《数字图像处理》
第9讲 图像压缩

冯建江

清华大学 自动化系

2017.11.23

内 容

- 基础
- 霍夫曼编码（Huffman Coding）
- 预测编码（Predictive Coding）
- 变换编码（Transform Coding）

为什么压缩图像

- 图像传感器的空间和时间分辨率越来越高
- 以手机为例，不少手机的后置摄像头已达到2000万像素
- 原始数字图像表示为二维数组
- 一幅彩色图像占$3MN$字节，M和N为宽和高
- 一张照片占60M字节
- 一段10分钟视频（1920×1080像素，30帧/秒）占1.86G字节

为什么压缩图像

- 以3D+t的心脏CT图像数据为例，一个心动周期采集20个时相
- 每个时相采集一个$512 \times 512 \times 402$矩阵
- CT值不同于一般灰度图像的灰度，其变化范围较大，一般为$-1000 \sim 1000$，因此采用有符号16位整型存储
- 因此，可算出一个时相占据内存：

$$512 \times 512 \times 402 \times 2 = 201 \text{ MB}$$

- 那么20个时相共计4.02 GB
- 同时，由于文件还要保存除了像素值以外的其他一些信息，实际图像大小达到4.10 GB。

为什么压缩图像

- 图像数据量越来越大
- 而存储空间有限
  - 主机5T硬盘：只够7.4小时手机视频（5场足球比赛）
  - 嵌入式系统存储空间非常有限：智能卡只给指纹1K字节的存储空间
- 通信带宽也有限，而且费用高。如果通过手机无线看视频的话，延时和费用都难以承受。

美剧《Silicon Valley》

- 讲述了Richard用他发明的颠覆性技术“通用压缩算法”进行创业的故事
- 为了可信、易懂，导演咨询了数据压缩专家Weissman

算法原理：先做有损压缩，然后对残差做无损压缩

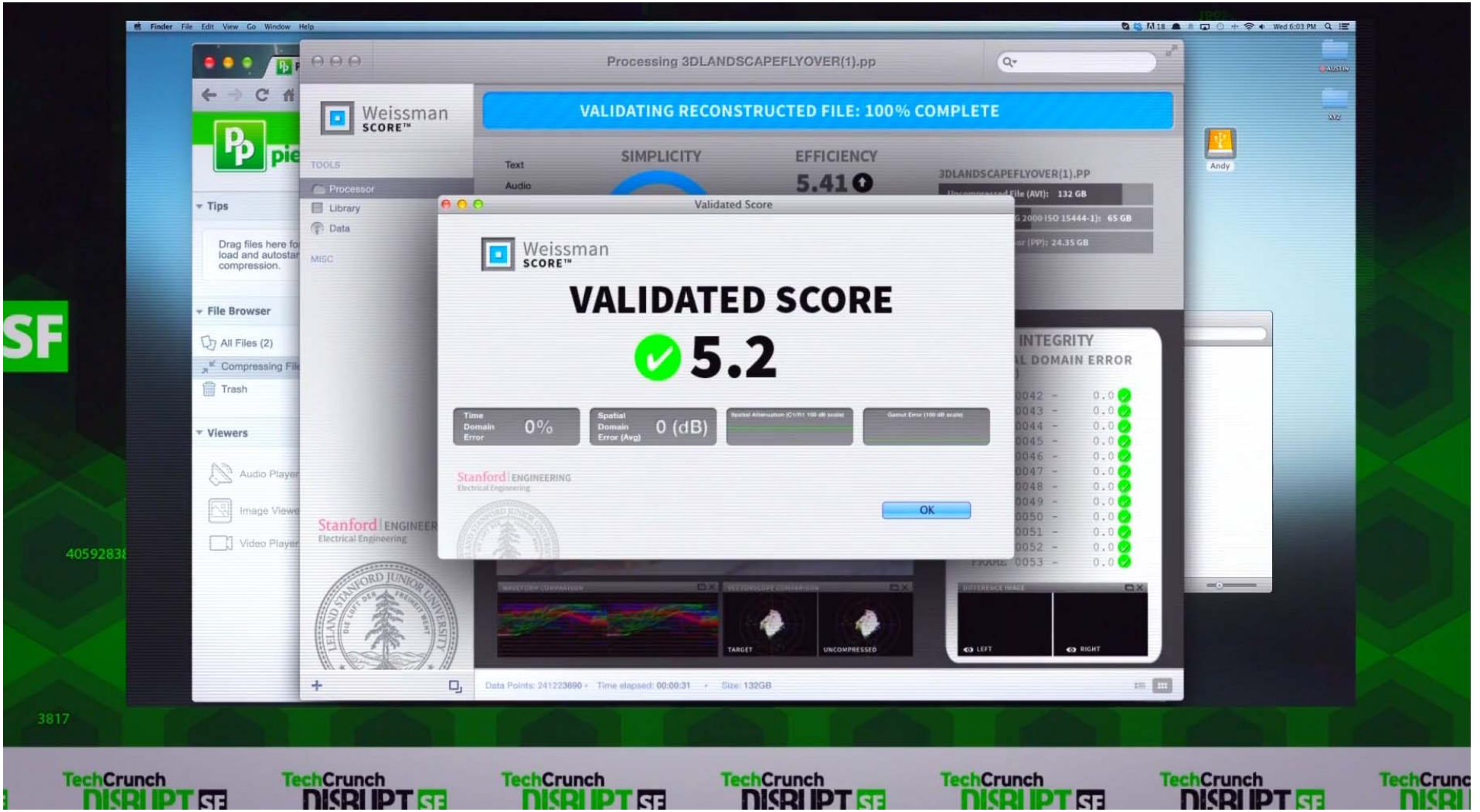
为了使压缩算法的评价简单易懂，还需要一个评价分数（scatter plot不直观）

Weissman SCORE ™

$$W = \alpha \frac{r}{\overline{r}} \frac{\log \overline{T}}{\log T}$$

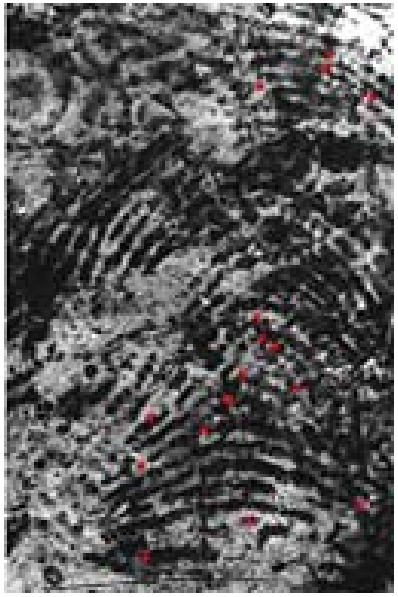
已经有论文在用这个指标了

NOTE: r and T refer to the compression ratio and time-to-compress for the target algorithm, $\overline{r}$ and $\overline{T}$ refer to the same quantities for a standard universal compressor (e.g. gzip or FLAC), and α is a scaling constant. By normalizing by the performance of a standard compressor, we take away variation in compressive performance between types of data.


Processing 3DLANDSCAPEFLYOVER(1).pp
Weissman SCORE™
VALIDATING RECONSTRUCTED FILE: 100% COMPLETE
SIMPLICITY
EFFICIENCY
5.41
3DLANDSCAPEFLYOVER(1).PP
Validated Score
VALIDATED SCORE
5.2
Time Domain Error
0%
Spatial Domain Error (Avg)
0 (dB)
Stanford ENGINEERING
Electrical Engineering
OK
INTEGRITY
DOMAIN ERROR
TARGET
UNCOMPRESSED
LEFT
RIGHT
Tips
File Browser
All Files (2)
Trash
Viewers
Audio Player
Image Viewer
Video Player
Processor
Library
Data
Text
Audio
Andy
TechCrunch DISRUPT SF

无损和有损图像压缩

- 图像压缩分为无损和有损
- 某些领域的图像需要无损压缩，如医疗图像（防止误诊、漏诊）、现场指纹（防止误识别、漏识别）
- 大多数领域的图像允许有损压缩，如手机照片、互联网图片、监控视频、聊天视频

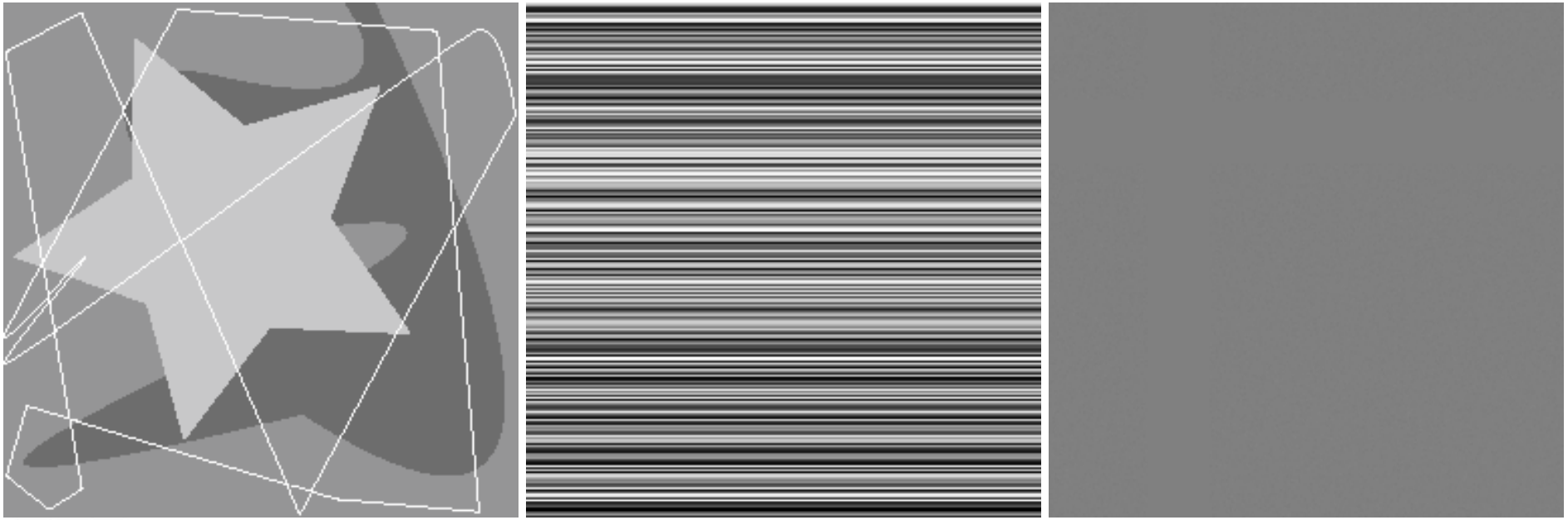


马德里火车爆炸案作案分子的指纹

图像为什么能压缩

图像的2D数组表示具有三种冗余：

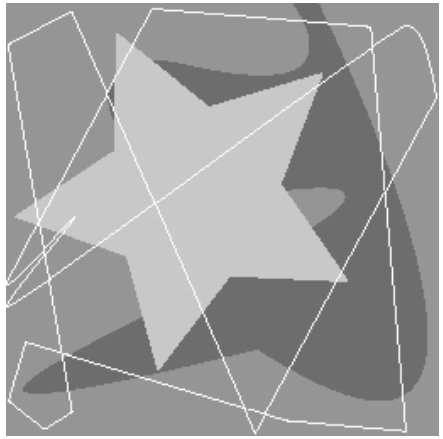
1. 编码冗余：对所有像素值做定长编码，未考虑不同值的概率
2. 空间冗余：对各像素独立编码，未考虑相邻像素值的相关性
3. 不重要信息：所有像素一视同仁，未考虑人视觉系统的特点



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.)

编码冗余（Coding Redundancy）



自然的8 bit二进制编码为定长编码（fixed-length code）

变长编码（variable-length code）

灰度r_k的概率

码长

r_k	$p_r(r_k)$	Code 1	$l_1(r_k)$	Code 2	$l_2(r_k)$
$r_{87} = 87$	0.25	01010111	8	01	2
$r_{128} = 128$	0.47	10000000	8	1	1
$r_{186} = 186$	0.25	11000100	8	000	3
$r_{255} = 255$	0.03	11111111	8	001	3
r_k for $k \neq 87, 128, 186, 255$	0	—	8	—	0

每像素平均比特数$L_{\text{avg}} = \sum_{k=0}^{L-1} l(r_k)p_r(r_k)$

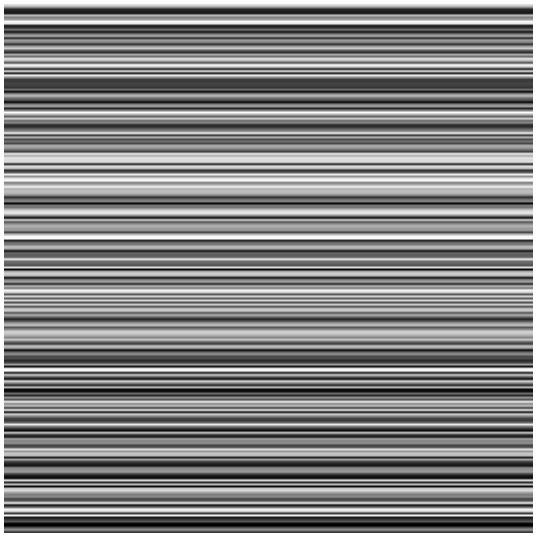
Code 1: $L_{\text{avg}} = 8$

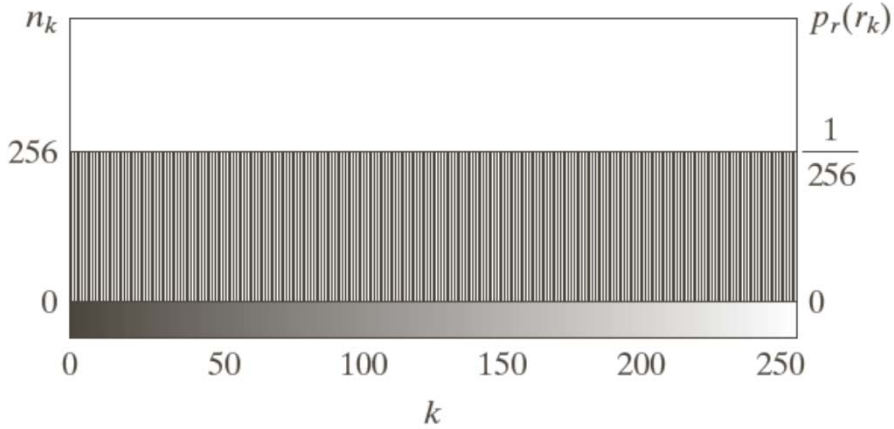
Code 2: $L_{\text{avg}} = 1.81$

$C \approx 4.42, R \approx 0.774$

Code 3：自然的2 bit二进制编码，$C = 4$

空间冗余（Spatial Redundancy）





灰度直方图

特点：

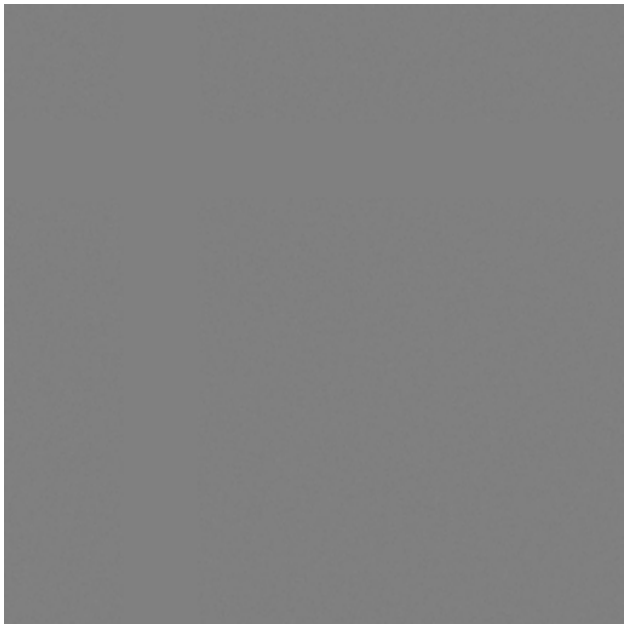
1. 灰度直方图为均匀分布。因此无法用前面的变长编码来压缩
2. 垂直方向的灰度值相互独立
3. 水平方向的灰度值恒定

游程编码（run-length coding）：用一对数字（灰度值，数量）表示一连串相同灰度值，达到压缩目的

压缩比为$\frac{256\times256\times8}{256\times2\times8}=128:1$

不重要信息

- 对于人眼，该图几乎为恒定灰度。
- 只需用一个平均灰度值表示。压缩比为65536:1
- 将所有[125,131]的灰度值表示为128，称为量化
- 不同于前两种冗余（编码冗余和空间冗余）的去除方法，量化会带来信息损失，是不可逆的



256 × 256

图像压缩的性能指标：压缩比

- 图像压缩属于数据压缩（视频、音频、文本等）
- 数据是用来表示信息的
- 相同信息可以用不同的表示方式，有的数据量较大，有的较小
- 假设图像原始的2D数组表示需要b bits，而另一种高效的表示需要b' bits
- 压缩比$C = b/b'$
- 原始表示的冗余度$R = 1 - 1/C$
- 例如，某压缩算法的压缩比为10:1（$C = 10$），那么原始表示的冗余度为0.9

