Watermark Embedding in Digital Camera Firmware

Peter Meerwald,
May 28, 2008

Application Scenario

- Digital images can be easily copied and tampered
- Active and passive methods have been proposed for copyright protection and integrity verification: watermarking and forensics

- This work: simple watermarking in camera firmware for copyright protection

Camera Forensics

- [Chen08a] Use Photo-Response Non-Uniformity (PRNU) for camera identification and integrity verification
- Detecting forgeries (lighting, interlacing, specular highlights on the eye, JPEG quantization, double compression, chromatic aberration, image statistics, resampling, region, duplication, CFA Interpolation [Popescu05a], [Hani Farid])

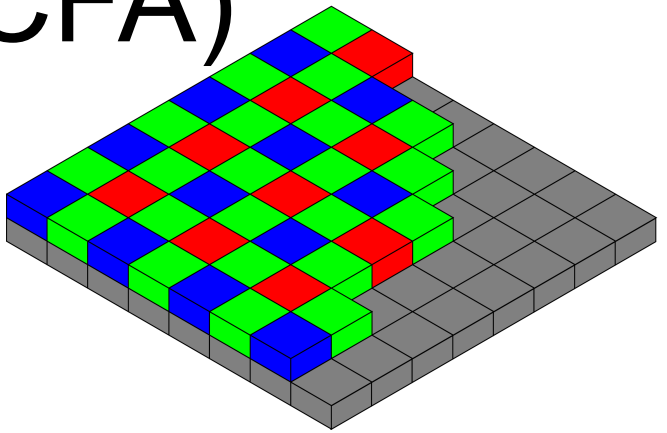
“Secure Digital Camera”: Watermarking

- [Blythe04a] Capture human iris image through viewfinder, embed in camera image together with camera identification and image hash
- [Lukac06a] Emboss visible watermark in CFA domain
- [Mohanty07c] VLSI architecture for robust and fragile watermarking
- [Nelson05a] CMOS image sensor adds pseudo-random watermark
- Kodak and Epson offered cameras with watermarking capabilities (2003, discontinued)

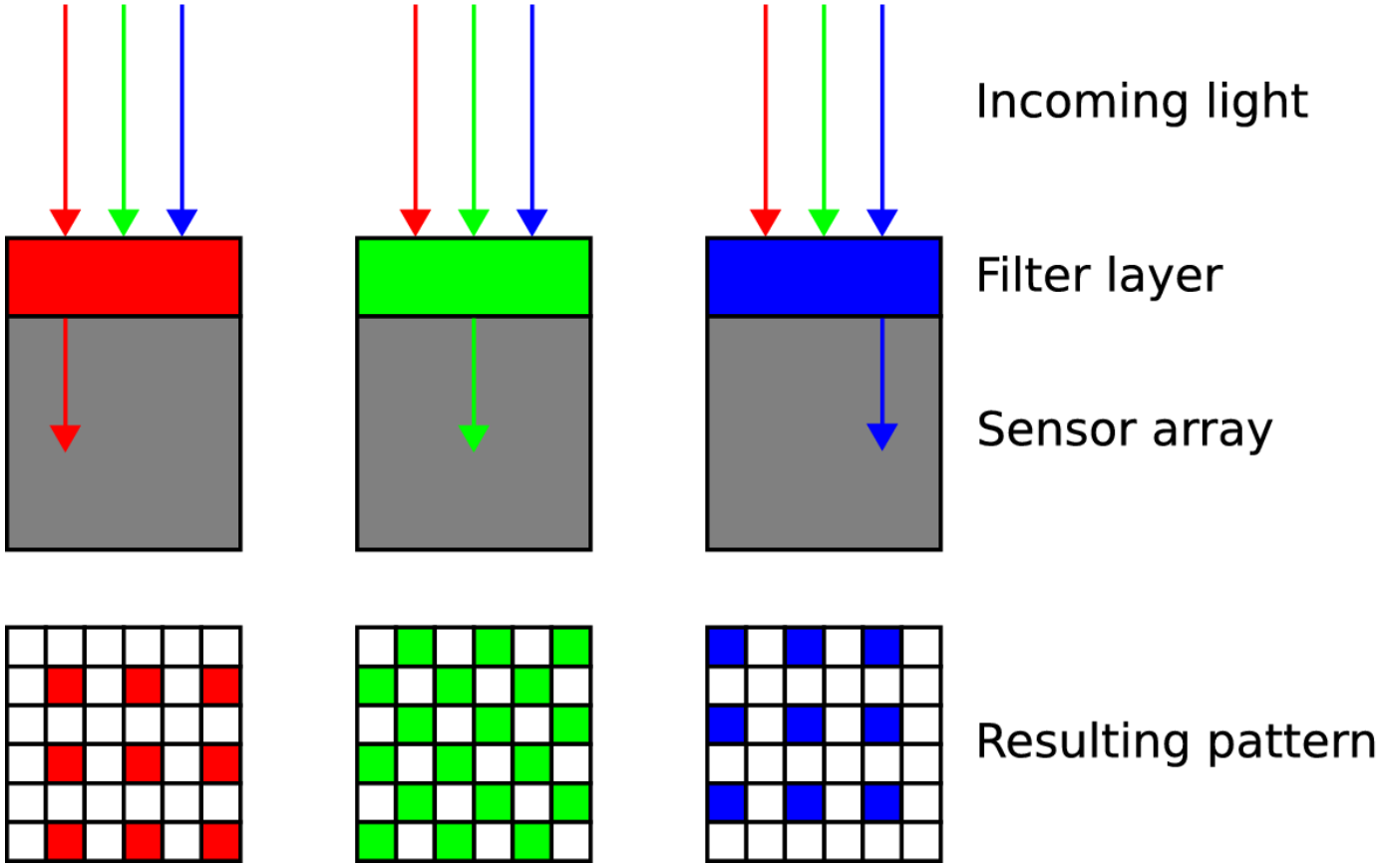
Patents!

