تمرین پنجم - امین قندی - 400202163

① در این سؤال تلاش گردید با انتخاب threshold مناسب ابتدا ستاره‌های کم نور حذف شوند و سپس با تعریف توابع erosion، dilation، opening در نهایت مورفولوژیک که ستاره‌های پرنور انتخاب شوند. معیار شده جهت رتبه‌بندی مساحت هر ستاره پس از پردازش‌ها بوده است.

①

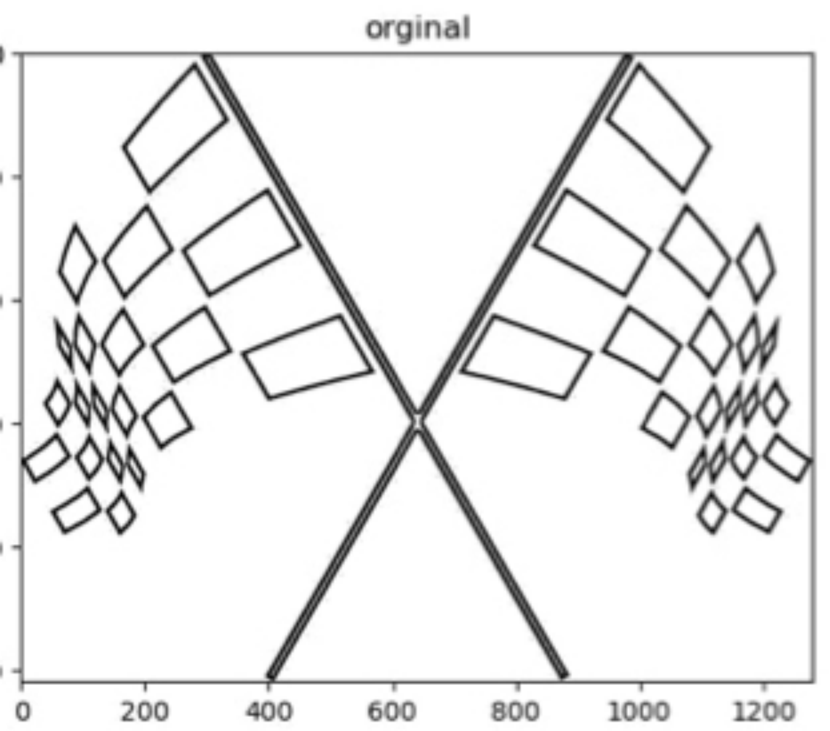
③

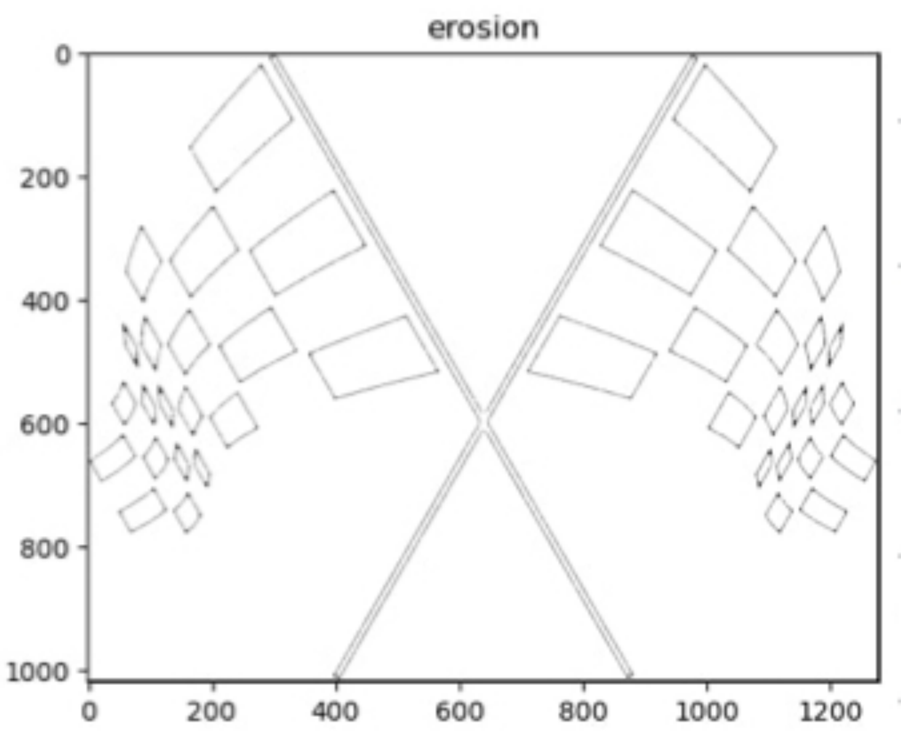
②

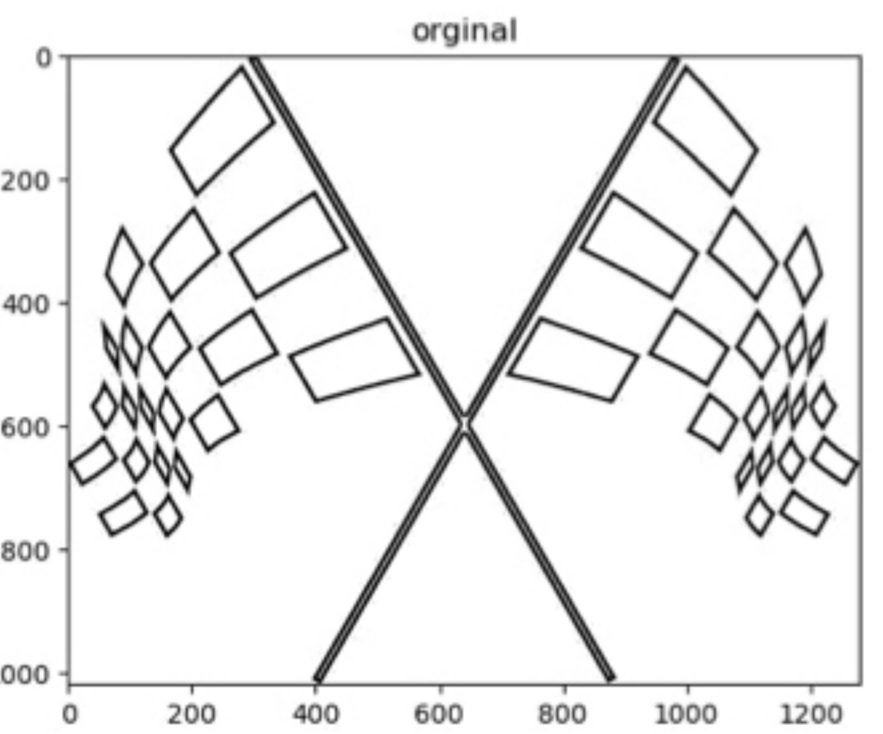
④

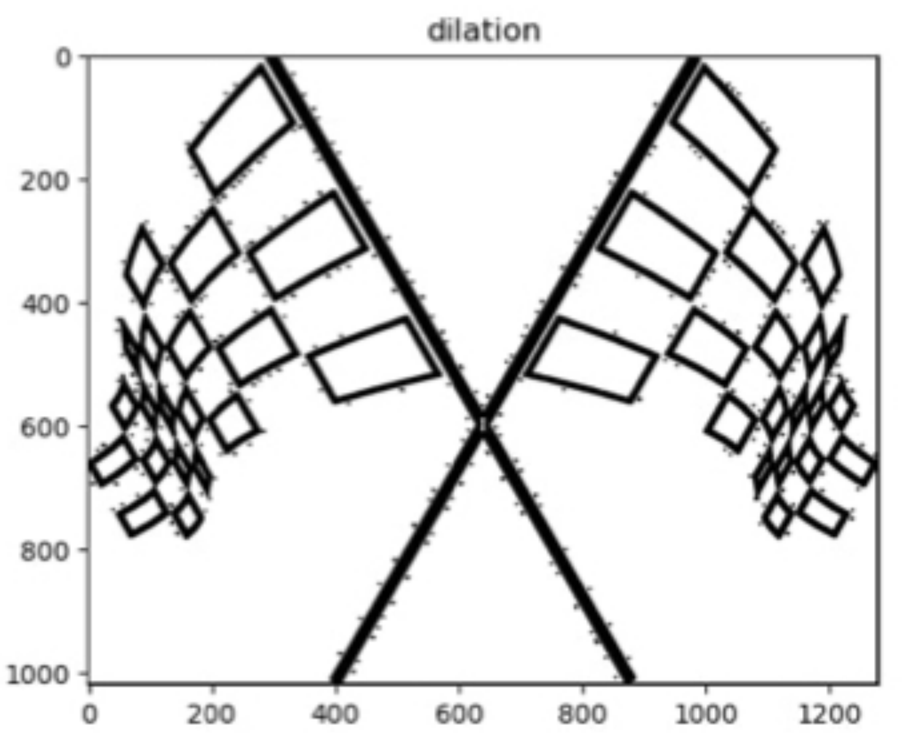
② ☆ عملگرهای مورفولوژیک گسترش و فرسایش
به نسبت کارایی بیشتری از عملگرهای کانولوشن
ندارد، اما به صورت کلی می‌توان عملگرهای
مورفولوژیک را با استفاده از عملگرهای کانولوشن
بازنمایی کرد که کد آن در فایل‌های پیوست
موجود است.

☆ عملگرهای گسترش و فرسایش برای تصاویر خاکستری
نیز قابل استفاده هستند. در این حالت مقادیر
