Lloyd Max Algorithm for Gray value quantization

Image processing, Retrieval, and Analysis II
Project 01 - 1

Quantization by Lloyd Max Algorithm

Find optimal quantization intervals and points which represent original image with given levels

Process:

initialize quantization intervals $a_1 = x_{\min},\ a_i = a_{i-1} + \frac{x_{\max} - x_{\min}}{L},\ a_{L+1} = x_{\max}$

initialize quantization points $b_i = \frac{a_i + a_{i-1}}{2}$

initialize iteration counter $t = 0$

compute $E_t = \sum_{i=1}^{L} \int_{a_i}^{a_i+1} (x - b_i)^2 p(x)\, dx$

repeat

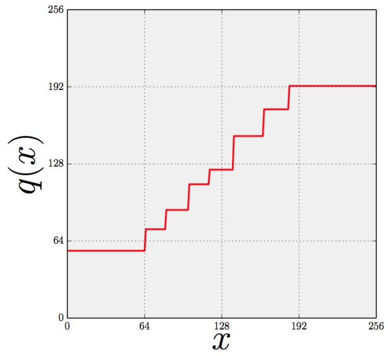
$t = t + 1$

update quantization intervals $a_i = \frac{b_i + b_{i-1}}{2}$

update quantization points $b_i = \frac{\int_{a_i}^{a_i+1} x\, p(x)\, dx}{\int_{a_i}^{a_i+1} p(x)\, dx}$

compute $E_t = \sum_{i=1}^{L} \int_{a_i}^{a_i+1} (x - b_i)^2 p(x)\, dx$

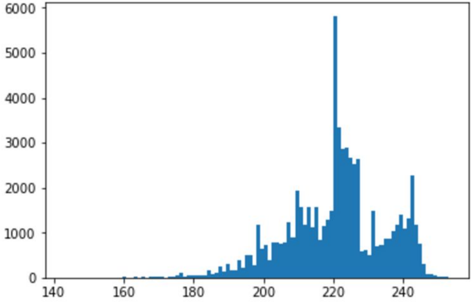
until $|E_t - E_{t-1}| \leqslant \epsilon$ or $t \geqslant t_{\max}$



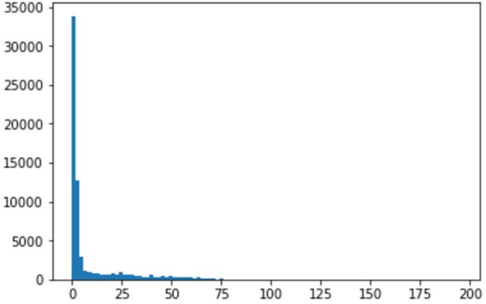
- Transform function q(x)
- Quantized Image (matrix)

Image, Histogram, Probability Density Function

$$p(\lambda) = \frac{h(\lambda)}{\sum_y h(y)}$$



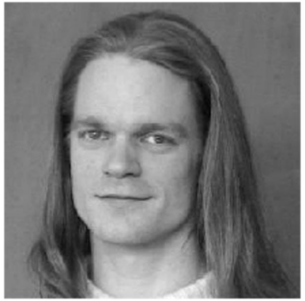
bauckhage_gamma_1

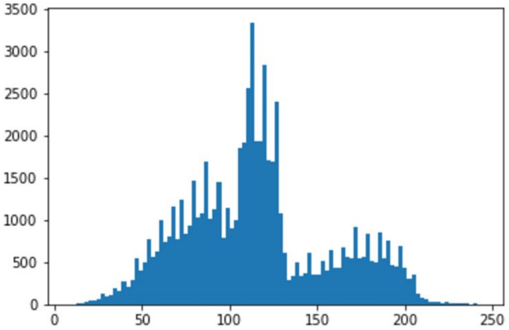


bauckhage_gamma_2

[histogram with bins=100]

Boundaries and Points





Two types of boundary initialization

- Set first and end boundaries to 0 and 256, respectively.

```
Init Boundaries:  [   0.    32.    64.    96.   128.   160.   192.   224.   256.]
Init Certer Points:  [  16.    48.    80.   112.   144.   176.   208.   240.]
```

- Set first and end boundaries to min(g(x)) and max(g(x)), respectively

```
Init Boundaries:  [    8.     37.5    67.     96.5   126.    155.5   185.    214.5   244. ]
Init Certer Points:  [  22.75    52.25    81.75   111.25   140.75   170.25   199.75   229.25]
```

