2016/9/29 数字图像处理第一次作业

1 Exercises

1.1 Storage

1. 7 bit planes. Since $128 = 2^7$
2. The first panel is the most visually significant one because the first/higher order bit plane contains the set of the most significant bit.
3. $1024 \times 2048 \times 7 \div 8 = 1835008$ bytes are required for storing this image.
   Proof: "1024 × 2048" means the width is 1024 pixels and the height is 2048 pixels(1024x2048=2Mpixels in total) and each pixel is represented by 7 bits and 1byte=8bits.

1.2 Adjacency

1. 4-path does not exist because neither of the 4-neighbors of q (0,4) is in V={1,2,3} so it's impossible to find a 4-path between p and q.
2. The length of the shortest 8-path between p and q is 4.

```
    3   4   1   2   0
    0   1   0   4   2 (q)
                  ↗
    2   2 → 3 → 1   4
      ↗
(p) 2   0   4   2   1
    1   2   0   3   4
```

3. The length of the shortest m-path between p and q is 5.

```
    3   4   1   2   0
    0   1   0   4   2 (q)
                  ↗
    2 → 2 → 3 → 1   4
    ↑
(p) 2   0   4   2   1
    1   2   0   3   4
```

1.3 Logical Operations

1. $A \cap B \cap C$
2. $(A \cap B) \cup (A \cap C) \cup (B \cap C)$
3. $\{B - [(A \cap B) \cup (B \cap C)]\} \cup [(A \cap C) - (A \cap B \cap C)]$

2 Programming Tasks

2.1 Pre-requirement

2.2 Scaling

1. 1) 192×128

2) 96×64

3) 48×32

4) 24×16

5) 12×8

2. 300×200

3. 450×300

4. 500×200

5. Process and algorithm

输出图像的每个像素(i, j)的灰度值都可以通过下面的计算过程得到：

1) 按缩放比例找到(i, j)在输入图像对应的点(r, c)

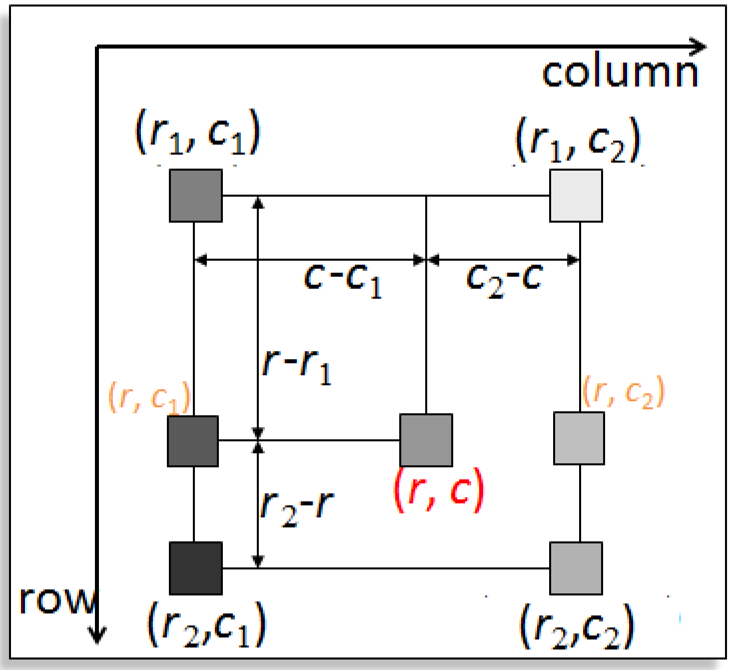
$$\begin{cases} \mathrm{r} = \mathrm{i} \times \mathrm{h\_scaleFactor} \\ c = j \times w\_scaleFactor \end{cases}$$

其中

$$\begin{cases} w\_scaleFactor = \dfrac{width\_in}{width\_out} \\ h\_scaleFactor = \dfrac{height\_in}{height\_out} \end{cases}$$

2) (r, c)在输入图像中一般不是一个整数点（若是，也包含在下面计算过程中），在输入图像中找到与(r, c)最邻近的四个像素点，构造单位正方形，进行双线性插值计算得到在(r, c)的灰度值。算法如下：

双线性插值(Bilinear Interpolation)



