10110100101…

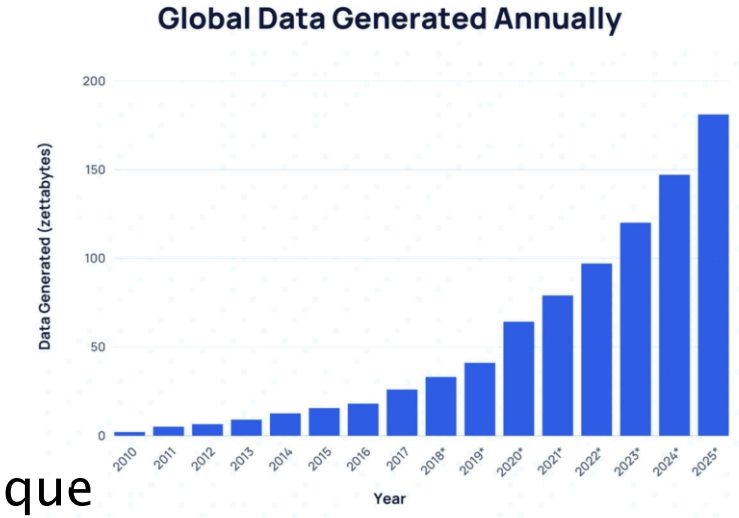
Compressão de imagem

André Damas Mora
FCT-UNL
2025-2026

NOVA SCHOOL OF SCIENCE & TECHNOLOGY

Introdução

- O que é a compressão?
  - O processo de encontrar representações mais eficientes (mais compactas) dos dados.
- Restrições impostas à compressão:
  - Sem perda de informação
  - Com perda de informação
- Estratégias para a compressão da informação:
  - Redução da redundância
  - Exploração das características da visão humana
- Qual a necessidade da compressão?
  - A produção de quantidades exorbitantes de informação leva a que a sua compressão seja essencial.
  - Aproximadamente 328,77 milhões de terabytes de dados são criados a cada dia
    - Um hospital típico (USA) gera cerca de 137 terabytes por dia
    - Duas horas de vídeo (HD 1080p @ 30fps) = 1.3 Terabytes ↔ 180Mb/segundo
      - MPEG2 consegue entre 4 e 10Mb/s atingindo uma compressão de cerca de 18-45x
      - MPEG4 consegue entre 10Kbs e 1Mb/s atingindo uma compressão entre 18000x a 180x



Número de cores numa imagem

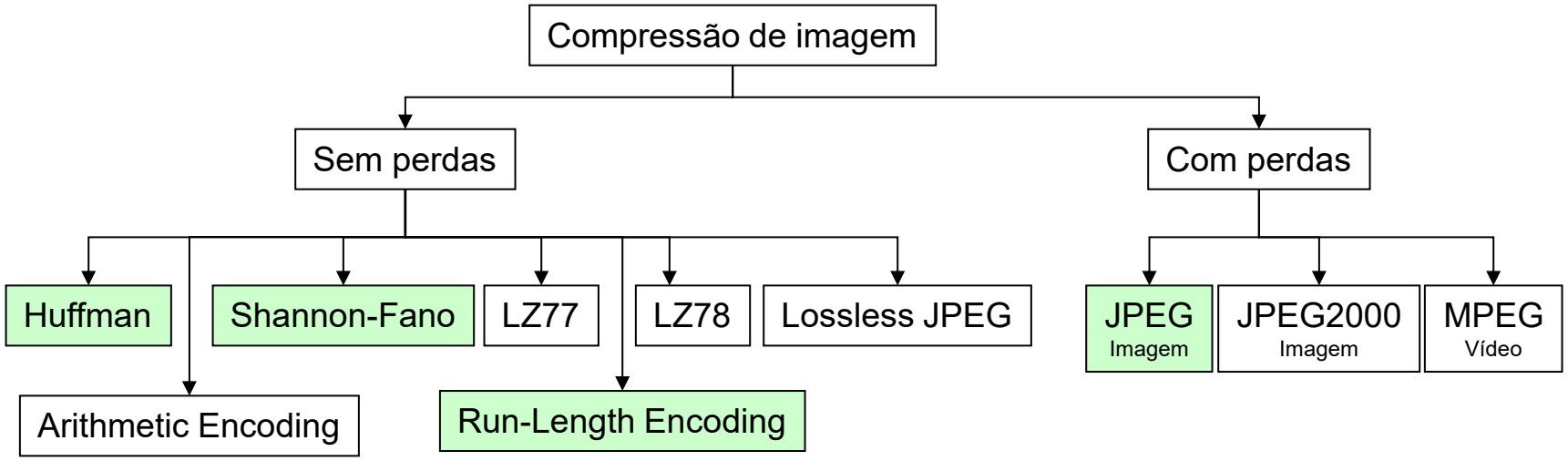
- O número de cores numa imagem é dado pelo número de bits utilizados para cada pixel.

