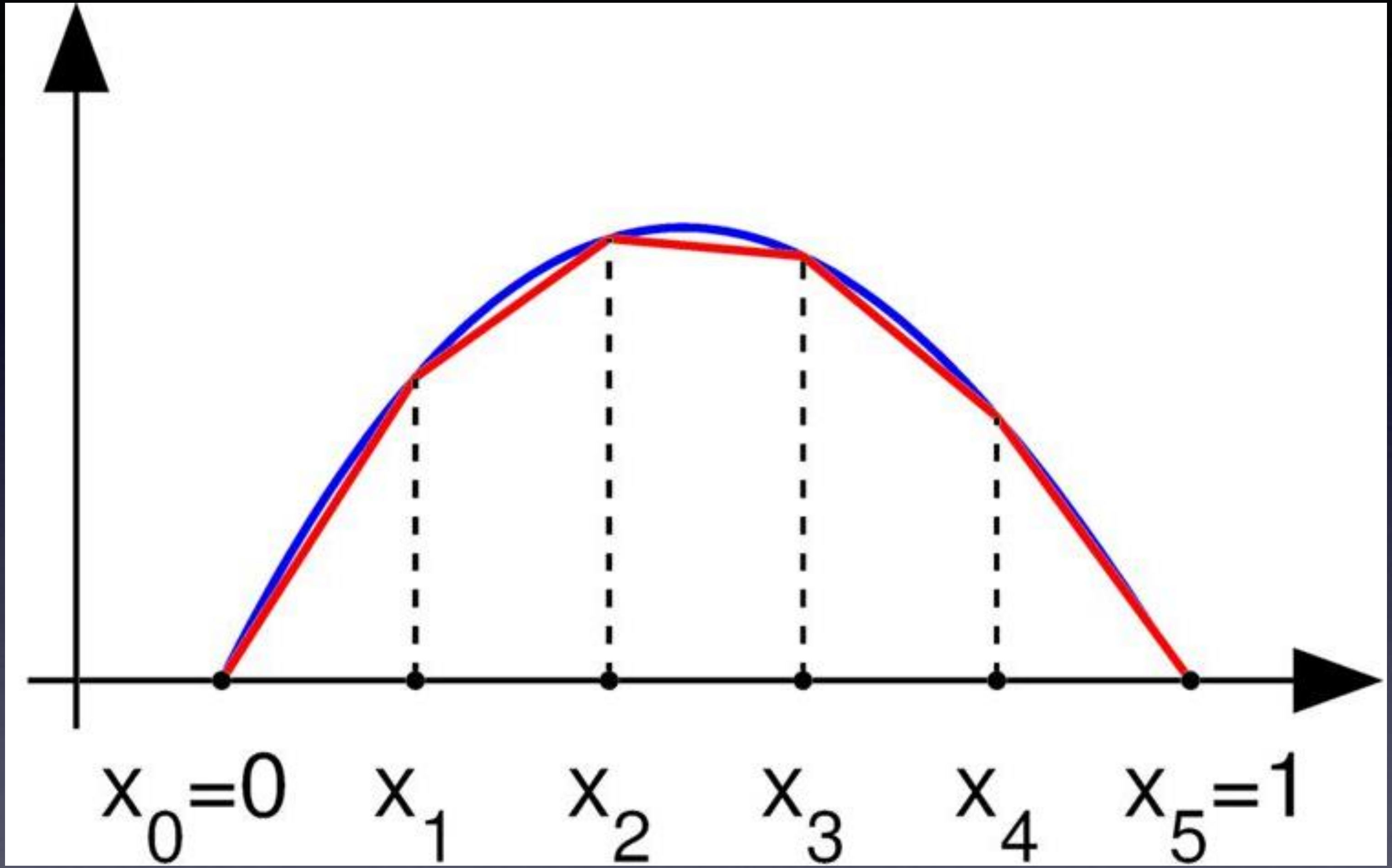
Genetic Piecewise Linear Approximation

Yonghye Kwon

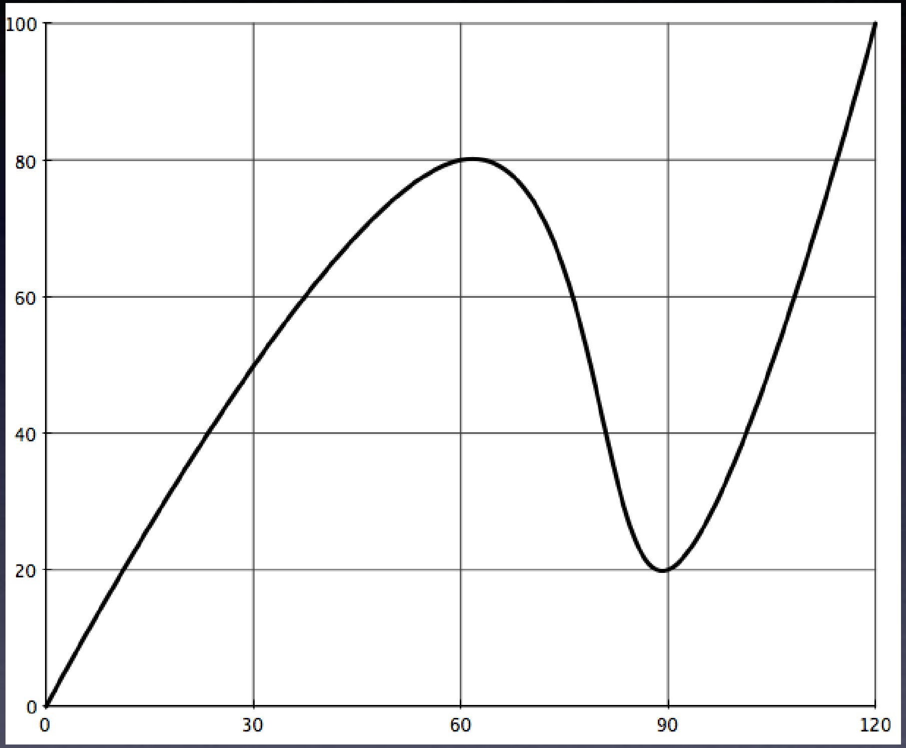
Piecewise Linear Approximation



A function (blue), and a piecewise linear approximation (red)