Boundaries and Points

Iterate Update boundaries and points until…

- Boundaries: $$a_\nu = \frac{b_\nu + b_{\nu-1}}{2}$$ if and only if (b_v != 0 and b_v-1 != 0)
- Points:

$$b_\nu = \frac{\int_{a_\nu}^{a_{\nu+1}} \lambda p(\lambda) d\lambda}{\int_{a_\nu}^{a_{\nu+1}} p(\lambda) d\lambda}$$

```
Init Boundaries:  [   0.   32.   64.   96.  128.  160.  192.  224.  256.]
Init Certer Points:  [  16.   48.   80.  112.  144.  176.  208.  240.]
Computed Error:  81.5736236572

Iteration: 1
Boundaries: [   0.   32.   64.   96.  128.  160.  192.  224.  256.]
Points: [  25.65057471   52.38218054   80.55665354  113.58613037  141.14585682
  175.91886434  199.61624396  230.51086957]
Computed Error:  73.6048634981

Iteration: 2
Boundaries: [   0.           39.01637763   66.46941704   97.07139195  127.3659936
  158.53236058  187.76755415  215.06355676  256.        ]
Points: [  30.59301015   55.1087129    81.83696546  113.42054869  138.64154216
  172.57586299  196.06087715  224.80769231]
Computed Error:  64.9355310102
```

Error computation

Mean squared quantization error

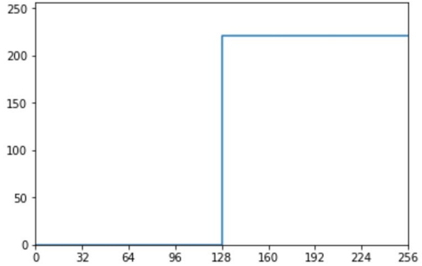
$$E = \sum_{\nu=1}^{L} \int_{a_\nu}^{a_{\nu+1}} (\lambda - b_\nu)^2 p(\lambda) d\lambda$$

Iteration condition

- iteration Count < max Iteration count
- Computed Error > threshold_1
- Difference between current error and previous error > threshold_2