- Resoluções comuns:
  - 1 bit por pixel – preto e branco puro
  - 8 bits por pixel – imagens em tons de cinzento ou em cores não realistas
  - 24 bits por pixel – imagens de "qualidade fotográfica" (3x 8bit)
  - 48 bits por pixel – imagens de qualidade "ultrahigh" (3x 16bit)

Formatos de ficheiros de imagem

Formato	Compressão	Sem perdas	Bits / pixel	Patenteado	Outros
BMP – Bitmap	✗	✓	24	✗	Imagem de 640x480 (0.3 MPixeis) com 24 bits/pixel ocupa: 640*480*24/8 = 921 600 bytes
GIF – Graphics Interchange Format	✓	✗ (+256 cores)	8 (Palete de 256 cores)	✓	GIFs animados
PNG – Portable Network Graphics	✓	✓	>1	✗	Imagens naturais e diagramas
TIFF – Tagged Image File Format	✓	✓	>1	✗	Alta qualidade
JPEG – Joint Photographic Experts Group	✓	✗	>1	✗	Imagens Naturais

Técnicas de compressão de imagem



Sem Perdas

A informação é totalmente recuperada na recepção. A compressão obtida é geralmente muito limitada dadas as características das imagens.

Com Perdas

A informação não é totalmente recuperada na recepção existindo perdas no processo.

A compressão obtida é geralmente boa pois tira-se partido das características do observador para "degradar" a imagem de forma não notória.

LZW (Lempel–Ziv–Welch)

- É uma evolução do LZ77
- Algoritmo
  - Inicializar o dicionário com os blocos de dimensão 1;
  - Procurar na imagem o maior bloco que já apareceu no dicionário;
  - Codificar o bloco pelo índice no dicionário
  - Adicionar ao dicionário uma nova entrada contendo o bloco atual e o símbolo seguinte;
  - Voltar ao passo 2

Data: a b b a a b b a a b a b b a a a a b a a b b a

Dictionary			
Index	Entry	Index	Entry
0	a	7	
1	b	8	
2		9	
3		10	
4		11	
5		12	
6		13	

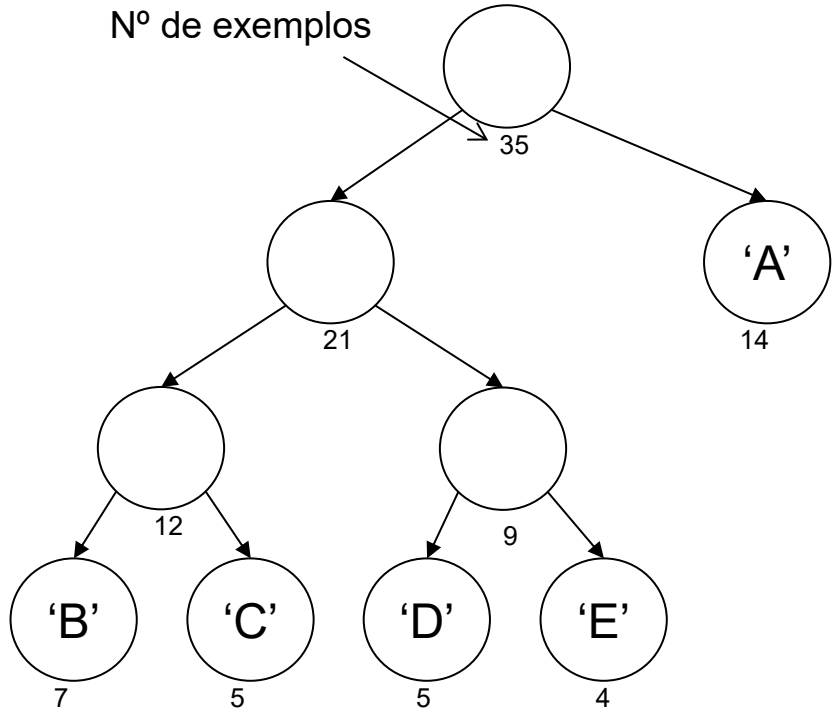
http://www.data-compression.com/lempelziv.shtml