پیکسل‌ها به عنوان شدت نور تفسیر می‌شوند.







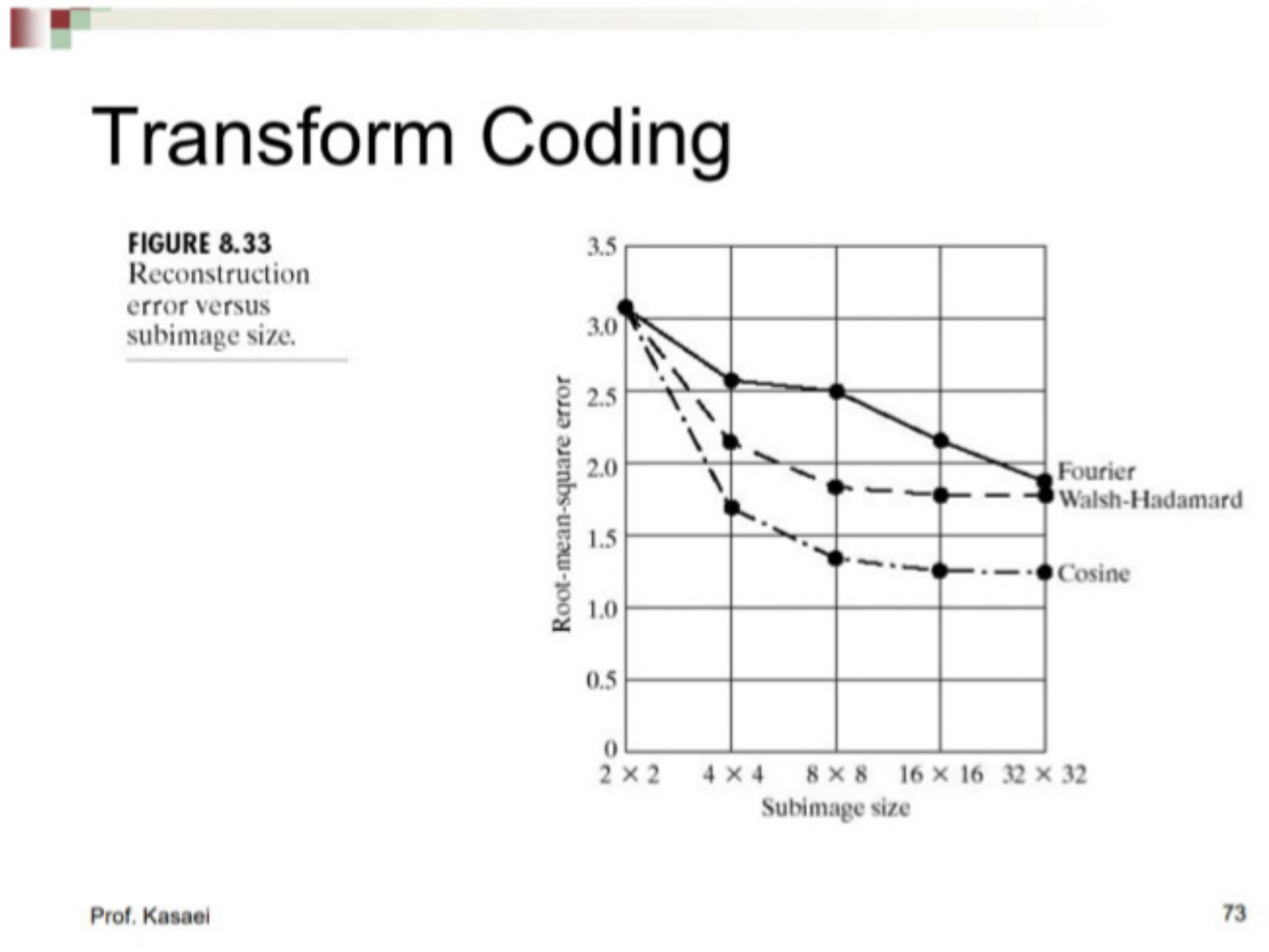


③ با توجه به توضیحات ارائه شده در فصل فشرده‌سازی با [illegible] از متد Block transform coding استفاده شدم. هسته اصلی آن از تابع DCT استفاده می‌کند و نتایج قابل قبولی را ارائه می دهد.

متد Block transform coding در موارد زیر نیز استفاده می‌شود:

Jpeg / M-jpeg / MPEG-1,2,4

H-261 , H-262 / VC-1



Transform Coding

FIGURE 8.33 Reconstruction error versus subimage size.

Prof. Kasaei

73

مقایسه به کارگیری DCT , W-H , DFT در بلوک های مختلف با معیار RMSE.

(از اسلاید)

✩

مراحل به کار رفته در برنامه پایتون:
(از کتاب)

In this section, we restrict our attention to square subimages (the most commonly used). It is assumed that the input image is padded, if necessary, so that both *M* and *N* are multiples of *n*.