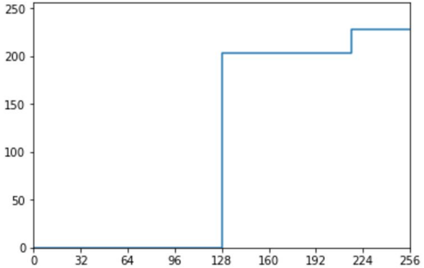
Results - Init boundaries from 0 to 256

bauckhage_gamma_1



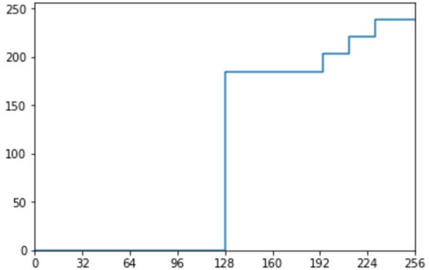
level = 2

Iteration: 2
Error : 197.30



level = 4

Iteration: 8
Error : 76.77

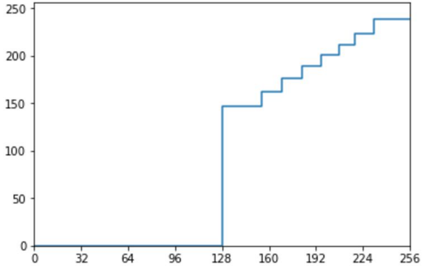


level = 8

Iteration: 17
Error : 23.78

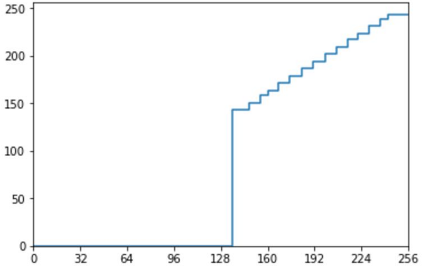
Results - Init boundaries from 0 to 256

bauckhage_gamma_1



level = 16

Iteration: 5
Error : 11.50

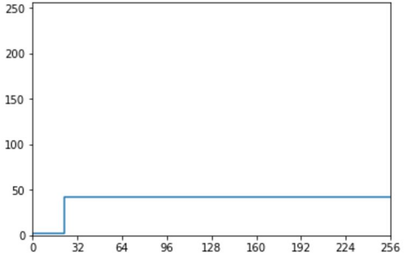


level = 32

Iteration: 4
Error : 4.29

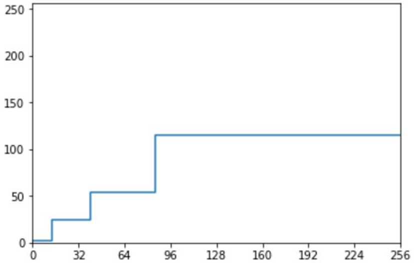
Results - Init boundaries from 0 to 256

bauckhage_gamma_2



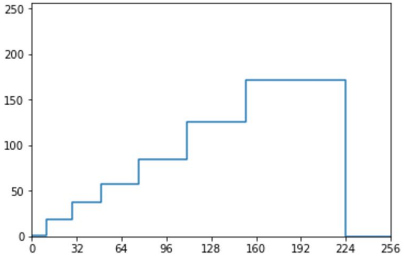
level = 2

Iteration: 9
Error : 64.40



level = 4

Iteration: 12
Error : 20.07

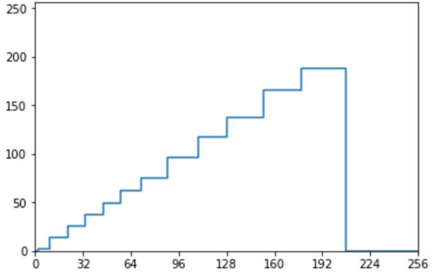


level = 8

Iteration: 12
Error : 9.90

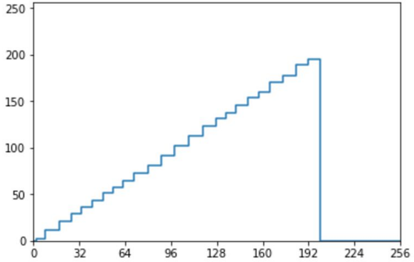
Results - Init boundaries from 0 to 256

bauckhage_gamma_2



level = 16

Iteration: 20
Error : 3.8

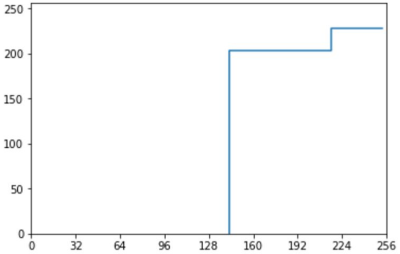


level = 32

Iteration: 7
Error : 2.06

Results - Init boundaries from min(g(x)) to max(g(x))

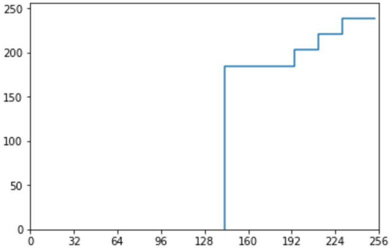
bauckhage_gamma_1



level = 2

Iteration: 7

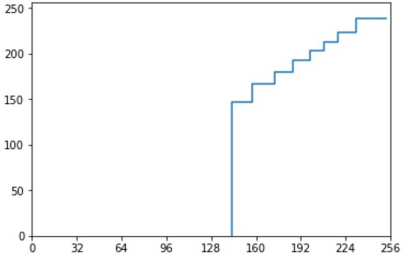
Error : 76.77



level = 4

Iteration: 12

Error : 23.77



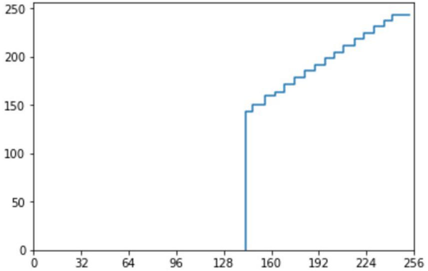
level = 8

Iteration: 6

Error : 10.76

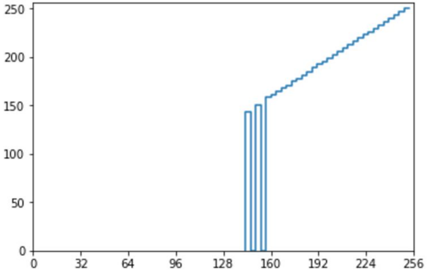
Results - Init boundaries from min(g(x)) to max(g(x))

bauckhage_gamma_1



level = 16

Iteration: 4
Error : 3.41

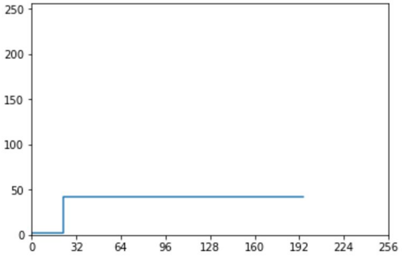


level = 32

Iteration: 1
Error : 0.93

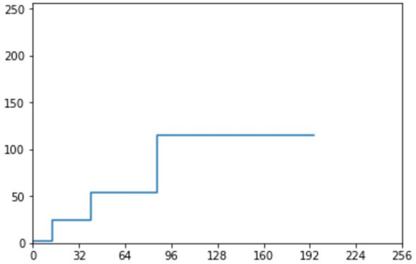
Results - Init boundaries from min(g(x)) to max(g(x))