Piecewise Linear Approximation

Divide a function to 3 pieces
for piecewise linear approximation

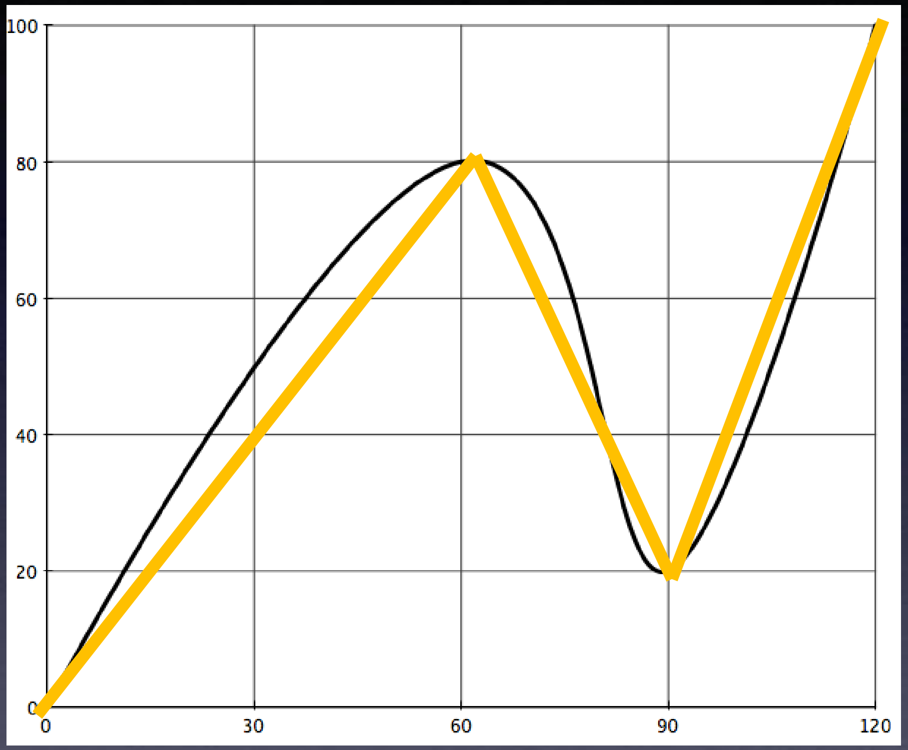


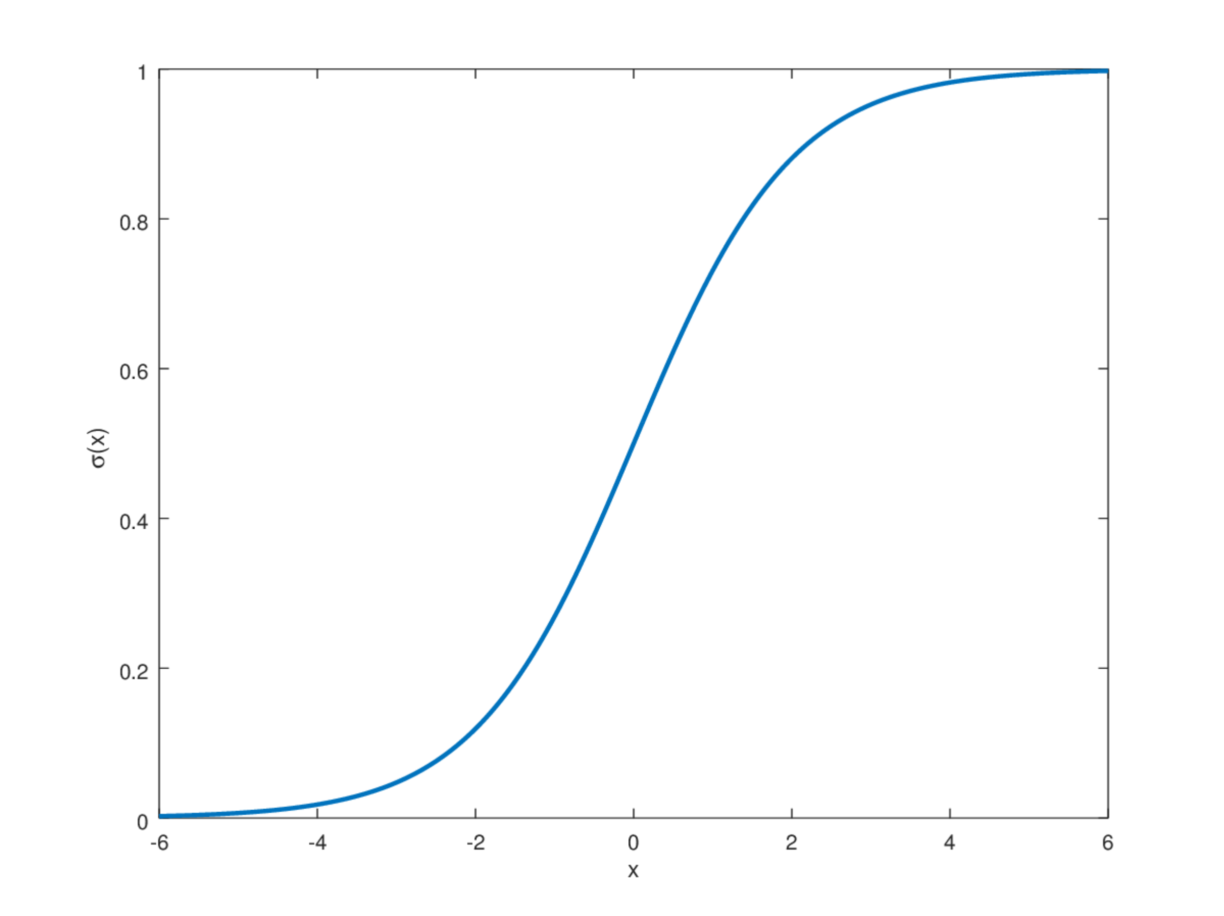
Piecewise Linear Approximation

Like this …?

Piecewise Linear Approximation

How about this one…?



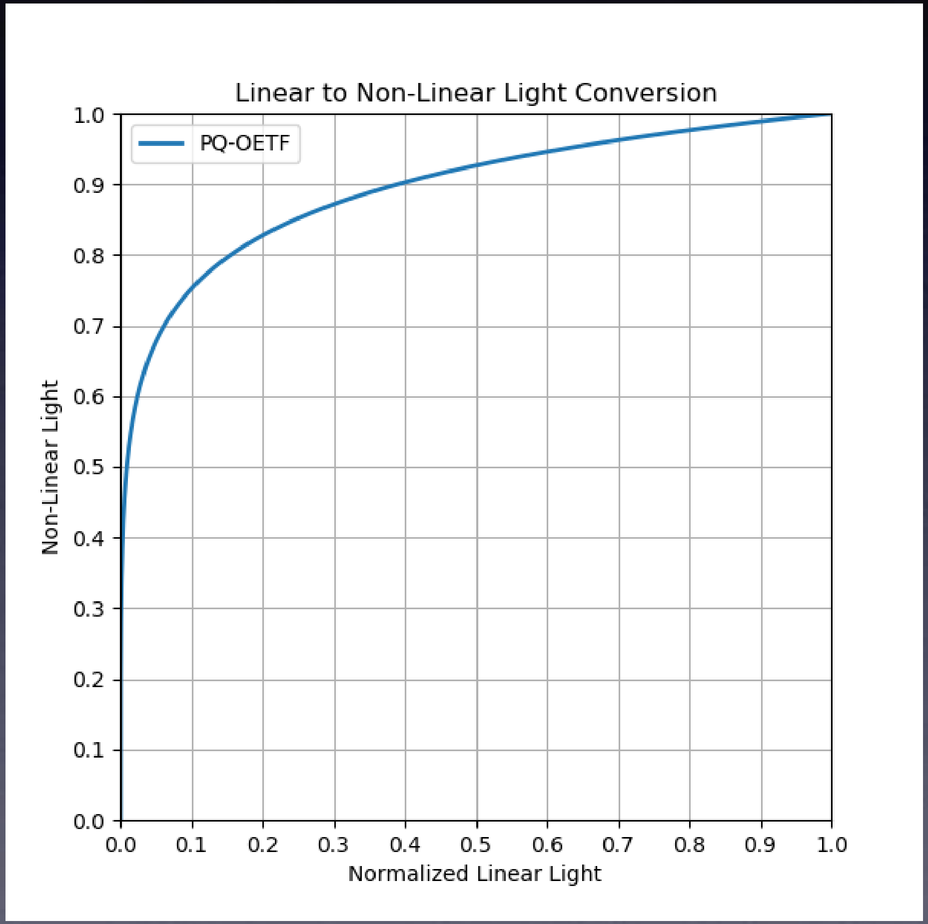


How to divide a function to multiple pieces for piecewise linear approximation?

Genetic Piecewise Linear Approximation

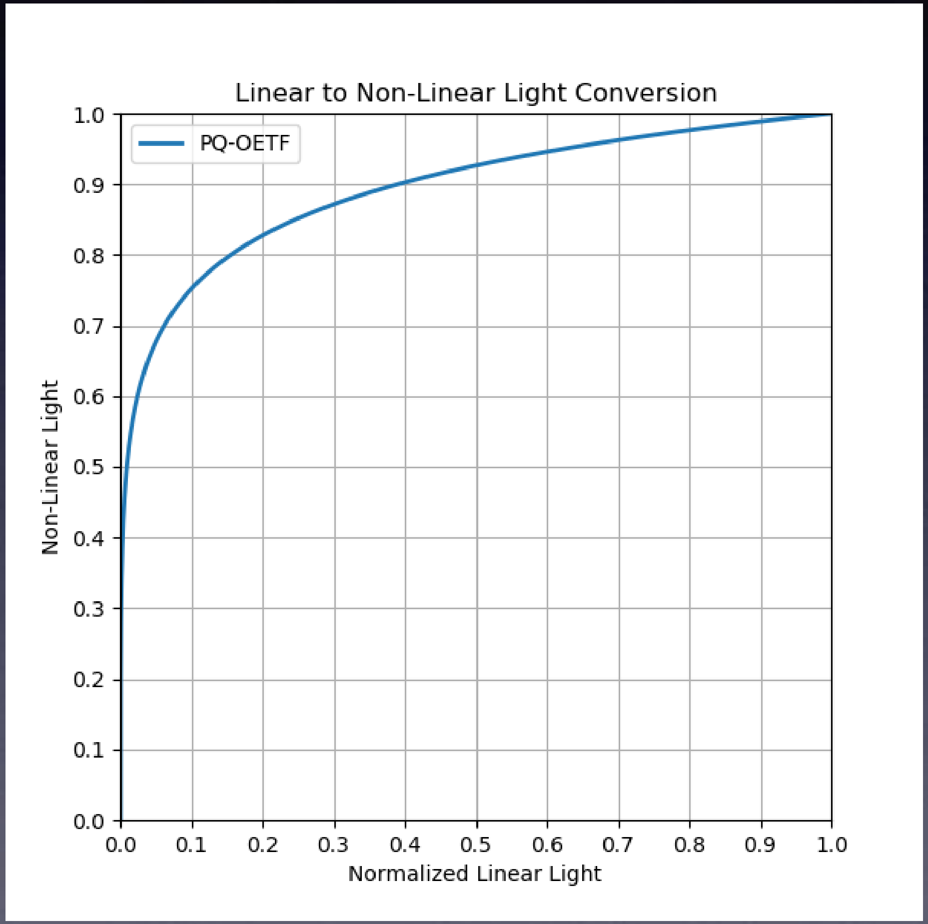
Function for Experiment

- **Opto-Electrical Transfer Function(OETF) used in HDR Image Acquisition**

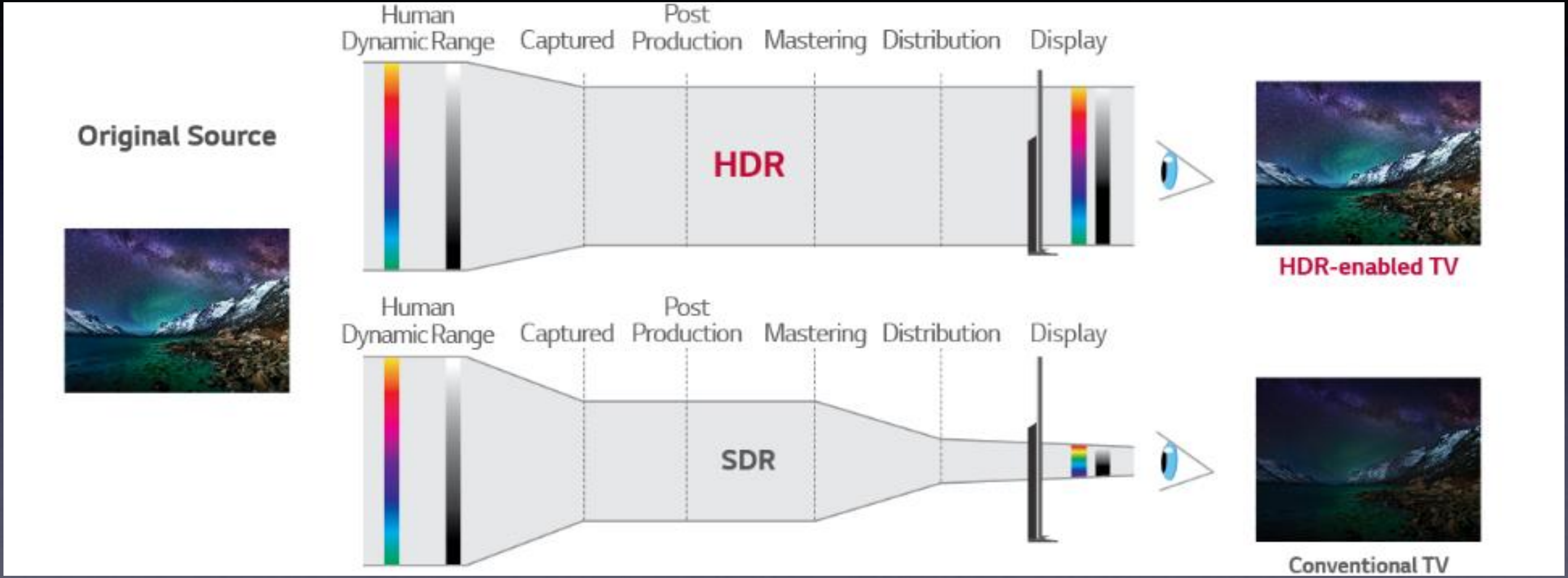


Function for Experiment

- **Opto-Electrical Transfer Function(OETF) used in HDR Image Acquisition**



HDR(High Dynamic Range)

