GPGPU-Assisted Nonlinear Denoising Filter Generation for Video Coding

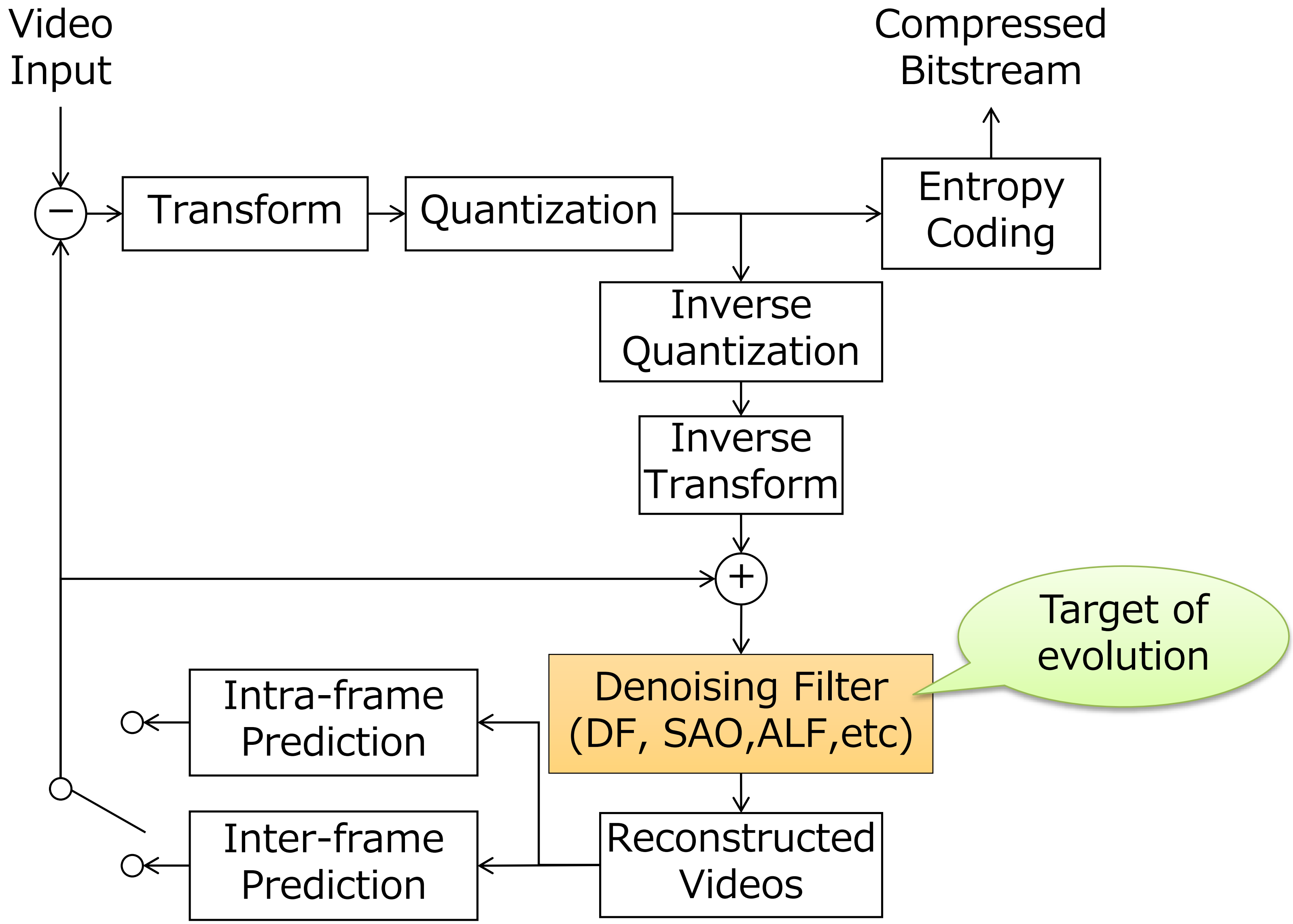
Innovative R&D by NTT

Seishi Takamura and Atsushi Shimizu
NTT Corporation, Japan

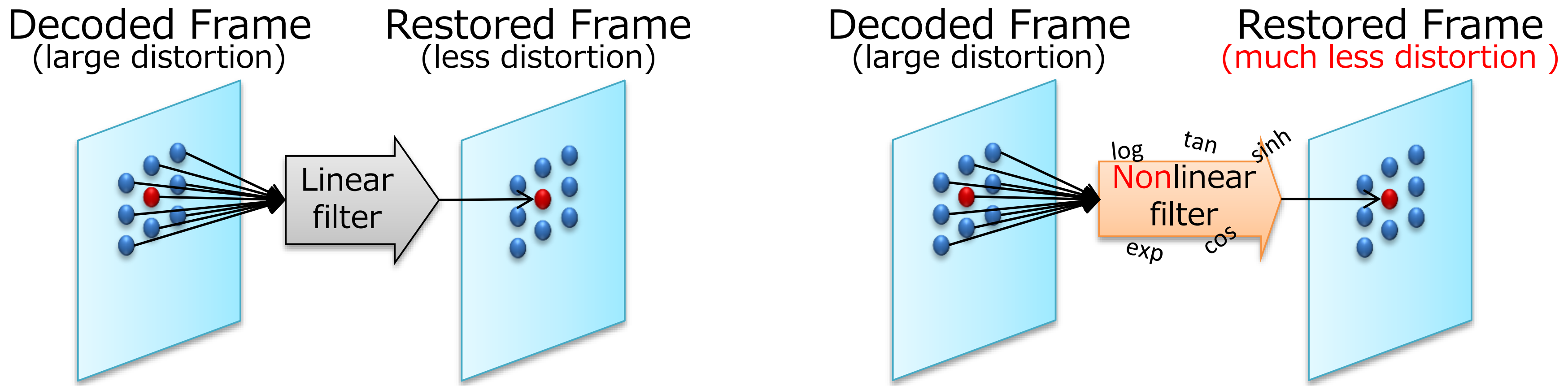
Summary

- State-of-the-art video coding technologies such as H.265/HEVC employ in-loop denoising filters.
- We have developed a new type of in-loop denoising filter with Genetic Programming (GP), which is heavily nonlinear and content-specific.
- To boost the evolution, GPGPU is utilized in filter evaluation process.
- Proposed method yielded better denoising filter in 100x less time.
- The bit rate reduction of 1.492-2.569% was achieved against the reference software of H.265/HEVC.

