ABSTRACT
Recent developments in the field of facial expression recognition advocate the use of feature vectors based on Local Binary Patterns (LBP). Research on the algorithmic side addresses robustness issues when dealing with non-ideal illumination conditions. In this paper, we address the challenges related to mapping these algorithms on smart camera platforms. Algorithmic partitioning taking into account the camera architecture is investigated with a primary focus of keeping the power consumption low. Experimental results show that compute-intensive feature extraction tasks can be mapped on a massively-parallel processor with reasonable processor utilization. Although the final feature classification phase could also benefit from parallel processing, mapping on a general-purpose sequential processor would suffice.

Index Terms— Facial Expression Recognition, Low power Smart Cameras, Video Scene Analysis, Parallel Processing.

1. INTRODUCTION
Advances in CMOS technology enable new application areas employing distributed sensing systems. Being part of this category, distributed smart camera systems (DSCS) help to add additional sensory input to a decision making process. Due to the computational-intensive nature of video processing algorithms, the impact of progress in CMOS process technology is more visible on the application of smart cameras. This is demonstrated with the low-cost yet good quality image sensors that are integrated in mobile devices and the advances in highly integrated smart camera modules that form the hardware infrastructure of DSCS.

The research challenges concerning DSCS cover a number of layers ranging from optimal hardware design for the smart camera nodes to developing applications on a network of sensing nodes. With respect to camera systems targeted at analyzing facial expressions, the following challenges need to be addressed:

- Computationally-efficient algorithms for robust facial-expression analysis under varying illumination and viewing angle.

- Optimal trade-off between on-node processing and streaming video for long operation times and small camera size.

- Energy-efficient partitioning of algorithms on the smart camera compute resources.

A number of approaches have been proposed in the field of facial expression recognition with varying performance levels and computational complexity. Methods based on geometric features and appearance models [1] and Gabor filters [2] have been shown to have good recognition rates. Although the Gabor filter based approach appears to be robust in real-world applications, its improved performance comes at the expense of high computational complexity. Recently in [3], application of Local Binary Patterns (LBP) has been proposed for extracting facial features primarily motivated by the low computational complexity for equivalent recognition performance. The complete expression recognition task involves a feature classification stage in which the extracted feature is labeled as one of expression classes (happy, sad, angry, etc.).

Depending on sensor resolution, compute resources on-board the camera and communication means, smart camera architectures take different forms [4, 5, 6, 7]. Despite architectural differences, the common principle in most designs is to bring video processing (analysis) as close as possible to the image sensor thereby reducing the amount of video (data) that would otherwise need to be transferred to a central decision point. Due to increased component and system integration, smart cameras are getting more compact and dissipate less power without sacrificing on functional performance.

In this work, we focus on issues related to mapping LBP-based facial expression recognition algorithms on a smart wireless camera platform. The primary objective is to identify proper partitioning of the algorithm on the compute resources available on the camera node in such a way that the overall power dissipation is kept minimal. Envisaged applications are in power constrained systems such as mobile or re-deployable camera networks. An example of the latter system is customer behavior (emotion) analysis in shopping malls. Since LBP features have been shown to be powerful in representing local and global object textures, the results reported here have relevance in the general sense.
The paper is organized as follows: in section 2, a brief description of current facial expression recognition techniques is given based on the LBP technique. The smart camera platform chosen for the mapping exercise is described in section 3 followed by a discussion on algorithm partitioning and mapping options in section 4. Experimental results are presented in section 5, based on which some conclusions are drawn in section 6.

2. FACIAL EXPRESSION RECOGNITION

Figure 1 shows a block diagram of an automatic facial expression recognition system. The two main stages in the system are: (i) feature extraction and (ii) feature classification. The feature extraction stage involves pre-processing stages such as face localization in the scene and scaling in addition to extraction of a feature of some kind for the region-of-interest (ROI). The feature classification process involves comparing the generated feature vector with vectors selected to represent a set of expressions.