- Many JPEG-domain algorithms that could be employed

Color Filter Array (CFA)

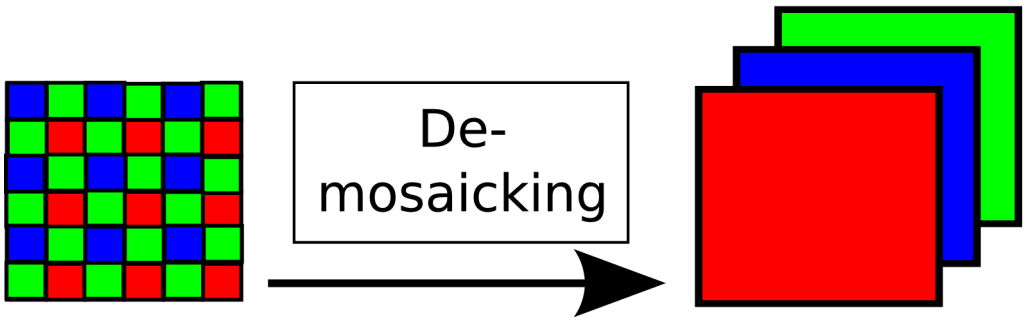


Bayer CFA pattern



Demosaicking

- Basically interpolation to get full-resolution RGB image



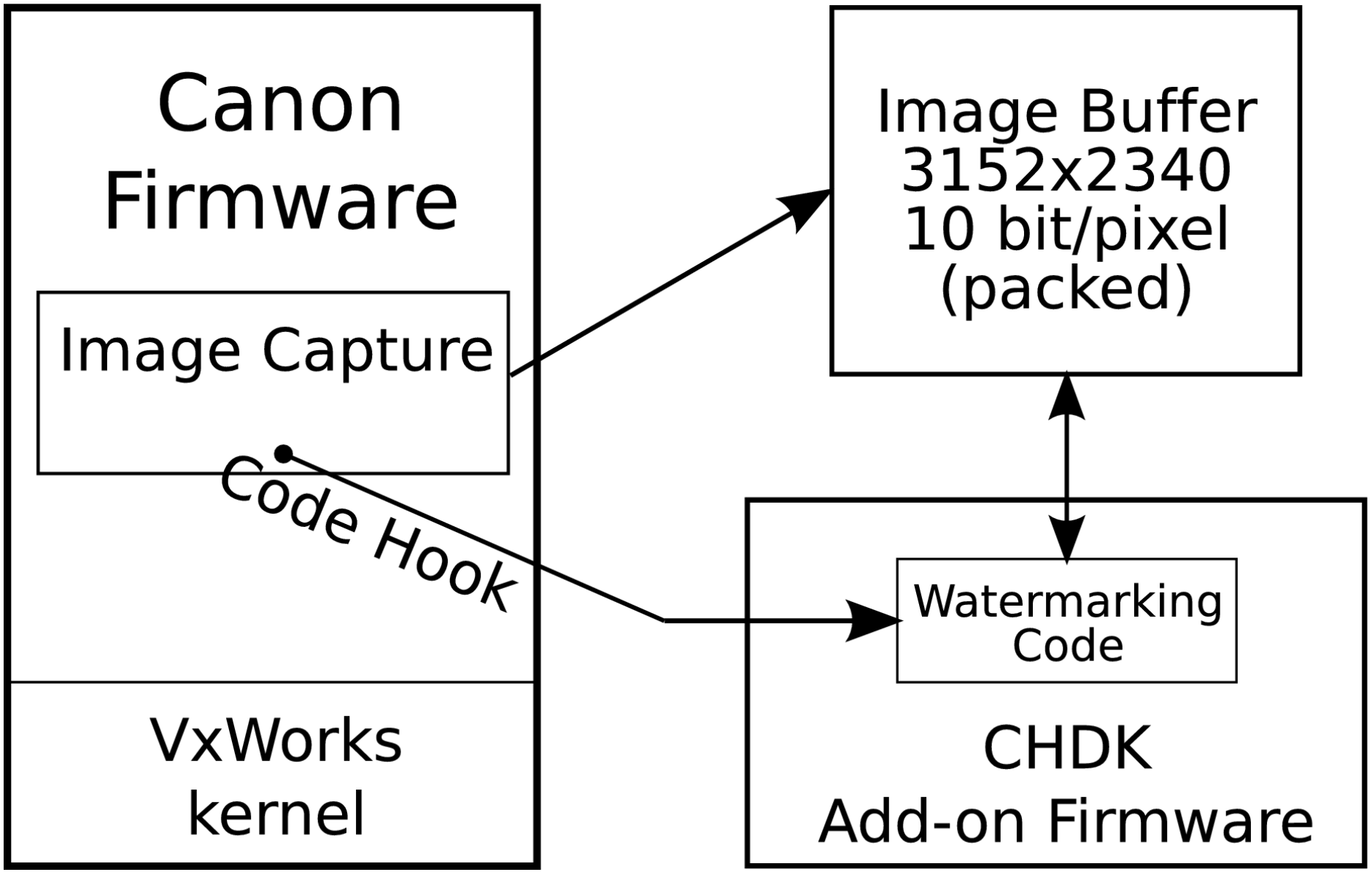
  Many different approaches, eg. Bilinear, bicubic, smooth hue transition, gradient-based, adaptive color plane, adaptive homogeneity-directed, ...

- Implementation, eg.
  http://www.cybercom.net/~dcoffin/dcraw/
- Camera implementation unknown