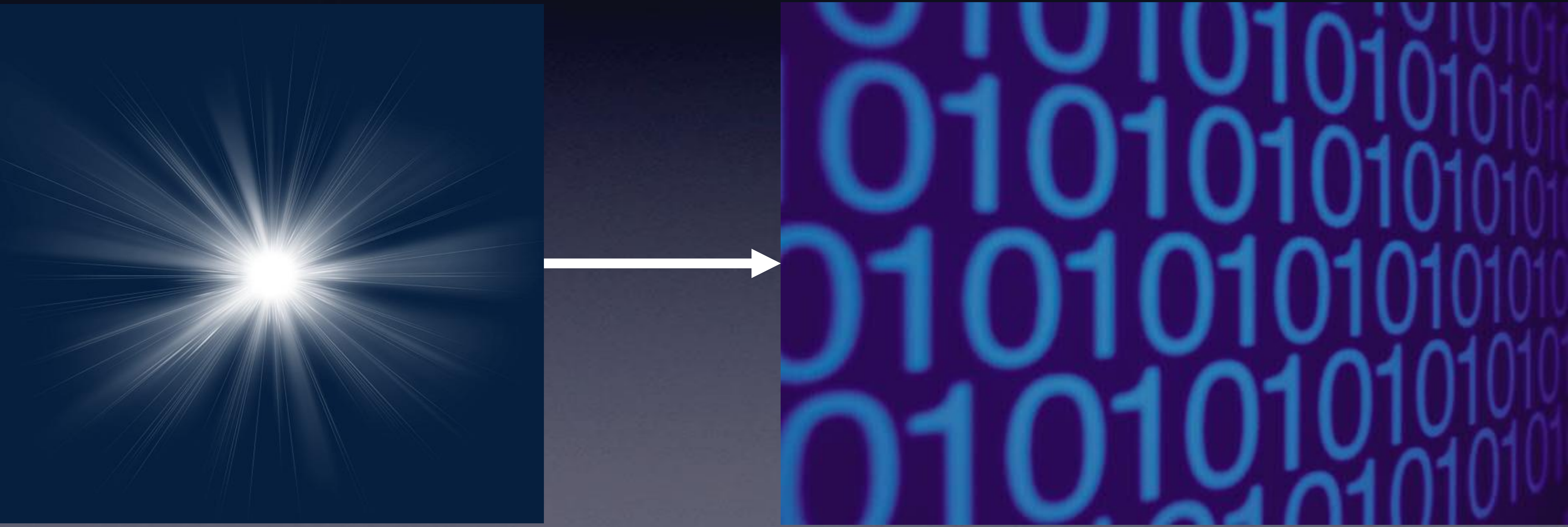


SDR Image

HDR Image

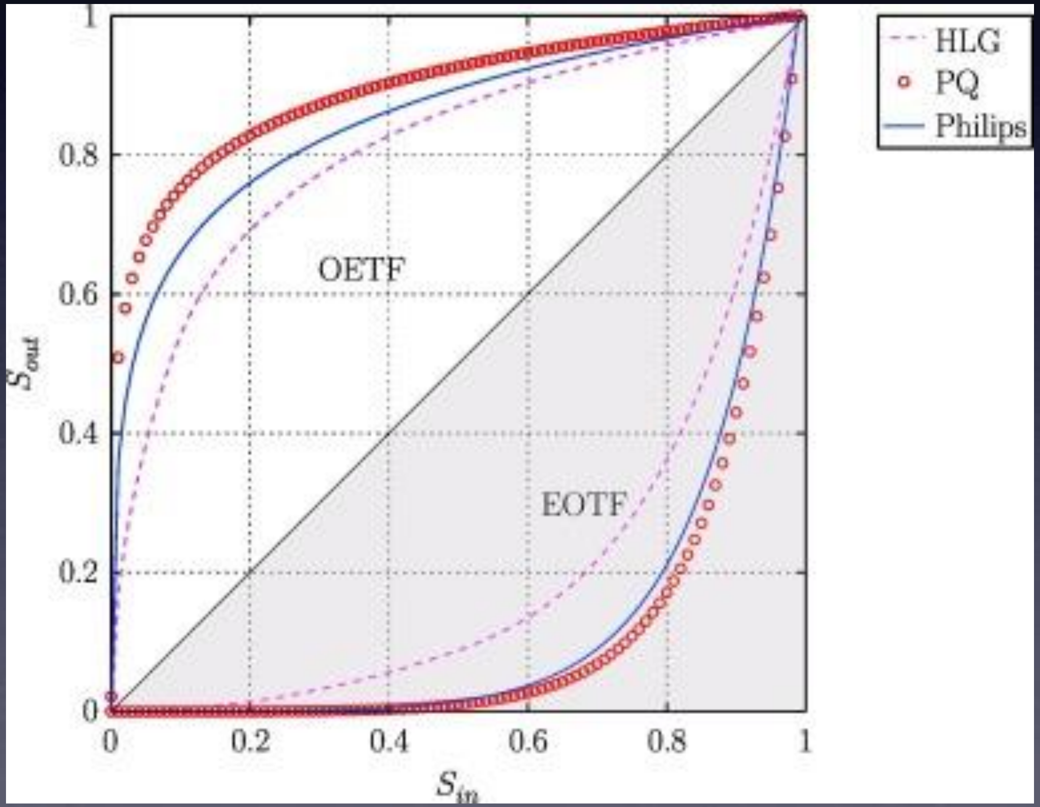
OETF

- **Non-linear function to convert an optical level signal to digital signal**



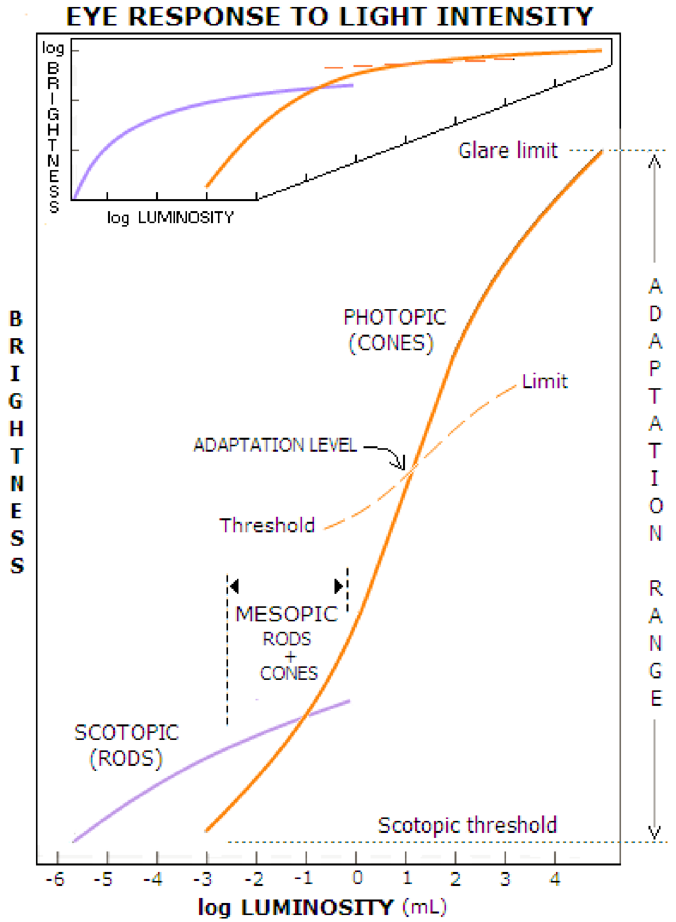
OETF

- **Non-linear** function to convert an optical level signal to digital signal

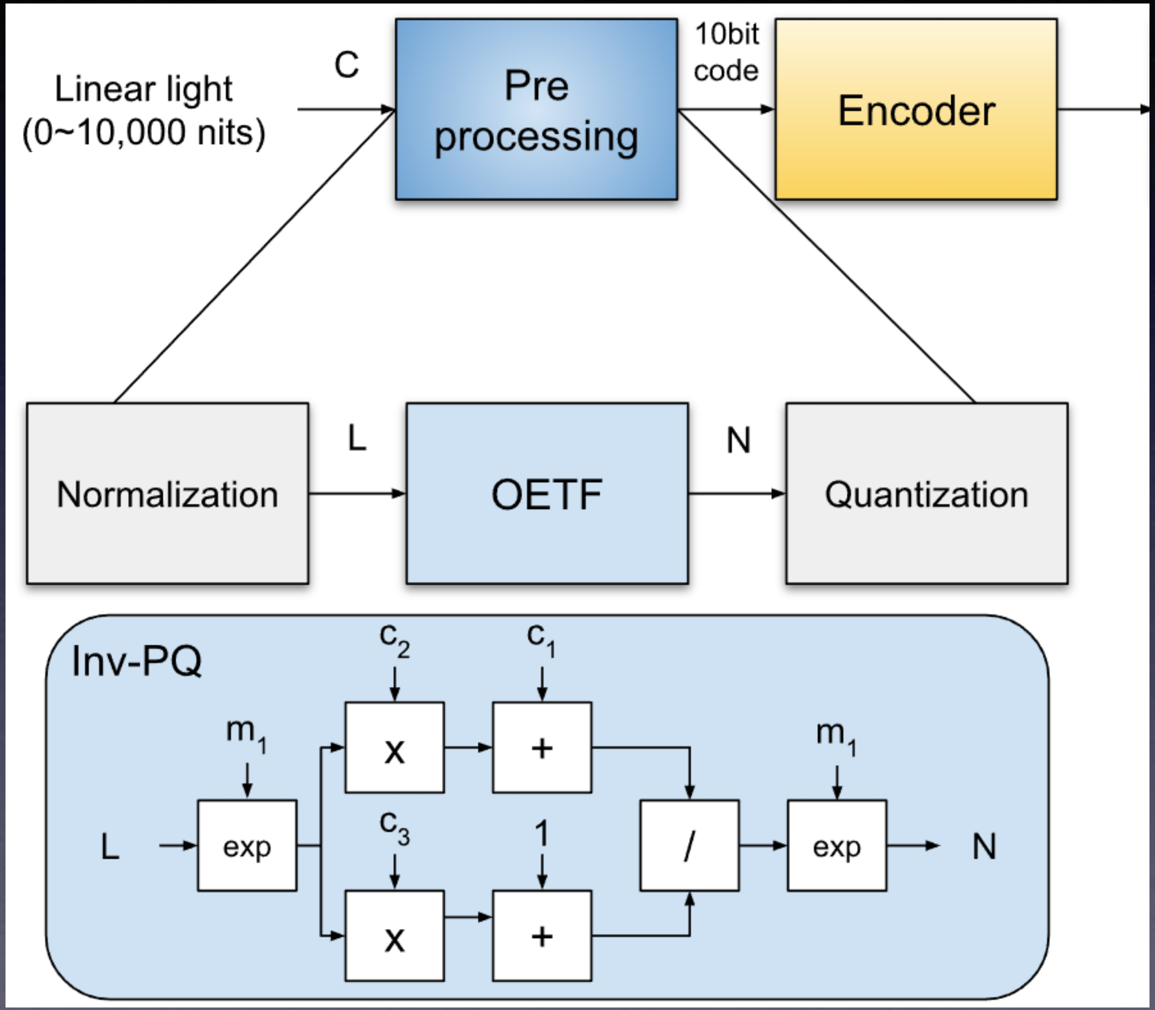


Why OETF is Non-linear?

- Human eye response to the optical signal intensity nonlinearly.



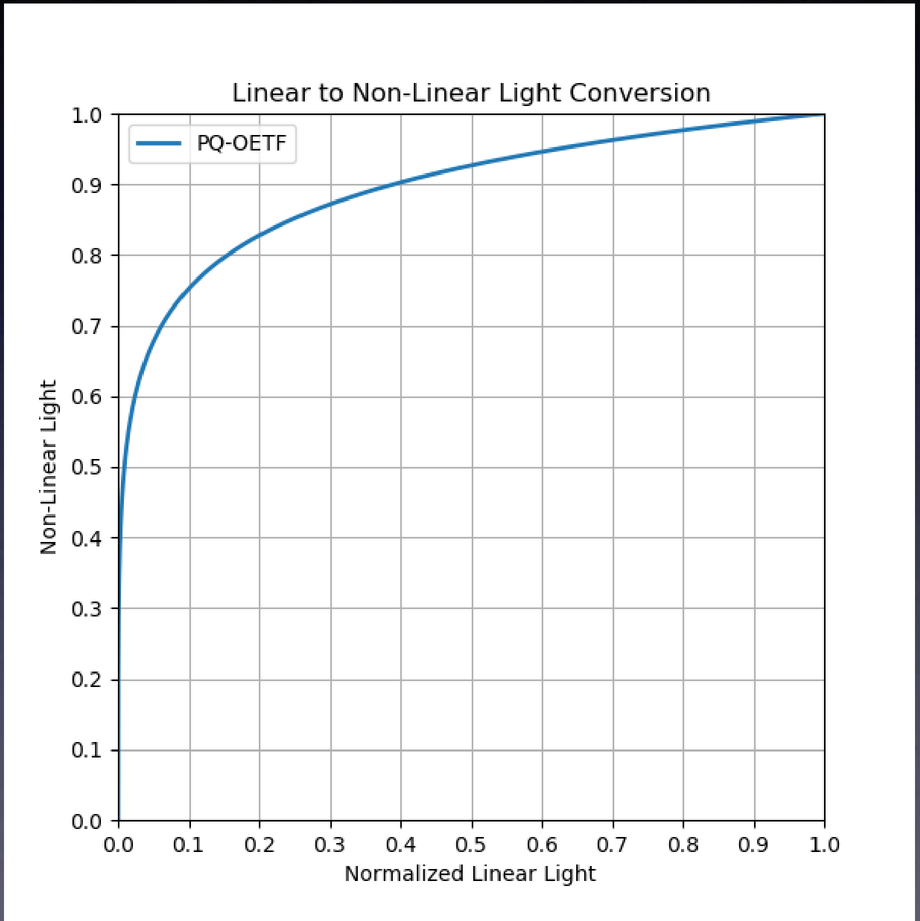
HDR Image/Video **Acquisition** Stage



PQ(Perceptual Quantization) OETF