bauckhage_gamma_2



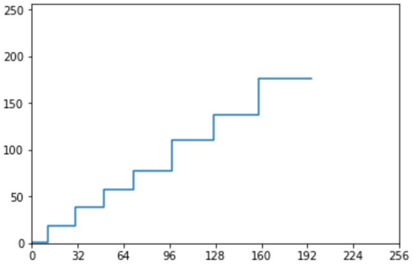
level = 2

Iteration: 8
Error : 64.05



level = 4

Iteration: 10
Error : 19.97

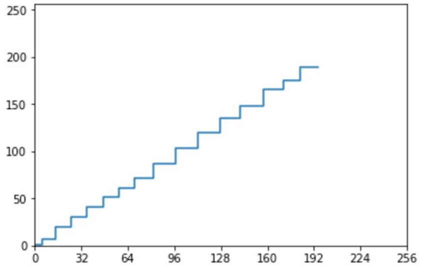


level = 8

Iteration: 8
Error : 9.56

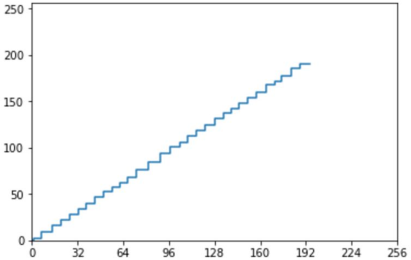
Results - Init boundaries from min(g(x)) to max(g(x))

bauckhage_gamma_2



level = 16

Iteration: 10
Error : 3.58



level = 32

Iteration: 5
Error : 1.32

Histogram transformation

Image processing, Retrieval, and Analysis II
Project 01 - 2

Transform to Weibull distribution

We know...

$$P_Y(y) = \int_0^y p_Y(z)dz = \int_0^y p_X(T^{-1}(z)) \left| \frac{d}{dz} T^{-1}(z) \right| dz$$

$$P_Y(y) = 1 - exp\{-(\frac{y}{l})^k\}$$

So..

$$\begin{aligned} &1 - exp\{-(\frac{y}{l})^k\} \\ &= \int_0^y p_X(T^{-1}(z)) \left| \frac{d}{dz} T^{-1}(z) \right| dz \\ &= \int_0^{T^{-1}(y)} p_X(z)dz \end{aligned}$$

Transform to Weibull distribution

Continue..

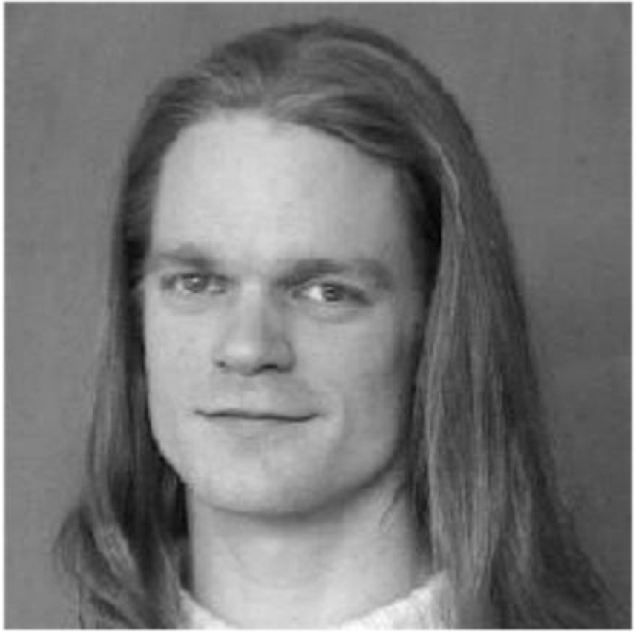
$$1 - exp\{-(\frac{T(x)}{l})^k\} \ = \int_0^x p_X(z)dz = \widetilde{H}(x)$$

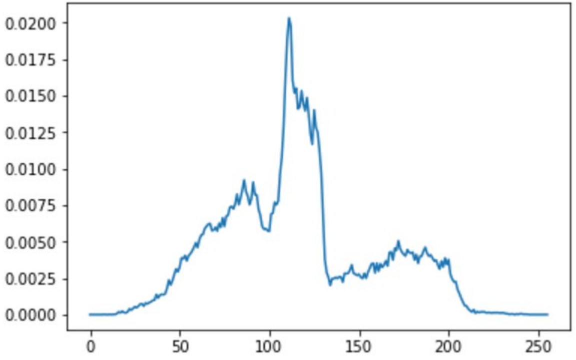
$$exp\{-(\frac{T(x)}{l})^k\} \ = 1 - \widetilde{H}(x)$$

$$(\frac{T(x)}{l})^k \ = ln(\frac{1}{1-\widetilde{H}(x)})$$

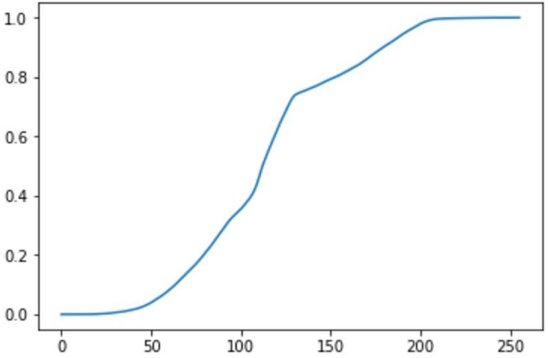
$$T(x) = l \cdot \{ln(\frac{1}{1-\widetilde{H}(x)})\}^{\frac{1}{k}}$$