Codificação de Huffman – Introdução

- A ideia de codificar letras com um diferente número de símbolos de acordo com a sua probabilidade de ocorrência foi inicialmente adoptada por **Morse** no código com o seu nome.
  - as letras mais frequentes como o e(·) e o a(· –) e as letras menos frequentes como o q(– – · –) e o j(· – – –) têm diferentes comprimentos para minimizar o tempo de transmissão.
- **Shannon and Fano**, em 1950, desenvolveram um código de compressão. Em 1952, **Huffman** publicou um artigo com uma optimização do algoritmo anterior que se viria a tornar mais popular.
- Os códigos de compressão de comprimento variável, tais como os códigos de **Huffman** e de **Shannon-Fano**, possuem as seguintes características:
  - Os códigos dos símbolos mais frequentes são mais curtos
  - Propriedade de prefixo – nenhum código é prefixo de outro. Essencial para que não haja ambiguidade na fase de descodificação

Codificação de Huffman – O conceito

- Para obter o código de Huffman, o primeiro é criar uma "Árvore de Huffman" tal como a apresentada.
- Para formar os códigos a partir da árvore de Huffman começa-se pela raiz e acrescenta-se um "1" sempre que se circula pela esquerda e um "0" sempre que se circula pela direita.
- Neste exemplo o código para o caracter 'A' será "0" enquanto que para o caracter 'D' será "101".



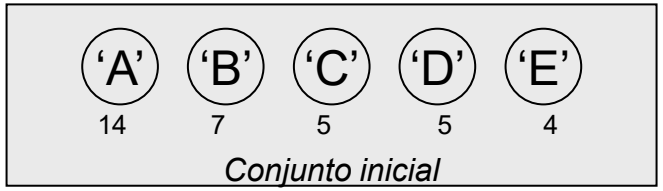
Codificação de Huffman – O algoritmo

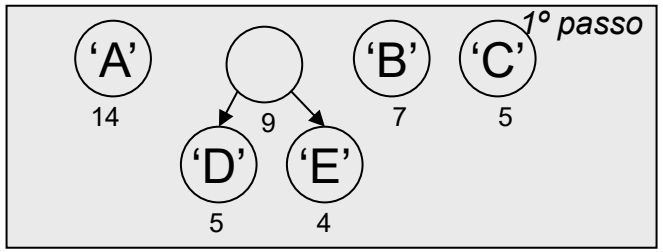
1. Contar o número de ocorrências de cada símbolo a codificar.
2. Criar um conjunto de N (em que N é o número de símbolos distintos existentes) árvores de nó único a que é atribuído um peso correspondente ao número de ocorrências do símbolo respetivo
3. Deste conjunto selecionar duas das árvores com menor peso e retirá-las do conjunto. Combinar estas árvores numa só árvore à qual se atribui um peso igual à soma dos pesos das duas que se juntaram. Juntar a nova árvore ao conjunto de árvores.
4. Repetir este processo até que reste apenas uma árvore.

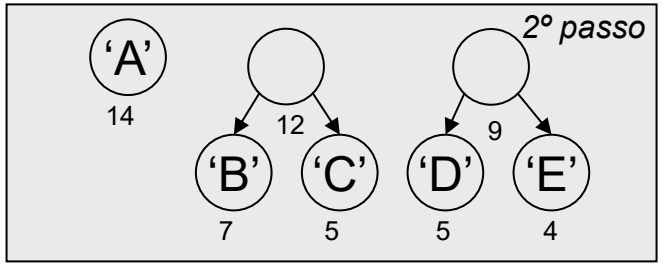
- Nota: Se em qualquer momento existirem mais de duas árvores por onde escolher a escolha é efectuada de forma aleatória.