图像压缩的性能指标：保真度（Fidelity）

- 如何评价不同的无损压缩算法？
- 如果不考虑算法的时空复杂度等指标，压缩比是唯一指标
- 如何评价不同的有损压缩算法（不考虑算法复杂度）？
  - 固定压缩比，看谁损失小
  - 固定损失，看谁压缩比大
- 术语为保真度准则。损失小，保真度高；损失大，保真度低。
- 保真度分为客观和主观

客观的保真度准则

- 一种常用的客观保真度：均方根（root-mean-square, rms）误差

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

原图$f(x,y)$，压缩并解压后的图像$\hat{f}(x,y)$

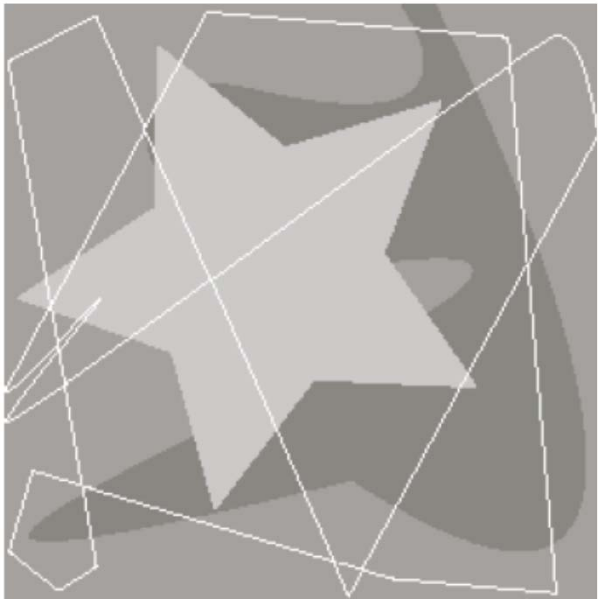
- 另一种客观保真度：将$\hat{f}(x,y)$视为$f(x,y)$与噪声之和；定义均方信噪比

$$\text{SNR}_{\text{ms}} = \frac{\sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \hat{f}(x,y)^2}{\sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \left[\hat{f}(x,y) - f(x,y) \right]^2}$$

- 均方根信噪比：$\text{SNR}_{\text{rms}} = \sqrt{\text{SNR}_{\text{ms}}}$

主观的保真度准则

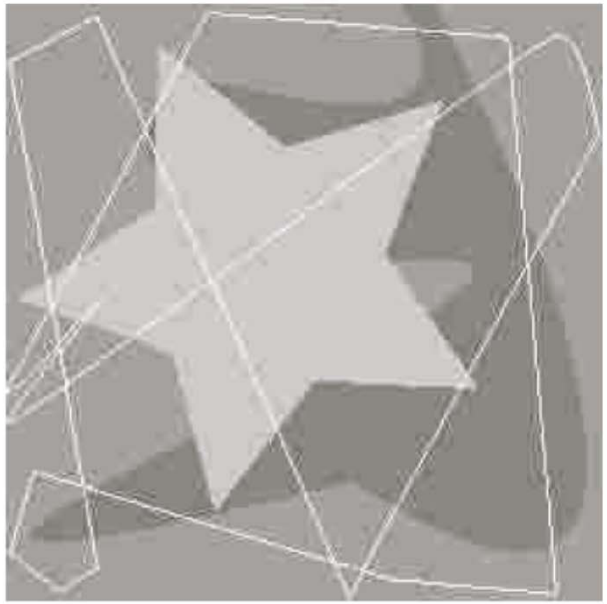
- 客观的保真度准则有2个不足：
  - 很多应用中，图像是给人看的，人的评价才是最终标准
  - 很难找到理想的客观的保真度准则。例如，轻微的图像几何变形会使均方根误差非常大
- 主观评价方法：邀请一组观察者，观看解压图像和原始图像，独立打分（比如5个质量等级：优秀、良好、可用、差、极差），最后取平均
- 如果是对比2种压缩算法，同时显示各自解压图像，由观察者评价相对优劣



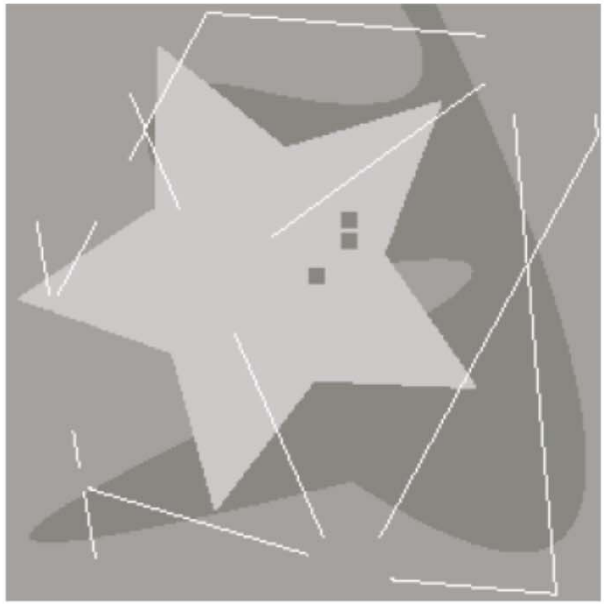
原图

三个不同算法压缩解压后的图像

均方根误差5.17



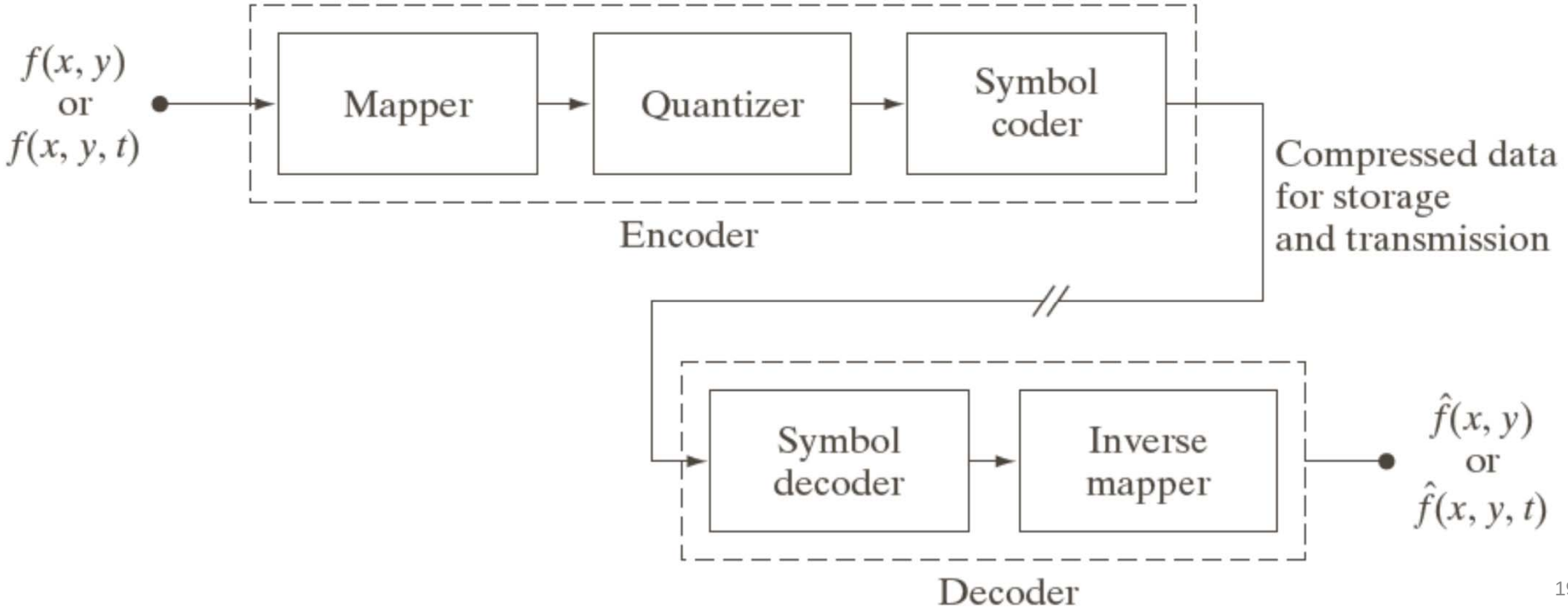
15.67



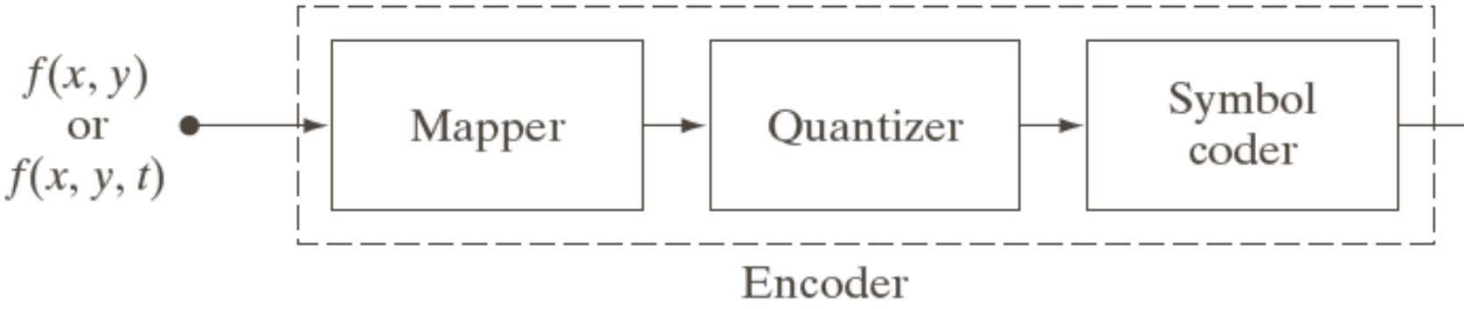
14.17

图像压缩系统框图

- 这是通用的图像/视频压缩系统的方框图
- 两大模块：编码器（encoder）和解码器（decoder）
- 图像查看软件只有解码器；图像编辑软件有编解码器（codec）
- 原始图像$f(x,y)$，原始视频$f(x,y,t)$
- 解压图像$\hat{f}(x,y)$，解压视频$\hat{f}(x,y,t)$

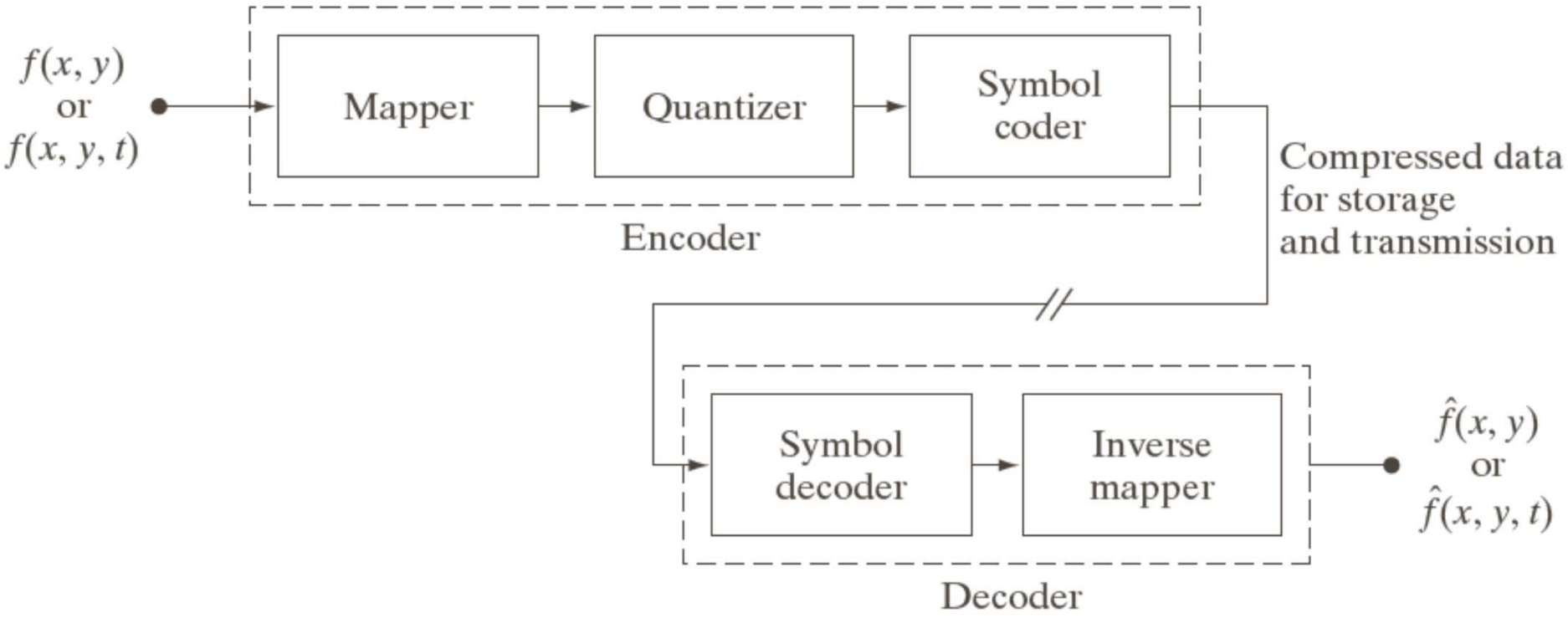


图像编码



- 映射器（Mapper）对图像进行变换以减少空时冗余。通常该变换是可逆的，有可能减少数据量（游程编码），也可能不减少、甚至增加数据量（变换编码：数组大小不变，BYTE --> DOUBLE）
- 量化器（Quantizer）估计保真度要求，降低映射器输出的精度。这步是不可逆的。
- 符号编码器（Symbol coder）按照适当的码本对量化器输出值进行编码（如霍夫曼编码）。这步是可逆的。

图像解码



- 符号解码器（symbol decoder）根据对应码本进行解码
- 逆映射器（inverse mapper）将数据恢复为空域图像（如离散余弦逆变换）
- 量化不可逆

压缩的极限

- 某算法工程师设计出一个无损图像压缩算法，压缩比为C_1
- 一个月后，他对算法进行了改进，将压缩比提高到C_2
- 问题：压缩比能一直提高吗？极限是多少？无穷大？
- 有损压缩存在同样问题：保真度不变，压缩比是否有极限？
- 香农信息论给出了数据压缩的极限
- 下面以图像和文本为例，简要介绍信息论

信源

- 信源是产生各类信息的实体。
- 信源产生的符号是不确定的，可用随机变量及其统计特性描述。
- 通信学科研究信源的目的：对已有信源产生的数据进行压缩。
- 最简单的信源模型：扔硬币，独立同分布。
- 计算机学科的一些领域也研究信源，目的是合成。
  - 图形学领域研究图像的合成
  - 自然语言处理研究文本的合成