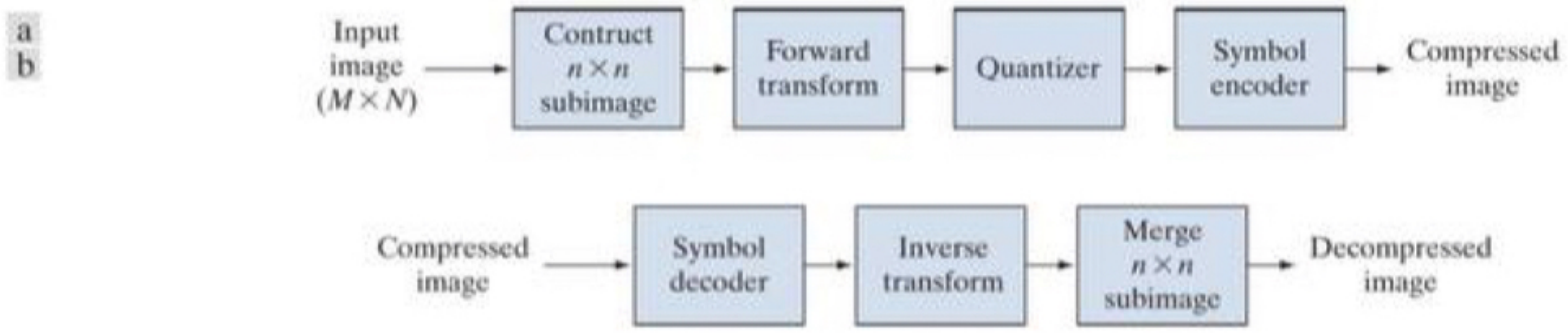
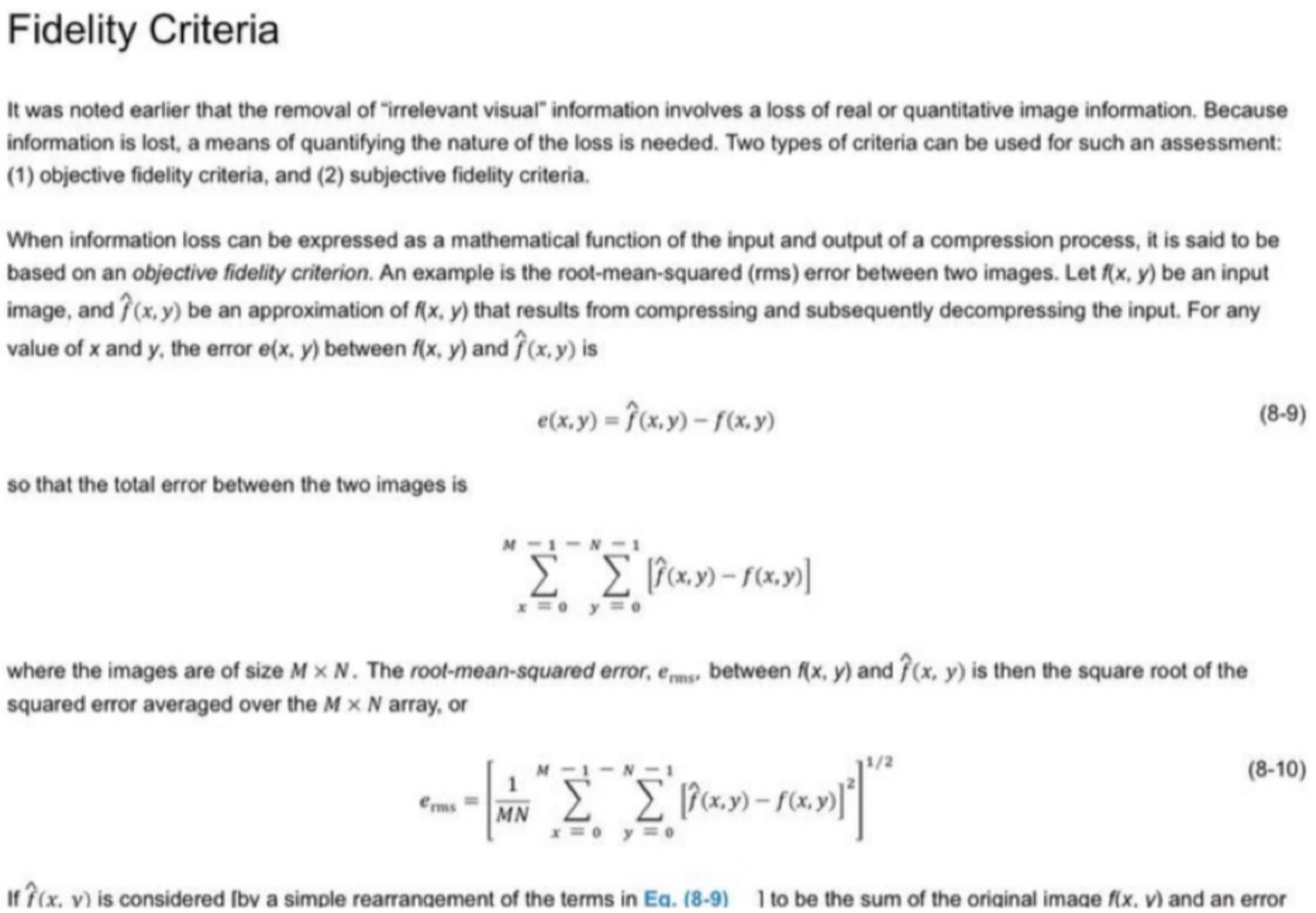


FIGURE 8.21
A block transform coding system: (a) encoder; (b) decoder.

تعریف RMSE:
(از کتاب)



Fidelity Criteria

It was noted earlier that the removal of "irrelevant visual" information involves a loss of real or quantitative image information. Because information is lost, a means of quantifying the nature of the loss is needed. Two types of criteria can be used for such an assessment: (1) objective fidelity criteria, and (2) subjective fidelity criteria.

When information loss can be expressed as a mathematical function of the input and output of a compression process, it is said to be based on an *objective fidelity criterion.* An example is the root-mean-squared (rms) error between two images. Let $f(x, y)$ be an input image, and $\hat{f}(x,y)$ be an approximation of $f(x, y)$ that results from compressing and subsequently decompressing the input. For any value of x and y, the error $e(x, y)$ between $f(x, y)$ and $\hat{f}(x,y)$ is

$$e(x,y)=\hat{f}(x,y)-f(x,y) \tag{8-9}$$

so that the total error between the two images is

$$\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\left[\hat{f}(x,y)-f(x,y)\right]$$

where the images are of size $M \times N$. The *root-mean-squared error,* e_{rms}, between $f(x, y)$ and $\hat{f}(x, y)$ is then the square root of the squared error averaged over the $M \times N$ array, or

$$e_{rms}=\left[\frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\left[\hat{f}(x,y)-f(x,y)\right]^2\right]^{1/2} \tag{8-10}$$

If $\hat{f}(x, y)$ is considered [by a simple rearrangement of the terms in Eq. (8-9)] to be the sum of the original image $f(x, y)$ and an error

نتیجه:

Block: 64 × 64 / truncation (%) = 10%

```
# test usage
compress_image("q3.jpg", block_size=64, target_rmse=1.0, truncation_percentage=10)

Compression successful. RMSE: 0.38
Compression Ratio: 4.44
```