Demo



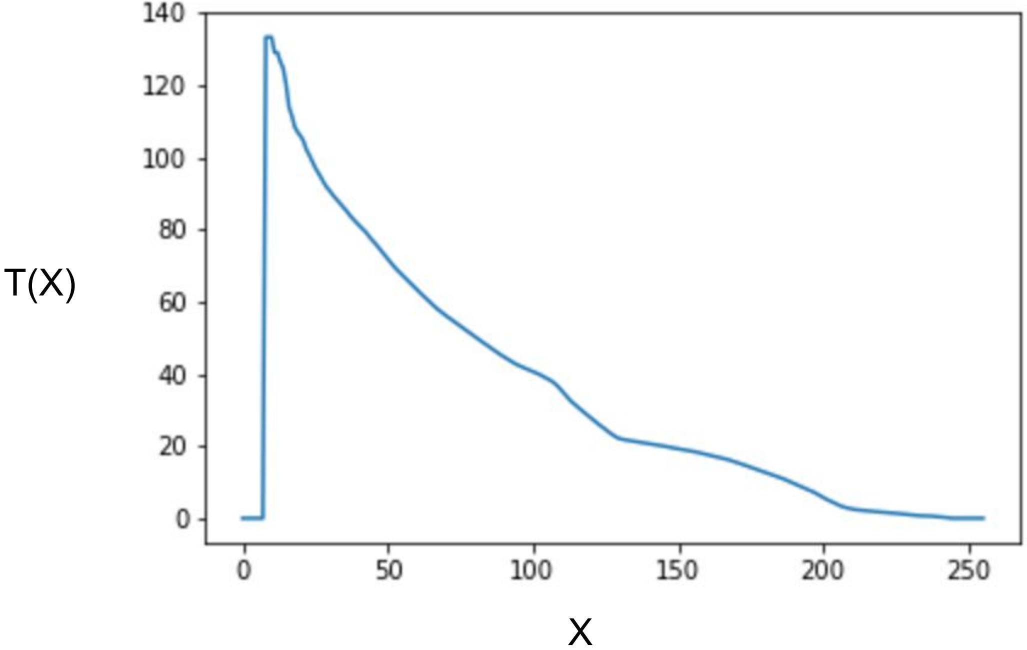

0.0200
0.0175
0.0150
0.0125
0.0100
0.0075
0.0050
0.0025
0.0000
0
50
100
150
200
250