取单位正方形上的四个顶点(r_1, c_1)、(r_1, c_2)、(r_2, c_1)、(r_2, c_2)如上图所示，有：

$$\begin{cases} c_2 - c_1 = 1 \\ r_2 - r_1 = 1 \end{cases} \quad ①$$

设在点(x, y)的亮度值为f(x, y)

首先对点(r, c_1)和点(r, c_2)在竖直方向做线性插值:

$$\begin{cases} \mathrm{f}(\mathrm{r}, c_1) \approx \frac{r_2 - r}{r_2 - r_1} f(r_1, c_1) + \frac{r - r_1}{r_2 - r_1} f(r_2, c_1) \\ \mathrm{f}(\mathrm{r}, c_2) \approx \frac{r_2 - r}{r_2 - r_1} f(r_1, c_2) + \frac{r - r_1}{r_2 - r_1} f(r_2, c_2) \end{cases} \quad ②$$

再对点(r, c)在水平方向做线性插值：

$$\mathrm{f}(\mathrm{r}, c) \approx \frac{c_2 - c}{c_2 - c_1} f(\mathrm{r}, c_1) + \frac{c - c_1}{c_2 - c_1} f(\mathrm{r}, c_2) \quad ③$$

将②代入③：

$$\mathrm{f}(\mathrm{r}, c) \approx \frac{c_2 - c}{c_2 - c_1}\left[\frac{r_2 - r}{r_2 - r_1} f(r_1, c_1) + \frac{r - r_1}{r_2 - r_1} f(r_2, c_1)\right] + \frac{c - c_1}{c_2 - c_1}\left[\frac{r_2 - r}{r_2 - r_1} f(r_1, c_2) + \frac{r - r_1}{r_2 - r_1} f(r_2, c_2)\right] \approx \frac{1}{(c_2 - c_1)(r_2 - r_1)}[f(r_1, c_1)(c_2 - c)(r_2 - r) + f(r_1, c_2)(c - c_1)(r_2 - r) + f(r_2, c_1)(c_2 - c)(r - r_1) + f(r_2, c_2)(c - c_1)(r - r_1)] \quad ④$$

将①代入④：

$$\mathrm{f}(\mathrm{r}, c) \approx f(r_1, c_1)(c_2 - c)(r_2 - r) + f(r_1, c_2)(c - c_1)(r_2 - r) + f(r_2, c_1)(c_2 - c)(r - r_1) + f(r_2, c_2)(c - c_1)(r - r_1) \quad ⑤$$

3) 输出图像的(i, j)的灰度值等于(r, c)的灰度值

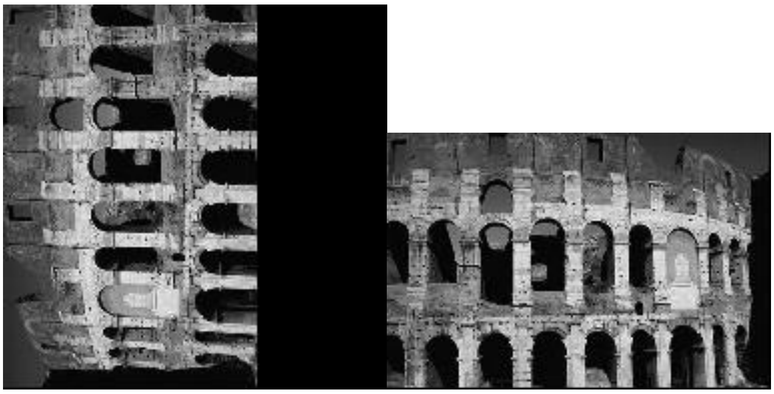
$$\mathrm{output\_img}(\mathrm{i}, \mathrm{j}) = \mathrm{f}(\mathrm{r}, \mathrm{c})$$