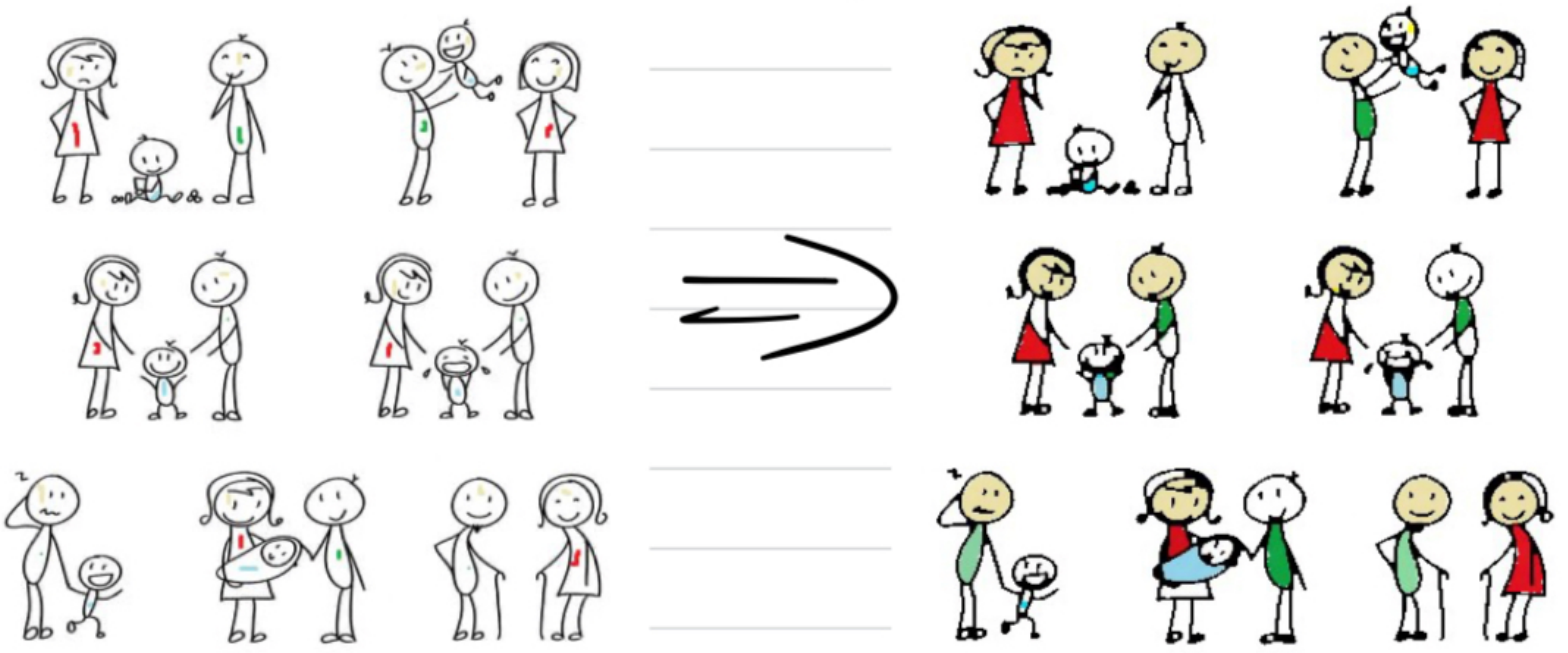
≤ 1 ☑

$\equiv 25\%$ ☑

۴

تصویر اول: در این تصویر ابتدا با استفاده از threshold مناسب مرز اشکال با رنگ مشکی جدا شد. سپس رنگ های تقسیم شده به عنوان قالبی جدا ذخیره گشت. (برای region filling نیاز لیست لوکیشن رنگ ها و کد آن ها ذخیره شود). در مرحله ای دیگر برای اطمینان از بسته بودن مرزها و جداسازی پشت صفحه اشکال از توابع مورفولوژیک استفاده شد. پس از آن با تعریف حلقه رنگ ها گسترش

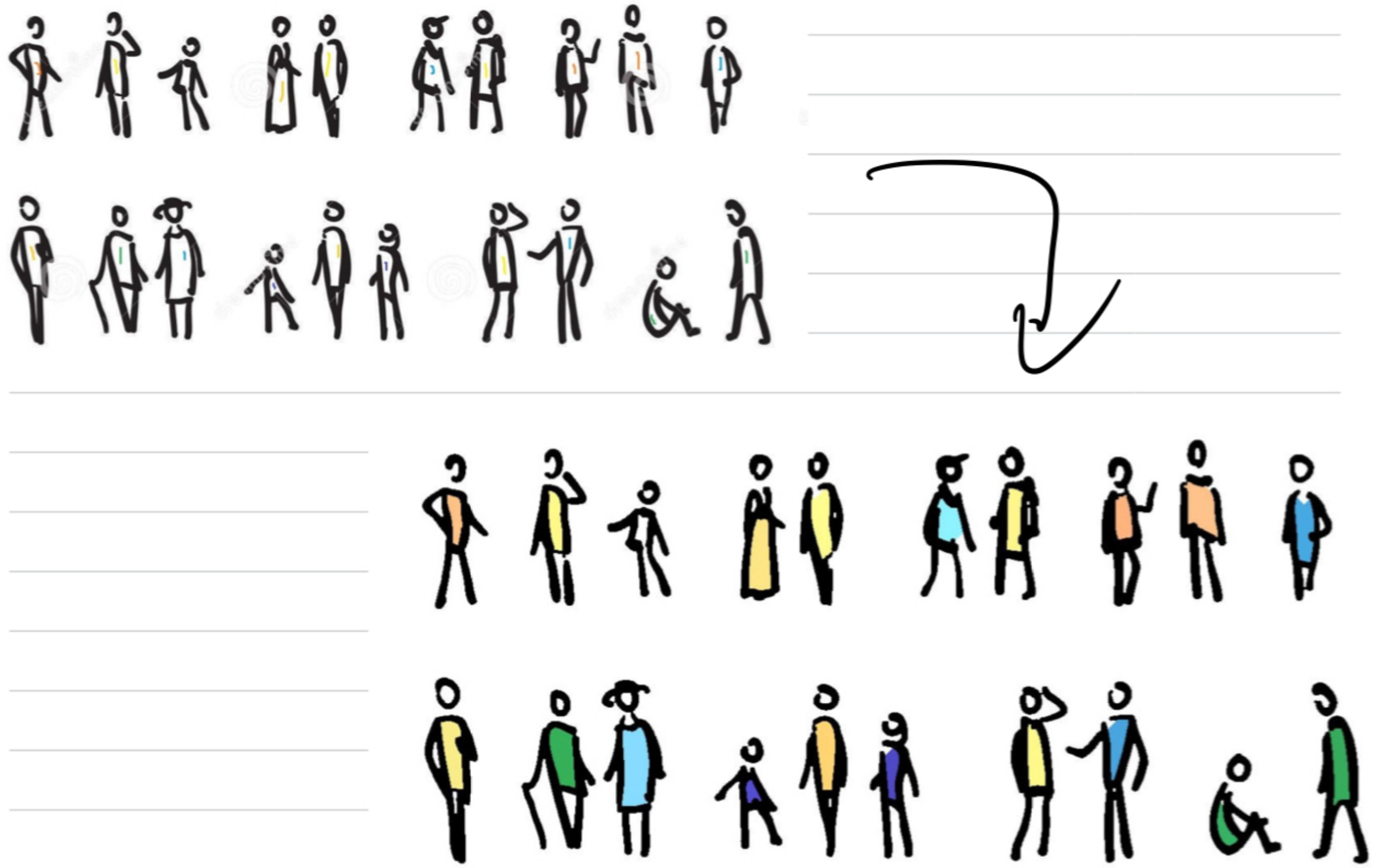
پیدا کردند و با استفاده از روش استخراج شده، و پشت صفحه به دست آمد رنگ های اضافی فیلتر شدند.



تصویر دوم:

اکثر مراحل به شکل قبل است. با این تفاوت که در حذف رنگ خاکستری علامت ها نیز نیاز است تا بتوان درست بازه رنگی آن استخراج بشود

و برای بستن مرزهای اشکال نیز
از توابع مورفولوژیک استفاده گردد .
در نهایت به مانند تصویر اول با
استخراج رنگ دانه ها و مرز اشکال و پشت
صفحه تصویر نهایی حاصل می شود :



۵ در تصاویر، افزونگی های مختلفی
را می توان نام برد برای مثال در تصویر
های زیر سه نمونه ای از آن ها اشاره شده است