$$N = \left(\frac{c_1 + c_2 L^{m_1}}{1 + c_3 L^{m_1}}\right)^{m_2}$$

N = non-linear light
L = normalized linear light [0, 1.0]
$c_1 = 0.8539375$
$c_2 = 18.8515625$
$c_3 = 18.6875$
$m_1 = 0.1593017578125$
$m_2 = 78.84375$



Problem

- OETF requires a lot of floating-point operations such as power, log, and division.
    - impractical for real time image/video acquisition

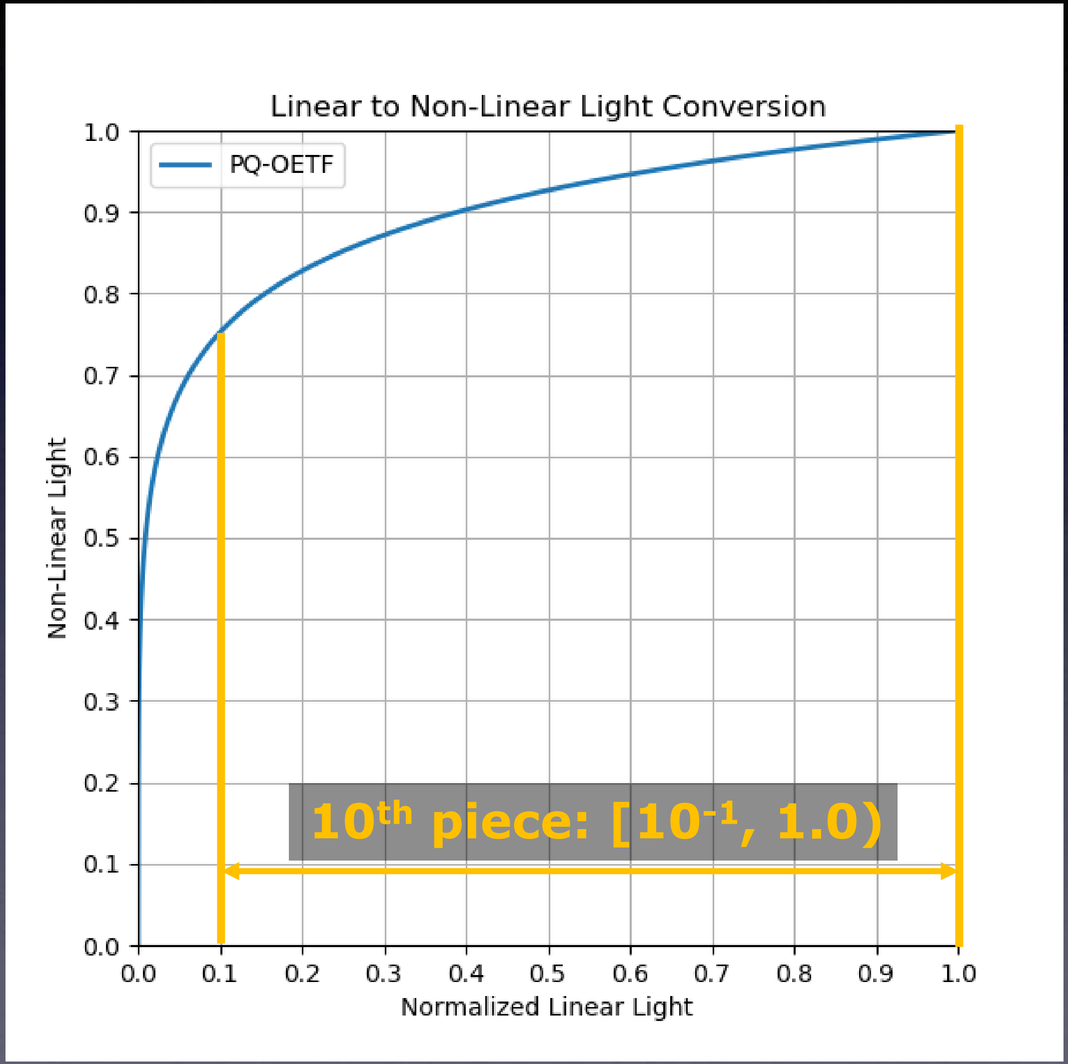
Related Works

- [1] proposed a look-up table based interpolation method to reduce computational complexity.

Related Works

- [1] proposed a look-up table based interpolation method to reduce computational complexity.

