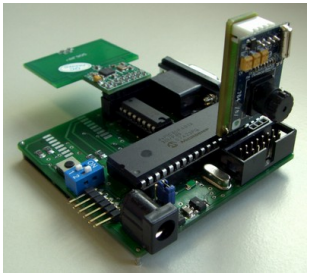
Fractional Wavelet Filter for Camera Sensor Node with extremely little RAM

Stephan Rein, Stephan Lehmann, Clemens Gühmann
Wavelet Application Group
Technische Universität Berlin

MobiMedia July 2008

Motivation

- Sensor networks are low-cost and free (Opensensor) ⇒ a nice basis for research activities in communications and signal processing
- Possible future application: Camera sensor network for moving vehicle navigation or space exploration



- Bandwidth is very limited ⇒ wavelet-based compression gives superior results for high compression rates while large RAM is required
- Related work mainly addresses transforms for FPGA-platforms; a very recent work [Oliver et al. 2008] implements a line-based transform in C++, but is still too memory intensive
- ⇒Use own SD-Card filesystem to design a low-memory picture wavelet transform – the *fractional wavelet filter*

Contents

1. Picture Wavelet Transform

1.1. FBI-filter

- Use Daubechies 9/7 wavelet as it gives superior compression results while computationally more complex
- Also employed for JPEG2000

i	lowpass Al(i)	highpass Ah(i)
0	0.852699	0.788486
±1	0.377403	-0.418092
±2	-0.110624	-0.040689
±3	-0.023849	0.064539
±4	0.037828	0

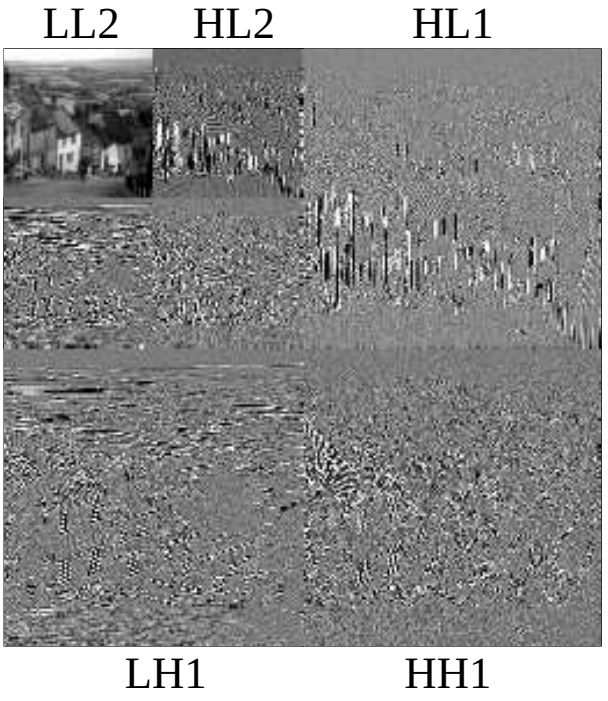
- Compute $L(i)$ and $H(i), i = 0 \ldots N-1$:

$$L(i) = \sum_{j=-4}^{4} line_{pic}(i+j) \cdot Al(j),$$

$$H(i) = \sum_{j=-3}^{3} line_{pic}(i+j) \cdot Ah(j)$$

1.2. Subband division

- Picture wavelet transform computes four subbands per level, the LL, HL, LH, and HH subband
- For each subband the filter is 1) applied on all rows and 2) on all columns
- Example HL-subband: First apply highpass on all rows; then take the result and apply the lowpass on all columns of it
- For the next level the LL-subband of the previous level is the source



2. Fractional Wavelet

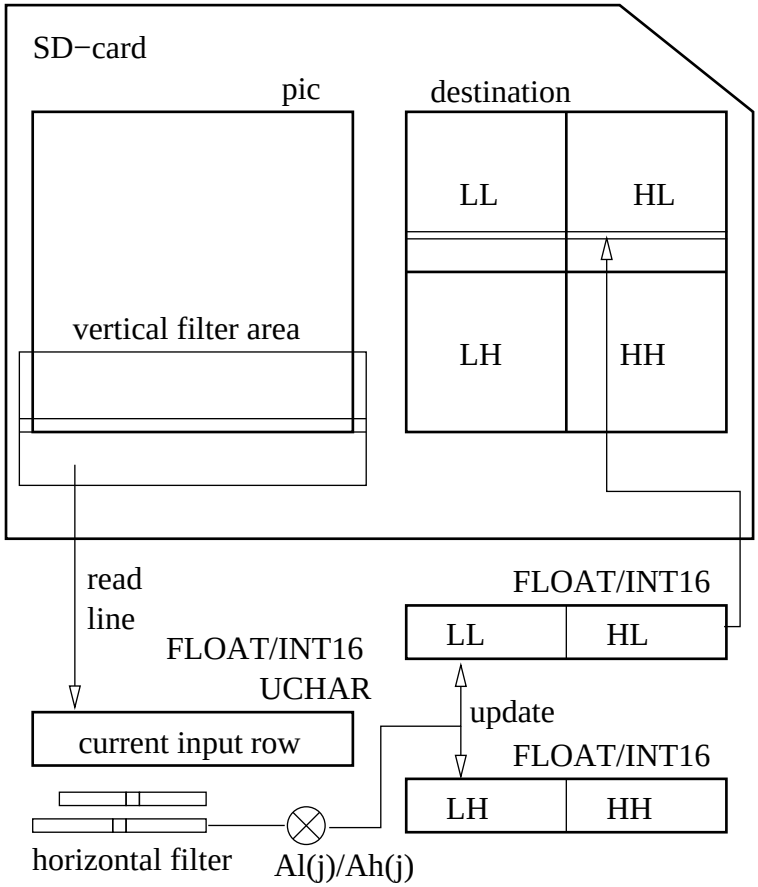
2.1. Problems

- The sensor has only 2 kByte of RAM
- It is only possible to line-wisely read or write data from/to the SD-card