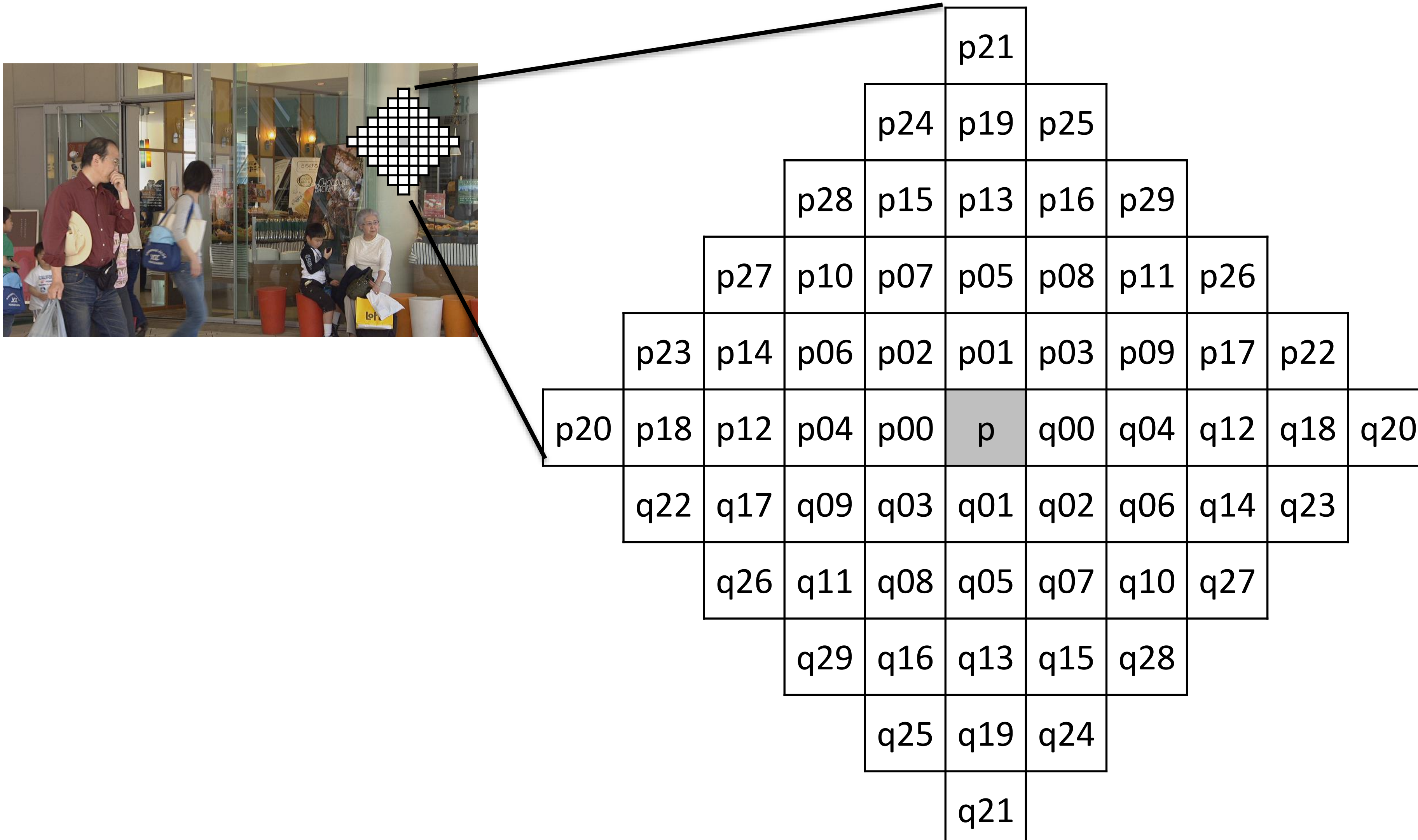
Video Coding Block Diagram



A Leap from Linear Denoising Filter



Denoising Filter Support



Nodes used by our Filter

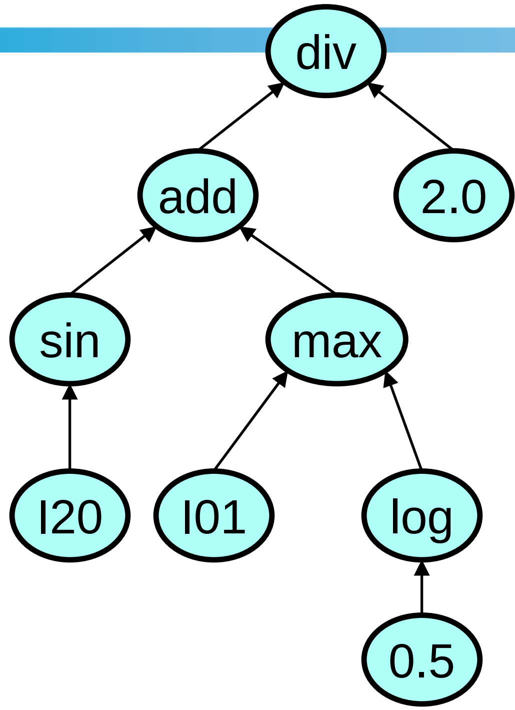
- Terminal nodes

I: pixel value of p
Ixx: (pxx + qxx) / 2,
Dxx: (pxx – qxx) / 2,.
Ils: least-square restored value, a linear combination of I, I00··· I11 with offset.
x, y: horizontal and vertical coordinate of the pixel.
value: immediate values such as “0.3”.

- Functional nodes

min, max, average, abs, /, *, +, −,
exp, pow, log, sqrt, sin, cos, tan, asin, acos, atan,
sinh, cosh, tanh, conditional branch
In addition, followings are defined
and(a, b) := (a>=0 && b>=0) ? (a+b)/2 : −(|a|+|b|)/2,
or(a, b) := (a>=0 || b>=0) ? (|a|+|b|)/2 : −(|a|+|b|)/2,
xor(a, b) := (ab<=0) ? (|a|+|b|)/2 : −(|a|+|b|)/2.