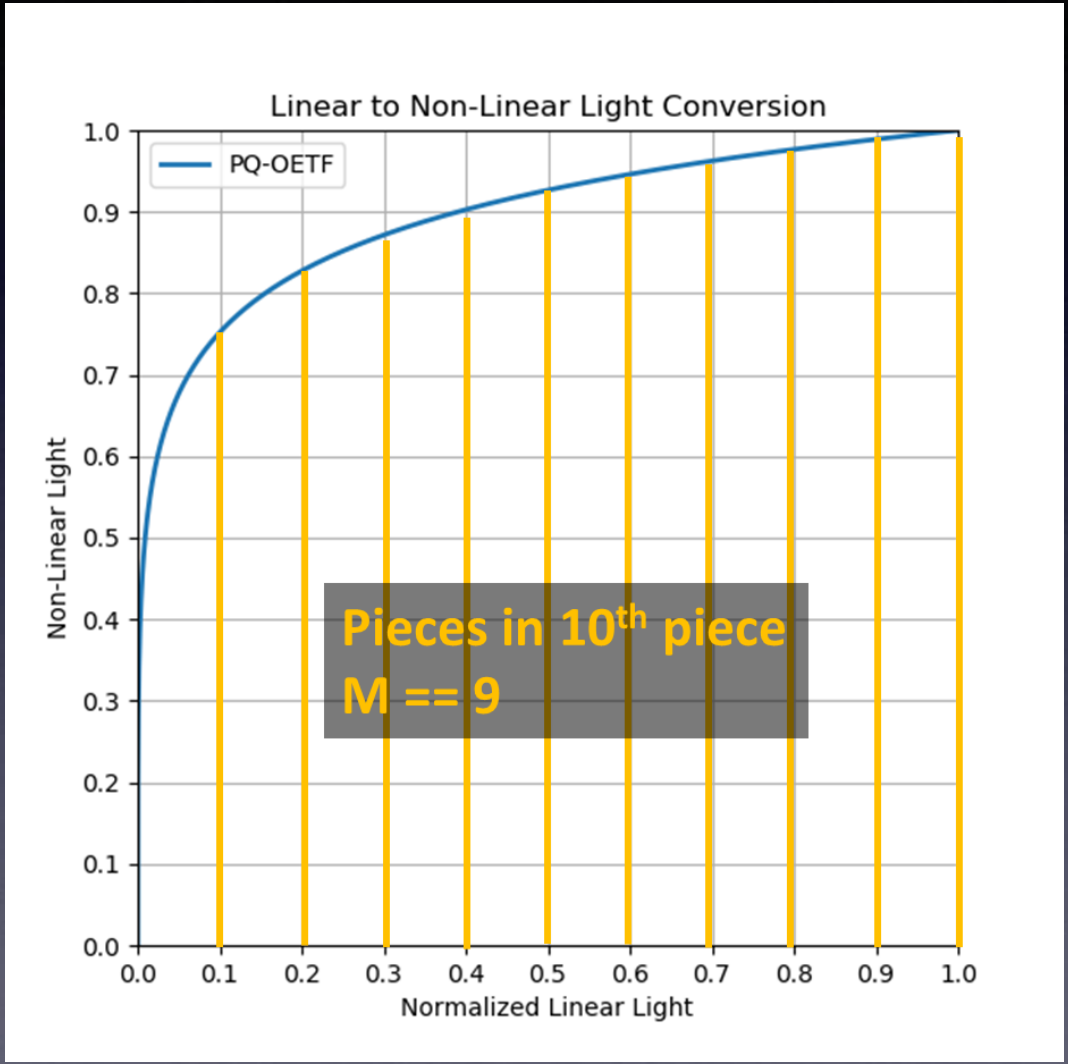
  - **First**, the normalized linear light value is divided into **10 pieces in $\log_{10}$ scales.**
    - [0.0, 10^{-9}), [10^{-9}, 10^{-8}), …, [10^{-1}, 1.0)



Related Works

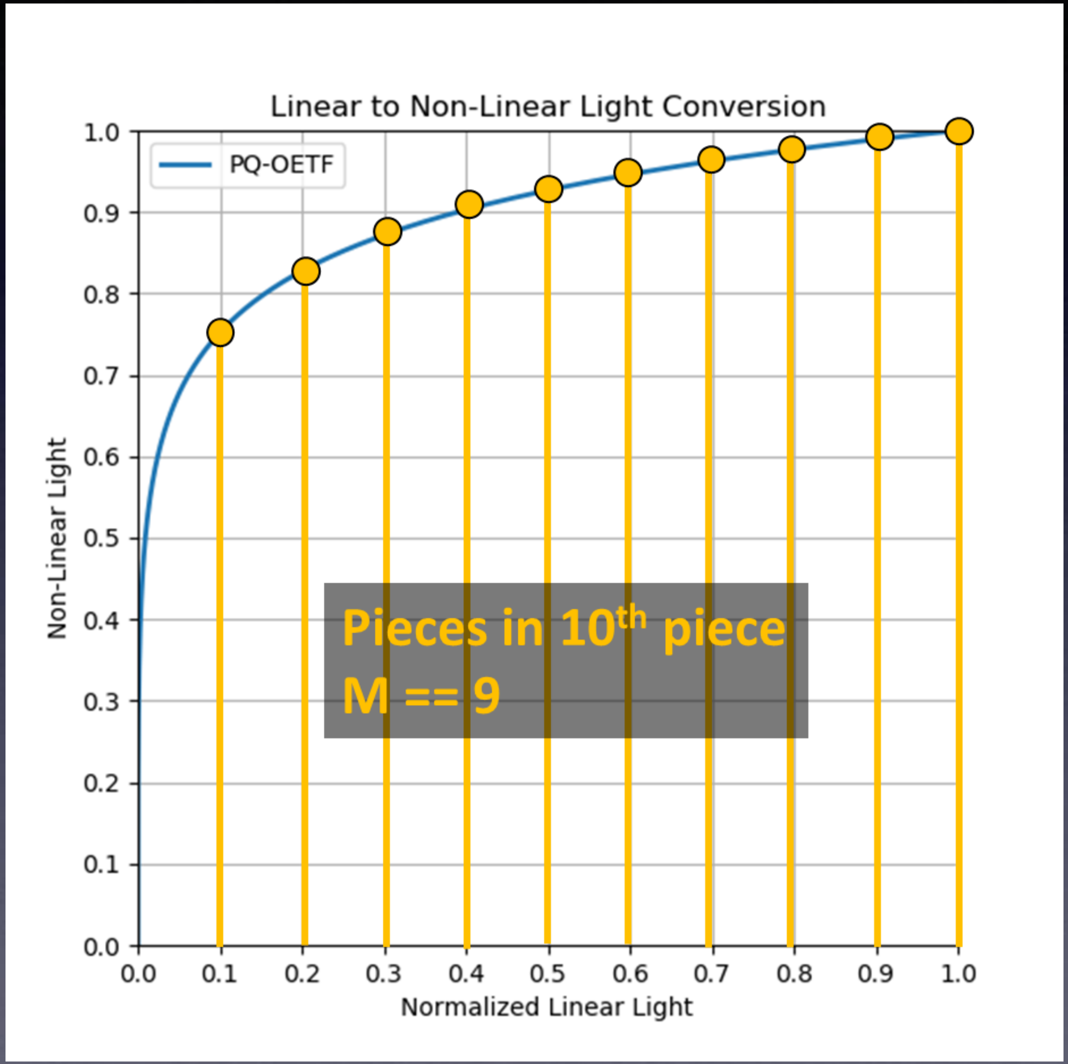
- [1] proposed a look-up table based interpolation method to reduce computational complexity.

  - **Second**, each piece is again divided into *M* **equal pieces**.



Related Works

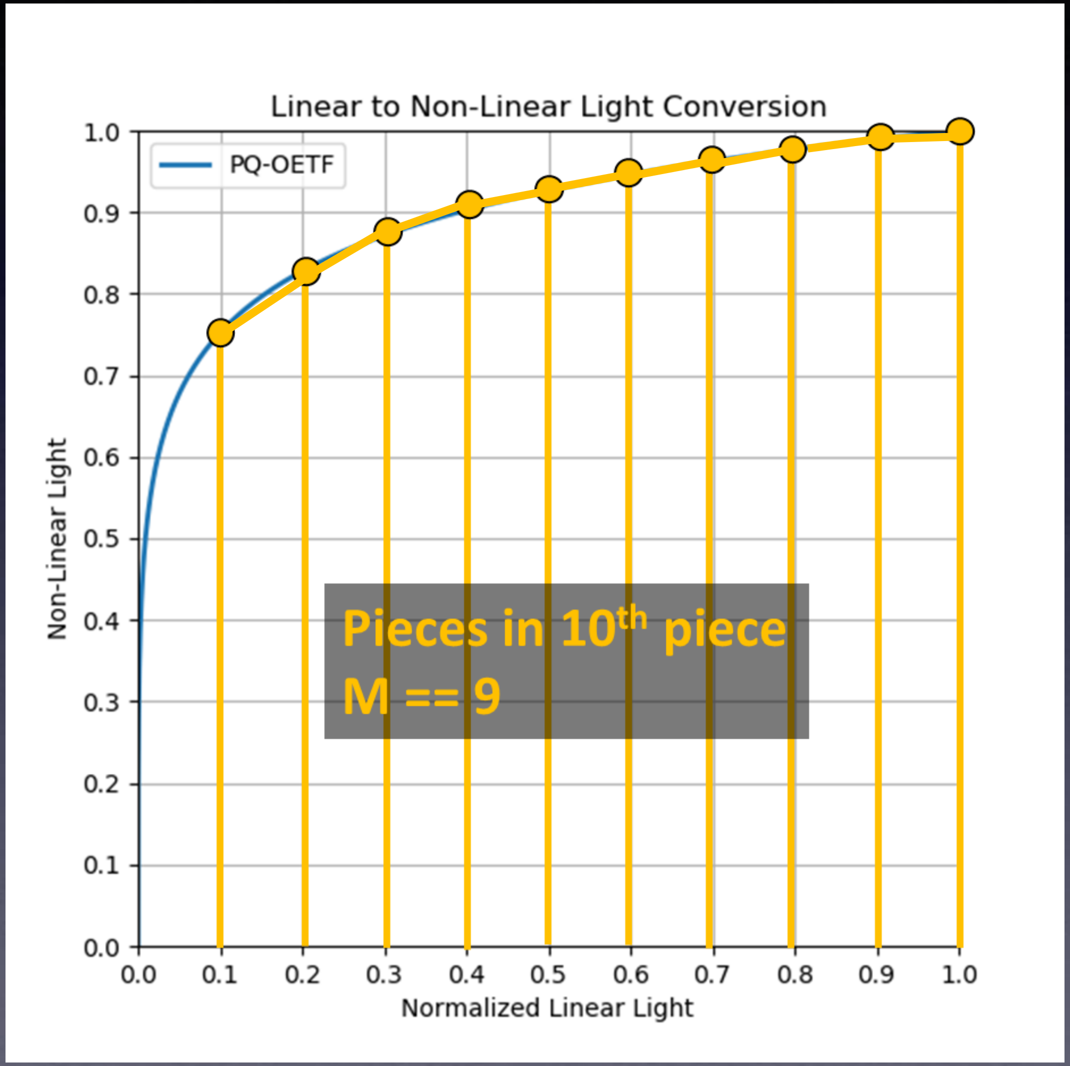
- [1] proposed a look-up table based interpolation method to reduce computational complexity.