PDF


1.0
0.8
0.6
0.4
0.2
0.0
0
50
100
150
200
250

CDF

Demo



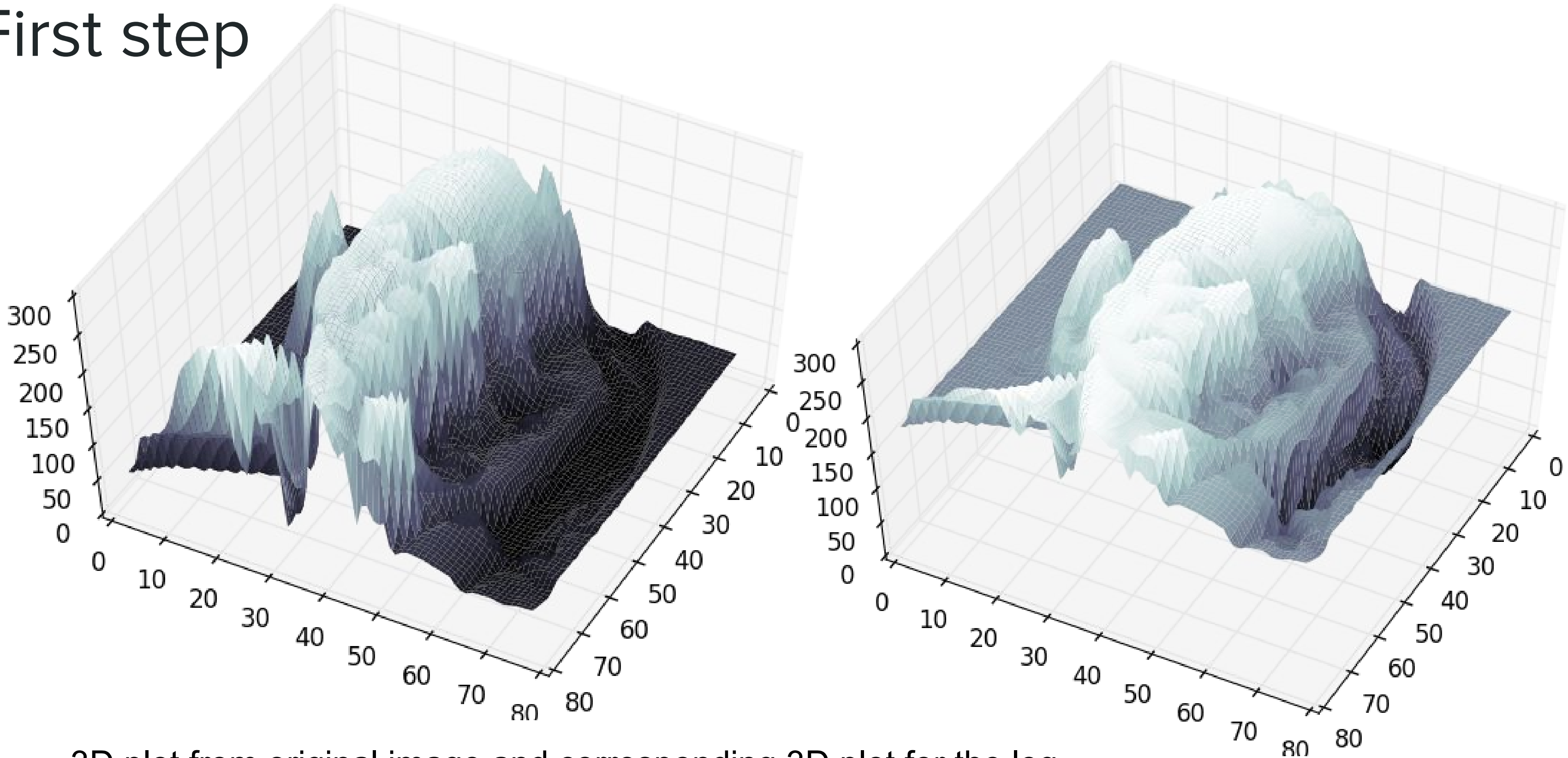
Illumination compensation

Image processing, Retrieval, and Analysis II
Project 01 - 3

Illumination compensation

- We get some image with decompensated illumination
- Our goal is to make more uniform the illumination

First step



3D plot from original image and corresponding 3D plot for the log

Second step: create the models

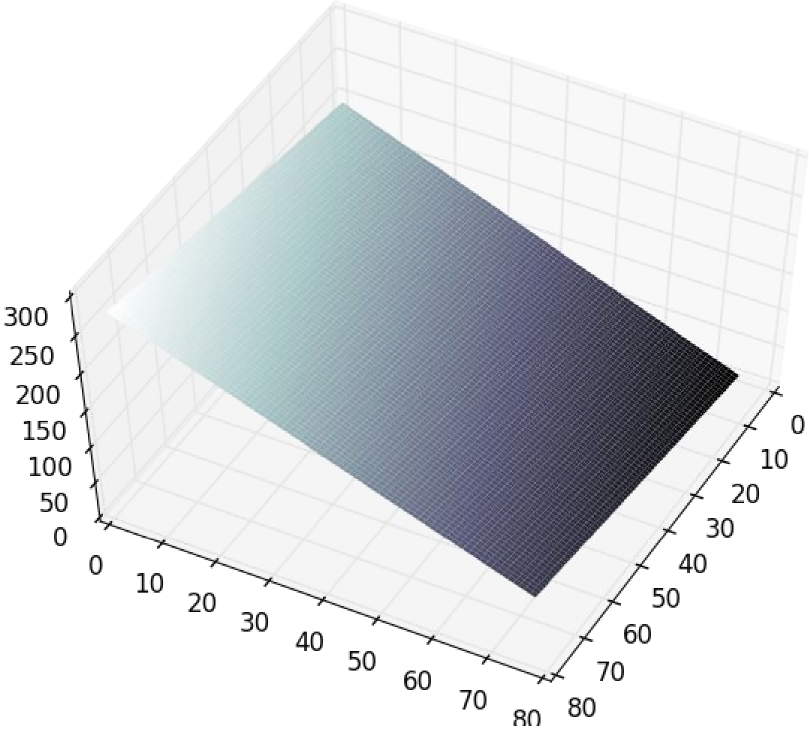
```
def linear_fitting(l):
    rows, cols = l.shape
    row_arr = np.array(range(rows))
    col_arr = np.array(range(cols))
    ones = np.ones(rows*cols)
    x1 = np.repeat(row_arr, cols)
    x2 = np.tile(col_arr, rows)

    X = np.vstack([x1, x2, ones]).T

    z = l.flatten()
    w = lsq_solution_V3(X, z)

    return np.reshape(np.dot(X, w), (rows, cols))


def bilinear_fitting(l):
    rows, cols = l.shape
    row_arr = np.array(range(rows))
    col_arr = np.array(range(cols))
    ones = np.ones(rows * cols)
    x1 = np.repeat(row_arr, cols)
    x2 = np.tile(col_arr, rows)

    X = np.vstack([x1 * x2, x1, x2, ones]).T

    z = l.flatten()
    w = lsq_solution_V3(X, z)

    return np.reshape(np.dot(X, w), (rows, cols))
```