合成图像

合成游戏、影视作品中的纹理（森林、沙漠、火焰、水波、人物头发和皮肤等）

合成文本

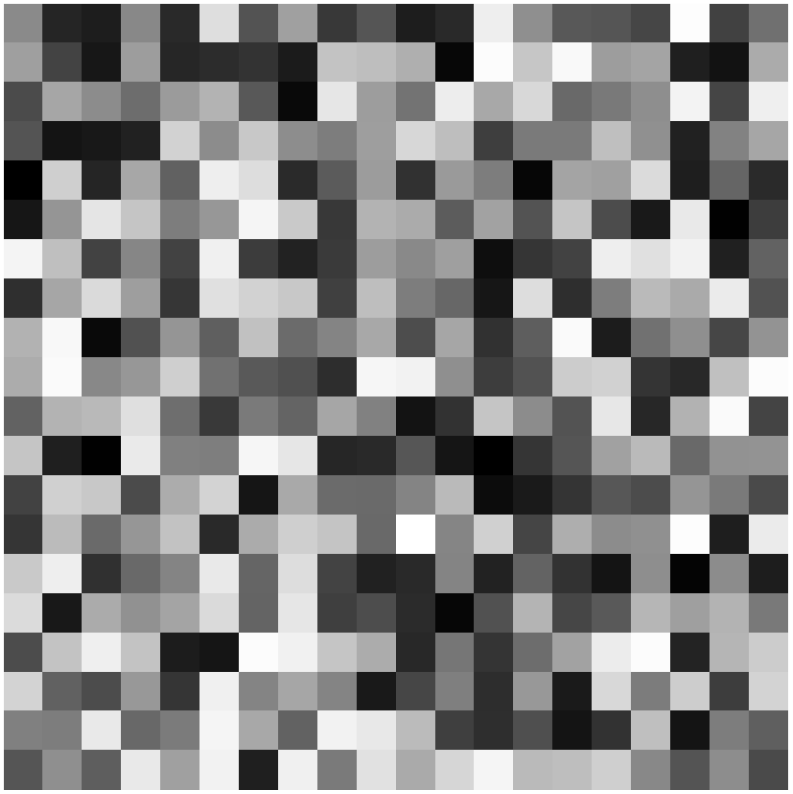
- 新闻（财经、体育）写作机器人
- 聊天机器人

图像信源

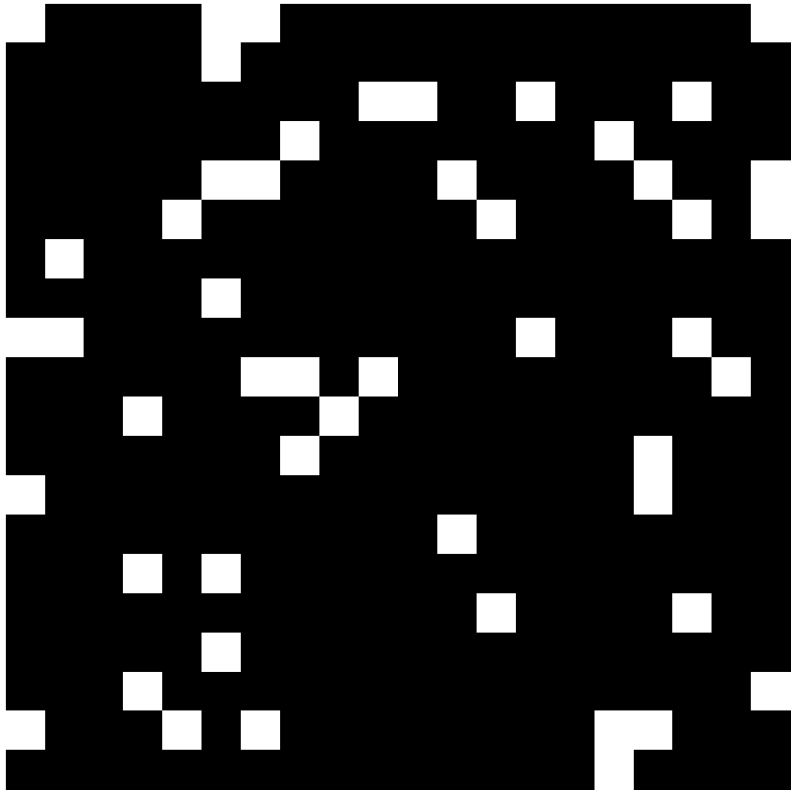
- 一幅灰度图像的像素值是随机变量
- 一幅图像看做是一个信源按照某概率模型产生的
- 信源按照其概率模型可以产生无数图像
- 这些图像的像素值不完全一样，但统计特性是相同的
- 最简单的图像信源模型就是，每个像素的产生是相互独立的，即无记忆信源

无记忆信源产生的图像

[0, 1]均匀分布



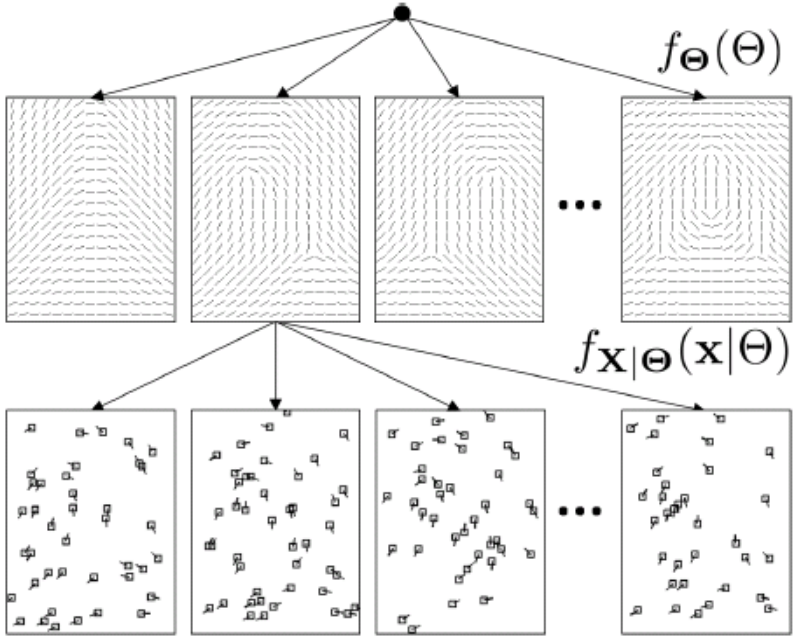
$p(0) = 0.9, p(1) = 0.1$

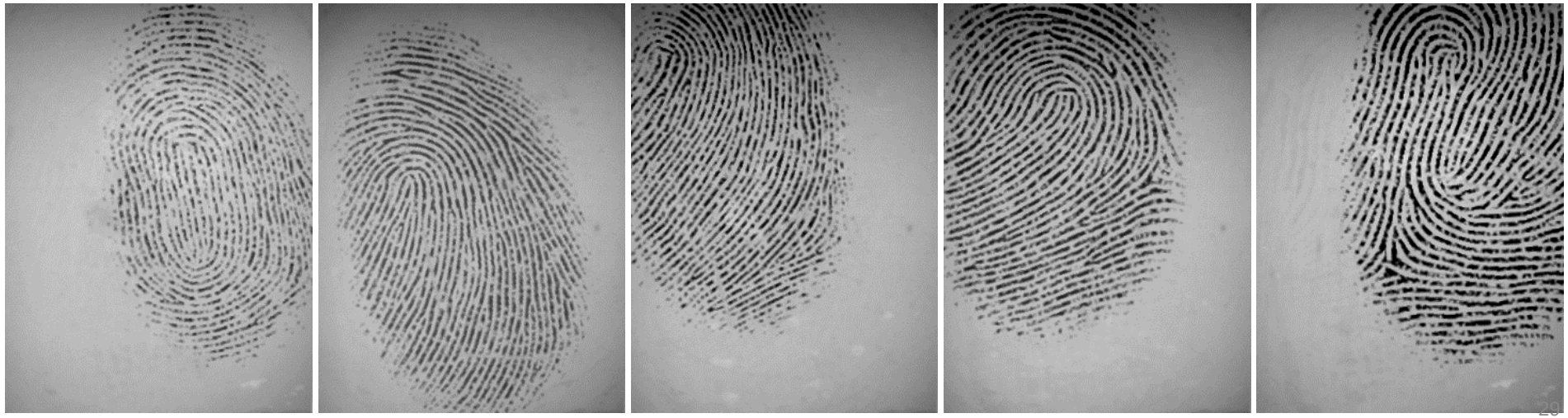


```
x = rand(20,20);
figure(1),imshow(imresize(x,50,'nearest'))
y = rand(20,20)>0.9;
figure(2),imshow(imresize(y,50,'nearest'))
```

利用复杂信源模型生成图像

利用复杂信源模型生成图像





Mark V Shaney

>From mvs Fri Nov 16 17:11 EST 1984 remote from alice

It looks like Reagan is going to say? Ummm... Oh yes, I was looking for. I'm so glad I remembered it. Yeah, what I have wondered if I had committed a crime. Don't eat with your assessment of Reagon and Mondale. Up your nose with a guy from a firm that specifically researches the teen-age market. ……

People, having a much larger number of varieties, and are very different from what one can find in Chinatowns across the country (things like pork buns, steamed dumplings, etc.) They can be cheap, being sold for around 30 to 75 cents apiece (depending on size), are generally not greasy, …..

Can anyone think of myself as a third sex. Yes, I am expected to have….

_-_-_-_-Mark

给3岁儿子讲故事

孩子晚上不睡，从1数到1000不管用。熵为0！

孩子非要听故事，还必须是关于恐龙。为什么？不知道恐龙，在幼儿园抬不起头。就像学信息专业的人不知道深度学习一样。

从前有一只绿色的恐龙，它有999颗牙，第1颗牙是红色的、三角形，第2颗牙是橙色的、三角形……第10颗牙是红色的、长方形……

从前有一只红色的恐龙，它有9999个脚趾头，第1个脚趾头是白色、铁做的，第2个脚趾头是红色、铁做的……第10个脚趾头是白色、豆腐做的……

从前有一只黑色的恐龙，它有99999张嘴，第1张嘴能喷红色的火，第2张嘴能喷绿色的火……第10张嘴能喷红色的岩浆……

改写课文

池塘边的叫声

小鱼在池塘里遇到了青蛙，就上前打招呼：“青蛙大哥，你好！你在干什么？”青蛙说：“我在岸上玩了半天，太热了，来洗个澡。”
小鱼好奇地问：“岸上什么样？我从来没去过。”
青蛙的大眼睛转了一下，说：“那好办。你趴在我的背上，我背着你去看看。”
小鱼连连摆着尾巴，说：“不行，离开水我会干死的！”
青蛙的大眼睛转了两下，说：“有办法了！每天傍晚我到池塘边来，给你讲讲岸上的故事。”
小鱼高兴地游到青蛙身边，亲了亲青蛙的白肚皮，说：“太好了！谢谢你了！”
从此，每天傍晚，池塘边就响起了“呱呱”的叫声，那是青蛙在给小鱼讲岸上的故事呢。

改写课文

animals = {'杨过','韦小宝','郭靖','虚竹','东方不败'};

animal_acts = {'走','蹿','跨','飘','飞','走','移','跳','跳','飞'};

sounds = {'支支吾吾','哈哈哈哈','磨磨唧唧','断断续续','哈哈哈哈','噢噢','旺旺','啊呜','呱呱','呀呀'};

places = {'古墓里','赌场里','桃花岛里','少林寺里','黑木崖里','原始森林里','非洲大草原上','山上','池塘里','树上'};

placematerials = {'姑姑','色子','七公','馒头','绣花针'};

KeSis = {'寂寞','穷','自责','饿','无聊'};

DaZhaoHus = {'趾高气扬地问','拐弯抹角道','嘻皮笑脸地问','鬼鬼祟祟地问','神秘兮兮地问','大义凛然地问','不耐烦地问','歇斯底里地问','悠悠地问','哭丧着脸问'};

ChengHus = {'小朋友','同学','奶奶','妹子','校长','老爷','大人','先生','小姐','女士','主席'};

HowLongs = {'一天','一个月','一辈子','大半辈子','一秒钟','一万年','一袋烟的工夫','一杯茶的时间'};

statuss = {'困','饿','渴','兴奋','惆怅','郁闷','窝火','爽','过瘾'};

act1s = {'洗个澡','抽根烟','放个屁','打个嗝儿','吃个饭','喝杯茶','打个拳','跳个舞','唱个歌','挠个痒','喘口气'};

改写课文

语文课本52页，“池塘边的叫声”的计算机随机改编版。

赌场里的叫声

韦小宝在赌场里遇到了郭靖，就上前趾高气扬地问：“郭靖小朋友，你好！你在干什么？”郭靖说：“我在桃花岛里玩了一秒钟，太爽了，来喘口气。”

韦小宝生气地问：“桃花岛里什么样？我从来没去过。”郭靖的大眼睛转了一下，说：“那好办。你靠在我的背上，我背着你去看看。”

韦小宝连连摆着尾巴，说：“不行，离开色子我会穷死的！”

郭靖的大眼睛转了两下，说：“有办法了！每天傍晚我到赌场里，给你讲讲桃花岛里的故事。”

韦小宝高兴地蹿到郭靖身边，亲了亲郭靖的肚皮，说：“太好了！谢谢你了！”

从此，每天傍晚，赌场里就响起了“磨磨唧唧”的叫声，那是郭靖在给韦小宝讲桃花岛里的故事呢。

改写课文

语文课本52页，“池塘边的叫声”的计算机随机改编版。

古墓里的叫声

杨过在古墓里遇到了东方不败，就上前不耐烦地问：“东方不败同学，你好！你在干什么？”东方不败说：“我在黑木崖里玩了一辈子，太惆怅了，来喘口气。”

杨过啰里啰嗦地问：“黑木崖里什么样？我从来没去过。”东方不败的大眼睛转了一下，说：“那好办。你躺在我的肩膀上，我扛着你去看看。”

杨过连连摆着尾巴，说：“不行，离开姑姑我会寂寞死的！”

东方不败的大眼睛转了两下，说：“有办法了！每天傍晚我到古墓里，给你讲讲黑木崖里的故事。”

杨过高兴地走到东方不败身边，亲了亲东方不败的肚皮，说：“太好了！谢谢你了！”

从此，每天傍晚，古墓里就响起了“哈哈哈哈”的叫声，那是东方不败在给杨过讲黑木崖里的故事呢。

我给研究生讲《信息论基础》

其中讲文本合成

如果到那时候，你还没脱单

或者需要再次脱单

欢迎选修！

无失真编码定理

像素值为离散随机变量X，其取值x的概率为$P(x)$，该事件的信息量为

$$h(x) = \log_2 \frac{1}{P(x)} \quad \text{bits}$$

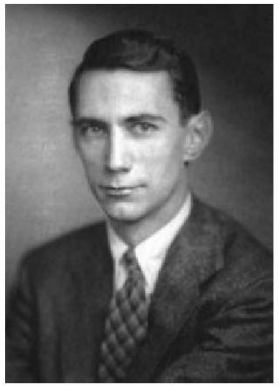
离散随机变量X的熵为平均信息量：

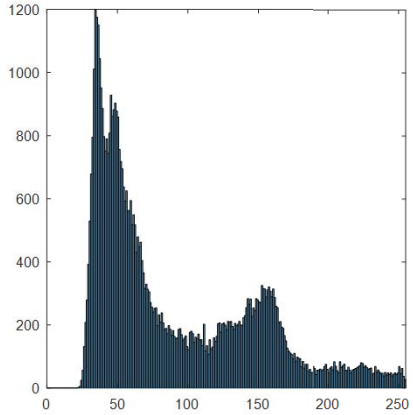
$$H(X) = \sum_{x \in A_X} P(x) \log_2 \frac{1}{P(x)}$$

无记忆信源输出的符号序列进行无失真编码的最小平均码长为熵

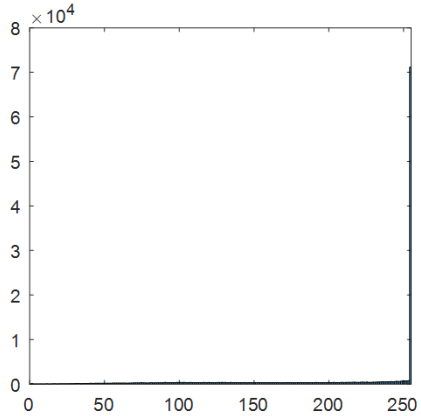
图像的熵

香农

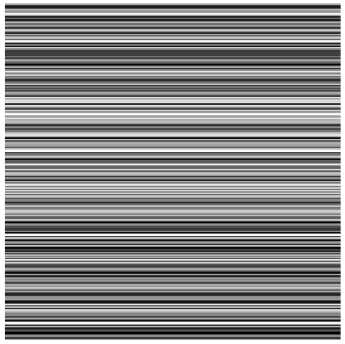


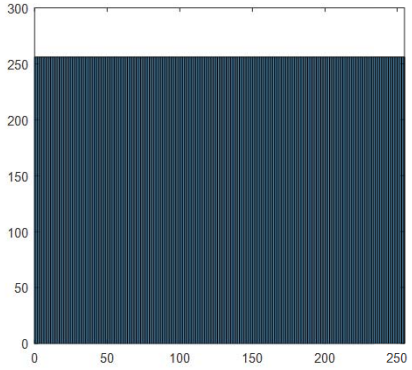


熵7.26 bits

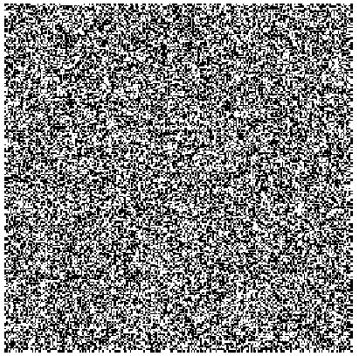


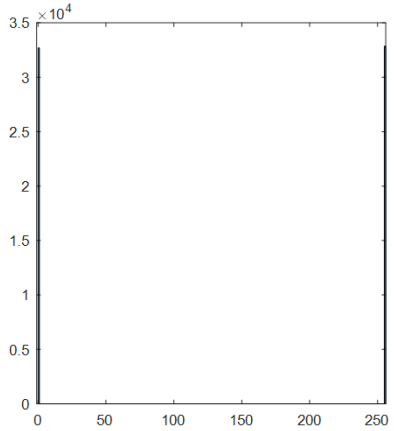
熵4.98 bits





熵8 bits





熵1 bit

```
function myEntropy_test()
% I = imread('..\data\shannon.jpg');
% I = imread('..\data\1_1.bmp');
% I = imread('..\data\Fig0801(b).tif');
I = 255*double(rand(256,256)>0.5);
close all
figure(1),
subplot(1,2,1),imshow(I)
subplot(1,2,2),
hist1 = histogram(I(:),0:256);
xlim([-1 256])
entropy1 = myEntropy(hist1.Values/sum(hist1.Values))
```

```
function e = myEntropy(p)
p(p==0) = []; % remove 0
p = p/sum(p);
e = sum(p.*log2(1./p));
```

Claude Shannon

Father of information theory and digital communications

Like building such devices as models of planes, a radio-controlled model boat and a telegraph system. Personal hero is Thomas Edison.