- **Then**, for each piece, *M* uniformly spaced lookup table entries could be specified.

Related Works

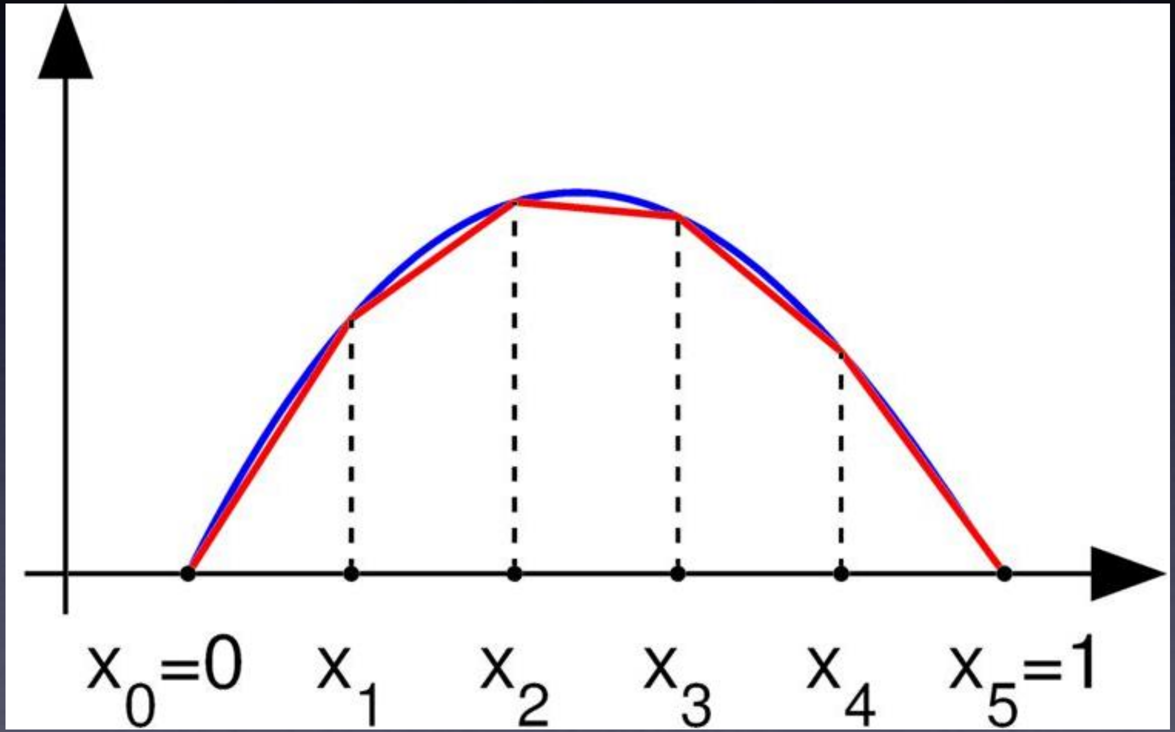
- [1] proposed a look-up table based interpolation method to reduce computational complexity.



- **As a final stage**, and to further improve precision, **linear interpolation** is used.

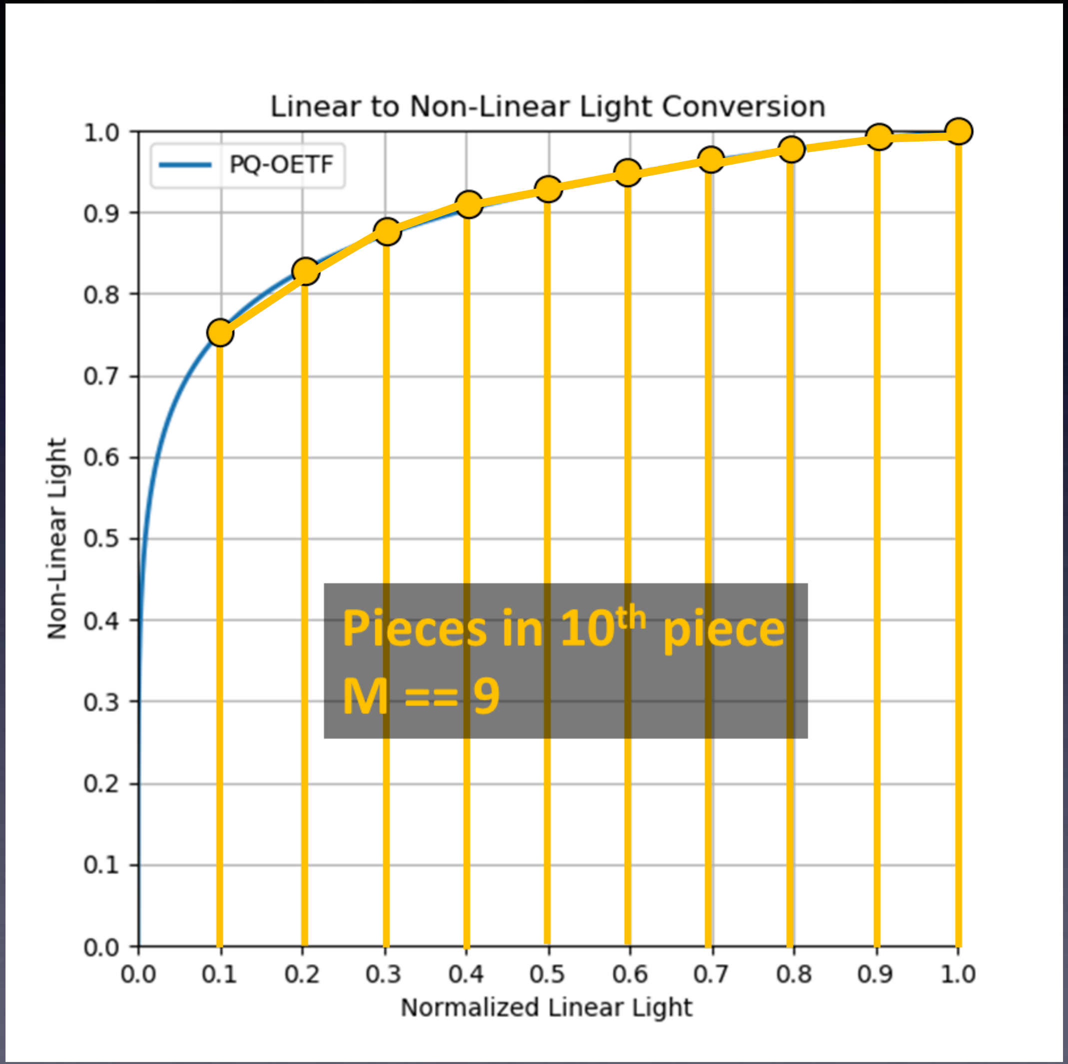
Related Works

- The method proposed in [1] can be considered as piecewise linear approximation(PLA).

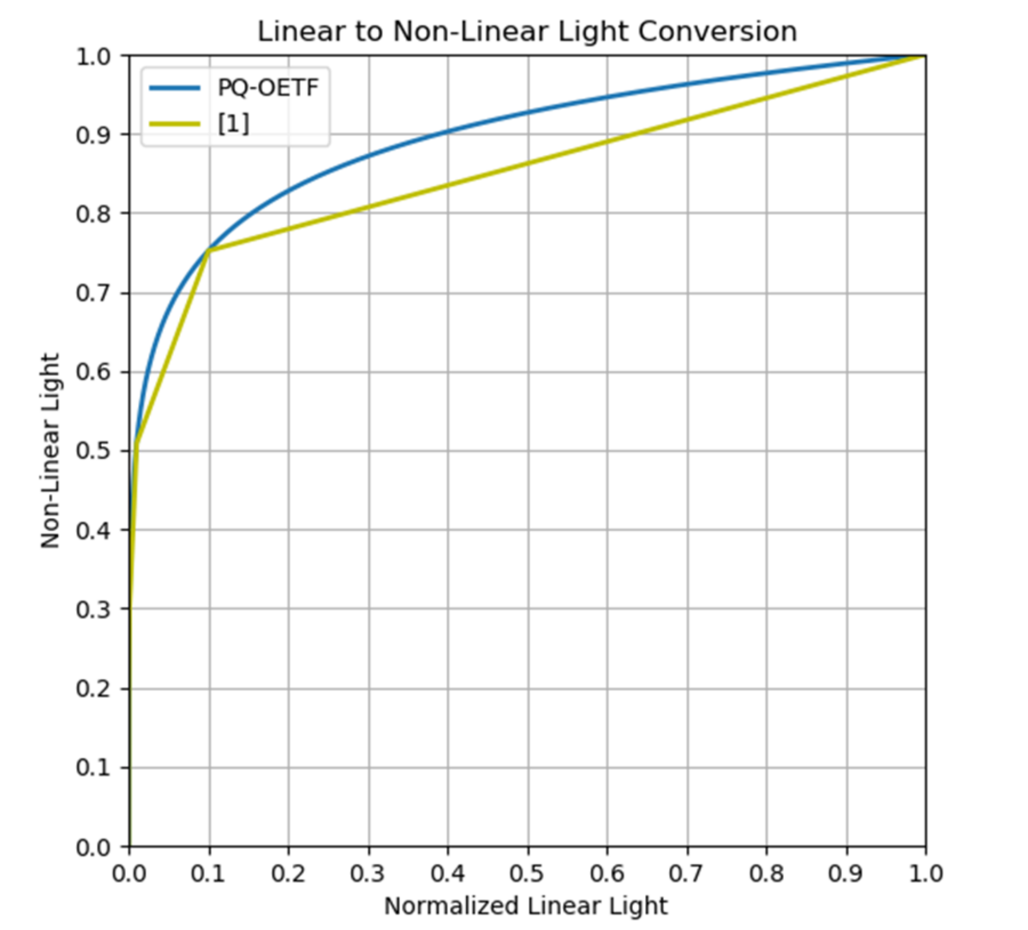


piecewise linear approximation

=



Related Works - Problem



- [1] divided a function into **10 pieces in $\log_{10}$ scales.**
  - $[0.0, 10^{-9}), [10^{-9}, 10^{-8}), \ldots, [10^{-1}, 1.0)$

→ Optimal for PLA ?