Insteresting phenomenons and analysis

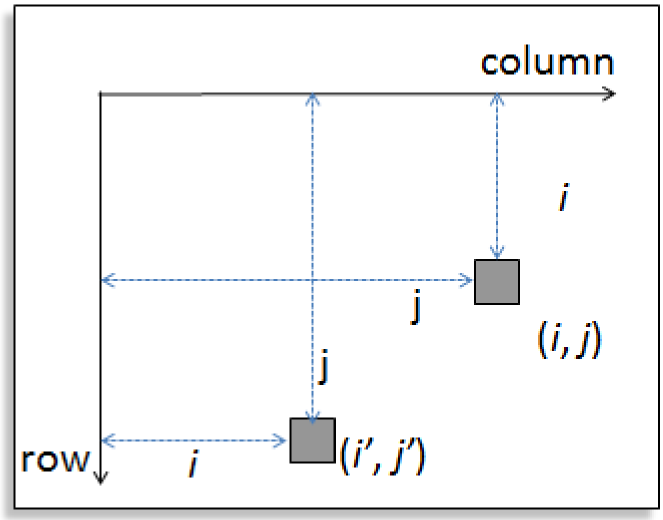
1) 颠倒输出图像的行列

```
output_img(j,i) = input_img(r1,c1) * (c2 - c) * (r2 - r) + ...
                  input_img(r1,c2) * (c - c1) * (r2 - r) + ...
                  input_img(r2,c1) * (c2 - c) * (r - r1) + ...
                  input_img(r2,c2) * (c - c1) * (r - r1);
```

得到的图像 2 及和原图像 1 的对比



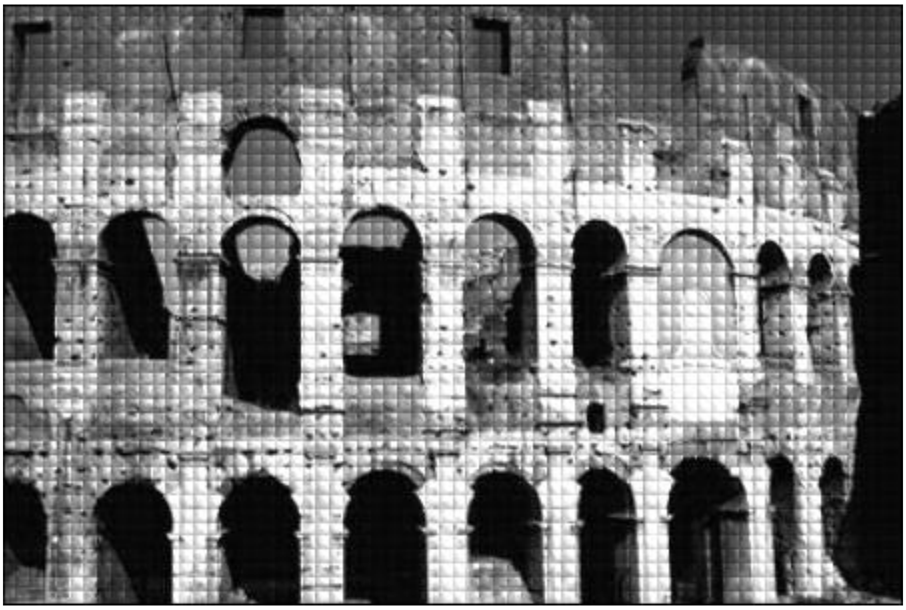
容易看出图像 2 可以从图像 1 逆时针旋转九十度得到。原因见下图



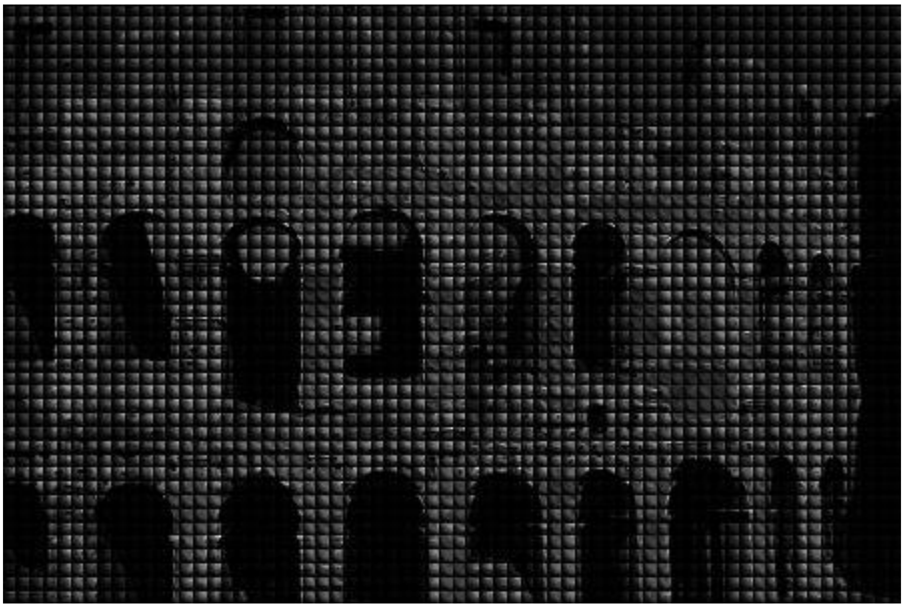
2) 尝试改变公式个别参数的值，如：

```
①output_img(i,j) = input_img(r1,c1) * (c2 - c) * (r2 - r1) + ...
                   input_img(r1,c2) * (c - c1) * (r2 - r) + ...
                   input_img(r2,c1) * (c2 - c) * (r - r1) + ...
                   input_img(r2,c2) * (c - c1) * (r - r1);
```