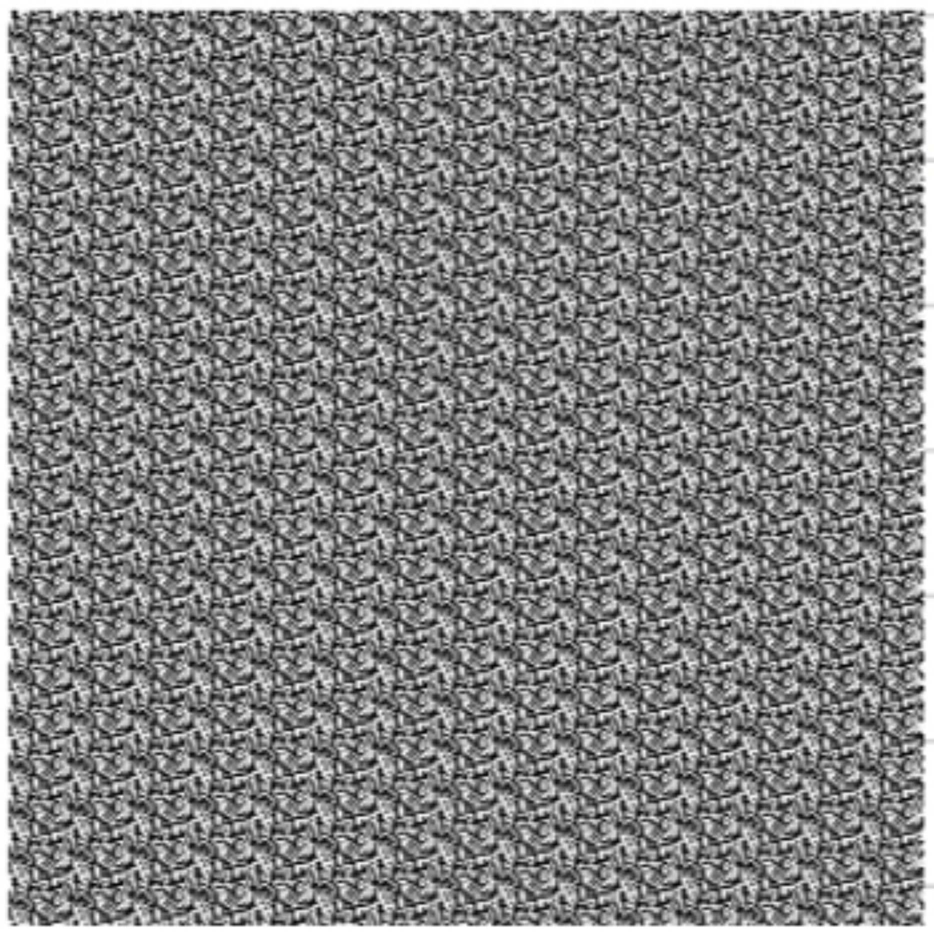
- In digital image compression there exist three basic data redundancies:

1. Coding redundancy
2. Spatial and Temporal redundancy
3. Irrelevant redundancy

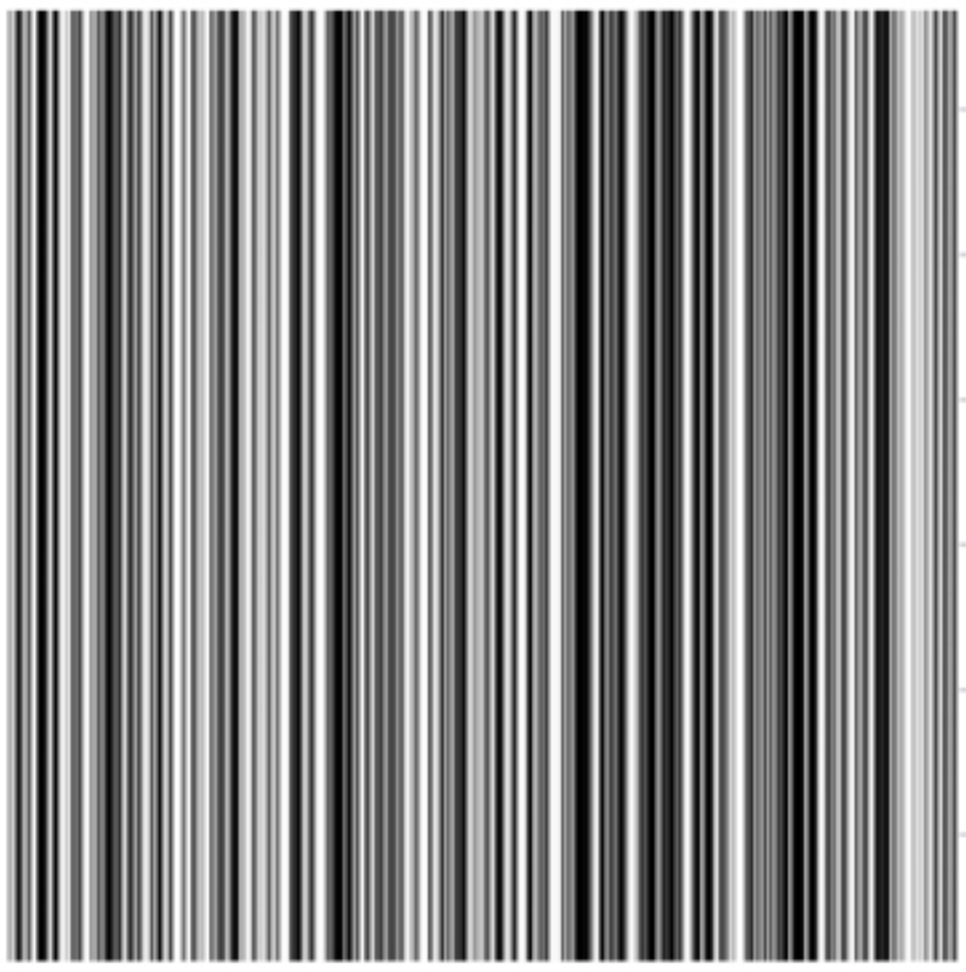
a b c

FIGURE 8.1 Computer generated 256 × 256 × 8 bit images with (a) coding redundancy, (b) spatial redundancy, and (c) irrelevant information. (Each was designed to demonstrate one principal redundancy but may exhibit others as well.)

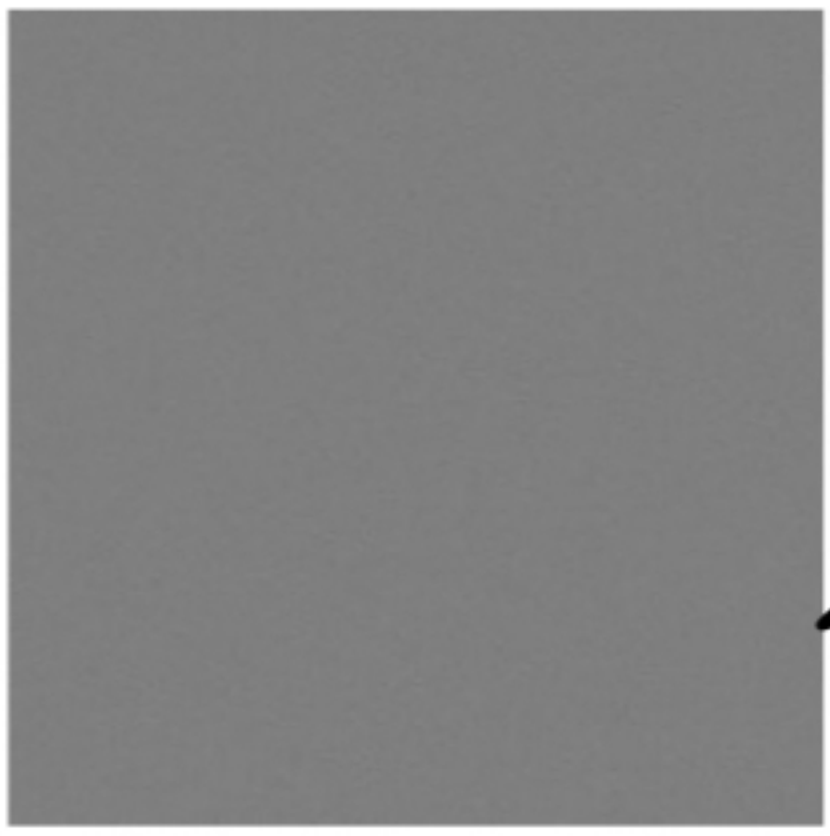
در تصاویر مورد سوال می توان گفت:



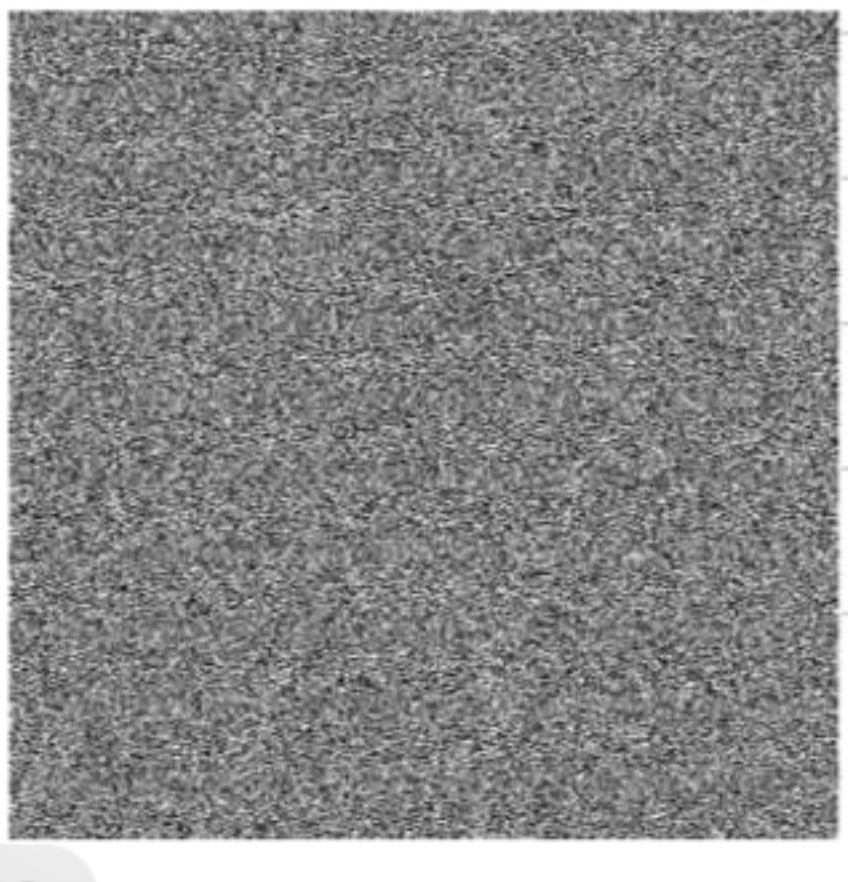
در این تصویر، المنت های مختلف به صورت متناوب وجود دارد. از طرفی دارای افزونگی های spatial و irrelevant می باشد و RLC می تواند گزینه خوبی باشد.



افزونگی spatial واضح است چرا که در جهت y مقادیر پیکسل ها تکرار شده اند و در کدگذاری می توان با ذخیره یک ردیف و تعداد موارد تکرار، کل را رمزگذاری کرد.



هرچند در طیف رنگی در تصویر البته بخش چپ دارای ناهمگونی می باشد. اما در کل می توان افزونگی های spatial irrelevant _ coding را به عنوان عامل اصلی داشت



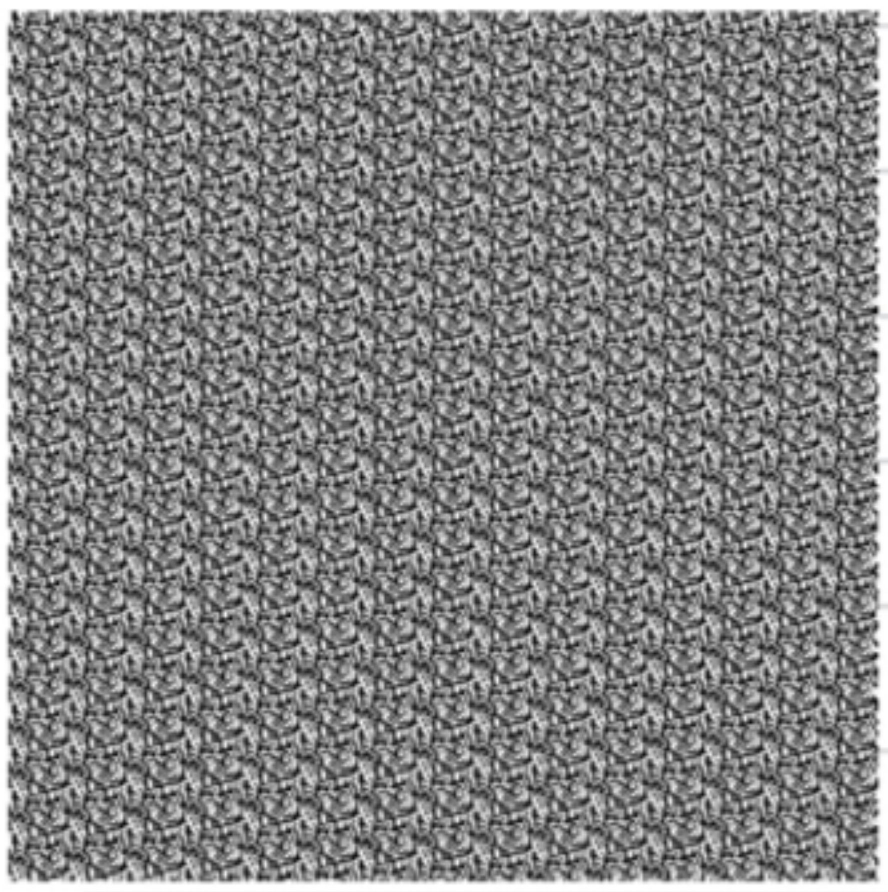
در این تصویر می توان افزونگی irrelevant را عامل اصلی داشت که متدهای مختلفی می توانند داشته باشند:

① color_subsampling

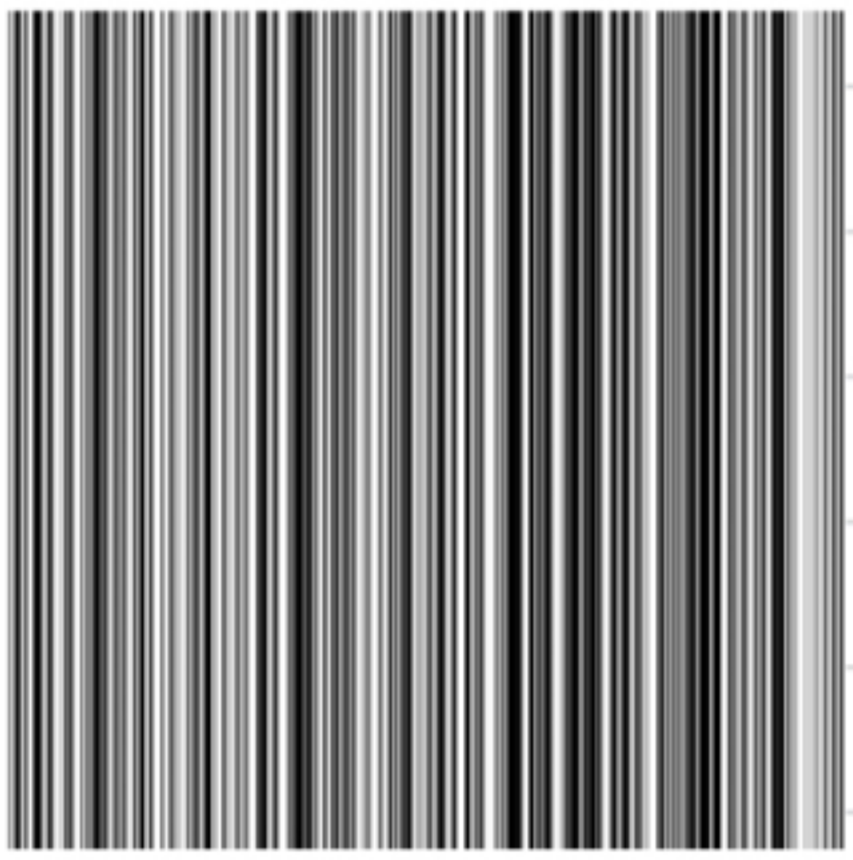
② quantization

③ transform coding

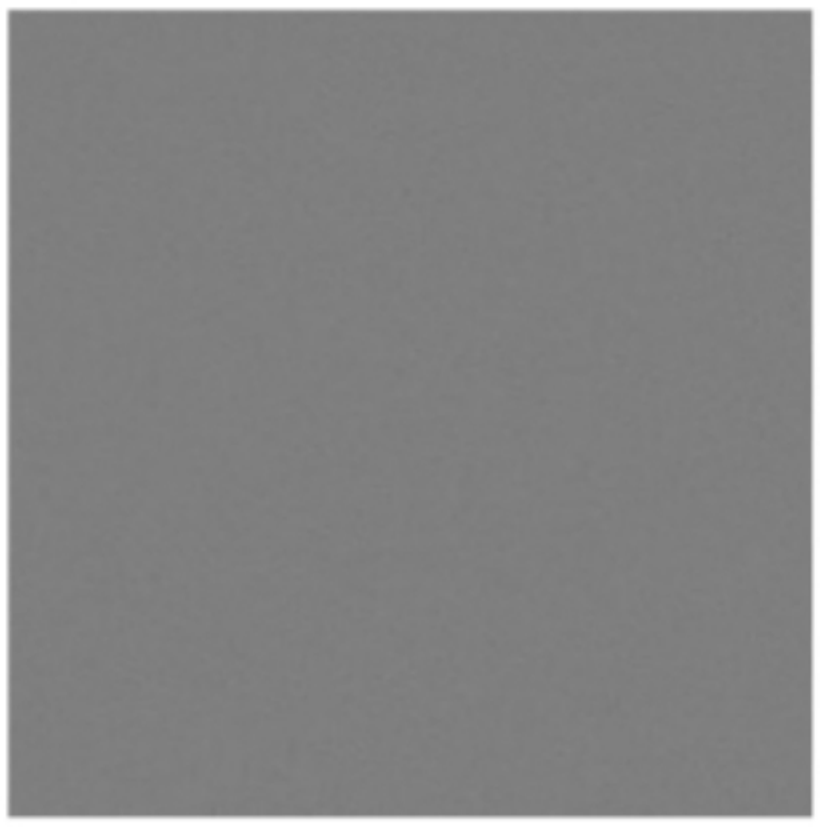
نتایج:



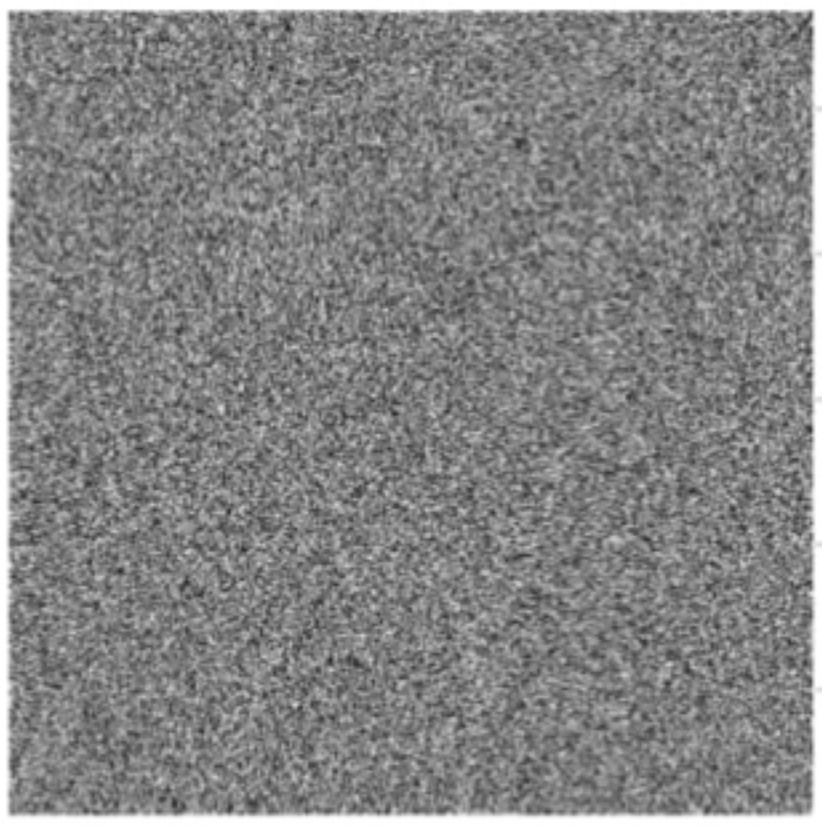
```
Compression Rate: 14.08
```



```
Compression Rate: 15.39
```



```
Compression Rate: 0.55
```



```
quantize_image:
Compression Rate: 0.50
```

```
transform_coding:
Compression Rate: 0.56
```

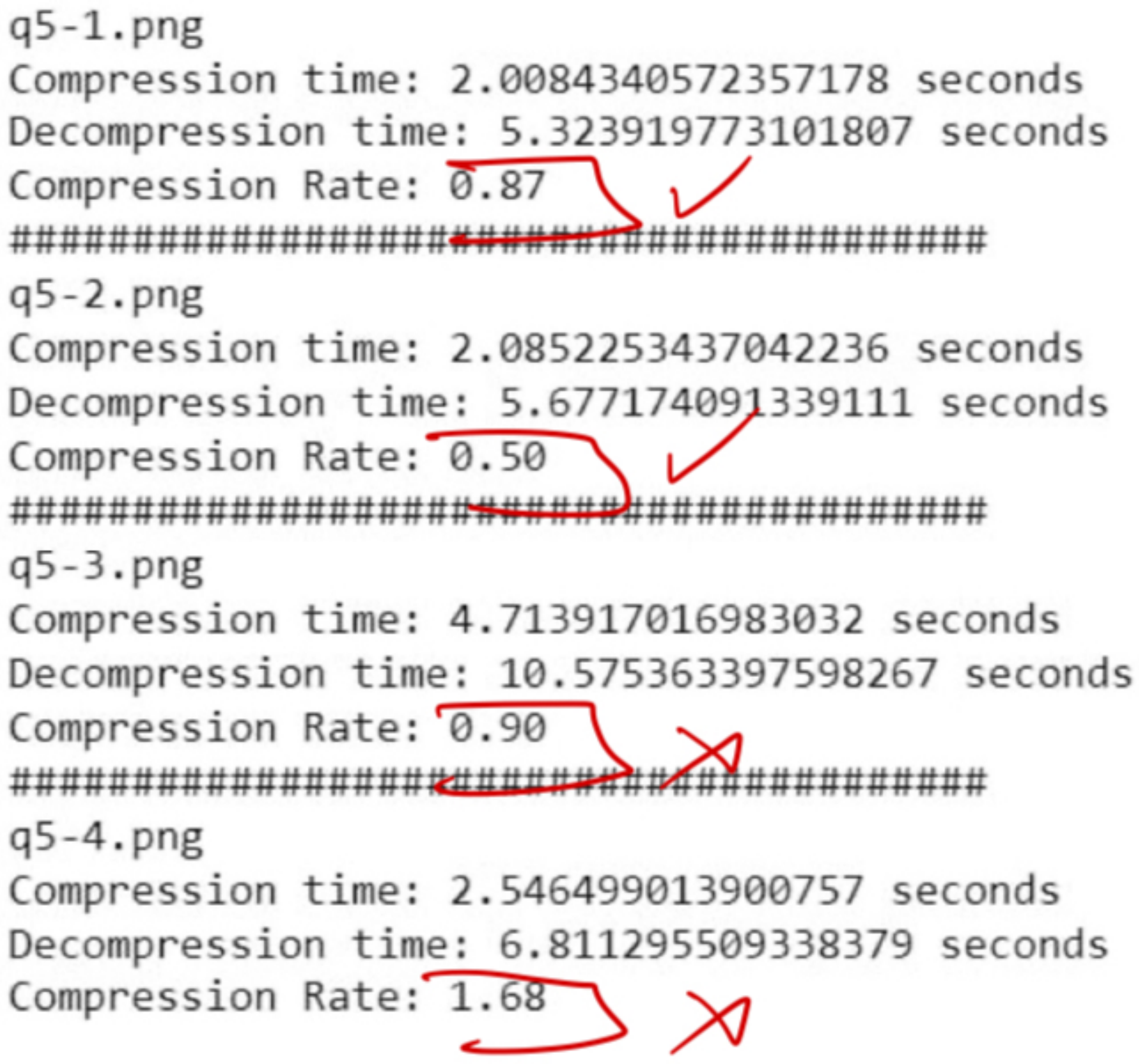
```
color_subsampling:
Compression Rate: 1.68
```