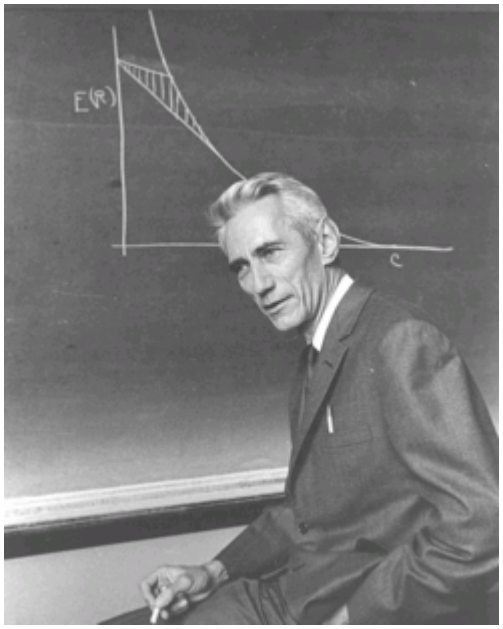
University of Michigan, two bachelor's degrees, electrical engineering & mathematics

MIT, master thesis, “A Symbolic Analysis of Relay and Switching Circuits”. “the most important master's thesis of all time.”

MIT, PhD thesis, “An Algebra for Theoretical Genetics”

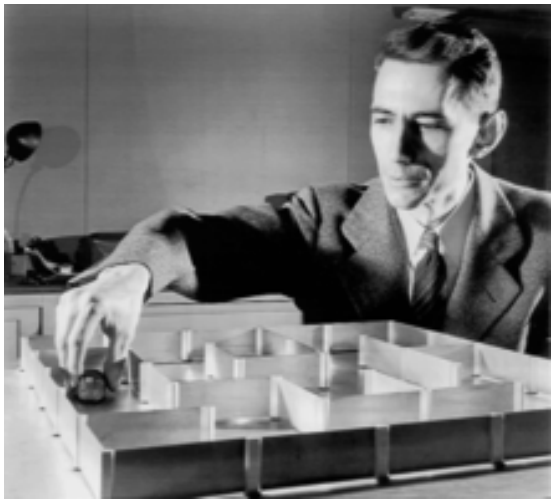
Shannon's Communication Theory

- In 1940, Shannon became a National Research Fellow at the Institute for Advanced Study in Princeton.
- Joined Bell Labs to work on fire-control systems and cryptography during World War II.
- In 1948, published "A Mathematical Theory of Communication" in the July and October issues of the Bell System Technical Journal.
- Co-authored with Warren Weaver, "The Mathematical Theory of Communication" to reach a wider audience.
- "Communication Theory of Secrecy Systems" changed cryptography from art to science.
- Returned to MIT as a professor in 1956. Only gave seminars, not taught regular course, did not supervise students.



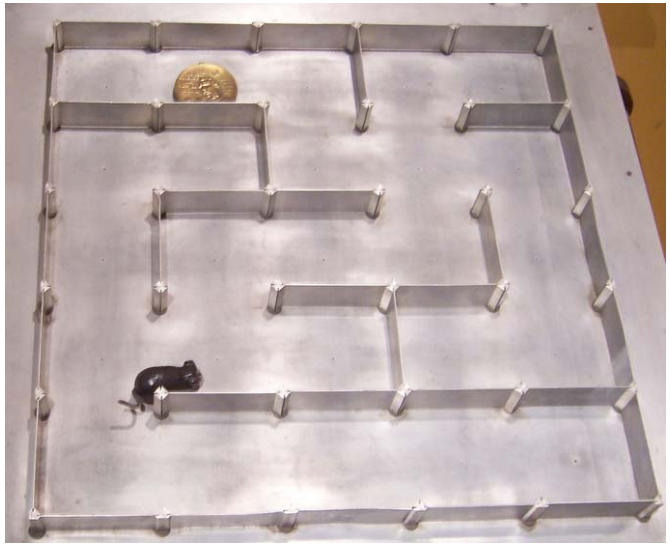
Shannon's Other Work

Theseus, a mechanical mouse, designed to solve mazes. Early example of artificial intelligence.



"Programming a Computer for Playing Chess", large impact on the computer chess programs.

Shannon also became interested in the stock market. He developed some theories about investment growth. In practice, he was also very successfully, both in the general market and, more particularly, in several companies started by talented friends.



内 容

- 基础
- 霍夫曼编码（Huffman Coding）
- 预测编码（Predictive Coding）
- 变换编码（Transform Coding）

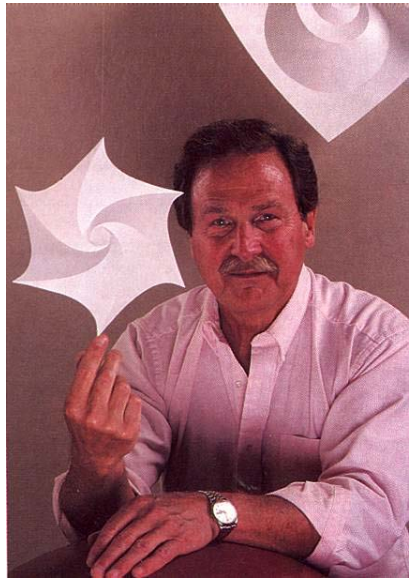
History of Huffman Code

Robert Fano was teaching information theory at MIT.

Both he and Shannon did not find the optimal symbol code.

So, he gave the students at his class two choices: final exam or solve this problem.

A student, Huffman, solved it.



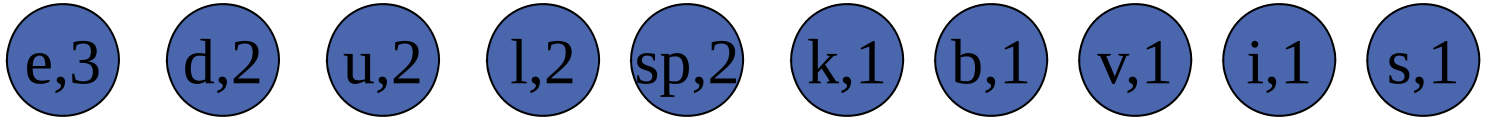
David Huffman

Huffman Coding

- As an example, lets take the string:

  "duke blue devils"

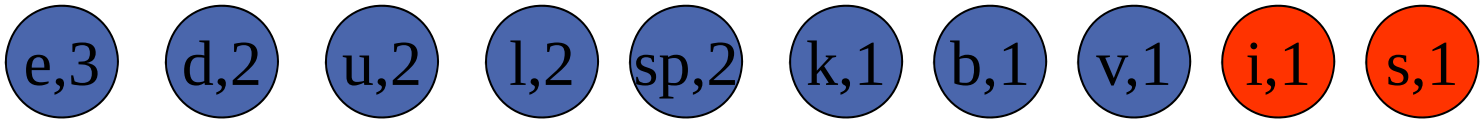
- We first compute count of the characters:
  - e:3, d:2, u:2, l:2, space:2, k:1, b:1, v:1, i:1, s:1
- Next we use a Greedy algorithm to build up a Huffman Tree
  - We start with nodes for each character



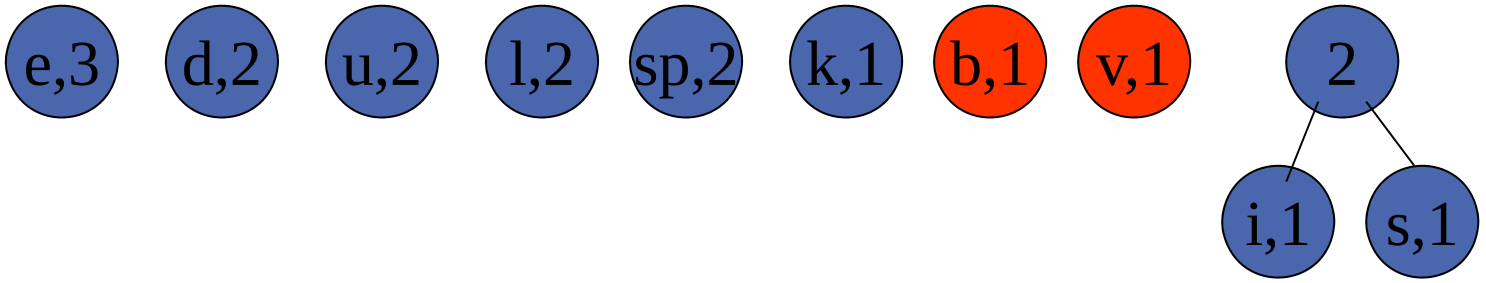
Huffman Coding

- We then pick the nodes with the smallest frequency and combine them together to form a new node (the selection of these nodes is the Greedy part)
- The two selected nodes are removed from the set, but replace by the combined node
- This continues until we have only 1 node left in the set