![Facial Expression Analysis](image)

Fig. 1. Facial Expression Analysis

There is active algorithmic research in the area of facial expression recognition with the objective of finding a robust recognition method with modest computational complexity. A recent proposal for facial expression recognition [3, 8] uses features derived from Local Binary Patterns. Originally proposed for texture analysis, LBP based methods have been successfully adopted for face recognition purposes [9]. While being computationally low-cost, LBP based features vectors have been shown to have a good descriptive power with respect to facial expressions. Motivated by these characteristics, this method has been chosen for the mapping exercise in this paper.

For the final feature classification stage, a number of approaches have been investigated in the literature, ranging from simple template matching using weighted Chi-square distance measure [9], to Support Vector Machine (SVM) [10]. In [3] and [8], SVM is shown to have better discriminative power when operating on LBP based feature vectors.

2.1. Feature Extraction

2.1.1. Local Binary Patterns

Originally, the LBP method was introduced for texture analysis using a 3x3 pixel neighborhood [11]. Further research on this topic has lead to kernels with different neighborhood sizes enabling application of LBPs to areas like face recognition [9].

In Figure 2, the generation of an LBP(8,2) code is shown, where 8 neighbors at a circular distance of 2 pixels from the pivot pixel (P) are considered. Bi-linear interpolation on a 2x2 pixel window is used to generate intensity values for samples that do not lie at the center of the grids (those marked with X). The samples located in the green boxes are directly used in the LBP computation.

![LBP Code Generation](image)

Fig. 2. LBP code generation; the binary pattern is built via clock wise scanning with the least-significant bit at A, i.e., (111100101) = (249)10.

Contribution \((C_i)\) of each neighbor \((N_i)\) to the \(LBP(8,2)\) code is determined through a simple comparison (Equation 1) and binary weighting (Equation 3). Bilinear interpolation for the off-grid samples, i.e., neighbors 1, 3, 5, and 7, is done as shown in Equation 2. The pixels \(b = \{b_1, b_2, b_3, b_4\}\) lie around the new sampling point with the corresponding weights \(w = \{0.1716, 0.2426, 0.2426, 0.3431\}\).

\[
C_i = \begin{cases} 
1 & \text{if } N_i \geq P, \ i = 0..7 \\
0 & \text{otherwise} 
\end{cases} \\
N_i = w \cdot b_i, \ i = 1, 3, 5, 7 \\
LBP(8,2) = \sum_{i=0}^{7} C_i \times 2^i
\]

2.1.2. Spatially Enhanced Histogram

The LBP codes are transformed into a feature vector by creating a so-called spatially-enhanced histogram. The vector is
built by concatenating local histograms from partitions of the region-of-interest. Figure 3 shows a partitioning into 42 (7 rows by 6 columns) regions of $21 \times 18$ pixels. The bins of the histogram are selected based on a uniformity criterion defining the maximum number of bit transitions (‘1’ to ‘0’ or vice versa) in the LBP code. For the case of uniformity order 2, there are 58 uniform codes that have at most 2 bit transitions. The remaining codes are lumped into one non-uniform bin. Thus, for the chosen partitioning, one finds a feature vector of length $2478$ ($= 7 \times 6 \times 59$).

Fig. 3. Region-of-interest partitioning for spatially-enhanced histogram generation.

2.2. Feature Classification

In [3], a simple approach with modest recognition performance is described in which feature templates are generated by taking the average histogram of different test images for a given class of expression. The weighted Chi square metric is then applied to determine to which expression template the measured facial feature comes closest to. The metric is given by Equation 4, where $S$ and $M$ represent the measured feature vector and one of the expression templates, respectively. The weighting vector $w$ accentuates feature-rich regions and helps to improve the recognition performance. The classification stage selects an expression associated to the template that gives the smallest $\chi^2$ value.

$$\chi^2(S, M) = \sum_{i,j} w_{i,j} \frac{(S_{i,j} - M_{i,j})^2}{S_{i,j} + M_{i,j}}$$

To improve the expression recognition accuracy further, [3, 8] propose feature classification based on Support Vector Machines. In this work, we restrict our focus to the template matching approach since the reasoning derived from the algorithm mapping experiment can be extended easily to other classification methods.