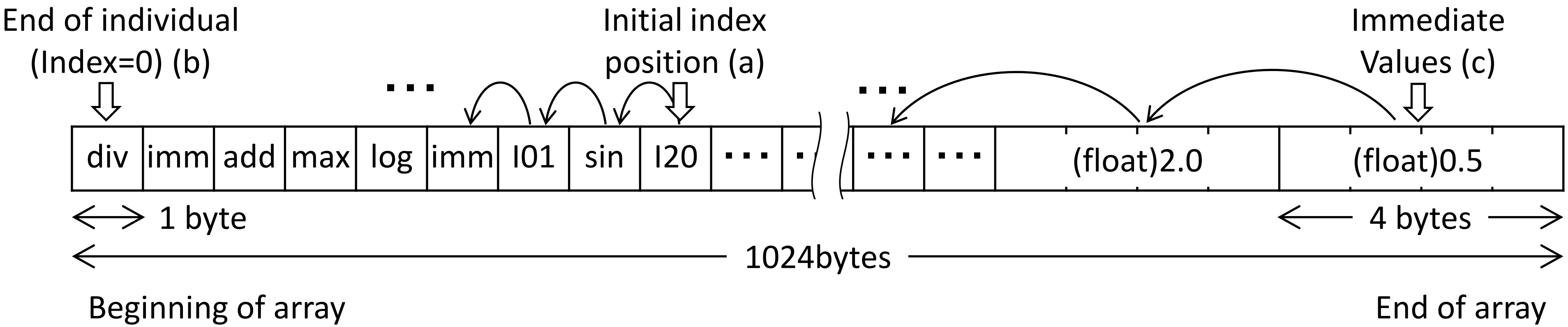
Serializations of a Tree



- Normal expression (or infix notation):
  (sin(I20) + max(I01, log(0.5))) / 2
- Lisp S-expression (or prefix notation):
  (div (add (sin (I20))(max (I01)(log 0.5))) 2)
- Reverse Polish notation (or postfix notation):
  I20 sin I01 0.5 log max add 2.0 div

- We used Reverse Polish notation (as described later).
- The fitness function in the evolution is D+λR, where
  - D is the squared sum of the errors between the filtered image and original image
  - R is the amount of tree information that represents the filter algorithm
  - λ is the same Lagrange multiplier as the encoder uses during rate-distortion optimization process

GPGPU implementation



- We convert the tree in Reverse Polish Notation (RPN) prior to the evaluation.
- Linearized instructions are stuffed from the middle of the array (a) toward the beginning.
- Immediate values are picked out and stuffed from the end (c).

- Filter evaluation procedure is like following:

```
for (index = 0; index < array_length; index++) {
  switch (funcIDs[index]) {
    case add: a=pop(); b=pop(); push(a+b); break;
    case sin: a=pop(); push(sin(a)); break;
    case imm: push(<the value>); break;
    case I: push(I); break;
    case I00: push(I00); break;
    …
  }
}
```

Simulation Conditions

CPU: Intel Core i7-3960X Extreme Edition, C2 stepping

Clock rate: 3.3GHz

Cores: 6 (one core is used for the CPU-experiment)

Hyper threading: on

Memory: 64 GB

OS: Ubuntu Linux 12.04.2 LTS x86_64 Desktop Edition

GPU: NVIDIA GeForce GTX 690

CUDA capability: 3.0

CUDA Cores: 1536

GPU Clock rate: 1.020 GHz

Global memory: 2048 MB

L2 Cache Size: 512 KB

CUDA: Driver version: 5.0.35, x86_64

SDK/Toolkit version: 5.0.35

C++ Compiler (as the backend for nvcc):

Intel C++ Compiler version: 12.1.5 20120612

Video sequences used

BQMall (832x480)

BQTerrace (1920x1080)

RaceHorces (416x240)