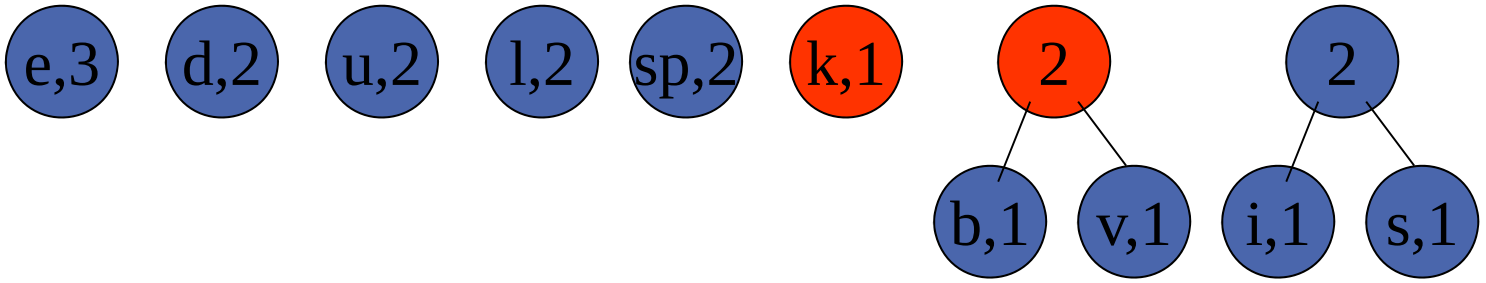
Huffman Coding



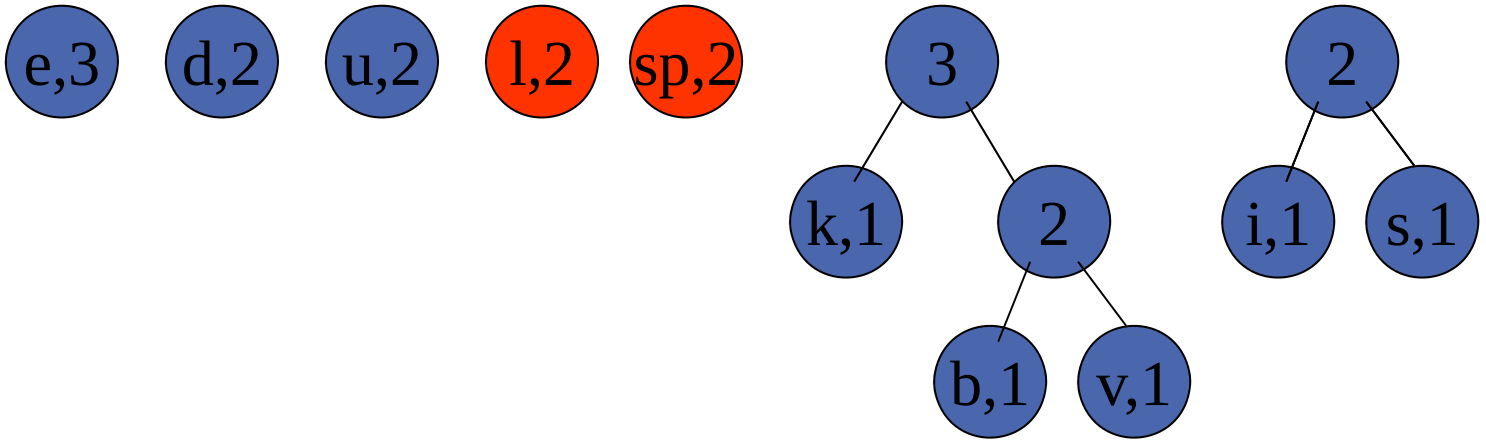
Huffman Coding



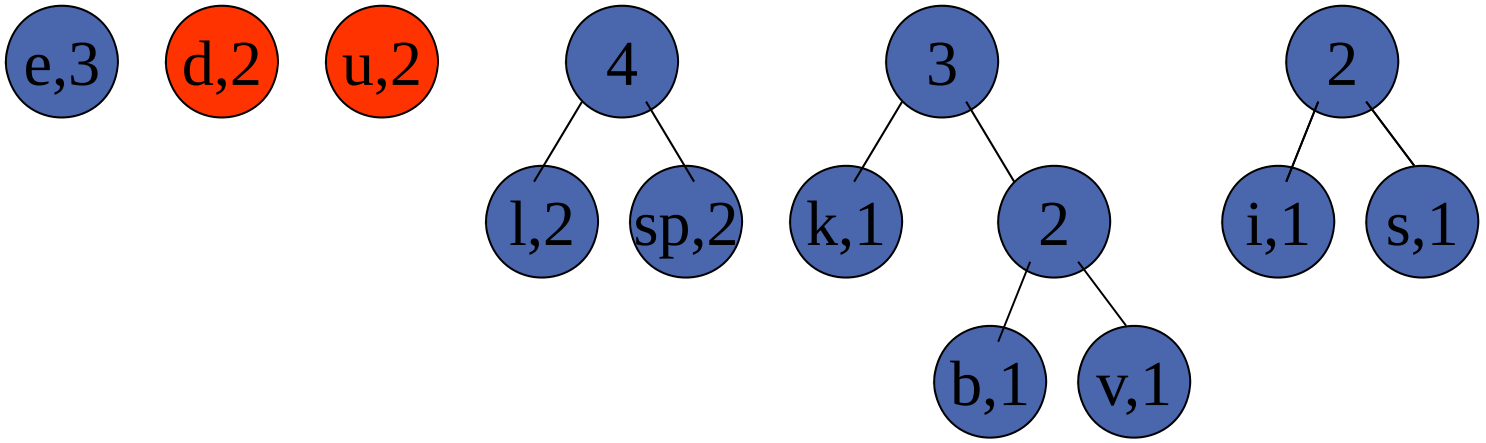
Huffman Coding



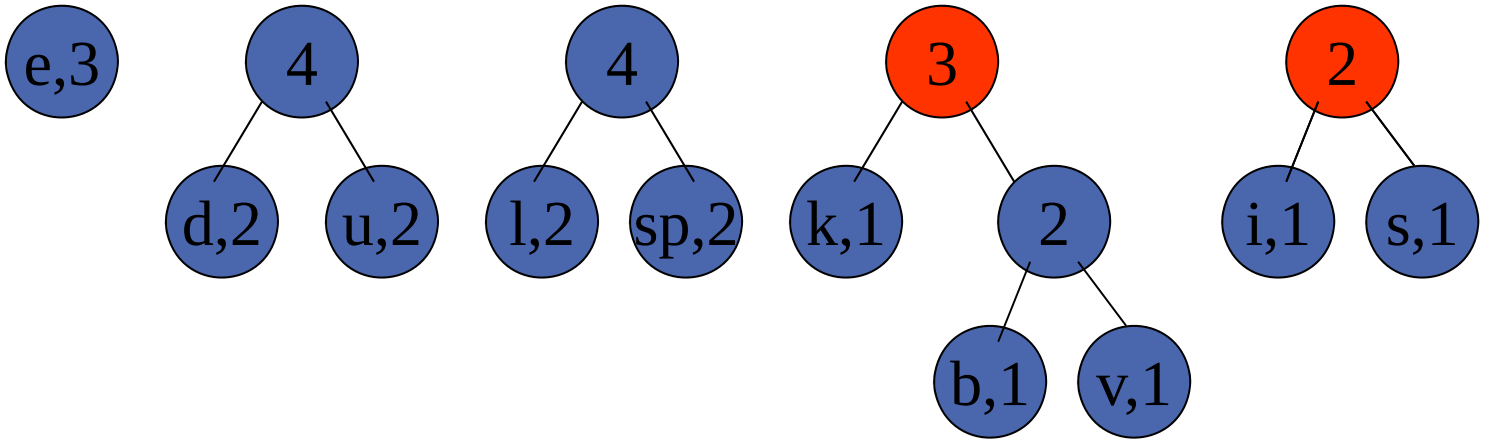
Huffman Coding



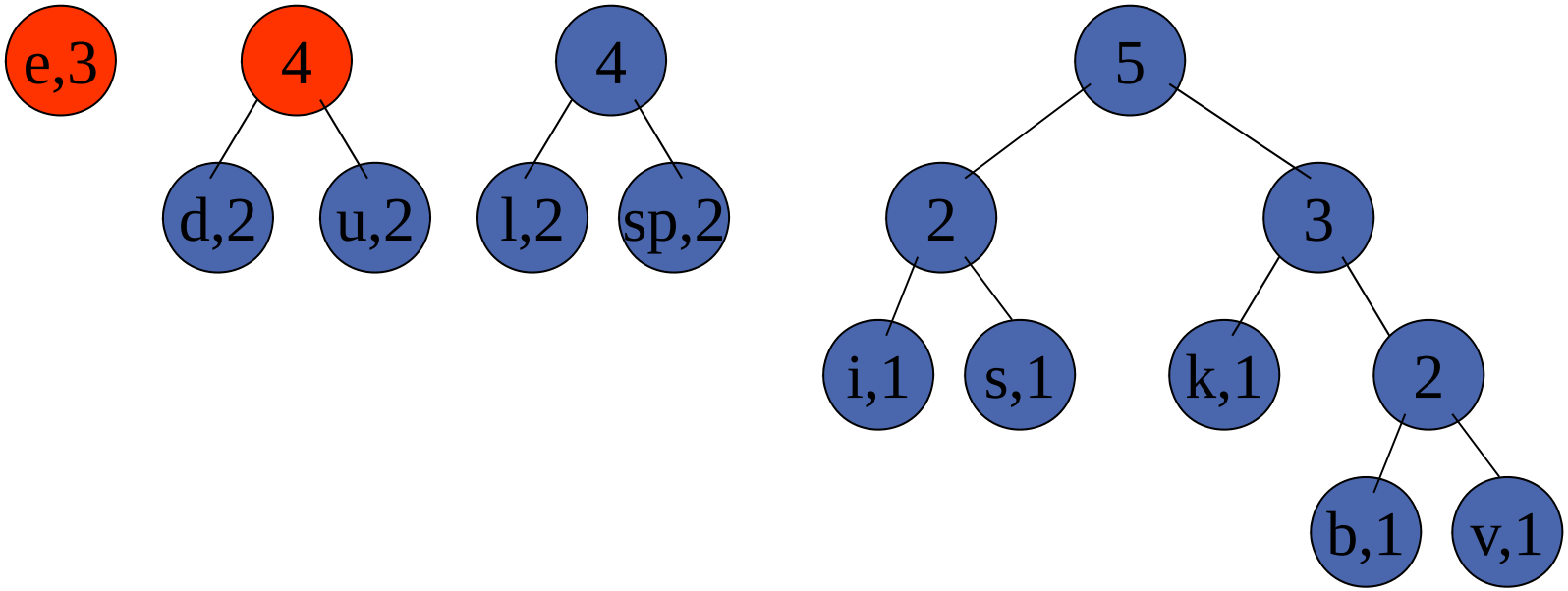
Huffman Coding



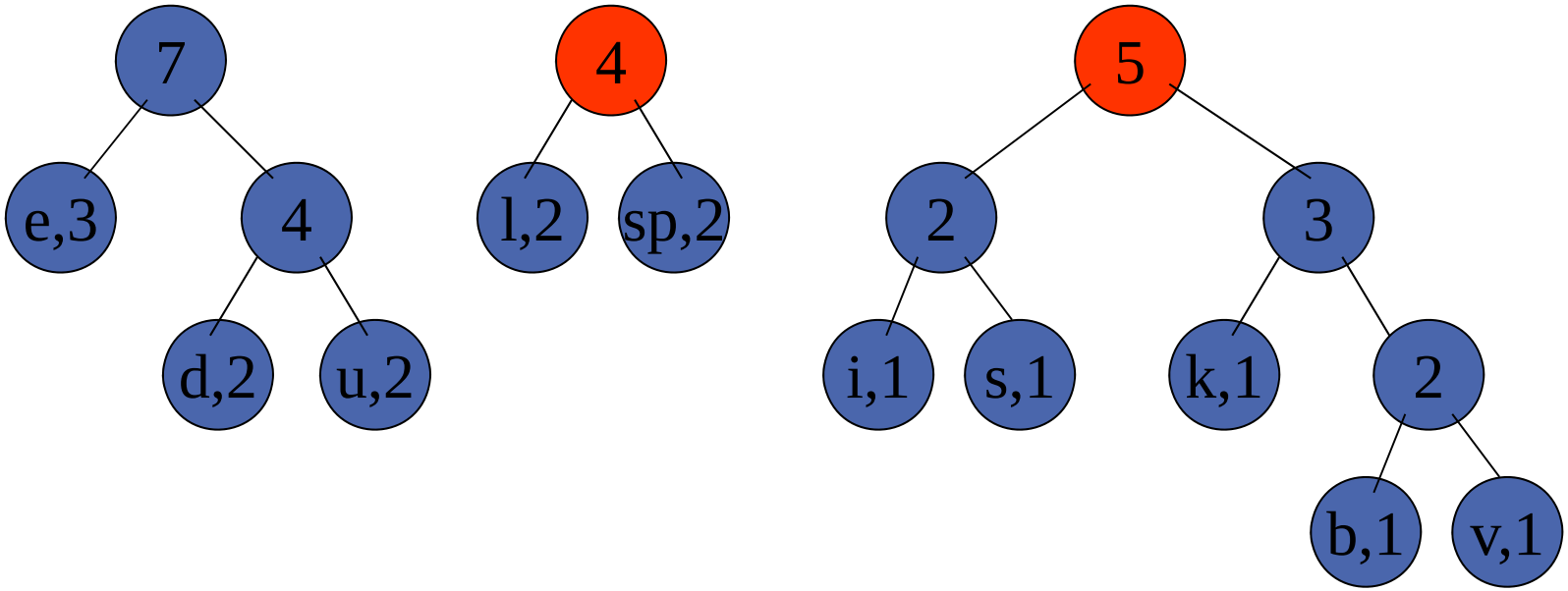
Huffman Coding



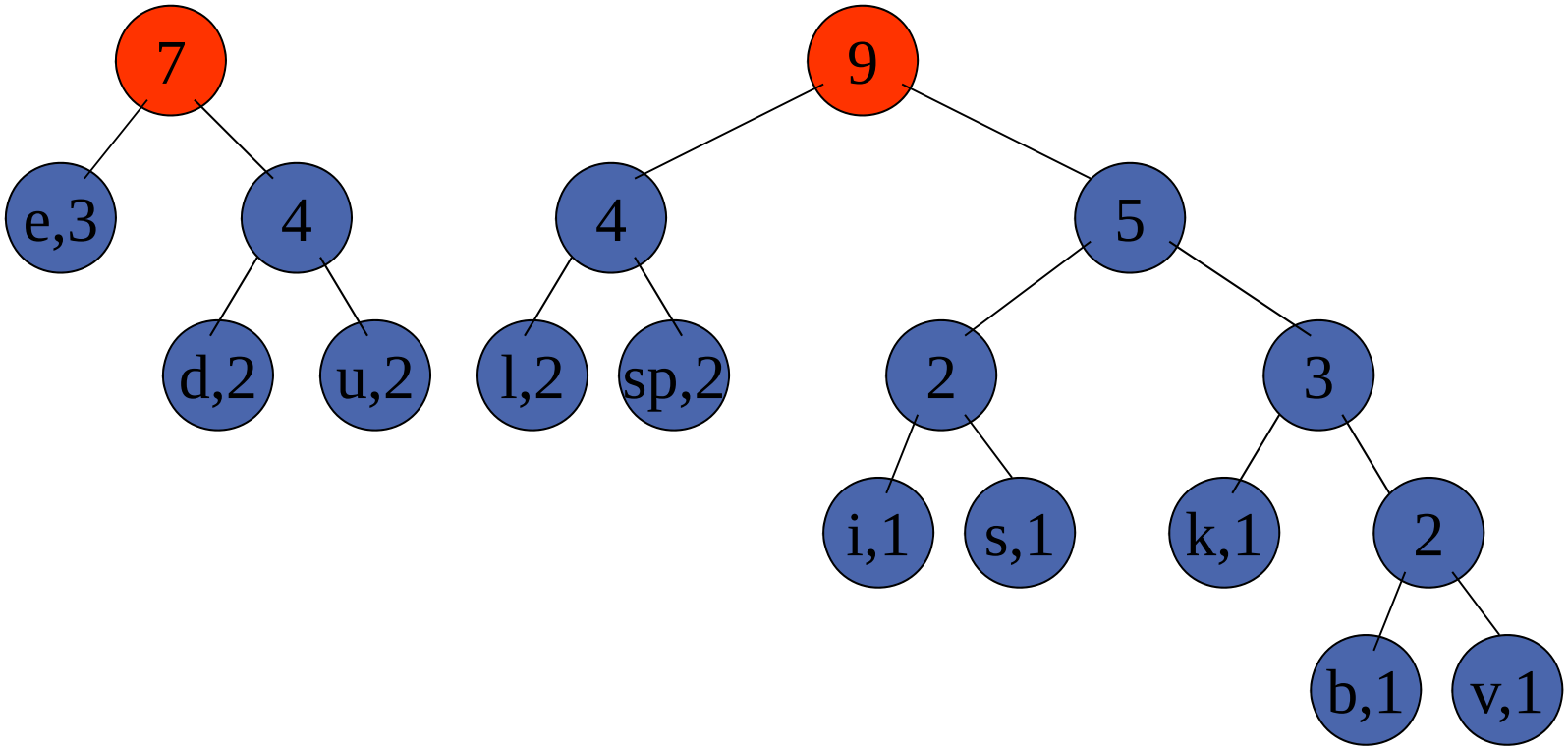
Huffman Coding



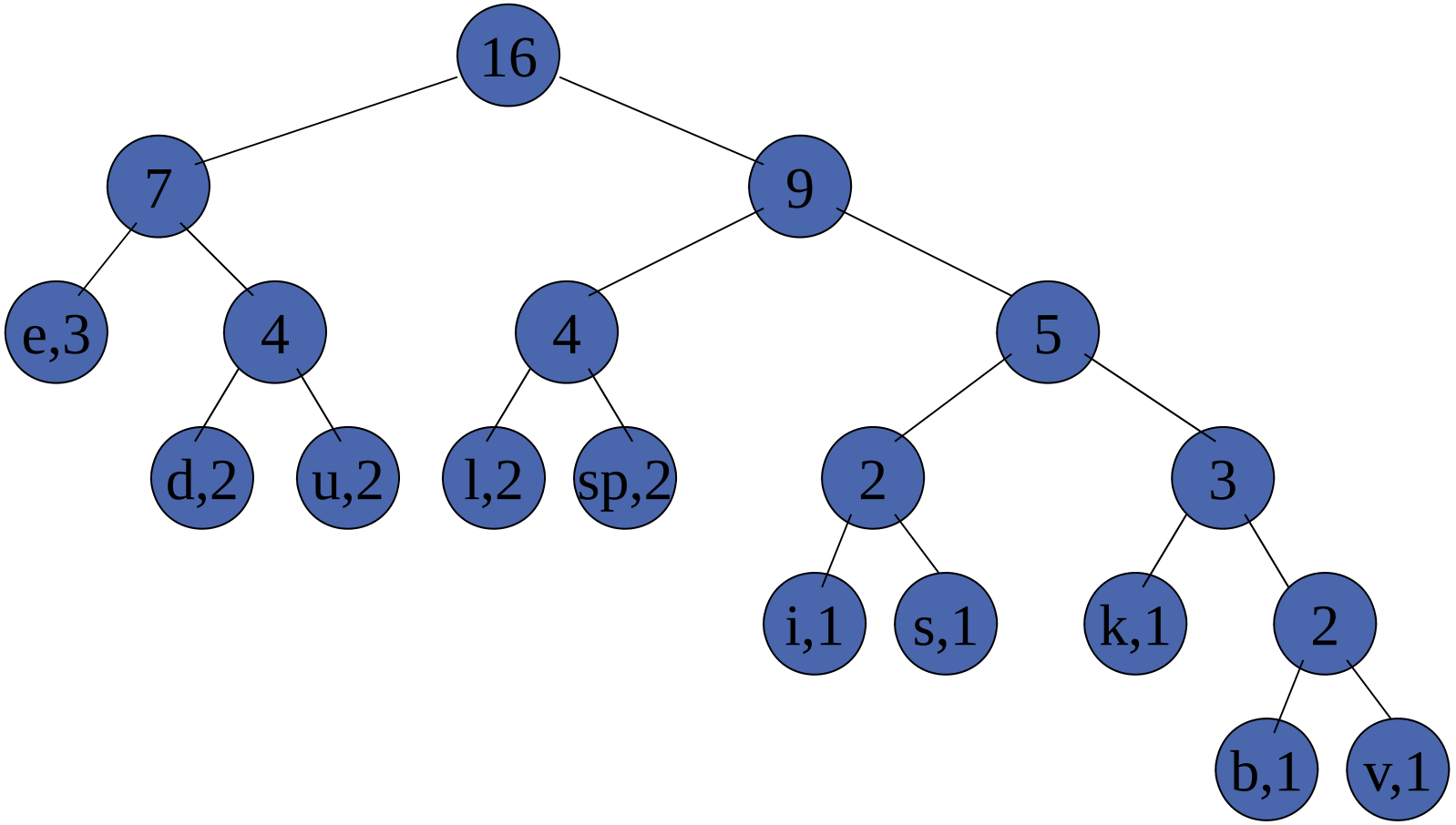
Huffman Coding



Huffman Coding



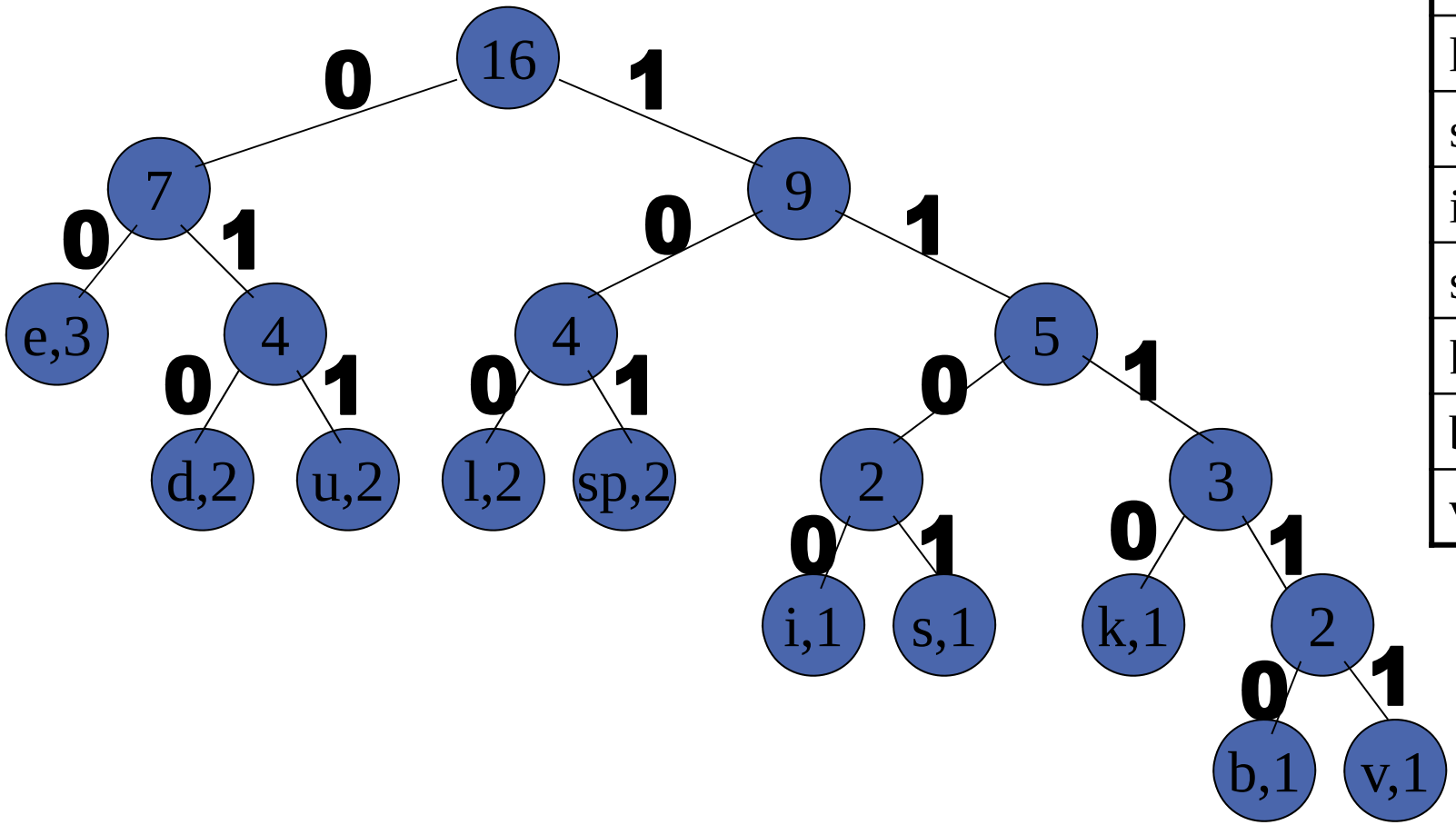
Huffman Coding



Huffman Coding

- Now we assign codes to the tree by placing a 0 on every left branch and a 1 on every right branch
- A traversal of the tree from root to leaf give the Huffman code for that particular leaf character
- Note that no code is the prefix of another code

Huffman Coding



e	00
d	010
u	011
l	100
sp	101
i	1100
s	1101
k	1110
b	11110
v	11111

Huffman Coding

- These codes are then used to encode the string
- Thus, “duke blue devils” turns into:

010 011 1110 00 101 11110 100 011 00 101 010 00 11111 1100 100 1101

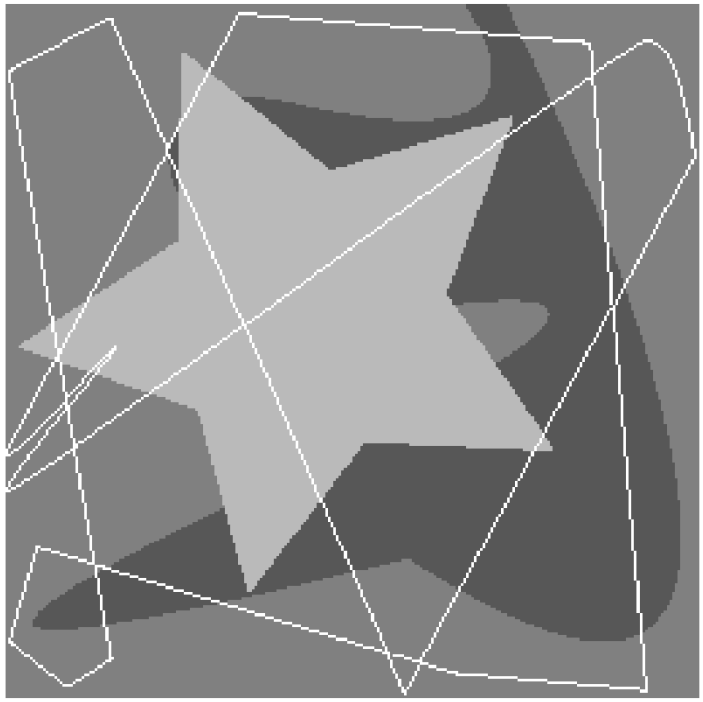
- When grouped into 8-bit bytes:

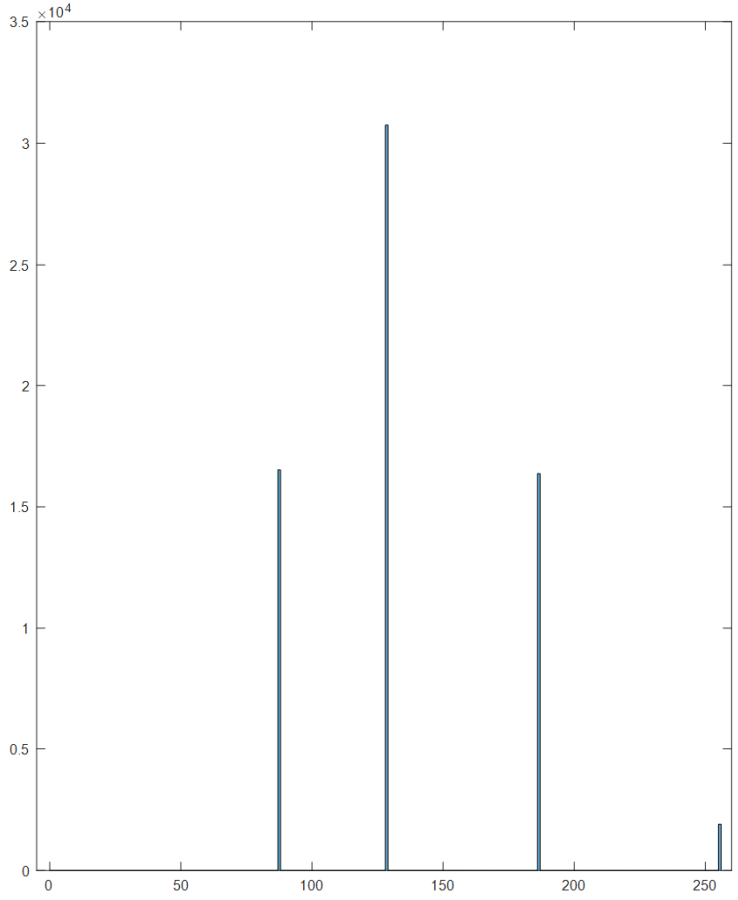
01001111 10001011 11101000 11001010 10001111 11100100 1101xxxx

- Thus it takes 7 bytes of space compared to 16 characters * 1 byte/char = 16 bytes uncompressed

Huffman Coding

- Uncompressing works by reading in the file bit by bit
  - Start at the root of the tree
  - If a 0 is read, head left
  - If a 1 is read, head right
  - When a leaf is reached decode that character and start over again at the root of the tree
- Thus, we need to save Huffman table information as a header in the compressed file
  - Doesn't add a significant amount of size to the file for large files (which are the ones you want to compress anyway)
  - Or we could use a fixed universal set of codes/freqencies



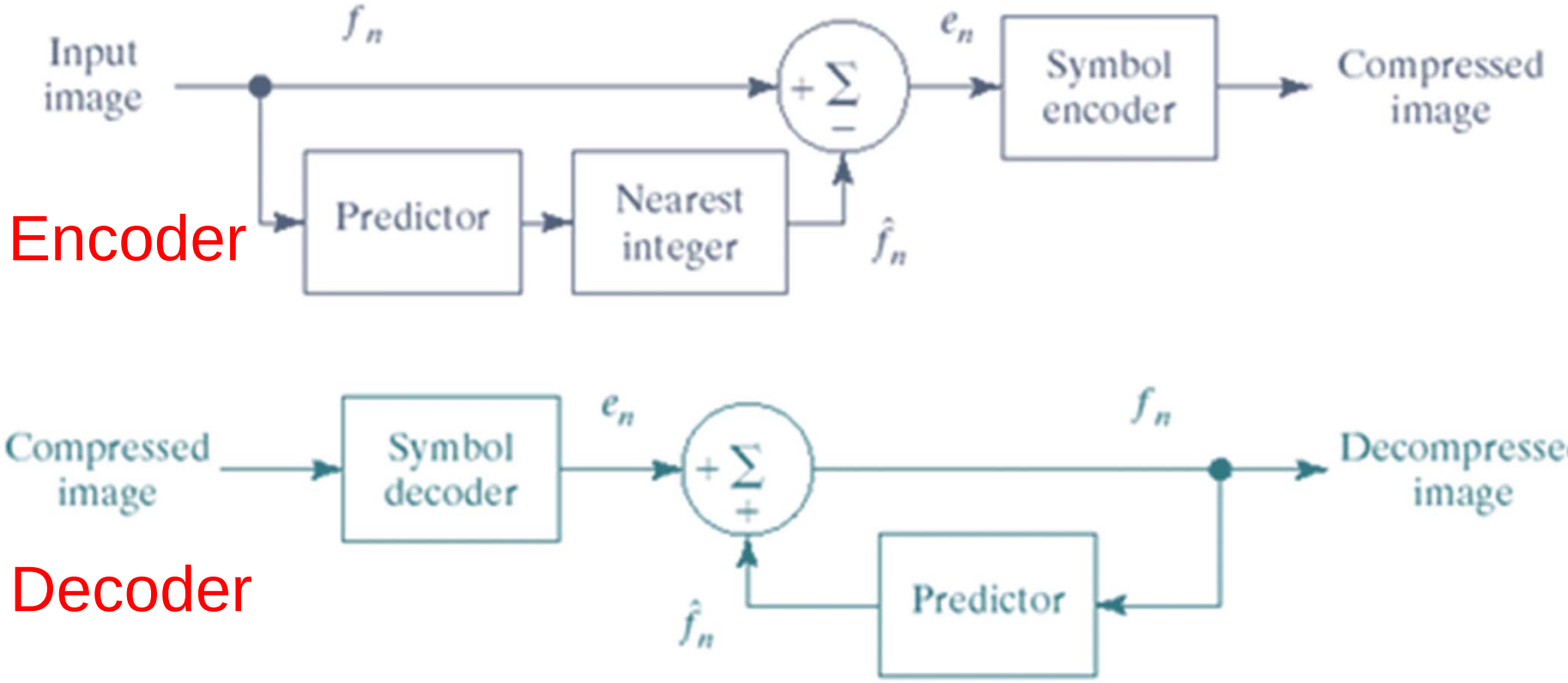

×10^4
3.5
3
2.5
2
1.5
1
0.5
0
0
50
100
150
200
250

```
% huffman_test()
% 2016-11-16

I = imread('..\data\Fig0801(a).tif');
close all
figure(1),
subplot(1,2,1),imshow(I)
subplot(1,2,2),
hist1 = histogram(I(:),0:256);
xlim([-5 260])
entropy1 = myEntropy(hist1.Values/sum(hist1.Values))

idx = find(hist1.Values>0);
symbols = hist1.BinEdges(idx);
prob = hist1.Values(idx)/sum(hist1.Values);
dict = huffmandict(symbols,prob);

disp('Show huffman dictionary')
for k = 1:size(dict,1)
    fprintf('%3d %.2f ',dict{k,1}, prob(k));
    fprintf('%d',dict{k,2});
    fprintf('\n');
end

sig = double(I(:));
hcode = huffmanenco(sig,dict);
dhsig = huffmandeco(hcode,dict);
isequal(sig,dhsig)
```

内 容

- 基础
- 霍夫曼编码（Huffman Coding）
- 预测编码（Predictive Coding）
- 变换编码（Transform Coding）

Lossless Predictive Coding



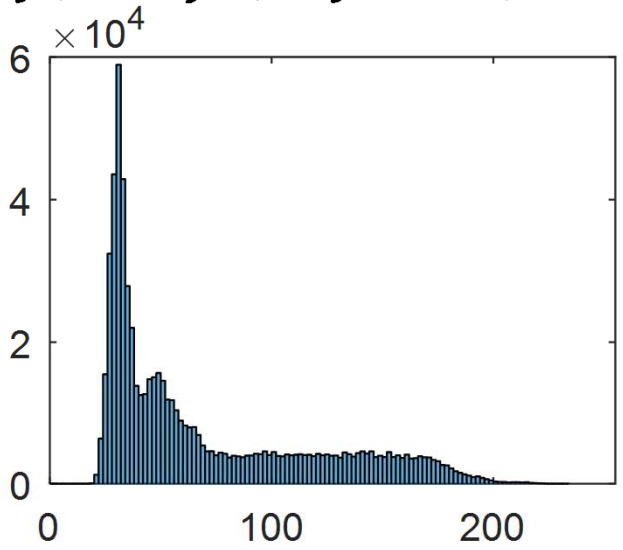
Predictor: $\hat{f}_n = \text{round}[\sum_{i=1}^{m} \alpha_i f_{n-i}]$

Prediction error: $e_n = f_n - \hat{f}_n$

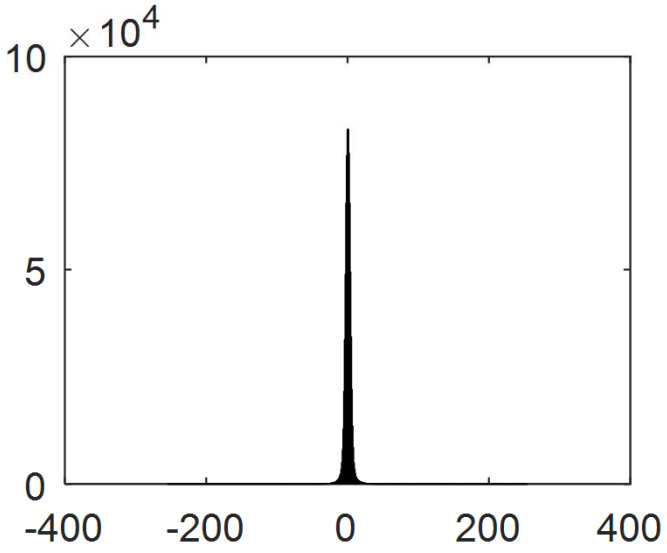
$$f_n = e_n + \hat{f}_n$$

为什么无损预测编码可以实现压缩？

Predictor: $\hat{f}(x,y) = f(x,y-1)$



熵6.7986



熵3.9673

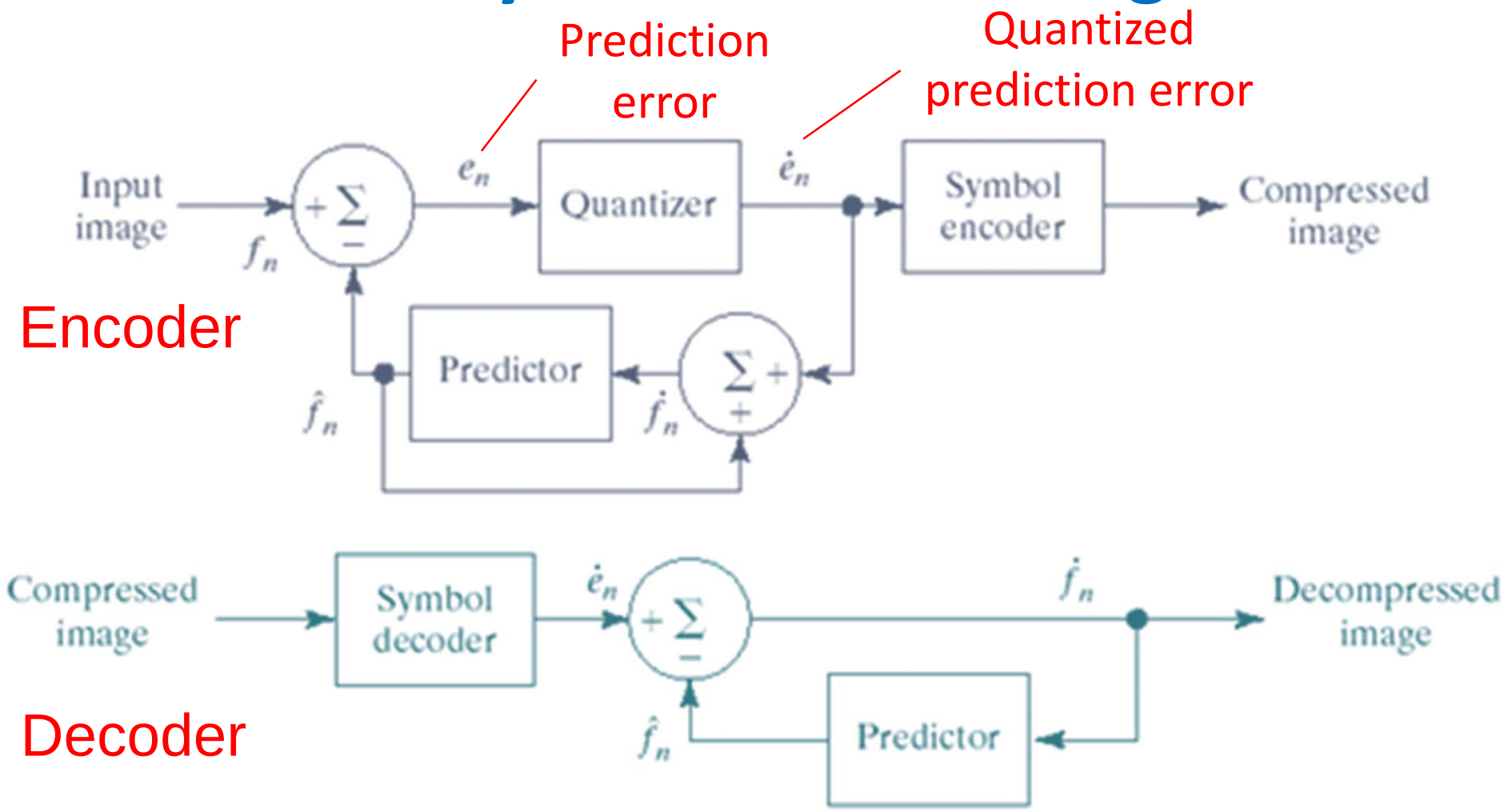
```
function im2lpc_test()

I = imread('..\data\Fig0819(a).tif');
I = double(I);
E = I(:,2:end)-I(:,1:end-1); % prediction error

close all
figure(1),
subplot(2,2,1),imshow(I,[])
subplot(2,2,3),imshow(E,[])
subplot(2,2,2),histogram(I(:));
xlim([0 255])
subplot(2,2,4),histogram(E(:));
xlim([-255 255])

std1 = std(I(:))
std2 = std(E(:))

hist1 = histogram(I(:),0:255);
entropy1 = myEntropy(hist1.Values/sum(hist1.Values))

hist2 = histogram(E(:),-255:255);
entropy2 = myEntropy(hist2.Values/sum(hist2.Values))
```

```
function Y = im2lpc(X, f)
if nargin<2
    f = 1; % default: previous pixel coding
end
X = double(X);
[M, N] = size(X);
P = zeros(M,N);
Xs = X;
for j = 1:length(f)
    Xs = [zeros(M,1) Xs(:,1:end-1)];
    P = P + f(j) * Xs;
end
Y = X - round(P);
```

```
function x = lpc2im(y, f)
if nargin<2
    f = 1; % default: previous pixel coding
end
f = f(end:-1:1);
[m, n] = size(y);
order = length(f);
f = repmat(f, m, 1);
x = zeros(m, n + order);
for j = 1:n
    jj = j + order;
    x(:, jj) = y(:, j) + round(sum(f(:, order:-1:1).*x(:, (jj-
1):-1:(jj-order)),2));
end
x = x(:, order+1:end);
```

```
function im2lpc_test2()
% Jianjiang Feng
% 2016-11-17
I = imread('..\data\Fig0834(a).tif');
I = double(I);

e = im2lpc(I);

[symbols, prob] = prob4huffman(e);
dict = huffmandict(symbols,prob);
hcode = huffmanenco(e(:),dict);
[huffmanCode,huffmanCodeLen] = huffmanDouble2Bin(hcode);

cr = imratio(uint8(I), huffmanCode)

hcode = huffmanBin2Double(huffmanCode,huffmanCodeLen);
e2 = huffmandeco(hcode, dict);
e2 = reshape(e2, size(I,1), size(I,2));
g = lpc2im(e2);
rmse = CompareImages(I, g)
```

Lossy Predictive Coding



Input to the predictor is $\dot{f}_n$; same for encoder and decoder.