3. LOW-POWER SMART WIRELESS CAMERA ARCHITECTURE

Smart cameras have been proposed over the years for real-time vision applications. While early smart vision systems relied on PC-based video processing, the availability of high-performance Digital Signal Processors (DSPs) and vision processors has led to true smart cameras capable of locally interpreting video scenes and generating high-level information. The motivation behind smart cameras includes cost and size reduction compared to PC-based systems, ease of deployment and low power operation due to reduction in video transfer.

Equipped with high-performance embedded processors and image sensors, current mobile devices (e.g., phones) are effectively becoming smart cameras allowing image and video processing at lower frame rates and resolutions. In [12], the feasibility of mapping of an LBP based face recognition algorithm on a mobile phone equipped with an ARM9 processor is described.

Functional blocks of a typical smart camera architecture are depicted in Figure 4. The processing power on the smart camera is partitioned into an image processing block for the compute-intensive part and a micro-processor for handling high-level decision making algorithms.

3.1. The WiCa Wireless Smart Camera Platform

Figure 5 shows a photo of the 2nd-generation wireless smart camera called WiCa [6]. The camera is capable of processing stereo images from the two image sensors and is equipped with a massively-parallel processor (Xetal-II [13]) for low-level and intermediate vision processing and an 8051 microcontroller for high-level decision making and camera control tasks. The two processors communicate via an SRAM and decision results can be communicated wireless to other nodes via a ZigBee radio. A low-power Complex Programmable Logic Device (CPLD) is used to realize interface and glue logic functionalities between the different components.

3.2. Xetal2-II Parallel Processor

Top-level architecture of Xetal-II is shown in Figure 6. It is a massively-parallel single instruction multiple data (MP-SIMD) processor realized using 320 processing elements (PEs) organized as a linear processing array (LPA). A total 10 Mbit on-chip frame memory is allocated to the LPA to facilitate frame-iterative video processing tasks. Video interfacing to the chip is provided via a data input processor (DIP) and
4. ALGORITHM MAPPING

In this section, we address issues related to mapping the LBP-based facial expression recognition algorithm on the WiCa platform. We will investigate a couple of task partitioning options and see how effectively the two compute resources, i.e., Xetal-II and the 8051 micro-controller, are utilized. The goal is to find a partition in which the overall operation leads to a low-power consumption.

Although we address the algorithms for expression recognition, it should also be noted that there is an earlier phase in which detection of face(s) is done to determine the region-of-interest. In relation to the chosen smart camera platform, this topic has been addressed in [16] where real-time face detection is demonstrated using a facial feature detection algorithm running mainly on the MP-SIMD processor.

4.1. LBP Computation

Since the LBP operations given in equations 1, 2 3 are executed on every pixel in the region-of-interest, they are naturally suited for parallel implementation. The code segment in Table 1 shows how the LBP computation can be coded in the XTC language of Xetal-II. The three operations, i.e., bi-linear interpolation, comparison and binary weighting are shown which generate a decimal LBP code from 8 neighbors of a pixel.

The left and right operators are used to access the left and right neighbors of the pivot pixel \( cr \), respectively. As shown in Figure 2, a total of 5 image lines \( (ff, fs, cr, pf, ps) \) are involved in computing LBP(8,2). The extensions \( .o \) and \( .e \) are used to access the odd and even pixels of an image line.

In principle, since the LBP generation can be performed on a line-basis, the amount of on-chip storage required for this algorithm is just the LBP row-span, i.e., 5 rows of VGA lines (640 pixels). This is equivalent to 10 Xetal-II line memories of 320 pixels each, which is only 0.5% of the available on-chip memory, implying that there is sufficient room for accommodating pre-processing (such as face detection) and post processing (histogram generation and feature classification).