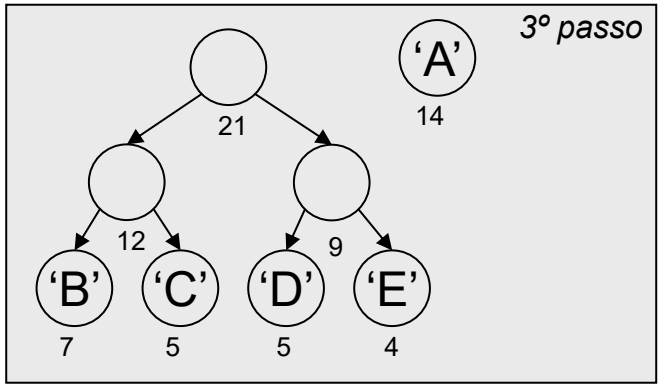
A árvore resultante da aplicação deste algoritmo é uma "Árvore de Huffman".

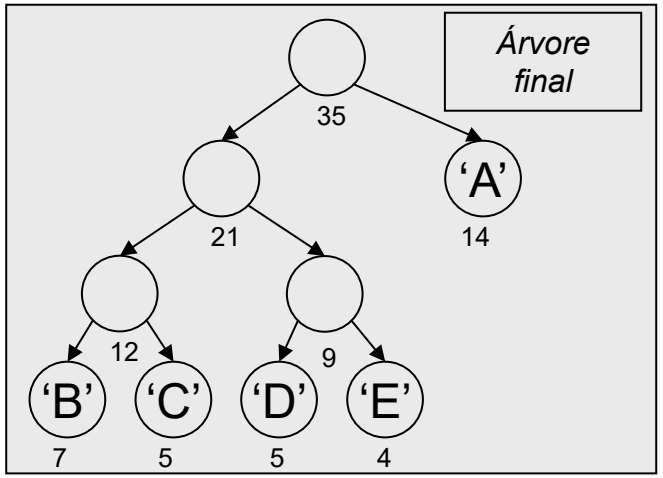
Codificação de Huffman – Exemplo









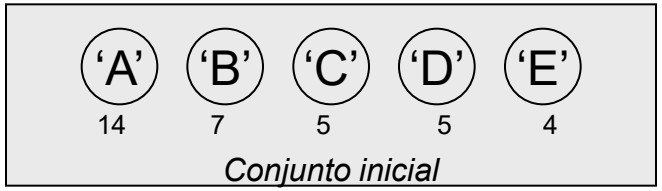


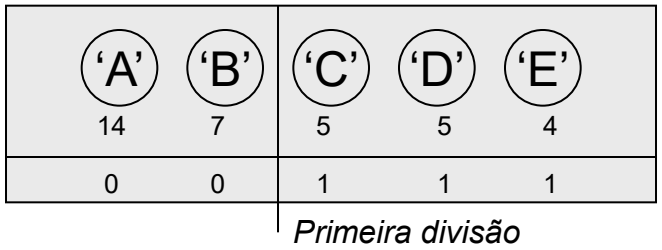
A – 0
B – 111
C – 110
D – 101
E – 100

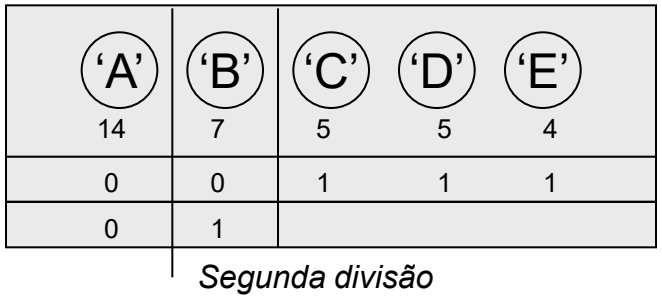
Codificação de Shannon-Fano – O algoritmo

