidence index trend for the tuning trace.](image)
leave the monitored ROI. A snapshot of the adopted tuning trace with the considered parking spaces is reported in Figure 7, while the time behavior ground-truth for each parking space is depicted in Figure 8. As it is possible to see from the plots, all the four parking spaces are characterized by status changes in the considered window time (more than 15 min) with shadows on neighboring parking spaces. Although the time related ground-truth is enough to evaluate the algorithm thresholds, a secondary outcome of the performed analysis is the parking space usage trend model. In fact, considering the frequencies of each event it is possible to evaluate all the probabilities of the Markov chain introduced in Section 0.3.2. Table 2 reports the parking spaces probabilities for the selected tuning trace.

<table>
<thead>
<tr>
<th>Parking space ID</th>
<th>$P_e$</th>
<th>$P_t$</th>
<th>$P_f$</th>
<th>$P_{et}$</th>
<th>$P_{ft}$</th>
<th>$P_{te}$</th>
<th>$P_{tf}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>P11</td>
<td>0.779</td>
<td>0.019</td>
<td>0.198</td>
<td>0.002</td>
<td>0.000</td>
<td>0.001</td>
<td>0.001</td>
</tr>
<tr>
<td>P12</td>
<td>0.657</td>
<td>0.008</td>
<td>0.331</td>
<td>0.002</td>
<td>0.000</td>
<td>0.001</td>
<td>0.001</td>
</tr>
<tr>
<td>P13</td>
<td>0.367</td>
<td>0.001</td>
<td>0.630</td>
<td>0.001</td>
<td>0.000</td>
<td>0.000</td>
<td>0.001</td>
</tr>
<tr>
<td>P14</td>
<td>0.000</td>
<td>0.000</td>
<td>1.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
</tbody>
</table>

Table 2: Ground-truth transition probabilities for the tuning trace.

Starting from a human-based ground-truth it is possible to tune all the algorithm thresholds by means of a comparison with the algorithm output. Although the thresholds introduced in Section 0.3.4 are four, only two of them must be properly tuned: $TH_D$ and $TH_N$. The two remaining thresholds, $TH_{BS}$ and $TH_{FD}$, are dependent on $TH_D$, hence it is possible to set them equal to a portion of the ROI area while tuning $TH_D$ appropriately. The $TH_{BS}$ threshold has been imposed equal to $1/4$ of the ROI area due to its requirements in detecting changes in the scene to trigger state transitions, while $TH_{FD}$ has been imposed equal to $1/8$ of the ROI area due to its requirements in guarantee event stability. The $TH_D$ and $TH_N$ have been jointly varied in the range from 50 to 60 and from 1 to 15 respectively, while evaluating the ground-truth similarity trend. In mathematical terms, the tuning procedure consists in finding from a set $\mathcal{I}$ of possible $TH_D$ and $TH_N$ thresholds combinations the pair $TH = (TH_D, TH_N) \in \mathcal{I}$ which minimizes the difference of the algorithm output from the human-based ground-truth. As similarity measure between algorithm output and ground-truth we adopted the relation reported in the following:
\[ S = \sqrt{\frac{1}{N} \sum_{k=1}^{N} (G_{gt}[k] - G[k])^2}, \]  

where \( G_{gt} \) is the ground-truth value, \( G \) is the algorithm output with a specific \( TH = (TH_D, TH_N) \) pair, and \( N \) is the total number of image frames. \( S \) is an averaged Euclidean distance among CI outputs where lower values indicate a better similarity between the considered outputs. To thresholds tuning purposes the similarity \( S \) has been calculated for all the four parking spaces selected in the tuning trace and then averaged among them in order to have an overall comparison value among \( TH = (TH_D, TH_N) \) combinations. A graphical representation of the performed analysis is reported in Figure 9, where for three \( TH_D \) values the similarity \( S \) as a function of \( TH_N \) is shown.

![Graph showing similarity trend analysis](image)

Figure 9: Similarity trend analysis.