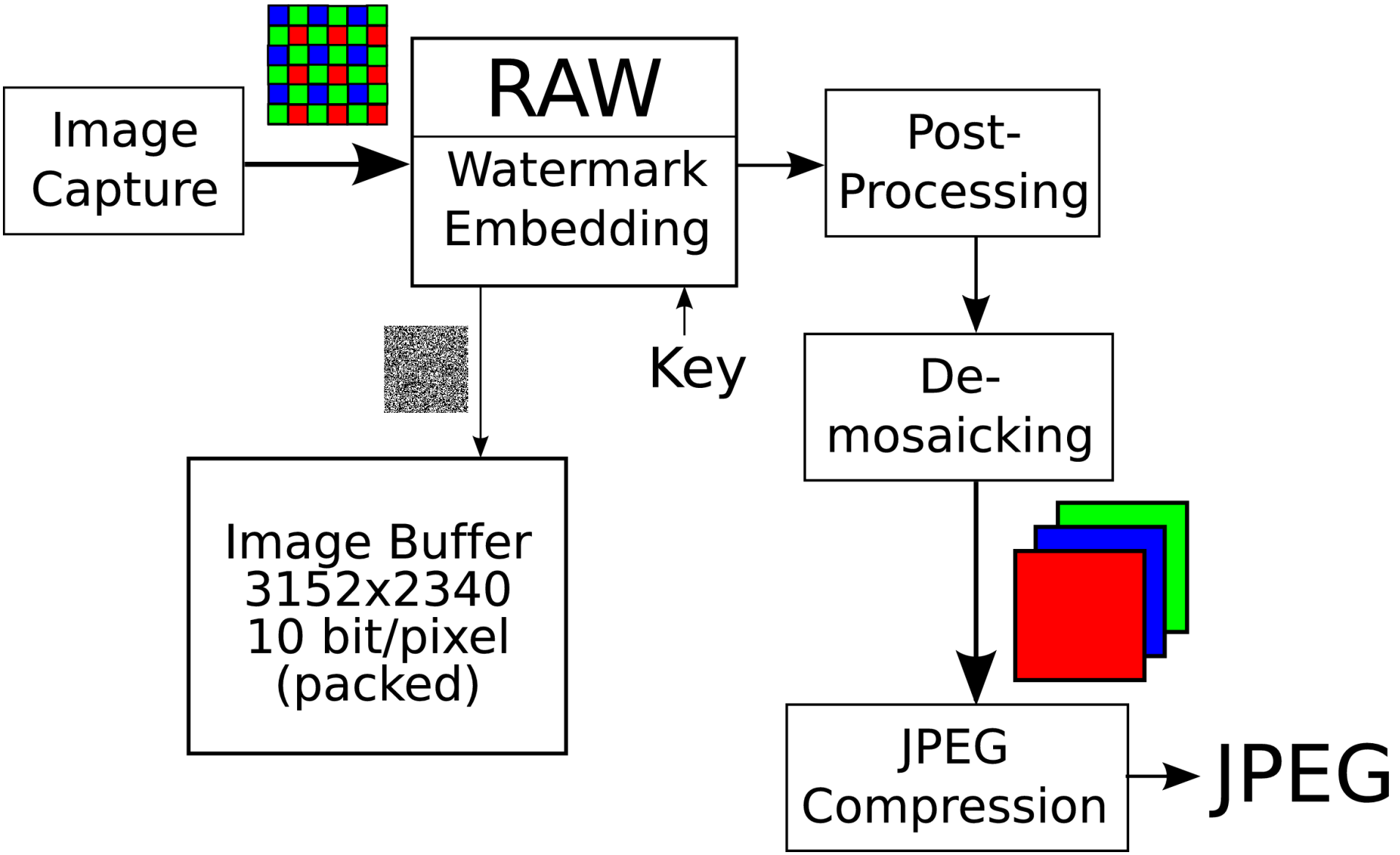
Enter CHDK

- CHDK: firmware add-on for Canon DIGIC II and DIGIC III cameras, http://chdk.wikia.com
- Adds bracketing of exposure, RAW file support, BASIC scripts, remote camera control, additional data display (histogram, battery life), longer exposure time, faster shutter speed, games, ...
- Linux-hosted cross-compilation, using arm-elf-gcc 3.4.6

CHDK enables Watermarking



Processing Pipeline



Camera Characteristics

- Canon IXUS 70, 7.1 MP
- DIGIC III chip (ARM9 core), unknown sensor
- Sensor resolution 3152x2340 pixels, usable 3112x2328 pixels
- Camera supports
  - 3072x2304, 2592x1944, 2048x1536, 1600x1200, 640x480 pixel JPEG images (3 quality settings)

Camera Hardware / Software

- Canon DIGIC III processor aka. TI 32-bit ARM9 core, ~ 200 MHz
- Lots of custom hardware for JPEG, scaling, histogram computation, color conversion, ...
- VxWorks operating system
- ~ 3.5 MB firmware
- ~ 1 MB usable free memory,
  ~45 MB / sec bandwidth
- ~ 150 KB CHDK add-on firmware

Watermark Embedding (1)

- Add pseudo-random sequence {-1, 1} generated by Mersenne Twister to blue channel pixels
- Want watermarking in 'real-time', before image is saved (so ~ 1 sec delay)
- 7.1 MP is 9.2 MB (10 bit/pixel, **packed**)
- Hard to match runtime requirement!
- Prefer to spend time on perceptual modelling rather than unpacking/packing bits!