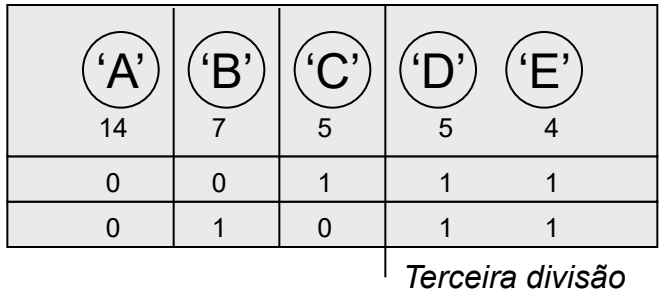
1. Ordenar os símbolos por ordem descendente de probabilidade de ocorrência.
2. Dividir a lista em duas partes em que o somatório das probabilidades de cada parte seja tão parecido entre si quanto possível.
3. Atribuir o bit ‘0’ à parte superior e o bit ‘1’ à parte inferior.
4. Repetir o processo para cada uma das partes até que apenas sobre um símbolo em cada delas.

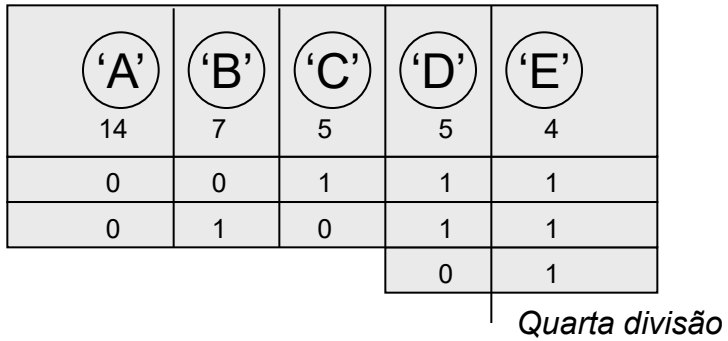
Codificação de Shannon-Fano – Exemplo











A – 00
B – 01
C – 10
D – 110
E – 111

Comparação Shannon-Fano/Huffman

		ASCII		Shannon-Fano		Huffman	
Símbolo	**Número de ocorrências**	**Comprimento dos símbolos**	**Ocupação total**	**Comprimento dos símbolos**	**Ocupação total**	**Comprimento dos símbolos**	**Ocupação total**
A	14	8	112	2	28	1	14
B	7	8	56	2	14	3	21
C	5	8	40	2	10	3	15
D	5	8	40	3	15	3	15
E	4	8	32	3	12	3	12
		TOTAL	**280**	TOTAL	**79**	TOTAL	**77**

Em geral, as codificações Shannon-Fano e Huffman são similares em dimensão.

No entanto, a codificação de Huffman possui uma eficiência pelo menos tão boa como a obtida pelo método de Shannon-Fano, pelo que se tornou um método preferido.

A Tabela de códigos (code book)

- A codificação pode funcionar de duas formas:
  - O **descompressor pode inferir** qual o code book usado pelo compressor a partir de um contexto anterior;
  - O **compressor** tem sempre **de comunicar** ao descompressor o code book. No entanto, este pode ter um tamanho significativo!

Codificação do Codebook

A – 0
B – 100
C – 101
D – 110
E – 111

- Solução Naive: Para cada símbolo é enviado o símbolo, o comprimento do código e o código

  (A,1,0) (B,3,100) (C,3,101) (D,3,110) (E,3,111)

- Se os símbolos forem ordenados, não é necessário enviá-los (códigos não utilizados podem ter comprimento 0)

  (1,0) (3,100) (3,101) (3,110) (3,111)

- Se for usada a forma canónica para os códigos apenas é preciso enviar o comprimento