4.2. Spatially-Enhanced Histogram Generation

4.2.1. LBP to Bin Conversion

This is the first step in which the LBP code at each pixel location is transformed into one of the 58 bins representing uniform codes or lumped into a bin for the non-uniform codes. This could be realized using a look-up-table (LUT) in which the LBP code is used as an address to a table filled with bin indexes. With regard to the WiCa platform the following three implementations are feasible: (i) using a bin table of 58 entries in the GCP data memory (ii) using the LUT memory in the data output processor (DOP) (iii) off-loading the LUT...
Table 1. LBP computation on a VGA size image line.

```c
void lbpo()
{
    lbpo = (((ff_o.left*0.1716 +
        ff_e.left*0.2426 +
        fs_o.left*0.2426 +
        fs_e.left*0.3431) >= cr_o)?128:0) +
        (( ff_o >= cr_o)?64:0 ) +
        (((fs_o.right*0.2426 +
        fs_e*0.3431 +
        ff_e*0.2426 +
        ff_o.right*0.1716) >= cr_o)?32:0 ) +
        (( cr_o.right >= cr_o)?16:0 ) +
        (((ps_e*0.2426 +
        pf_e*0.3431 +
        ps_o.right*0.1716 +
        pf_o.right*0.2426) >= cr_o)?8:0 ) +
        (( ps_o >= cr_o)?4:0 ) +
        (((ps_o.left*0.1716 +
        ps_e.left*0.2426 +
        pf_e.left*0.2426 +
        pf_o.left*0.3431) >= cr_o)?2:0 ) +
        (( cr_o.left >= cr_o)?1:0 ) ;
}
```

computation to the sequential (8051) processor. We discuss the first option further to see how well histogram computation could be done by combined GCP-LPA operation on the MP-SIMD processor.

4.2.2. Row-wise Aggregation

In this phase, each LBP line is compared with the 58 uniform bins and every time the LBP code matches a bin, the corresponding histogram entry is incremented. For this purpose, one Xetal-II line is reserved per bin to keep the histogram count. Since the feature vector construction requires separate histogram per region, in total, 406 lines are needed to hold histograms for all the 7 horizontal partitions of the region-of-interest. The storage requirement can be reduced at the expense of more computation if the column-wise aggregation is done right after the row-wise aggregation per horizontal partition.

4.2.3. Column-wise Aggregation

Here, the objective is to cluster the bin contributions of all the columns of a region. Since each region has a horizontal span of 18 pixels which are folded into 9 columns of odd and even pixels, the column-wise aggregation has effectively to collect pixels over 9 columns. The contribution of odd and even pixels is already merged during the row-wise aggregation.

Due to the limited neighbor connectivity of Xetal-II processing elements, i.e., direct left and right neighbors, column-wise operations are in-general executed less efficiently. However, this can be managed to a certain extent by writing instructions taking into account the hardware connectivity and pipeline latency.

4.2.4. Sequential Histogram Implementation

This alternative shifts part or all of the histogram computation to a sequential processor, in the WiCa case, to the 8051 microcontroller. By making use of the LUT in the DOP module of Xetal-II, the LBP code can be transformed to the 59 histogram bins as it streams out. In this case, the task of the sequential processor becomes building separate histograms for the 42 partitions of the region-of-interest. Since data streams out of Xetal-II line by line, the sequential processor needs to identify the boundaries of 21 × 18 LBP values per partition. Another option is to stream out the result of row-wise aggregation, so that the less SIMD friendly part of column aggregation is done on the 8051 core.

4.3. Feature Classification

In principle, template matching based feature classification using the weighted $\chi^2$ distance metric (see Equation 4) can be realized on the parallel processing hardware; which makes sense specially when the histogram calculation is also done on the parallel engine.

Assuming a LUT-based division operation, evaluating Equation 4 for a feature vector of length 2478 involves around 12390 basic arithmetic and memory operations. When checking for 7 facial expression classes, and a video speed of 25 frames per second, the raw computational load amounts to about 2.2 MOPS. Depending on the compute power available, mapping on the sequential processor could be a possibility.