We propose the method to search multiple pieces optimal for PLA using genetic algorithm.

- We encode the boundary values of each piece and define several standard genetic operation, i.e., selection, crossover and mutation.

Encoding

- **Real-valued Encoding**

Multiple pieces		Encoded pieces
[0, 0.1)		0
[0.1, 0.2)		0.1
[0.2, 0.3)		0.2
[0.3, 0.4)		0.3
[0.4, 0.5)	Encoding →	0.4
[0.5, 0.6)		0.5
[0.6, 0.7)		0.6
[0.7, 0.8)		0.7
[0.8, 0.9)		0.8
[0.9, 1.0)		0.9
		1.0

We encode the boundary values of each piece.

Encoding

- ## Real-valued Encoding

Encoded pieces

0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0

→ individual[ ] = { 0., 0.1, … , 0.9, 1,0}

Each value of encoded pieces is stored to array.

Initialization

- **First**, the normalized linear light value is divided into **7 pieces in $\log_{10}$ scales.**
  - $[10^{-7}, 10^{-6}), [10^{-6}, 10^{-5}), \ldots, [10^{-1}, 1.0)$
  - The output of PQ OETF with input lower than 10^{-7} is 0.
    - So, we neglect the normalized light value lower than 10^{-7}.

individual[ ] = $\{10^{-7}, 10^{-6}, \ldots, 10^{-1}, 1\}$

Initialization

- **2. 10^{-7} 과 1.0 을 제외한 각 경계 값은 아래의 Pseudo-code 에 따라 재정의된다.**

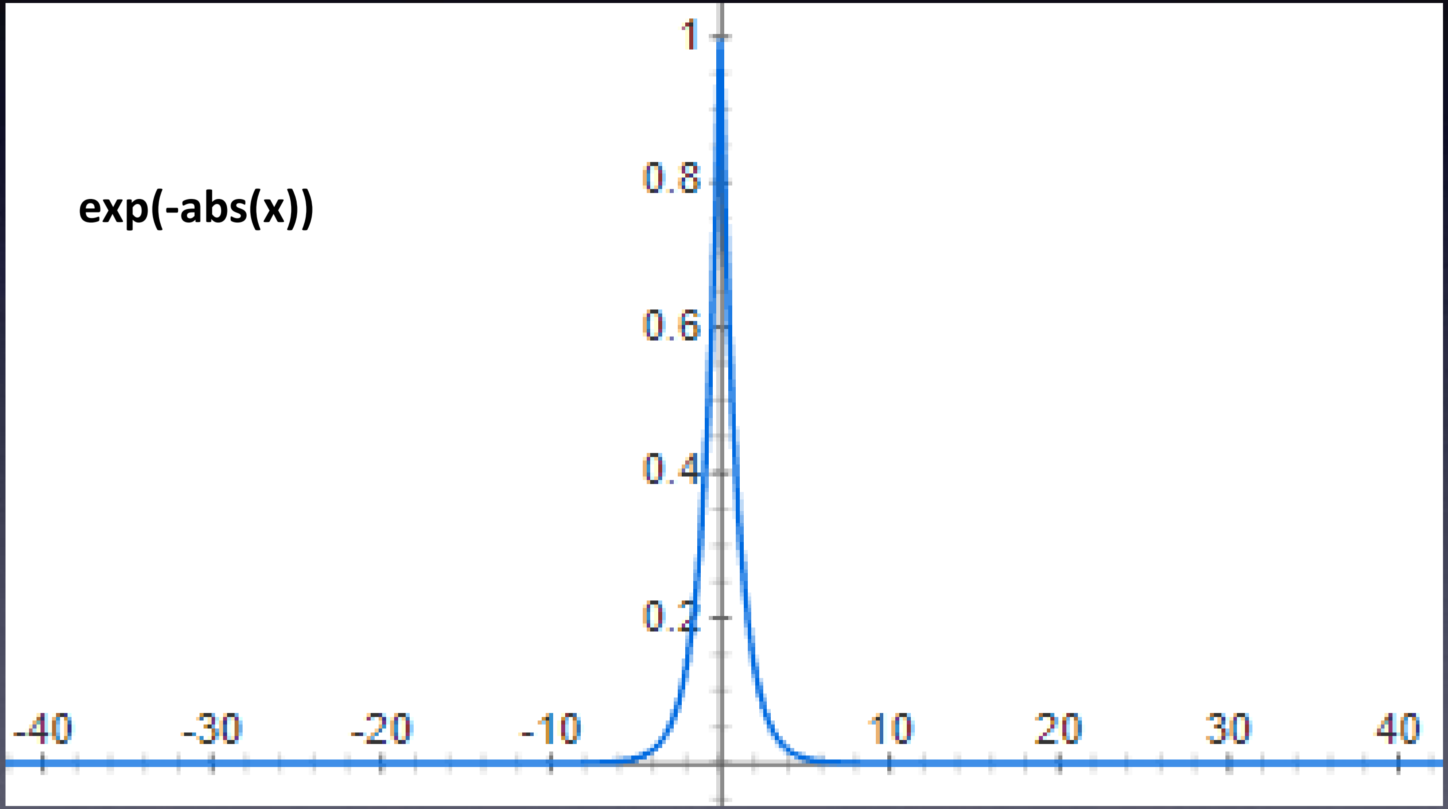
```
individual[ ] = {10^-7, 10^-6, … , 10^-1, 1} // boundary values of each piece

for( i = 1 to length(individual)-1) // zero-based index
{
    p= individual[i-1] //prev_boundary_value
    c = individual[i] //cur_boundary_value
    n = individual[i+1] //next_boundary_value

    individual[i] = rand() // [(c+p)/2, (c+n)/2]
}
```

Fitness

Fitness = $\sum exp^{-|PQ\_OETF - PLA\_PQ\_OETF|}$



Selection

- **Roulette-Wheel Selection**

Cross Over

- 각 individual 이 갖는 경계 값 들을 인덱스에 맞게 서로 교환
  - 어떤 individual 의 값을 물려받을지는 50% 확률로 무작위 선택됨
  - 10^{-7}, 1 은 변하지 않음

parentA[ ] = {10^{-7}, 0.001, 0.002 , 0.097 , 0.12 , 0.47 , 0.78, 1}

parentB[ ] = {10^{-7}, 0.003, 0.024 , 0.081 , 0.25 , 0.67 , 0.84, 1}

Offspring = CrossOver(parentA, parentB)

Offspring = {10^{-7}, 0.003, 0.024 , 0.097, 0.25 , 0.47, 0.84, 1}

Mutation
아래의 Pseudo code 에 따라 mutation 적용

- **10^{-7}, 1 은 변하지 않음**

```
individual // boundary values of each piece

for( i = 1 to length(individual)-1) // zero-based index
{
    p= individual[i-1] //prev_boundary_value
    c = individual[i] //cur_boundary_value
    n = individual[i+1] //next_boundary_value

    individual[i] = rand() // [c - (c-p)/100.0 , c + (n-p)/100.0]
}
```

Algorithm

```
NUM_INDIVIDUALS = 5
NUM_GENERATIONS = 100

Individuals = Initialize(NUM_INDIVIDUALS)

for (gen = 0 to NUM_GENERATIONS-1)
{
    GET_FITNESS

    NewIndividuals = null

    for(idx_offspring = 0 to NUM_INDIVIDUALS-1)
    {
        Offspring = SELECTION + CROSS_OVER +MUTATION
        NewIndividuals.push(Offspring)
    }
    Individuals = NewIndividuals
}
```