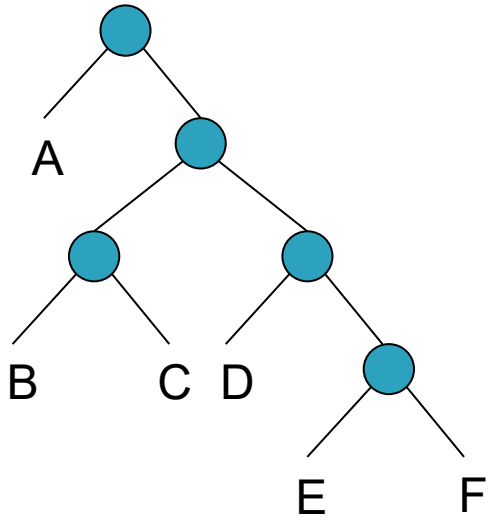
  (1) (3) (3) (3) (3)

Codificação canónica de Huffman

- Determinar a árvore de Huffman
- Baseada nos códigos de Huffman previamente calculados obtenha a versão canónica fazendo:
  - Ordene os símbolos a codificar por comprimento do código (use o valor do símbolo para desempatar).
  - Cada um dos códigos existentes é trocado por um novo com o mesmo comprimento usando o seguinte algoritmo:
    - O primeiro símbolo recebe um código com o mesmo comprimento que o original contendo apenas zeros.
    - A cada símbolo subsequente é atribuído o próximo valor binário em sequência, garantindo que são sempre maiores em valor.
    - Quando é atingido um código maior, após incrementar o valor são acrescentados 0's até perfazer o comprimento do novo código

Codificação canónica de Huffman – exemplo

1 - Atribuir ao primeiro símbolo um código com tantos 0's como o seu comprimento
2 - Para o novo símbolo incrementar o código anterior e se for de comprimento superior preencher com 0's
3 - Repetir o último passo até que todos os símbolos estão codificados



A – 1
B – 011
C – 010
D – 001
E – 0001
F – 0000

Não canónica

A – 0
B – 100
C – 101
D – 110
E – 1110
F – 1111

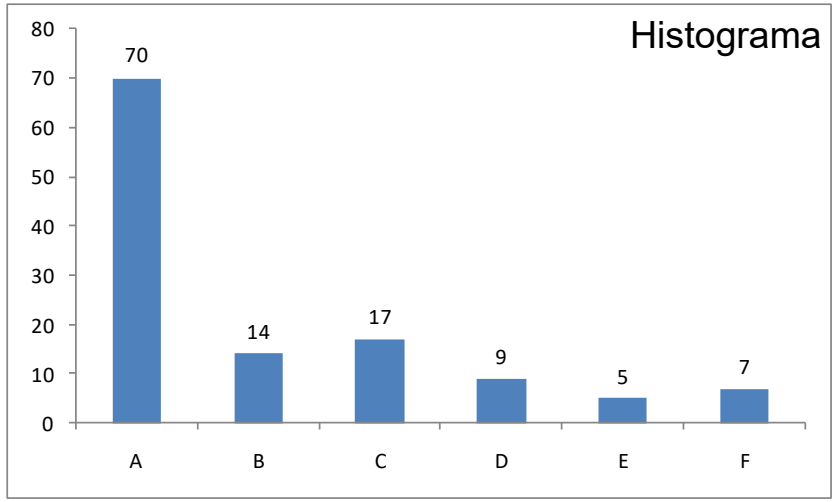
Versão Canónica

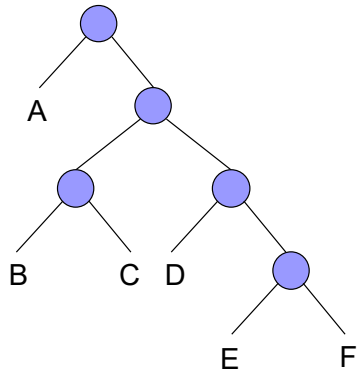
Apenas esta informação tem de ser enviada:
1,3,3,3,4,4 + símbolos (ABCDEF)

Codificação de Huffman

Codificar o texto:

A A C C A A A B A C D A C A C A A A C B A A C E A B D C A A A A B D B E A C B A
A E A A A A A C D A D C A C A A A A B A A A D A F A A D A A B F A A A B A A F C B
F D A B A D A F A A B E E A A A A A A A A A A A A F A C A A F A C A A A A A B C C B