The performed similarity analysis shows as \( TH_D \) must be set larger than 55. Adopting the lowest selected value, 50, the problem pointed out at the beginning of this section occurs, so that the parking space P14 loses the state/background synchronization due to luminosity variations caused by shadows (Figure 10). This behavior is confirmed by higher values of \( S \) for \( TH_D \) equal to 50 (Figure 9a). Regarding \( TH_N \), a suitable value coming from the realized analysis is bigger than 1, even if bigger values can be adopted due to a possible increase in spurious transition filtering capabilities with no sensitive differences in similarity. As a consequence of the performed analysis results we selected \( TH_D \) and \( TH_N \) equal to 60 and 5 respectively in order to better filter spurious transition and guarantee a correct state
0.4. **ALGORITHM THRESHOLDS TUNING AND PERFORMANCE EVALUATION**

Figure 10: Wrong state/background synchronization behavior in P14.

stabilization. Algorithm CI outputs with $TH_D = 60$ and $TH_N = 5$ are depicted in Figure 11 for all considered parking spaces, to be noticed the strong similarity with the human-based ground-truth showed in Figure 8.

A validation process regarding the chosen thresholds values can be easily performed by evaluating the Markov chain transition probabilities coming out from the algorithm and comparing them with the one obtained by the human ground-truth analysis. The whole

Figure 11: Algorithm confidence index trend for the tuning trace ($TH_D = 60, TH_N = 5$).
algorithm transition probabilities with the adopted thresholds values are reported in Table 3. Comparing such results with the one reported in Table 2 by means of the overall Euclidean distance between the vector of ground-truth probabilities and the one of the algorithm probabilities, the distances for the considered parking spaces are minimum: 0.019 for P11, 0.004 for P12, 0.007 for P13, and 0.000 for P14. Moreover, must to be noticed that the differences among the probabilities in Table 2 and Table 3 are minimum for the stable states ($P_e$ and $P_f$), while the biggest differences are reached in the transition state ($P_t$) where the human ground-truth is substantially different from the algorithm output.

<table>
<thead>
<tr>
<th>Parking space ID</th>
<th>$P_e$</th>
<th>$P_t$</th>
<th>$P_f$</th>
<th>$P_{et}$</th>
<th>$P_{ft}$</th>
<th>$P_{te}$</th>
<th>$P_{tf}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>P11</td>
<td>0.780</td>
<td>0.005</td>
<td>0.211</td>
<td>0.002</td>
<td>0.000</td>
<td>0.001</td>
<td>0.001</td>
</tr>
<tr>
<td>P12</td>
<td>0.656</td>
<td>0.005</td>
<td>0.333</td>
<td>0.003</td>
<td>0.000</td>
<td>0.002</td>
<td>0.001</td>
</tr>
<tr>
<td>P13</td>
<td>0.366</td>
<td>0.005</td>
<td>0.623</td>
<td>0.002</td>
<td>0.001</td>
<td>0.001</td>
<td>0.002</td>
</tr>
<tr>
<td>P14</td>
<td>0.000</td>
<td>0.000</td>
<td>1.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
</tbody>
</table>

Table 3: Algorithm transition probabilities for the tuning trace ($TH_D = 60, TH_N = 5$).

0.4.2 Algorithm occupancy status detection performance

The detection performance of the developed algorithm have been evaluated by means of simulations using an algorithm implementation suitable to run in real embedded devices. By adopting the thresholds values selected in Section 0.4.1 the algorithm sensitivity and specificity [29] have been evaluated using IPERDS traces characterized by high movements in the scene with luminosity variations and regular shadows.

Figure 12: Parking spaces considered in a testing trace.
0.4. ALGORITHM THRESHOLDS TUNING AND PERFORMANCE EVALUATION

Figure 13: Ground-truth confidence index trend for a testing trace.