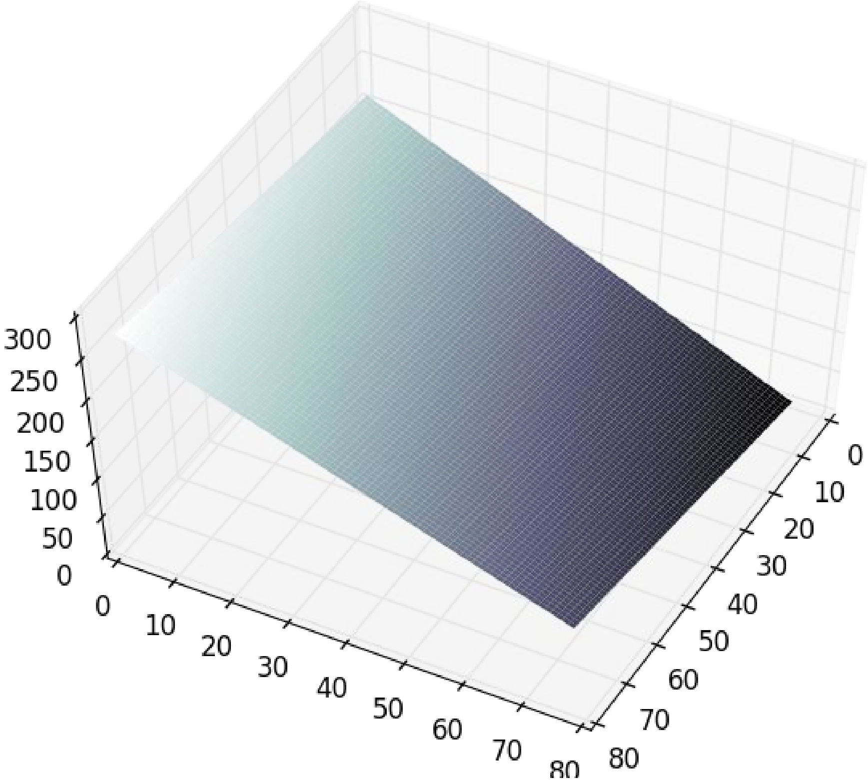
$$i_l(x, y) = ax + by + c$$

$$\boldsymbol{w} = \left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T\boldsymbol{z}$$