Delta Modulation (增量调制)

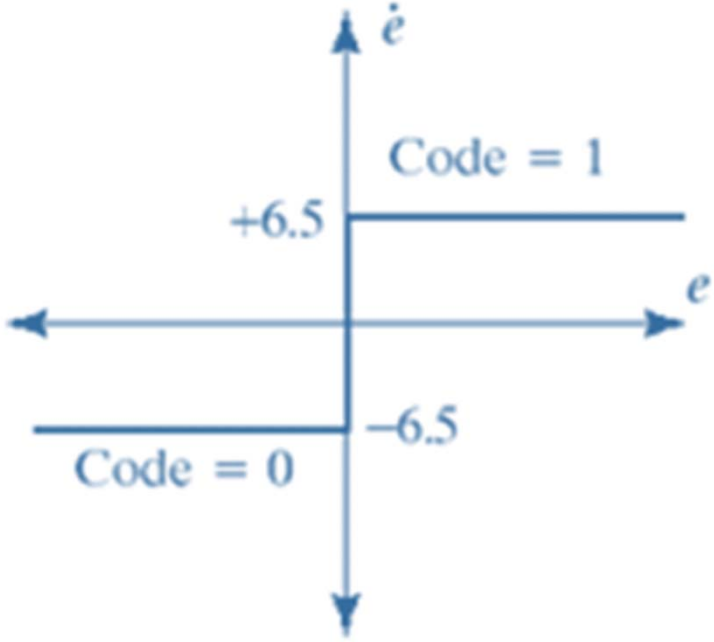
A simple but well-known lossy predictive coding method.

Predictor:

$$\hat{f}_n = \dot{f}_{n-1}$$

Quantizer:

$$\dot{e}_n = \begin{cases} 6.5 & \text{for } e_n > 0 \\ -6.5 & \text{otherwise} \end{cases}$$

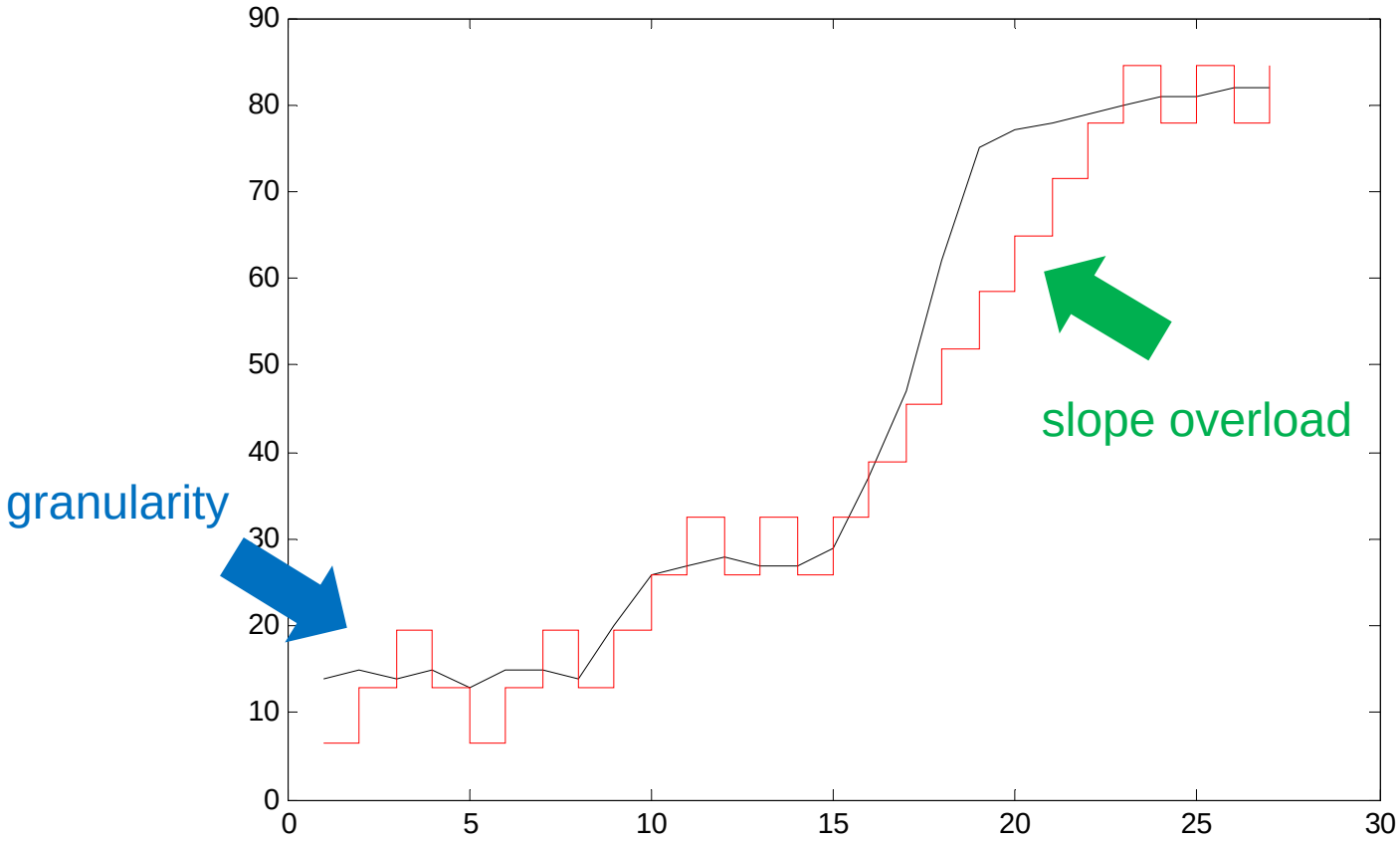


Prediction error is represented by a single bit.

Delta Modulation

```
predictor = [0 1]; % y(k)=x(k-1)
partition = [0];
codebook = [-6.5 6.5];
x = [14,15,14,15,13,15,15,14,20,26,27,28,27,27,29,37,47,
62,75,77,78,79,80,81,81,82,82]; % Original signal
% Quantize x using DPCM.
encodedx = dpcmenco(x,codebook,partition,predictor);
% Try to recover x from the modulated signal.
decodedx = dpcmdeco(encodedx,codebook,predictor);
t = 1:length(x);
figure(1),clf,plot(t,x,'k-'),hold on;
stairs(t,decodedx,'r-');
```

Delta Modulation



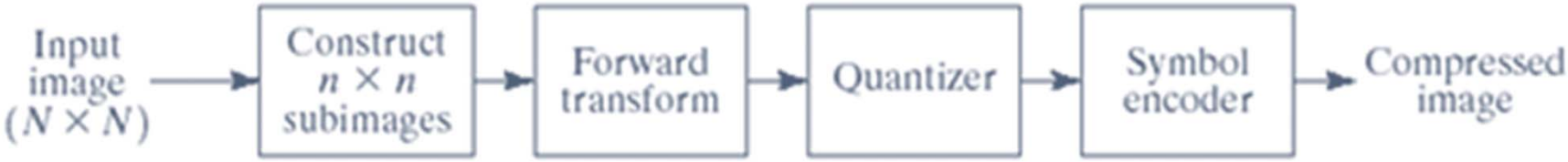
内 容

- 基础
- 霍夫曼编码（Huffman Coding）
- 预测编码（Predictive Coding）
- 变换编码（Transform Coding）

Transform Coding

Predictive coding operates on spatial domain.

Transform coding operates on transform domain.



Encoder

Decoder

Compression is achieved during quantization step.

Transform

Forward transform

$$T(u,v) = \sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x,y)g(x,y,u,v)$$

Image

Basis functions

Transform coefficients

Inverse transform

$$f(x,y) = \sum_{u=0}^{N-1}\sum_{v=0}^{N-1} T(u,v)h(x,y,u,v)$$

Transforms

Discrete Fourier Transform

$$g(x,y,u,v) = \frac{1}{N^2} e^{-j2\pi(ux+vy)/N}$$

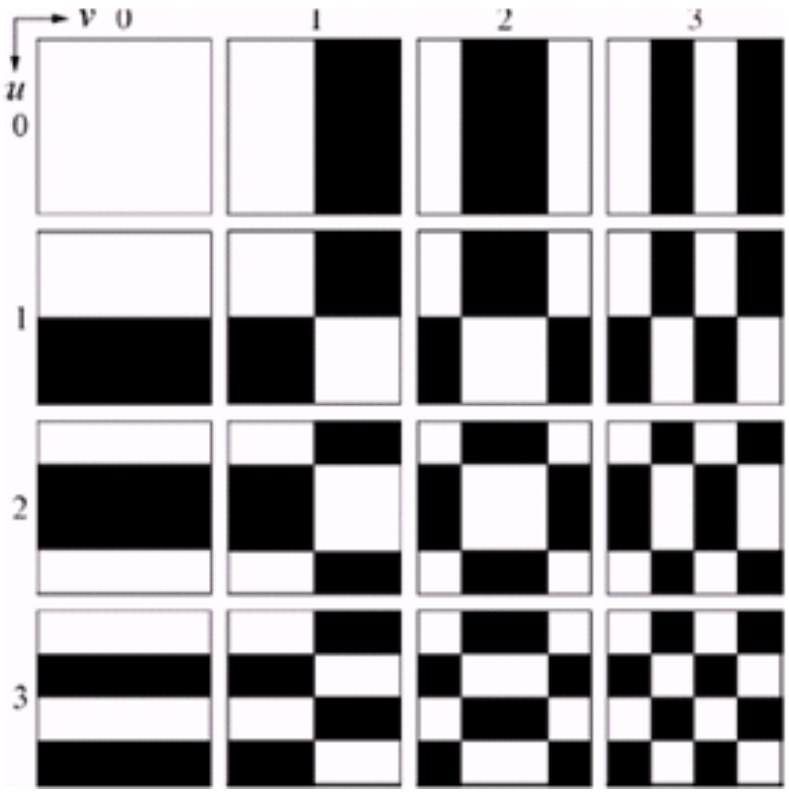
Walsh-Hadamard Transform

$$g(x,y,u,v) = \frac{1}{N}(-1)^{\sum_{i=0}^{m-1}\lfloor b_i(x)p_i(u)+b_i(y)p_i(v)\rfloor}$$

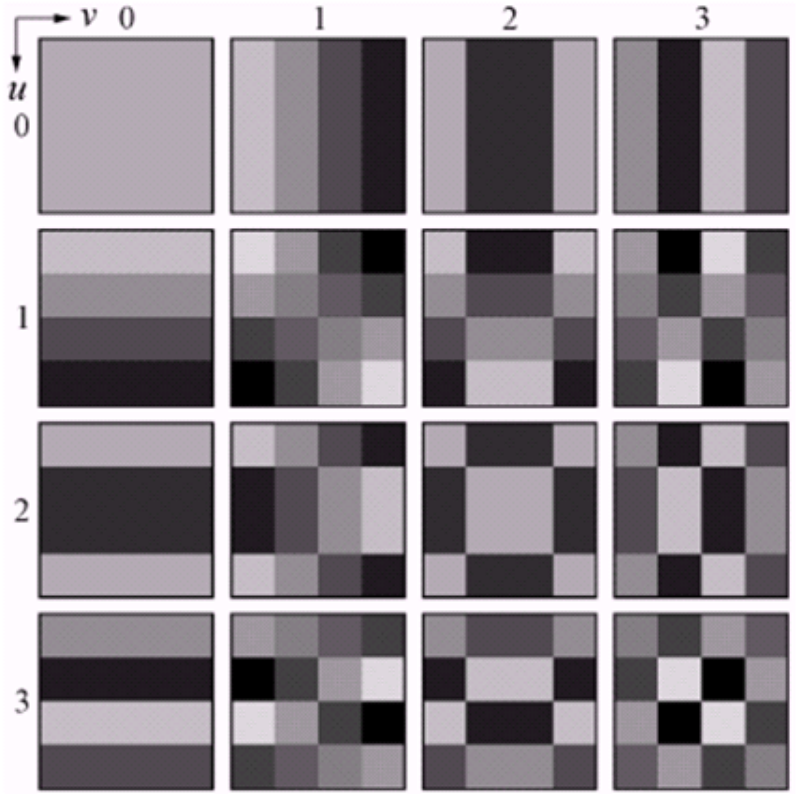
Discrete Cosine Transform

$$g(x,y,u,v) = \alpha(u)\alpha(v)\cos\left[\frac{(2x+1)u\pi}{2N}\right]\cos\left[\frac{(2y+1)v\pi}{2N}\right]$$

基函数（Basis Functions）

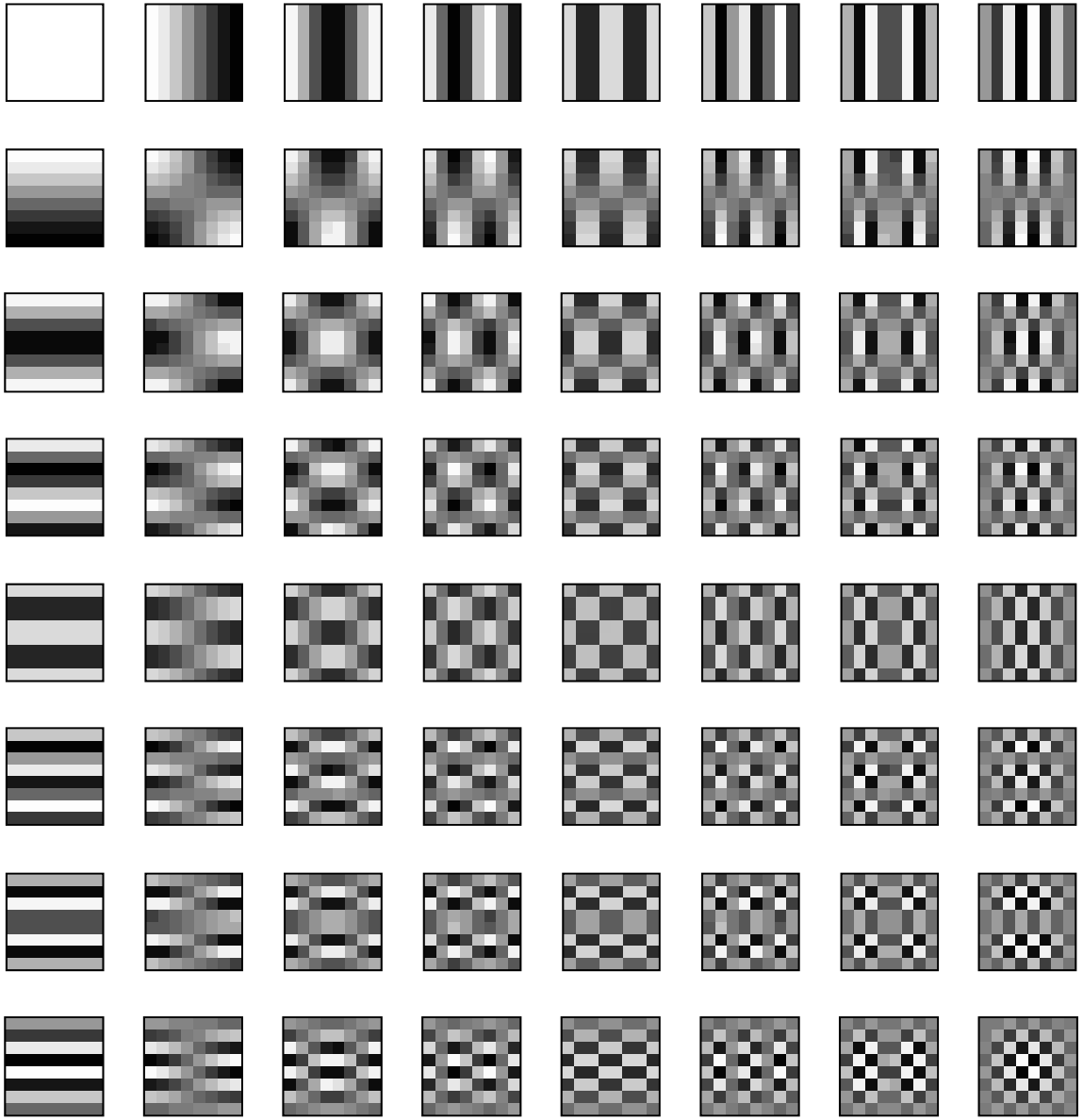


Walsh-Hadamard Transform



Discrete Cosine Transform

8×8 DCT基函数



```
function DctCheck()
% Jianjiang Feng
% 2016-11-14

M = 8; N = 8;

% I = zeros(M,N,1,M*N);
[DX,DY] = meshgrid(0:M-1);
close all
figure(1)
for p = 0:M-1
    for q = 0:N-1
        base = cos(pi*(2*DX+1)*p/(2*M)).*cos(pi*(2*DY+1)*q/(2*N));
        index = sub2ind([M N],p+1,q+1);
%         I(:,:,1,index) = base;
        subplot(M,N,index),
        imshow(0.5*(base+1));
    end
end

% montage(I);
```