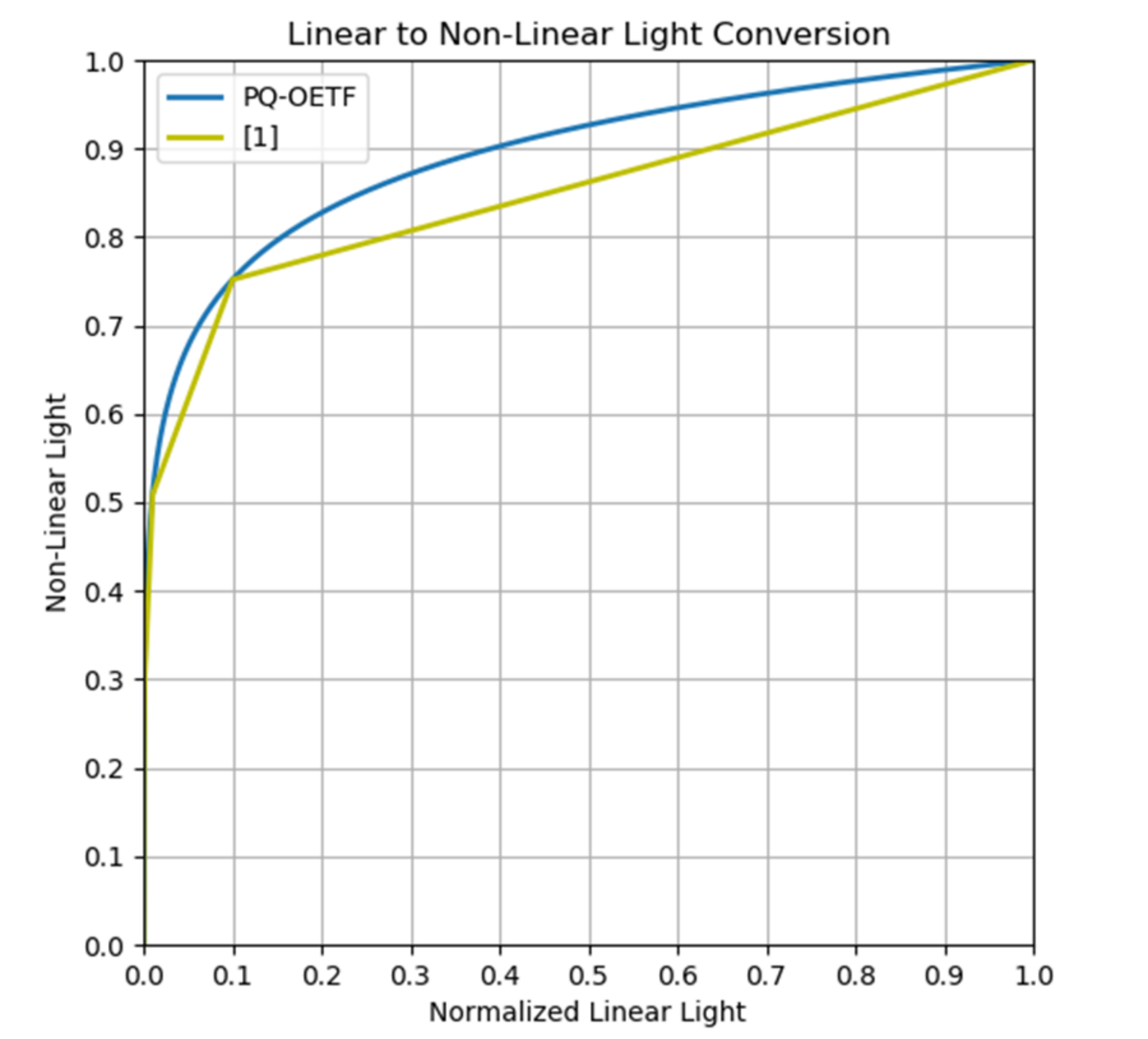
Experimental Results

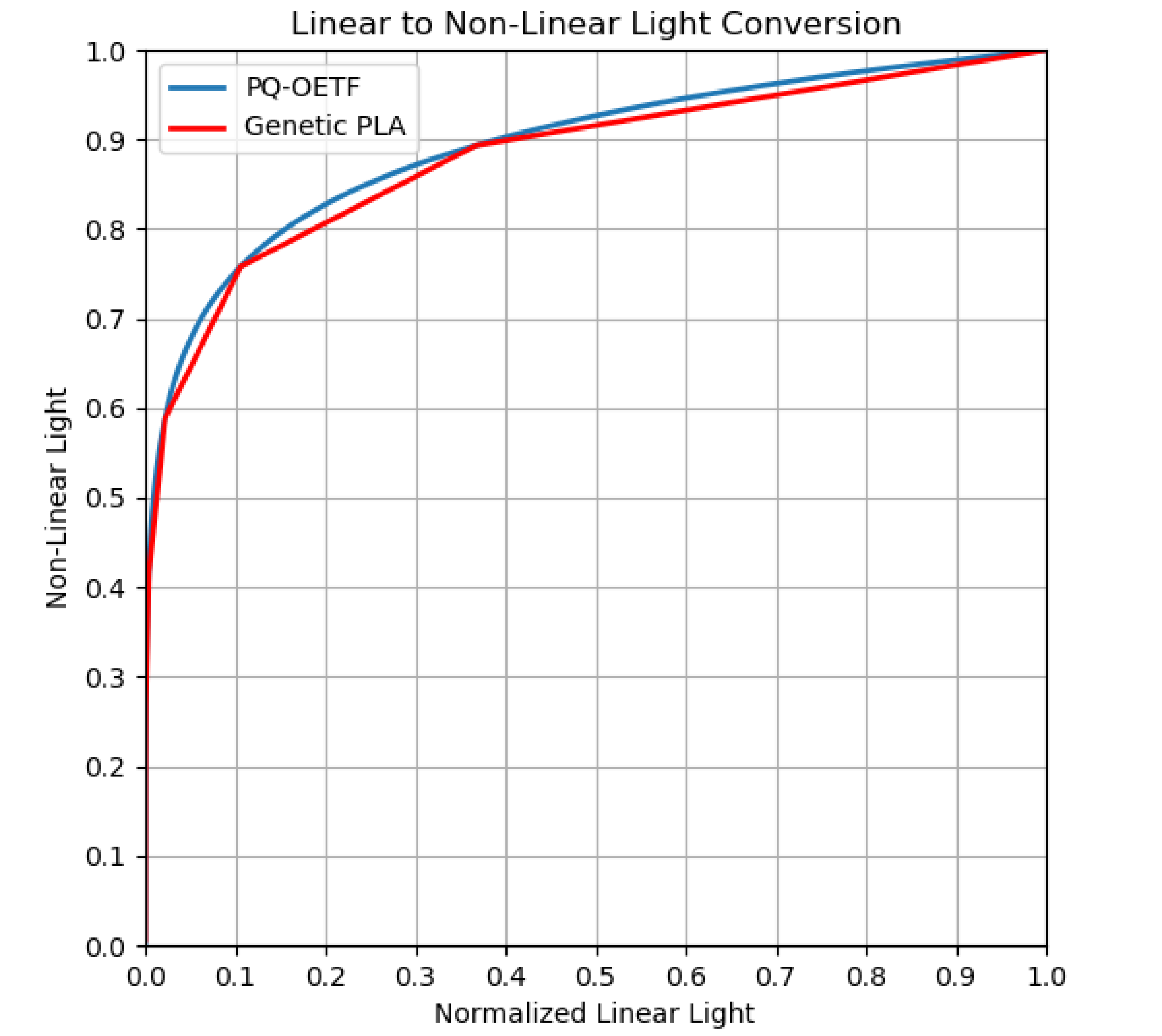
Generation	Average Fitness of individuals	Average MAE of individuals
1	58525	1695
2	58547	1673
3	58512	1695
4	58654	1597
5	58719	1560
6	58680	1571
…	…	
100	**58806**	**1299**

Best MAE of all individuals: 1297

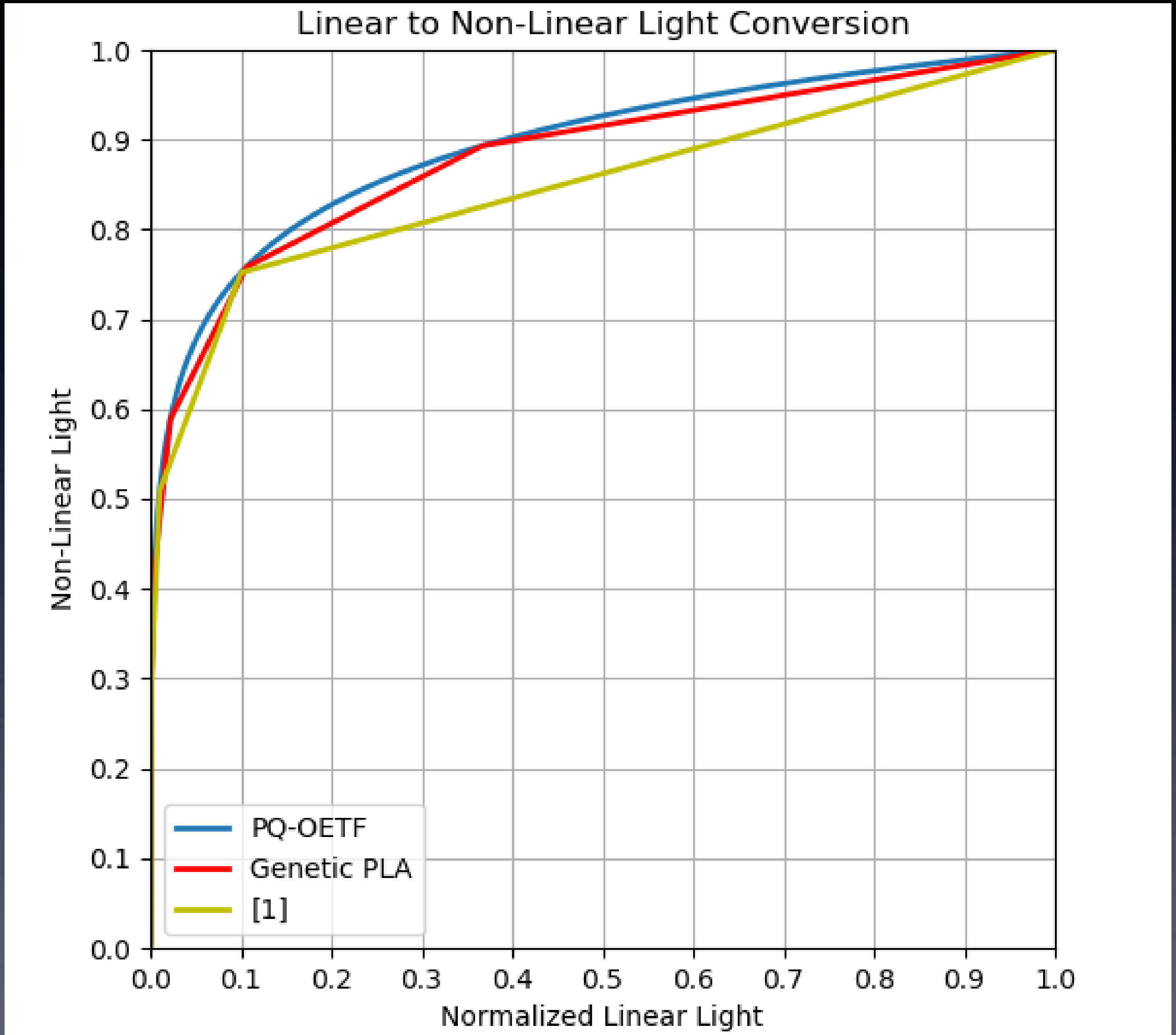
- **Generation: 96**

Experimental Results





Experimental Results



Conclusions

When approximating function using PLA, we can search multiple pieces optimal for PLA using genetic algorithm.

All the code is online at

https://github.com/developer0hye/Genetic-Piecewise-Linear-Approximation

Reference

[1] J. Ström, K. Andersson, M. Pettersson, P. Hermansson, J.Samuelsson, A. Segally, J. Zhaoy, S.-H. Kimy, K. Misray, A. M. Tourapisz, Y. Suz and D. Singerz, "High quality HDR video compression using HEVC main 10 profile," 2016 Picture Coding Symposium (PCS), Nuremberg, 2016. DOI:10.1109/PCS.2016.7906372