$$i_l(x, y) = axy + bx + cy + d$$

3D plots from linear and bilinear models

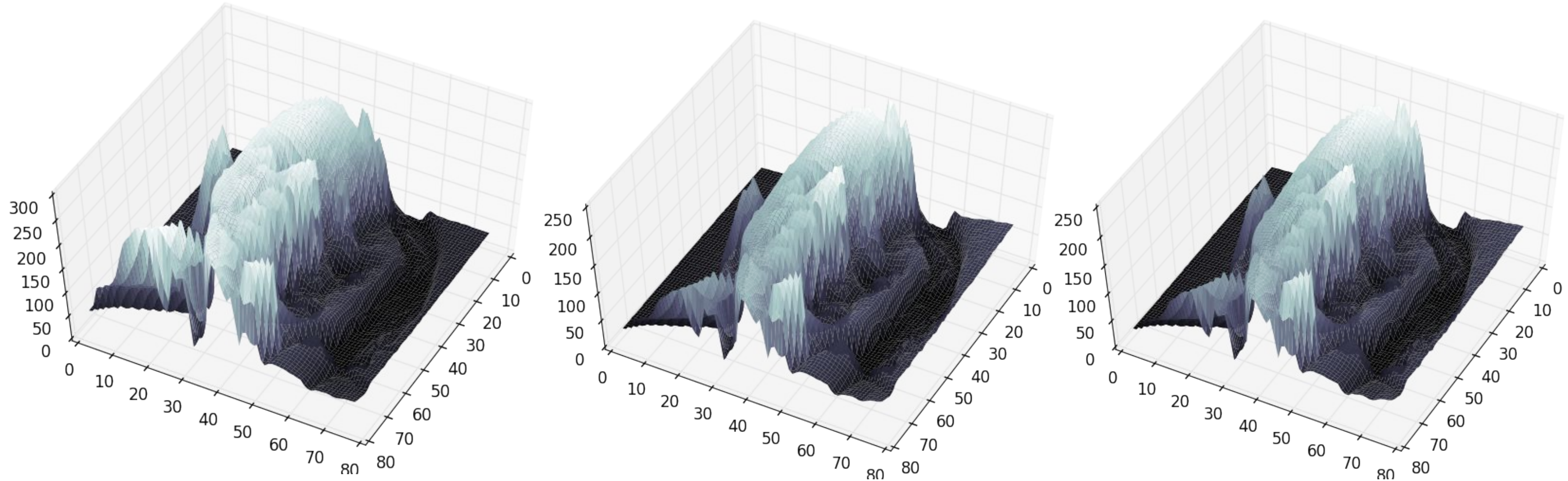




Resulting images

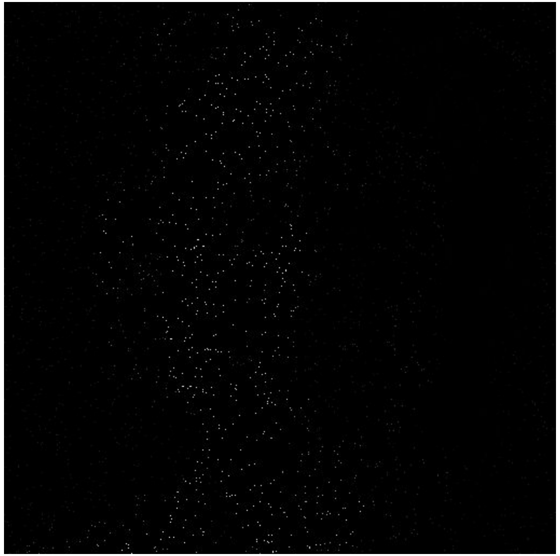
Original image, result of linear model, and result of bilinear model.

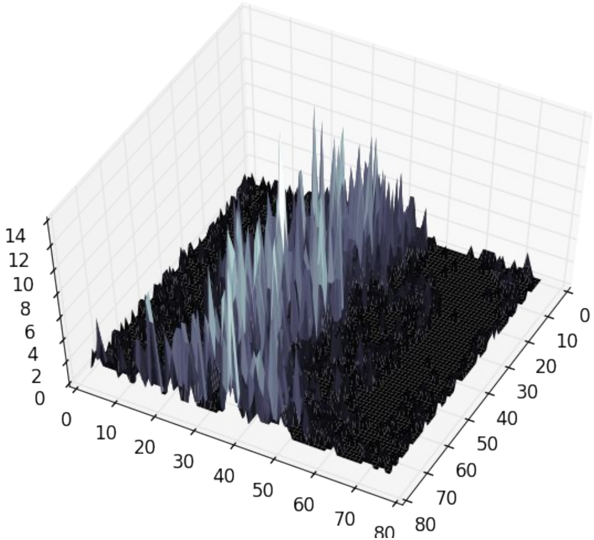
Resulting 3D plots



3D plots for original image, result of linear model, and result of bilinear model.

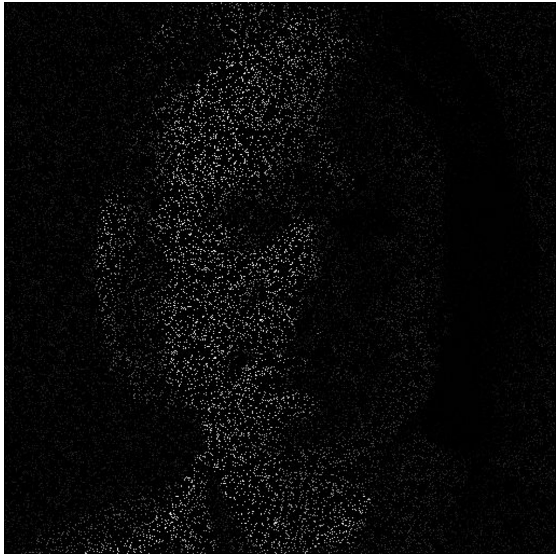
Experiments with different sample size

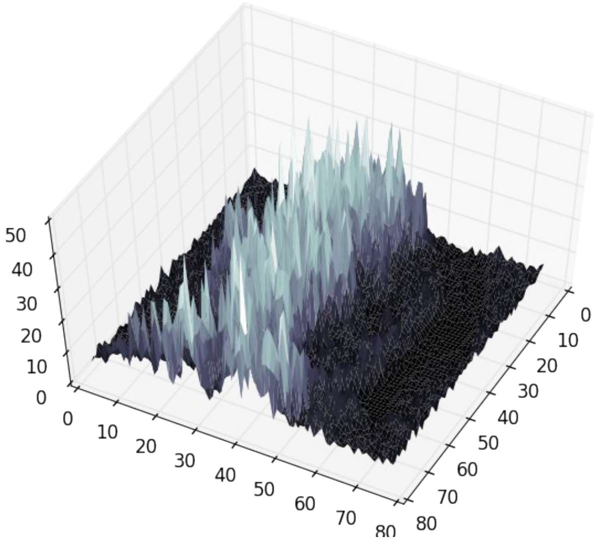




2500 pixels

Experiments with different sample size





25000 pixels