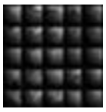
input_img(r1,c1)在 row 的比重加大，得到



和原图做差

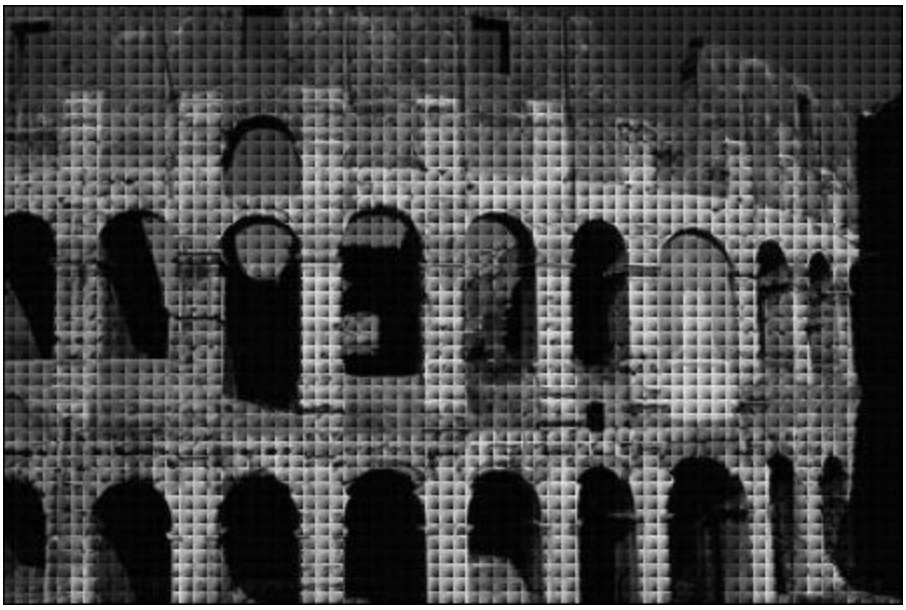


细节放大图：

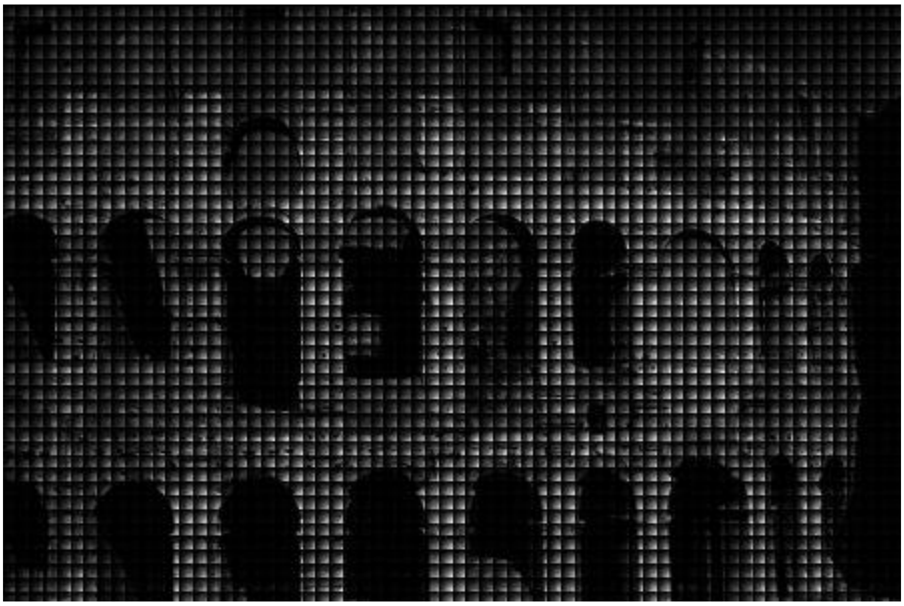


```
②output_img(i,j) = input_img(r1,c1) * (c2 - c) * (r1 - r) + ...
                   input_img(r1,c2) * (c - c1) * (r2 - r) + ...
                   input_img(r2,c1) * (c2 - c) * (r - r1) + ...
                   input_img(r2,c2) * (c - c1) * (r - r1);
```

