This paper describes the AddressLib, which is part of the MPEG-7 Reference Software. Frame-based image processing algorithms required by the feature extraction modules of normative MPEG-7 Ds and DSs as well as algorithm computational complexity are easily performed and measured with the AddressLib. This is achieved by defining and implementing an image processing model which consists of three schemes to address the pixel data: the inter-, intra-, and segment addressing. For the segment addressing (e.g., used by the watershed algorithm) a new implementation based on a hierarchical dual stack (HDLIFO) is presented. In a hardware implementation, the HDLIFO will reduce the area significantly compared to a hierarchical FIFO.

1. **Introduction**

One purpose to build the MPEG-7 Reference Software [1] (XM software) was to help to understand the standard by allowing the analysis of the code. Especially, the non-normative components of the software (feature extraction and computation of descriptor distances) are important because they are representing the normative semantic meaning of low-level descriptors. To make the extraction process from visual media data more transparent, the XM software provides a generic image processing library, the AddressLib. Using this library, it is not required to understand the image processing functions as a whole, but only the concept of the AddressLib and the function to process one pixel. It also leads to a homogeneous implementation style allowing to compare and to estimate the computational complexity of feature extraction tools[2].

The following chapters describe the concept of the AddressLib, which is not only applicable to MPEG-7 visual feature extraction but to frame-based video and image processing in general.

---

* This paper is in the context with the COST211 project, the QIMERA activity, the SCHEMA Network of Excellence and the IST project BUSMAN.
2. Addressing methods

2.1. Concept

The *load-process-store* processing scheme can be applied to nearly all computations in all application domains. In terms of the AddressLib, the processing is done in a way that the results for each pixel in the result image are computed and stored. This seems to be obvious, but it must be clear when the input image(s) and the result image have different resolutions.

The second basic observation is that in video processing a fixed operation is applied to all pixels (or a set of pixels) in the image (ca. 100,000 times for CIF and ca. 25,000 for QCIF). The scheduling of the pixels to be processed is similar in different image processing functions. However, the operations to process them may differ considerably. As a conclusion, image processing functions are split into the addressing and the processing of pixel data. This results in a generic and limited function set for addressing pixels which is specific for the application domain *video processing*. On the other hand, the pixel processing functions are specific for one individual application and the know-how of the algorithm developer. We found, that there are three different types of addressing functions: the inter-, intra-, and segment addressing. At this point, different image-processing functions using a given addressing scheme (inter, intra or segment) might use variations of this addressing scheme through a set of parameters; i.e., the image-processing function can use different input/output channels or pixel scan schemes (plain/meander horizontal/vertical scan). Other parameters are specific to certain addressing schemes, which are described in the following corresponding sections.

2.2. Inter Addressing

The inter addressing is used when a pixel result is computed from two pixels of two different images or image regions as shown in figure 1a. An example is the computation of absolute difference images. Here, the inter addressing is combined with a processing function computing the absolute difference of two pixel values. The results can also be accumulated in a global variable to compute the sum of absolute differences (SAD), typically used for block matching motion estimation.

The inter addressing allows to select a rectangular region of the image to be processed by specifying the size and the location of the regions in the result and input images. In addition, it is possible to apply a co-ordinate transformation for the second input image by giving a pointer to the transformation function.
2.3. Intra Addressing

The intra processing allows implementing all kind of filters. As shown in figure 1b, a pixel and its neighbourhood are processed to compute the value of the output pixel. The kind of neighbourhood system can be specified (no neighbours, 1D horizontal/vertical, or 2D). Also the treatment of neighbour pixels, which are outside the valid image area can be selected. Finally, the intra processing function can be used to perform a simple 1:2 or 2:1 scaling.

2.4. Segment Addressing

Image analysis and feature extraction often requires analysing the spatial structure of the content by using expansion processes. As shown in figure 2a, the input is a 2D neighbourhood similar to the intra addressing, but the pixels are scheduled for processing in a recursive way. An overview about segment addressing and its options can be found in [3]. Thus, we describe here the new implementation style using a hierarchical dual stack (HDLIFO).

Compared with the well know FIFO implementation [4][5] for expansion processes the dual stack (DLIFO) has the same feature of processing pixels in a spatial distance order. Here, two alternating stacks are used to store the pixels of following distance levels. Figure 2c shows that the processing order alternates for the distance levels compared to the FIFO implementation (figure 2b). Anyway, the order within one distance level is irrelevant.

In terms of implementation and computational complexity the FIFO and DLIFO are equivalent. The FIFO needs one start and one end pointer, while the DLIFO needs two top-of-stack pointers. In case the hierarchical versions are required (watershed algorithm or a skeleton expansion), the HDLIFO requires only one pointer for the higher level stacks, while the HFIFO still requires one start and one end pointer for the higher level FIFOs. As a result, the HDLIFO requires nearly half of the amount of pointers. This is relevant for hardware implementations, because the area needed to implement these pointers is one of the major factors for the total chip area of a HFIFO. Thus, the HDLIFO is significantly better than the HFIFO regarding to the required chip area. Besides
the chip area, in a pipelined system also the temporal behaviour can be improved due to bypassing the stack memory, especially when its fill level is low and the pipeline stalls. Bypassing reduces the read and write time. Here, the HDLIFO can bypass the stack memory at each change of the distance level, while the HFIFO can do this only when the hierarchy level is changed.

Figure 2. a) Segment addressing. Starting from one or more start pixels a connected area is processed. Processing order of b) FIFO and c) DLIFO expansion processes.

3. Summary

This paper discussed the basic idea of the AddressLib and its corresponding image processing model. It describes the three addressing schemes of the AddressLib, including their possible parameterisation. Due to previous discussion of the parameterisation of the segment addressing, it is only referenced in this case. For the segment addressing a new implementation scheme using a hierarchical dual stack (HDLIFO) was presented. The new scheme was briefly discussed and compared to the well-known hierarchical FIFO scheme (HFIFO).

References