روش های Huffman و Golomb-rice هر دو از روش های lossless هستند اما مبنای متفاوتی دارند.

☆ Huffman از دسته VLC ها هستند و براساس فرکانس حضور علائم مختلف در داده، به آن ها یک کد اختصاص می دهد. هرچه مقدار حضور بیشتر باشد، کد کوتاهتری به آن علامت تعلق می گیرد. در رابطه با عملکرد Huffman می توان گفت در مقادیری که الگوی تکراری یا علامت های با فرکانس بالا وجود دارد، روش Huffman بسیار مناسب است.

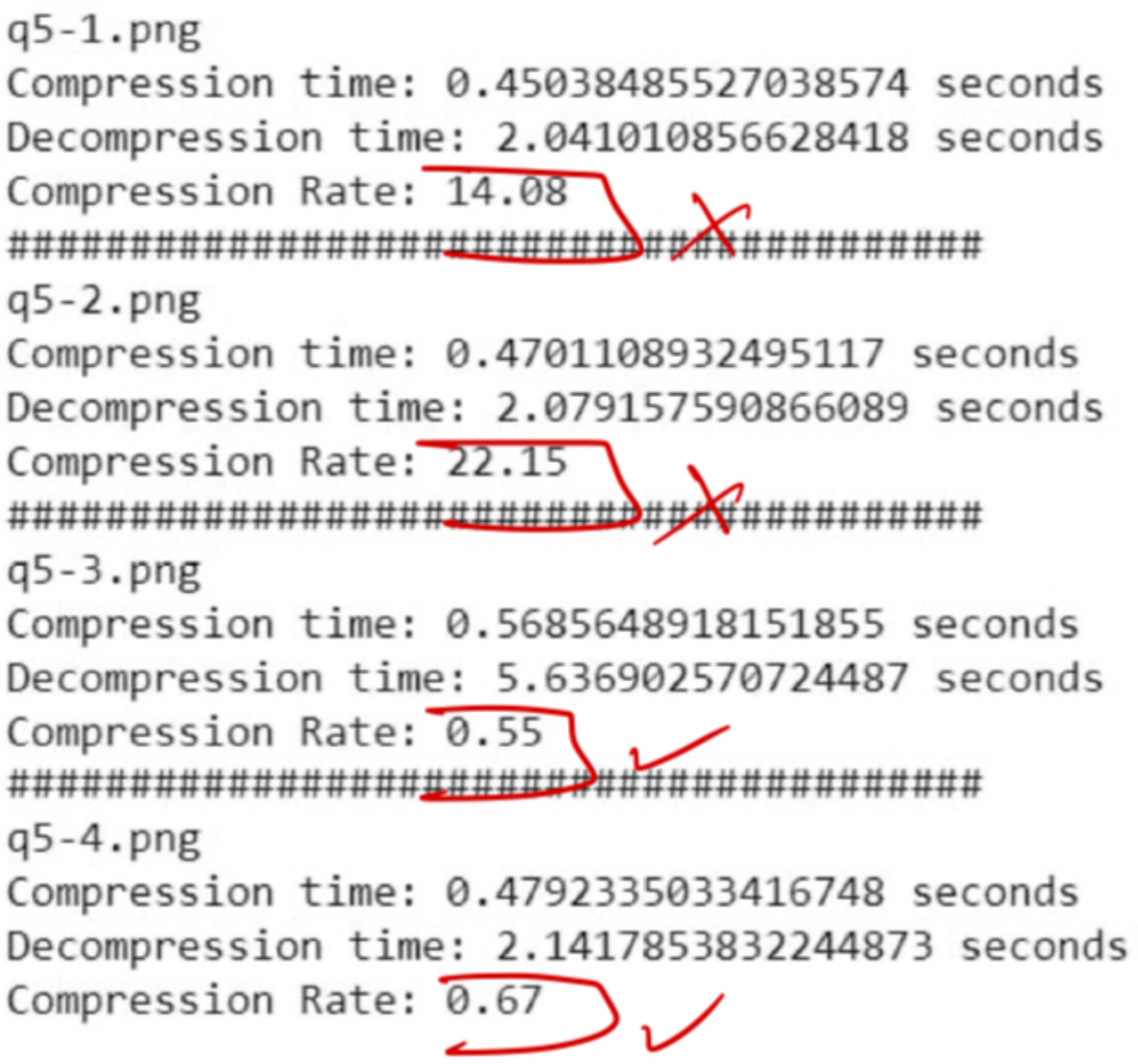
Golomb-rice از دسته متدهای universal entropy coding می باشد که برای مقادیر صحیح با توزیع geometric به کار می رود. در این متد، پارامتری به نام m تعریف می شود که با افزایش آن، متوسط طول کدها افزایش می یابد ولی در عوض برای مقادیر بزرگ فشرده‌سازی بهتری انجام می دهد. این متد در مقادیر دارای توزیع geometric کاربرد دارد.

Huffman



```
q5-1.png
Compression time: 2.0084340572357178 seconds
Decompression time: 5.323919773101807 seconds
Compression Rate: 0.87
########################################
q5-2.png
Compression time: 2.0852253437042236 seconds
Decompression time: 5.677174091339111 seconds
Compression Rate: 0.50
########################################
q5-3.png
Compression time: 4.713917016983032 seconds
Decompression time: 10.575363397598267 seconds
Compression Rate: 0.90
########################################
q5-4.png
Compression time: 2.546499013900757 seconds
Decompression time: 6.811295509338379 seconds
Compression Rate: 1.68
```

Golomb-rice



```
q5-1.png
Compression time: 0.45038485527038574 seconds
Decompression time: 2.041010856628418 seconds
Compression Rate: 14.08
########################################
q5-2.png
Compression time: 0.4701108932495117 seconds
Decompression time: 2.079157590866089 seconds
Compression Rate: 22.15
########################################
q5-3.png
Compression time: 0.5685648918151855 seconds
Decompression time: 5.636902570724487 seconds
Compression Rate: 0.55
########################################
q5-4.png
Compression time: 0.4792335033416748 seconds
Decompression time: 2.1417853832244873 seconds
Compression Rate: 0.67
```

همانطور که بالاتر گفته شد، هافمن در فشرده سازی تصاویر با الگوی تکراری و فرکانس بالا بسیار خوب عمل می‌کند. (تصاویر یک و دو) و در مقابل روش Golomb در تصاویر با توزیع geometric خوب عمل می‌کند (تصاویر سه و چهار)

نکته دیگر آن که زمان پردازش هافمن بسیار بیشتر از Golomb می‌باشد.