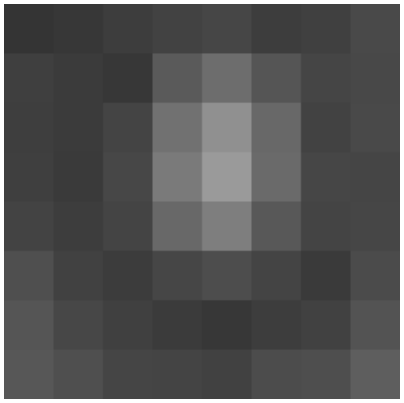
Retain 50% important transform coefficients

Fourier

Hadamard

cosine

JPEG图像压缩标准

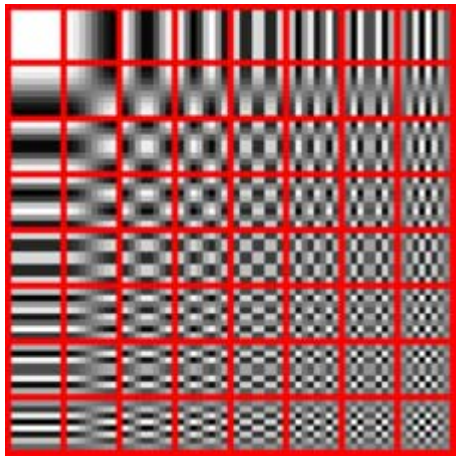


$$\begin{bmatrix} 52 & 55 & 61 & 66 & 70 & 61 & 64 & 73 \\ 63 & 59 & 55 & 90 & 109 & 85 & 69 & 72 \\ 62 & 59 & 68 & 113 & 144 & 104 & 66 & 73 \\ 63 & 58 & 71 & 122 & 154 & 106 & 70 & 69 \\ 67 & 61 & 68 & 104 & 126 & 88 & 68 & 70 \\ 79 & 65 & 60 & 70 & 77 & 68 & 58 & 75 \\ 85 & 71 & 64 & 59 & 55 & 61 & 65 & 83 \\ 87 & 79 & 69 & 68 & 65 & 76 & 78 & 94 \end{bmatrix}.$$

-128, to reduce the dynamic range

$$f = \begin{bmatrix} -76 & -73 & -67 & -62 & -58 & -67 & -64 & -55 \\ -65 & -69 & -73 & -38 & -19 & -43 & -59 & -56 \\ -66 & -69 & -60 & -15 & 16 & -24 & -62 & -55 \\ -65 & -70 & -57 & -6 & 26 & -22 & -58 & -59 \\ -61 & -67 & -60 & -24 & -2 & -40 & -60 & -58 \\ -49 & -63 & -68 & -58 & -51 & -60 & -70 & -53 \\ -43 & -57 & -64 & -69 & -73 & -67 & -63 & -45 \\ -41 & -49 & -59 & -60 & -63 & -52 & -50 & -34 \end{bmatrix}$$

DCT



$$T = \begin{bmatrix} -415.38 & -30.19 & -61.20 & 27.24 & 56.13 & -20.10 & -2.39 & 0.46 \\ 4.47 & -21.86 & -60.76 & 10.25 & 13.15 & -7.09 & -8.54 & 4.88 \\ -46.83 & 7.37 & 77.13 & -24.56 & -28.91 & 9.93 & 5.42 & -5.65 \\ -48.53 & 12.07 & 34.10 & -14.76 & -10.24 & 6.30 & 1.83 & 1.95 \\ 12.12 & -6.55 & -13.20 & -3.95 & -1.88 & 1.75 & -2.79 & 3.14 \\ -7.73 & 2.91 & 2.38 & -5.94 & -2.38 & 0.94 & 4.30 & 1.85 \\ -1.03 & 0.18 & 0.42 & -2.42 & -0.88 & -3.02 & 4.12 & -0.66 \\ -0.17 & 0.14 & -1.07 & -4.19 & -1.17 & -0.10 & 0.50 & 1.68 \end{bmatrix}$$

Note the top-left corner entry with the rather large magnitude. This is the DC coefficient.
The remaining 63 coefficients are called the AC coefficients.

The DCT temporarily increases the bit-depth of the data

Quantization

$$\hat{T}(u,v) = \text{round}\left[\frac{T(u,v)}{Z(u,v)}\right]$$

$$T = \begin{bmatrix} -415.38 & -30.19 & -61.20 & 27.24 & 56.13 & -20.10 & -2.39 & 0.46 \\ 4.47 & -21.86 & -60.76 & 10.25 & 13.15 & -7.09 & -8.54 & 4.88 \\ -46.83 & 7.37 & 77.13 & -24.56 & -28.91 & 9.93 & 5.42 & -5.65 \\ -48.53 & 12.07 & 34.10 & -14.76 & -10.24 & 6.30 & 1.83 & 1.95 \\ 12.12 & -6.55 & -13.20 & -3.95 & -1.88 & 1.75 & -2.79 & 3.14 \\ -7.73 & 2.91 & 2.38 & -5.94 & -2.38 & 0.94 & 4.30 & 1.85 \\ -1.03 & 0.18 & 0.42 & -2.42 & -0.88 & -3.02 & 4.12 & -0.66 \\ -0.17 & 0.14 & -1.07 & -4.19 & -1.17 & -0.10 & 0.50 & 1.68 \end{bmatrix}$$

$$\begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

Normalization matrix Z

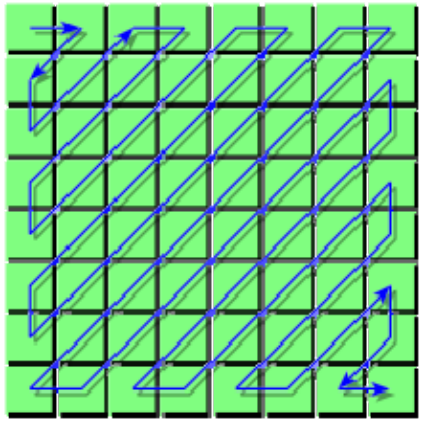
$$\begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -3 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Normalized coefficients $\hat{T}$

Loseless Coding

$$\begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -3 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Normalized coefficients $\hat{T}$



[-26 -3 0 -3 -2 -6 2 -4
1
-3 1 1 5 1 …-1 -1
EOB]

DC (-26) is coded using DPCM (predicted by DC of previous block)
AC is coded using Huffman code

直流成分的编码

DC就是每个块的均值，存在空间冗余，可用LPC编码

Decode

$$\begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -3 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Normalized coefficients $\hat{T}$

$$\begin{bmatrix} -416 & -33 & -60 & 32 & 48 & -40 & 0 & 0 \\ 0 & -24 & -56 & 19 & 26 & 0 & 0 & 0 \\ -42 & 13 & 80 & -24 & -40 & 0 & 0 & 0 \\ -42 & 17 & 44 & -29 & 0 & 0 & 0 & 0 \\ 18 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$\hat{T}Z$

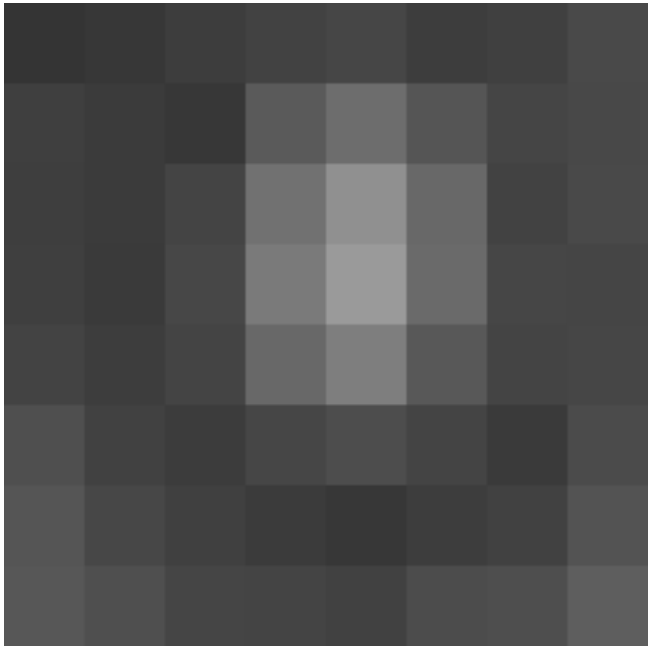
$$\begin{bmatrix} -66 & -63 & -71 & -68 & -56 & -65 & -68 & -46 \\ -71 & -73 & -72 & -46 & -20 & -41 & -66 & -57 \\ -70 & -78 & -68 & -17 & 20 & -14 & -61 & -63 \\ -63 & -73 & -62 & -8 & 27 & -14 & -60 & -58 \\ -58 & -65 & -61 & -27 & -6 & -40 & -68 & -50 \\ -57 & -57 & -64 & -58 & -48 & -66 & -72 & -47 \\ -53 & -46 & -61 & -74 & -65 & -63 & -62 & -45 \\ -47 & -34 & -53 & -74 & -60 & -47 & -47 & -41 \end{bmatrix}$$

Round(IDCT($\hat{T}Z$))

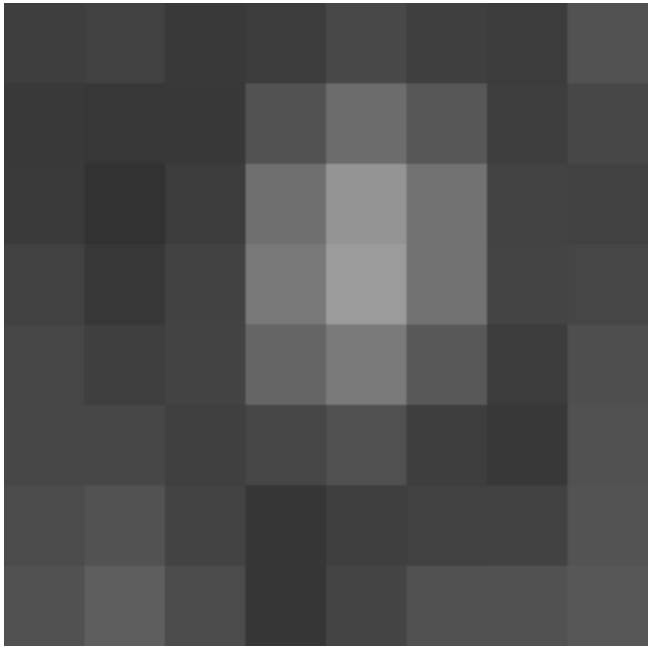
$$\begin{bmatrix} 62 & 65 & 57 & 60 & 72 & 63 & 60 & 82 \\ 57 & 55 & 56 & 82 & 108 & 87 & 62 & 71 \\ 58 & 50 & 60 & 111 & 148 & 114 & 67 & 65 \\ 65 & 55 & 66 & 120 & 155 & 114 & 68 & 70 \\ 70 & 63 & 67 & 101 & 122 & 88 & 60 & 78 \\ 71 & 71 & 64 & 70 & 80 & 62 & 56 & 81 \\ 75 & 82 & 67 & 54 & 63 & 65 & 66 & 83 \\ 81 & 94 & 75 & 54 & 68 & 81 & 81 & 87 \end{bmatrix}$$

Round(IDCT($\hat{T}Z$))+128

Decode



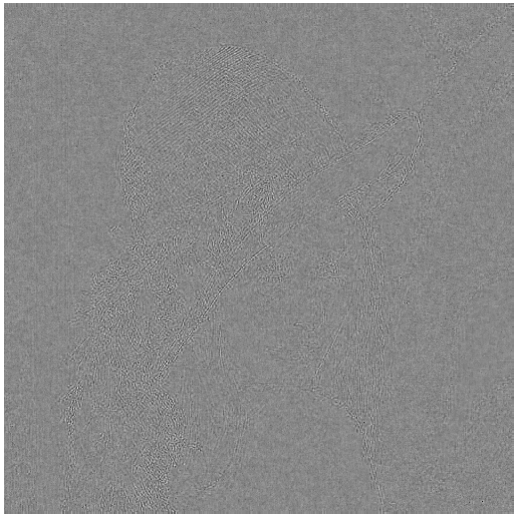
Original image

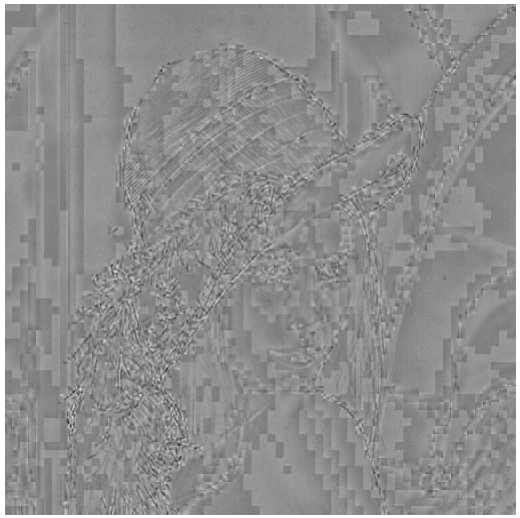


Decoded image

不同质量

将归一化矩阵Z乘以quality可以得到不同压缩性能



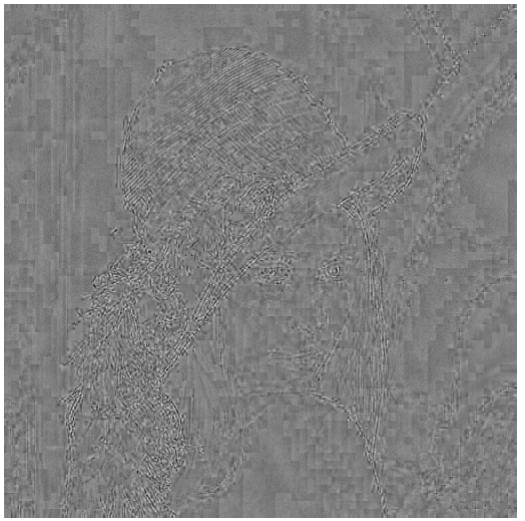


原图	quality 1	quality 10
压缩比,rmse	5:1, 3.8	33:1, 10.7

```
% im2jpeg_test
f = imread('..\data\Fig0809(a).tif');

c1 = im2jpeg(f, 1);
f1 = jpeg2im(c1);
cr1 = imratio(f, c1)
rmse1 = CompareImages(f, f1)

c4 = im2jpeg(f, 10);
f4 = jpeg2im(c4);
cr4 = imratio(f, c4)
rmse4 = CompareImages(f, f4)

close all
figure(1),
ax(1)=subplot(1,3,1);imshow(f);
ax(2)=subplot(1,3,2);imshow(f1);
ax(3)=subplot(1,3,3);imshow(f4);
linkaxes(ax);

figure(2),
ax2(1)=subplot(1,3,1);imshow(f);
ax2(2)=subplot(1,3,2);imshow(double(f1)-double(f),[]);
ax2(3)=subplot(1,3,3);imshow(double(f4)-double(f),[]);
linkaxes(ax2);

figure(3),
ax3(1)=subplot(1,3,1);imshow(f);
ax3(2)=subplot(1,3,2);imshow(f4);
ax3(3)=subplot(1,3,3);imshow(double(f4)-double(f),[]);
linkaxes(ax3);
```

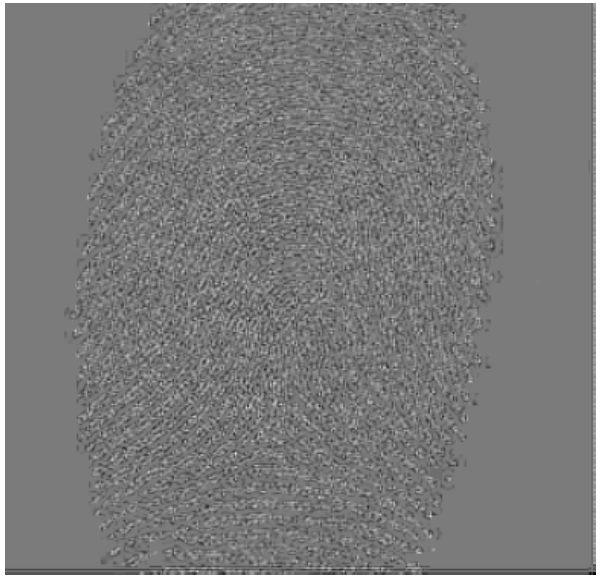
不同图像（quality=4）

压缩比16:1



rmse 6.6

压缩比6:1



rmse 8.7