input_img(r1,c1)比重减小为负数



和原图做差：



篇幅所限，不再粘贴尝试的结果。

根据上面实验看出改变参数得到和原图的差异和变化的是 row 方向还是 column 方向、变化的正负、变化的幅度、在哪个方向变化均有关系，具体的关系可以从公式推导模块推导。

2.3 Quantization

1. 128 levels

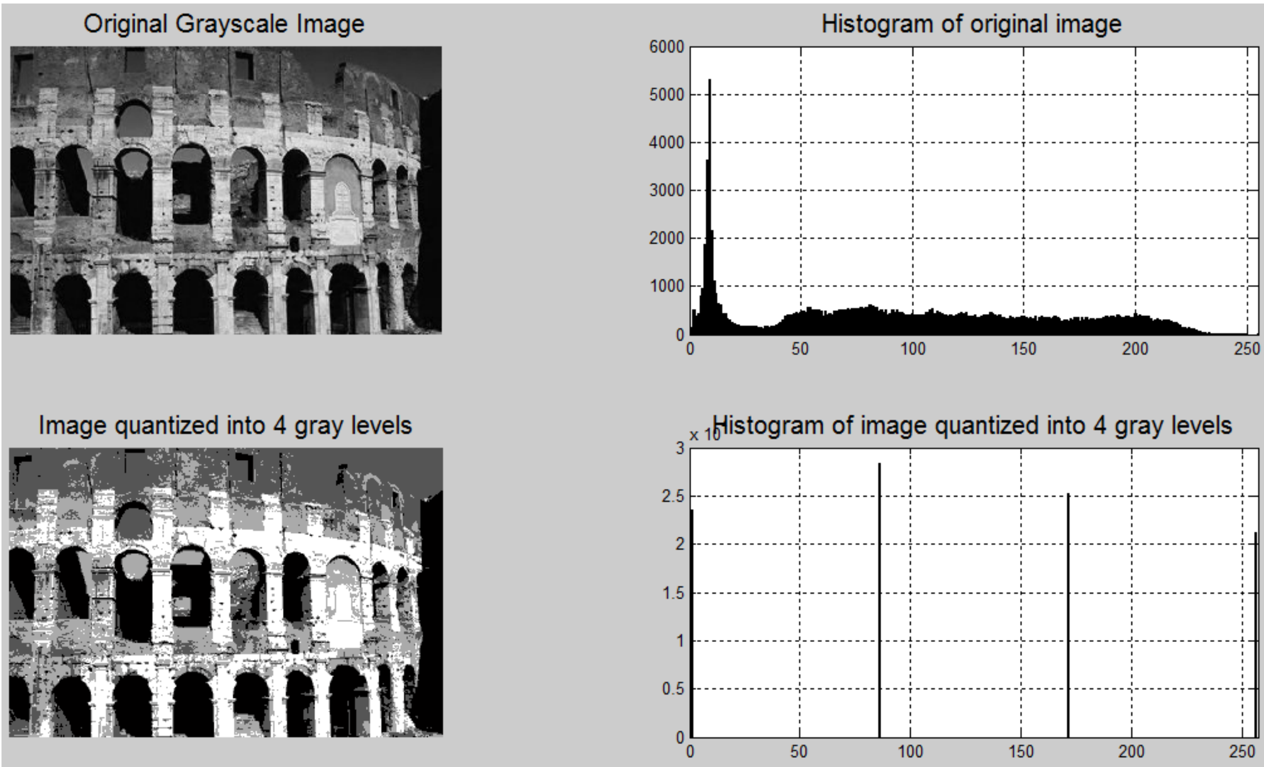
32 levels

8 levels

4 levels

2 levels

2. The process of implementing the quantization operation



为达到题目要求的“当灰度分辨率减小到 4 个级时，输出图像包含 0,85,170,255 个像素而不是 0,1,2,3”，可以采用这样的算法：

将 256 个灰度值均分为 level 份

$$\text{pixels\_per\_level} = \text{floor}(\frac{256}{\text{level}})$$