The sensitivity of the algorithm has been evaluated as the number of true positive events over the sum of true positive and false negative events, as reported in Equation 4, and indicates the ability of the algorithm of correctly detecting full state events. Regarding the

Figure 14: Algorithm confidence index trend for a testing trace \((TH_D = 60, TH_N = 5)\).
specificity, this performance parameter has been evaluated as the number of true negative events over the sum of true negative and false positive events, see Equation 4, and indicates the ability of the algorithm of correctly detecting empty state events. The performance of the developed algorithm in respect of these two metrics are: 99.92% for the sensitivity and 95.59% for the specificity. As it is possible to see from the reported results the proposed algorithm with a properly tuning process can correctly detect the status of a parking space both in full and empty conditions.

\[
Sensitivity = \frac{TP}{TP + FN}, \quad Specificity = \frac{TN}{TN + FP}
\] (4)

Considering a testing trace in which four parking spaces are monitored, Figure 12, a graphical comparison analysis between the human-based ground-truth and algorithm output, Figure 13 versus Figure 14, confirms the algorithm capabilities in detecting the effective parking spaces occupancy status.

0.4.3 Algorithm performance in the SEED-EYE camera network

The algorithm performance in terms of execution time and memory occupancy has been evaluated by means of a real implementation in the SEED-EYE [30] camera network device. The SEED-EYE board, depicted in Figure 15, is an innovative camera network device developed by Scuola Superiore Sant’Anna and Evidence within the IPERMOB [28] project. The board is equipped with a Microchip PIC32 microcontroller working at a frequency of 80 MHz and embedding 512 KB of Flash and 128 KB of RAM. The device mounts a CMOS camera which can be programmed to acquire images at various resolutions (up to 640x480) and frame rates (up to 30 fps). As network interfaces an IEEE802.15.4 compliant transceiver and an IEEE802.3 module have been installed in order to enable wireless communications among peer devices and possible connections to backhauling networks. The SEED-EYE board has been specifically designed to support high demanding multimedia applications while requiring low power consumption during image acquisition and processing. Performance evaluation executed in laboratory has shown that the board can acquire and process 160x120 images
at 30 fps while experiencing a maximum power consumption equal to 450 mW when all the peripherals are activated. Lower power consumption values can be achieved reducing the image acquisition frame rates.

To evaluate the algorithm execution time in the SEED-EYE camera network device the algorithm implementation adopted in Section 0.4.2 has been ported as a custom application on the top of the Erika Enterprise (EE) [31, 32] OS, an innovative real time operating system for small microcontrollers that provides an easy and effective way for managing tasks. More in detail, Erika is a multi-processor real-time operating system kernel, implementing a collection of Application Programming Interfaces (APIs) similar to those provided by the OSEK/VDX standard for automotive embedded controllers. The algorithm execution time on top of EE OS has been measured performing several execution runs while considering a single parking space. The whole performance evaluation results are presented in Figure 16 in terms of execution time distribution. As it is possible to see from the plot the algorithm execution time is not constant due to the priority-based scheduling policies adopted in EE OS giving higher priorities to basic operating system tasks. As overall result, the algorithm shows an average execution time of 1.37 ms with a standard deviation of 0.05 ms.
Regarding the memory occupancy on both Flash and RAM, these values have been obtained by Microchip tools and are equal to 80.5 KB and 26.7 KB respectively for Flash and RAM. The percentage of total required memory is equal to 16.75%.

Figure 16: Execution time distribution.
0.5 Conclusions

In this chapter we focus on the development of on-board image processing techniques for detecting the occupancy status of parking spaces. The developed techniques are presented as an effective solution for vehicle parking lot monitoring applications in the domain of Intelligent Transportation Systems. Starting from the adoption of classical background subtraction techniques we propose a modeling background process specially designed for the considered application in order to follow a low-complexity approach. Moreover, in the chapter the process for appropriately tuning all the algorithm parameters is exhaustively presented starting from a human-based ground-truth behavioral comparison. In a real implementation on a camera network device the developed algorithm can reach 99.92% sensitivity and 95.59% specificity in detecting the parking spaces occupancy status, while showing an average execution time of 1.37 ms with a memory occupancy of 80.5 KB in Flash and 26.7 KB in RAM.
References