Watermark Embedding (2)

- First naïve implementation
  - ~ 45 sec: GetPixel()/PutPixel()
  - ~ 15 sec: reduce calls to PRNG
- Unrolled pixel unpack/pack
  - ~ 2 sec: shifting through two bit buffers
- Not yet
  - Assembler code, SIMD instructions (?)
  - Use some specialized hardware (?)

Watermark Embedding (3)

// process even row

```
p_row_buf = p_out_row_buf =
    (uint16 *) &buf_pos[PIXTOBYTES(RAW_LEFT_MARGIN+4)];

bit_buf = *p_row_buf++;
out_bit_buf = bit_buf >> 6;

bit_buf = (bit_buf << 16) + *p_row_buf++;
pixel = bit_buf >> 12 & 0x3ff;
WATERMARK_PIXEL(pixel);
out_bit_buf = (out_bit_buf << 10) + pixel;
*p_out_row_buf++ = out_bit_buf >> 4;

out_bit_buf = (out_bit_buf << 10)
    + (bit_buf >> 2 & 0x3ff);
...
```

ARM9 can move & shift in one instruction!
`and` implemented by shifts

```
ldrh    r7, [sl], #2
ldr     r2, .L102
ldrh    r3, [sl], #2
mov     r6, r7, asr #6
ldr     r5, [r2, #12]
add     r7, r3, r7, asl #16
tst     r9, r1
mov     r3, r7, asr #12
mov     r4, r3, asl #22
rsbeq   r5, r5, #0
movs    r1, r1, lsr #1
mov     r4, r4, lsr #22
bne     .L50
```

Loading Firmware Add-on

Watermarking Menu

Watermarking Options

Image Quality (RAW vs. JPEG)

Image Quality (RAW vs. Watermarked)

Detection Results

Image	Watermark	Response	Format
3112×2328	strength 48	0.701	RAW
3112×2328	strength 96	1.314	RAW
3112×2328	no watermark	-0.027	RAW
3072×2304	strength 48	0.970	JPEG (SF)
3072×2304	strength 48	0.891	JPEG (F)
3072×2304	strength 48	0.568	JPEG (N)
2592×1944	strength 48	0.441	JPEG (SF)
2048×1536	strength 48	0.448	JPEG (SF)
1600×1200	strength 48	0.299	JPEG (SF)
640×480	strength 48	0.018	JPEG (SF)

SF ... superfine, F ... fine, N ... normal (JPEG quality)

Threshold ~ 0.03

Further Work

- Speedup
- Simple perceptual shaping
- Exploit interpolation for detection
- Image authentication
- Test different image mosaicking algorithms
- Compare with PRNU noise [Chen08a]

References

- [Blythe04a] P. Blythe, J. Fridrich, Secure Digital Camera, Digital Forensic Research Workshop, Aug. 11-13, 2004; Baltimore, MD, USA
- [Chen08a] M. Chen, J. Fridrich, M. Goljan, J. Lukas, Determining Image Origin and Integrity Using Sensor Noise, IEEE TIFS, 3(1):74-90, Mar. 2008
- [Lukac06b] R. Lukac, K. Plataniotis, Camera Image Watermark Transfer by Demosaicking, Proc. ELMAR '06, 9-12, Jun. 2006
- [Mohanty07c] S. Mohanty, E. Kougianos, N. Ranganathan, VLSI Architecture and Chip for Combined Invisible Robust and Fragile Watermarking, IET Computers & Digital Techniques, 1(5):600-611, Jun. 2007
- [Nelson05a] G. Nelson, G. Jullien, O. Yadid-Pecht, CMOS Image Sensor with Watermarking Capabilities, ISCAS '05, 5326-5329, May. 2005
- [Popescu05a] A. Popescu, H. Farid, Exposing Digital Forgeries in Color Filter Array Interpolated Images, IEEE TSP, 53(10):3948-3959, Oct. 2005