每份对应一个灰度值

$$\text{interval\_value} = \text{floor}(\frac{255}{\text{level} - 1})$$

将输入图像中的每一个像素进行映射，得到输出图像对应像素的灰度值

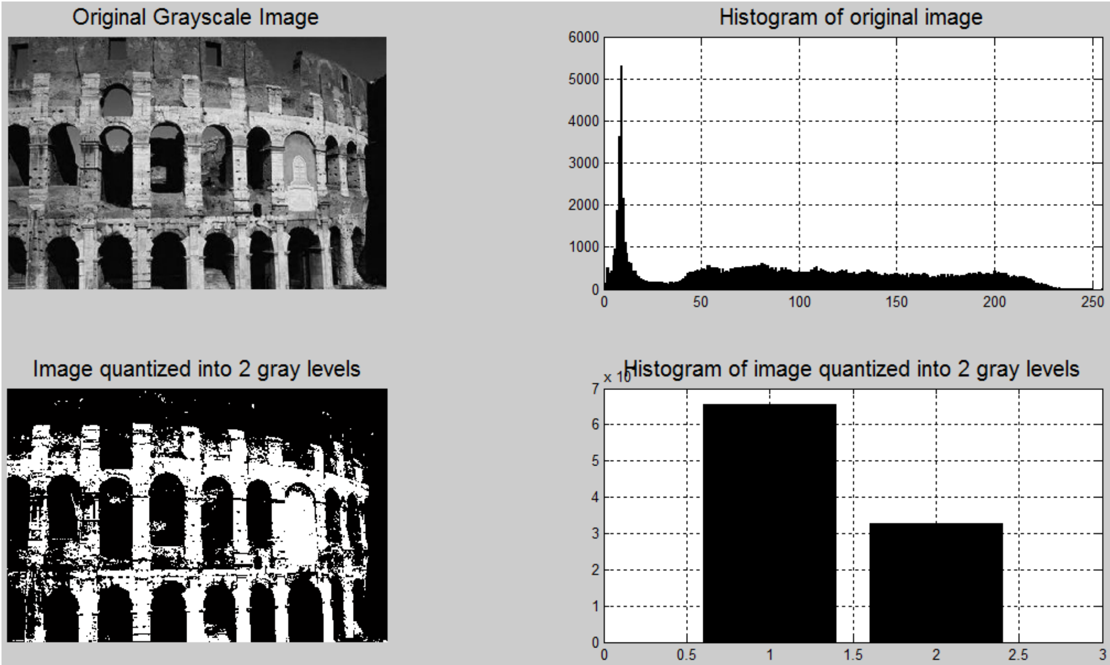
$$\text{output\_img}(r, c) = \text{floor}(\frac{\text{input}_{\text{img}(r,c)}}{\text{pixels}_{\text{per}_{\text{level}}}}) \times \text{interval\_value}$$

如灰度分辨率级别为 4 时，256 分 4 份，每份 64，即在区间 0-63,64-127,128-191,192-255 的灰度值，分别对应灰度 0,85,170,255，进行这样的映射得到输出图像。