2.2. Solution provided by fractional filter

- Allocate three arrays, one for the current input line and two destination buffers
- The horizontal filter coefficients are computed on the fly
- Only fractions of the vertical coefficients are computed and used to update the destination arrays
- The final destination arrays are written to the card

2.3. Scheme of fractional filter



3. Implementation (floating/fixed)

Two versions of the multi-level fractional wavelet filter were implemented:

- A floating-point version with high precision for 128x128 pictures that needs 1152 Bytes of RAM
- A fast fixed-point version for 256x256 pictures that needs 1280 Bytes RAM; the data format (Texas Instruments) for the 16 bit wavelet coefficients is given as

level	1	2	3	4	5
format	Q10.5	Q 11.4	Q12.3	Q13.2	Q12.2

and the real filter coefficients were transformed to integers in the Q0.15 data format.

Stephan Rein , Stephan Lehmann, Clemens Gühmann
Wavelet Application Group
stephan.rein@tu-berlin.de

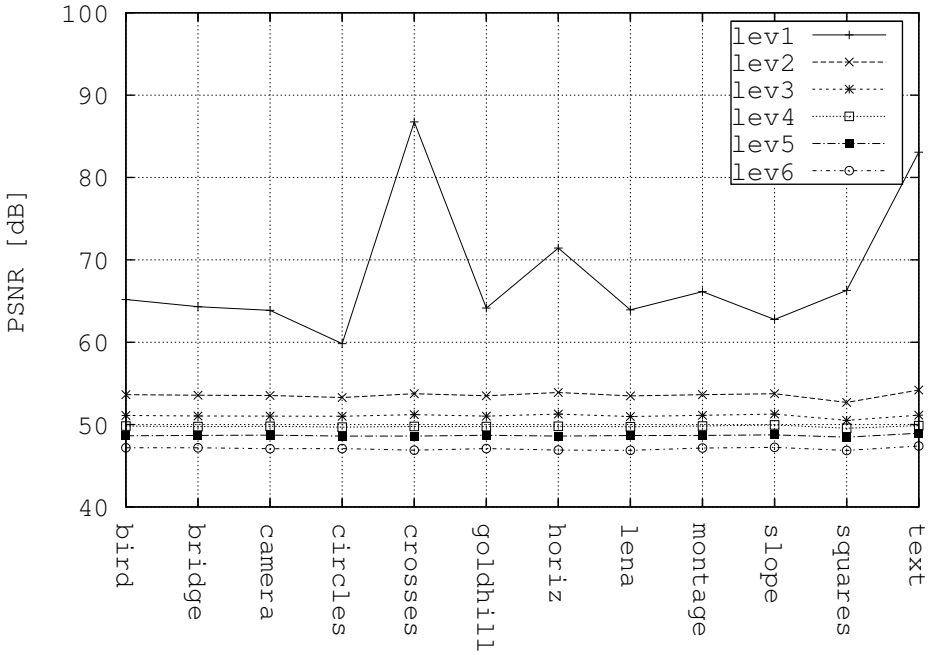
4. Results

4.1. Time for six-level transform

	time [sec]			
	T_{read}	T_{write}	$T_{compute}$	T_{total}
float128	1.421	0.5425	14.22	16.18
fix256	2.839	1.085	7.716	11.64

- Floating-point is very slow because of no hardware support; computations are seven times slower than for the fixed-point
- Reading time is more than twice the writing time because the rows are read repetitively
- Computing takes the largest amount of time $\Rightarrow$ the SD-card is not the bottleneck

4.2. PSNR results fixed-point for six levels



5. Summary and Demo

- Designed and implemented a scheme that allows to compute a picture wavelet transform of 256x256 pictures on a sensor with 2 kByte RAM
- The filter is relatively slow, however it may be applied for still images
- Little work has been devoted to wavelet picture transform on limited platforms $\Rightarrow$ the fractional filter may be a good starting point for future investigations
- Our future work concerns the inverse transform, the integration of the lifting scheme and a low-complexity version of SPIHT

Stephan Rein , Stephan Lehmann, Clemens Gühmann
Wavelet Application Group
stephan.rein@tu-berlin.de

demo

1. Took a picture with the sensor and stored it as *stern* on the SD-card
2. Will compute a 2-level wavelet transform on the dsPIC with

```
u w 2 stern
```

3. Save the level 1 and level 2 pictures on the laptop as *lv1.int* and *lv2.int* with

```
mmc save file stern_lv1 lv1
mmc save file stern_lv2 lv2
```

4. Start a demo-script on the laptop to illustrate the transform