Non canonical codes	Canonical codes
A – 1	A – 0
B – 011	B – 100
C – 010	C – 101
D – 001	D – 110
E – 0001	E – 1110
F – 0000	F – 1111

Code book: 1,3,3,3,4,4

Informação transmitida (com codebook)

00000110|00000001|00000011|00000011|00000011|00000100|00000100
001011010001000101111001010101000101110000…

Número de símbolos **Dimensão dos Símbolos** **Dados**

Huffman – Descodificação

Informação transmitida (com codebook)

```
00000110000000010000001100000011000000110000010000000100
00101101000100010111001010101000101101000...
```

De 00000110 sabemos que existem 6 símbolos

De 000000010000001100000011000000110000010000000100 sabemos que os símbolos são de dimensão:1, 3, 3, 3, 4, 4

Com esta informação retiramos o code book:

A – 0, B -100, C – 101, D – 110, E – 1110, F – 1111

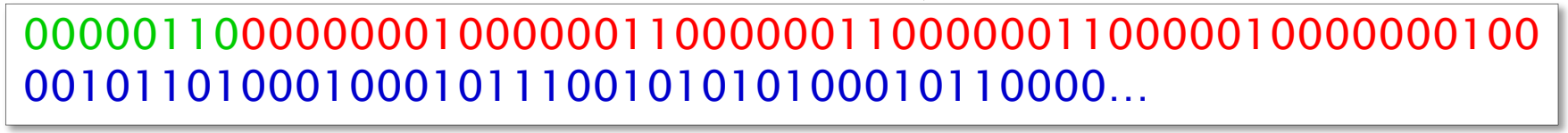
Agora Descodificando a mensagem:

0	0	1 0 1	1 0 1	0 0 0	1 0 0 0	1 0 1 1 1 0 0	1 0 1 0	1 0 1 0	0 0 0	1 0 1 1 0 0 0 0	0 0
A	A	C	C	A	A	A	B	A	C	D	A	C	A	C	A	A	A	C	B	A	A

```
A A C C A A A B A C D A C A C A A A C B A A C E A B D C A A A A B D B E A C B A A
E A A A A A C D A D C A C A A A A B A A A D A F A A D A A B F A A A B A A F C B F
D A B A D A F A A B E E A A A A A A A A A A A A F A C A A F A C A A A A A B C C B
```

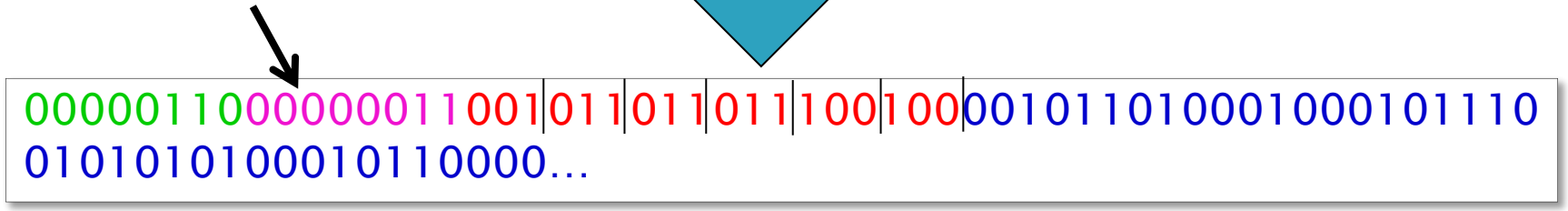
Codificação de Huffman – Optimização

Estão a ser usados 8 bits
para representar números até 4



Número de símbolos **Dimensão dos Símbolos** **Dados**

Enviamos o número de bits com que vamos enviar o code book e reduzimos o code book ao essencial



Número de símbolos **Bits usados para representar a dimensão símbolo** **Dimensão dos Símbolos** **Dados**

Formas de melhorar a compressão

Como reduzir a entropia da imagem?