Insteresting phenomenons and analysis

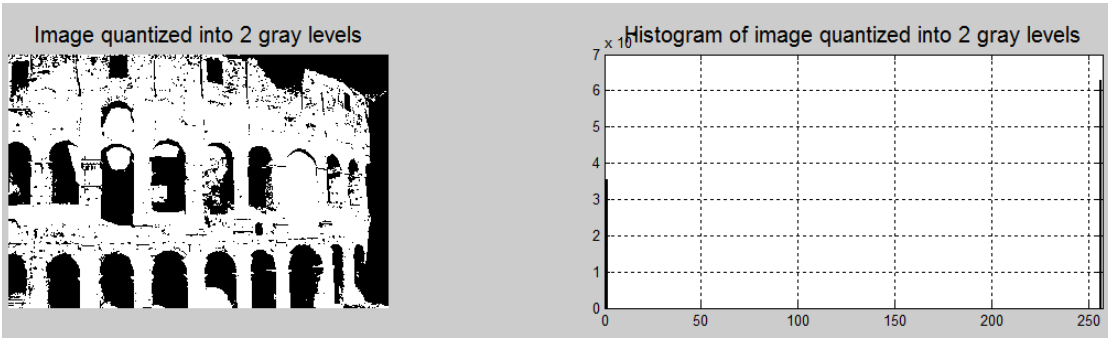
①用 MATLAB 自带的函数进行量化的结果：

```
quantizedImage = uint8(mat2gray(grayImage) * (numberOfGrayLevels-1));
```



②和我的量化方法作对比：

```
quantizedImage = quantize(grayImage, numberOfGrayLevels);
```



比较发现，两者的输出图像有较大不同，原因可以从直方图中看出。方法①的灰度值分布在 0-3，即二级对应的2^2，而我的方法根据题目要求，灰度值分布在 0-255 的 0 和 255，从而导致了图像的差异。

但是从下图的对比可以看到，在$\text{level} = 32$的时候，差异已经非常不明显了。

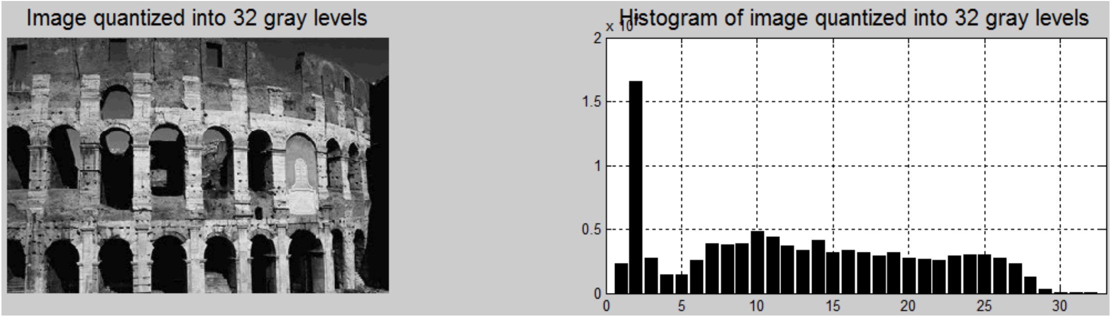


Image quantized into 32 gray levels
Histogram of image quantized into 32 gray levels
2
1.5
1
0.5
0
0
5
10
15
20
25
30

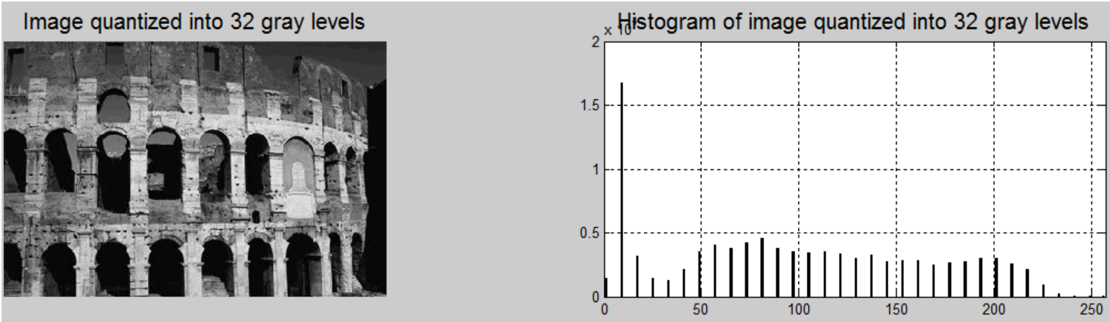


Image quantized into 32 gray levels
Histogram of image quantized into 32 gray levels
2
1.5
1
0.5
0
0
50
100
150
200
250