5. EXPERIMENTAL RESULTS

Table 2 gives a summary of the execution cycle count for different parts of LBP-based facial expression recognition algorithm on a region-of-interest of 108-by-147 pixels. Due to the limited compute power available in the micro-controller, all tasks other than the final template matching are mapped on the SIMD processor. The histogram computation task is split into two: Hist(1) where LBP code to histogram bin conversion and row-wise aggregation are done, and Hist(2) where column-wise aggregation is done to build the final feature vector.

While LBP computation maps efficiently on Xetal-II, due to high data-level parallelism, both histogram computation tasks are executed less efficiently. Due to the search over 58 bins during the mapping phase, Hist(1) takes up a large number of clock cycles. The column on array utilization gives a percentage of PEs that perform a useful task, which is rather low during the histogram computation phases. The figure obviously increases when multiple ROIs are handled in parallel.
Related to the array utilization, the column on compute performance gives a measure of useful computations delivered by the SIMD processor during execution of each task.

The last column gives an indication on power consumed by the respective processor when working on the assigned tasks. In case of Xetal-II, a total of 56588 execution cycles in one frame time (1/25Hz = 40 ms) amount to an average operation frequency of 1.41 MHz. When combined with a compute efficiency of (785 mW/110MHz = 7.14 mW/MHz) for the processor, this gives an average power dissipation of about 10 mW. It should be noted that the dissipation will be higher when a face detection task is also included. Since a large portion of the chip area is taken up by low-leakage frame memories, the idle power of Xetal-II is limited to about 1.7 mW.

The weighted $\chi^2$ based template matching was mapped on to the 8051 micro-controller. The cycle count obtained from a simulator when matching the generated feature vector with 7 expression templates is around 29 MCycles, which translates into 1.2 seconds of processing time at an operating frequency of 24 MHz. The large cycle count is a consequence of the 8051 processor architecture, which has a simple 8-bit datapath and instructions that cost 12 or 24 clock cycles. Replacing the micro-controller with a high performance general-purpose processor would improve the performance. For example, simulation on an ARM9E family processor with 32 kB data cache and running at 100 MHz shows processing time of 9 ms for the template matching task, which is within a frame interval of 40 ms.

### Table 2. Execution profile of facial expression recognition on the WiCa platform.

<table>
<thead>
<tr>
<th>Processor</th>
<th>Algorithm (ROI)</th>
<th>Cycle Count</th>
<th>Array Utiliz. [%]</th>
<th>Perf. @ 25 fps [MOPS]</th>
<th>Aver. Power [mW]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xetal-II</td>
<td>LBP</td>
<td>15582</td>
<td>17</td>
<td>22</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Hist (1)</td>
<td>34104</td>
<td>1.8</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Hist (2)</td>
<td>6902</td>
<td>1.8</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>56588</td>
<td>6</td>
<td>32</td>
<td>12</td>
</tr>
<tr>
<td>8051</td>
<td>Template</td>
<td>29.1e6</td>
<td>–</td>
<td>–</td>
<td>73</td>
</tr>
</tbody>
</table>

### 6. CONCLUSIONS

In this paper, the issue of mapping facial expression recognition algorithms on a low-power smart camera platform is addressed. A recognition algorithm based on Local Binary Patterns (LBP) and spatially-enhanced histogram has been investigated to determine how efficiently it maps on a platform with a massively-parallel single-instruction multiple-data (SIMD) processor (Xetal-II) and a general-purpose micro-controller (8051 family). Due to its pixel parallel operation, the LBP algorithm maps well on the SIMD processor. However, with respect to spatially-enhanced histogram computation, the SIMD processor utilization is about 1.8%, indicating the sequential nature of the algorithm. For the final phase, i.e., feature classification, a template matching algorithm based on the weighted $\chi^2$ metric is considered. Since mapping this algorithm on the 8051 micro-controller reduces the achievable frame rate, upgrading to a high-performance general-purpose processor would be necessary.

### 7. ACKNOWLEDGMENTS

We would like to thank R. Kleihorst and B. Schueler of NXP Research for support regarding the WiCa smart camera platform.

### 8. REFERENCES