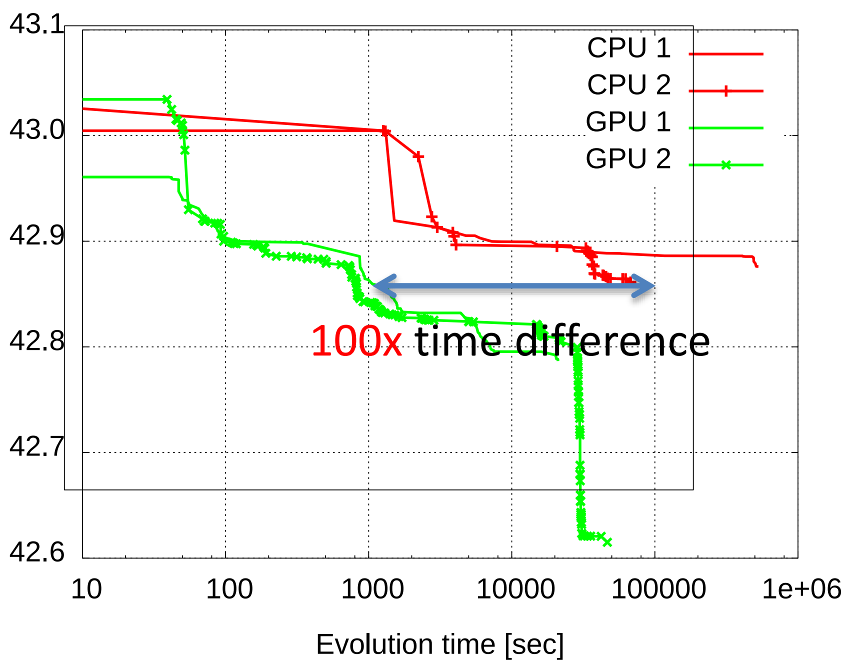
CPU vs. GPU Comparison

- Filter (of 121 nodes) evaluation time over BQMall (832x480)

	Time [sec]	Speed-up (vs.CPU)
CPU (1 core)	0.336489	
GPU	0.002674	125.8x

- Filter evolution speed for BQMall (832x480)

Better fitness



Coding Performance Comparison (vs. original H.265/HEVC)

Negative values mean better performance

Sequence	QP	HM-7.2-3164 rate (a) [bits]	HM-7.2-3164 Y-PSNR [dB]	ALF* BD-rate vs. HM	LS fiter** Y-PSNR [dB]	LS fiter** BD-rate vs. HM	Propsal filter info (R) [bits]	Propsal total rate (a+R)[bits]	Propsal Y-PSNR [dB]	Propsal BD-rate vs. HM
BQSquare (ALF off)	22	210,720	41.53		41.54	0.135%	626	211,346	41.71	-1.492%
	27	138,152	37.16		37.17		315	138,467	37.27	
	32	88,288	33.30		33.33		329	88,617	33.46	
	37	55,048	29.65		29.70		418	55,466	29.93	
BQSquare (ALF on)	22	210,944	41.53	-0.022%	41.54	0.28%	520	211,464	41.69	-1.437%
	27	138,352	37.16		37.17		445	138,797	37.30	(vs.ALFon)
	32	88,504	33.33		33.35		279	88,783	33.48	-1.455%
	37	55,392	29.71		29.72		315	55,707	29.95	(vs.ALFoff)
RaceHorses (ALF off)	22	174,448	42.19		42.30	-1.202%	1195	175,643	42.47	-2.569%
	27	109,264	37.97		38.10		698	109,962	38.18	
	32	63,848	34.08		34.21		750	64,598	34.35	
	37	34,696	30.57		30.71		536	35,232	30.86	
RaceHorses (ALF on)	22	174,936	42.26	-1.755%	42.29	0.428%	321	175,257	42.36	-0.843%
	27	109,536	38.12		38.14		36	109,572	38.13	(vs.ALFon)
	32	64,128	34.26		34.26		376	64,504	34.39	-2.580%
	37	34,992	30.73		30.74		236	35,228	30.85	(vs.ALFoff)

HM: H.265/HEVC reference software (used as an anchor)
*ALF: adaptive loop filter (state-of-the-art loop filter)
**LS filter: least square filter. Filter info(R) = 448 bits

Example of Generated Filter

RaceHorses, QP=22, ALF-off, filter information (R) = 1,195 bits

(add (add (add (add (mul (I) 0.932803332806)(mul (I01) 0.087968140841))(add (mul (I02) −0.051799394190)(mul (I00) 0.095137931406)))(add (add (mul (I03) −0.050682399422)(mul (I04) −0.040202748030))(add (mul (I05) −0.052293013781)(mul (ave (I02)(tan (I12))) 0.017782183364))))(add (add (add (mul (I07) 0.025515399873) (mul (I08) 0.025515399873))(sub (mul (sin (atan (and (I09)(I21)))) 0.016251996160)(mul (tanh (tanh (tanh (mul (I02)(asin (log (sinh (sqr (div (mul (I05) (sqr (div (atan (mul (mul (asin (asin (sqr (I))))(sqr (sqr (div (I05) (I13)))))(sqr (div (sin (I19)) (I01)))))(sqr (I01)))))(I03)))))))))) 0.005235218443)))(mul (I29) −0.005818639882)))

Conclusion

- A novel method to generate denoising filter that enhances the coding performance is proposed.
- GPGPU accelerated the evolution by around 100 times than the CPU.
- Generated filters outperformed least square filter and state-of-the-art filter, i.e., ALF.