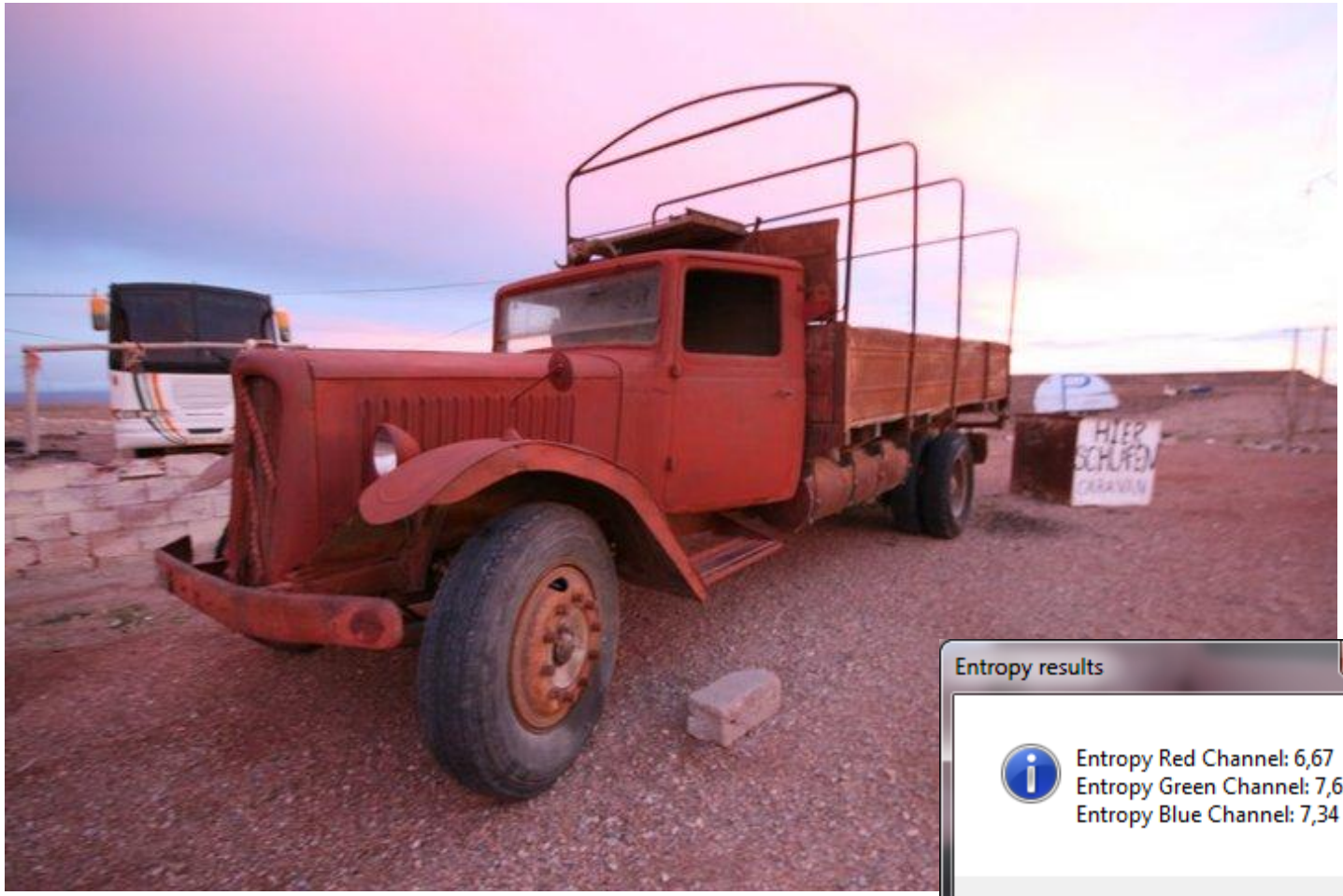
- O que é a entropia de uma imagem?
- Entropia de Shannon:

$$E = -\sum_{i=0}^{GrayLevels} p(i)\log_2\big(p(i)\big)$$

- O rácio de entropia de uma base de dados é número médio de bits por símbolo necessário para codificá-lo.

- O source coding theorem demonstra que é impossível de comprimir os dados de forma a que o número médio de bits por símbolo seja menor que a entropia de Shannon, sem que virtualmente nenhuma informação seja perdida.

Entropia exemplos

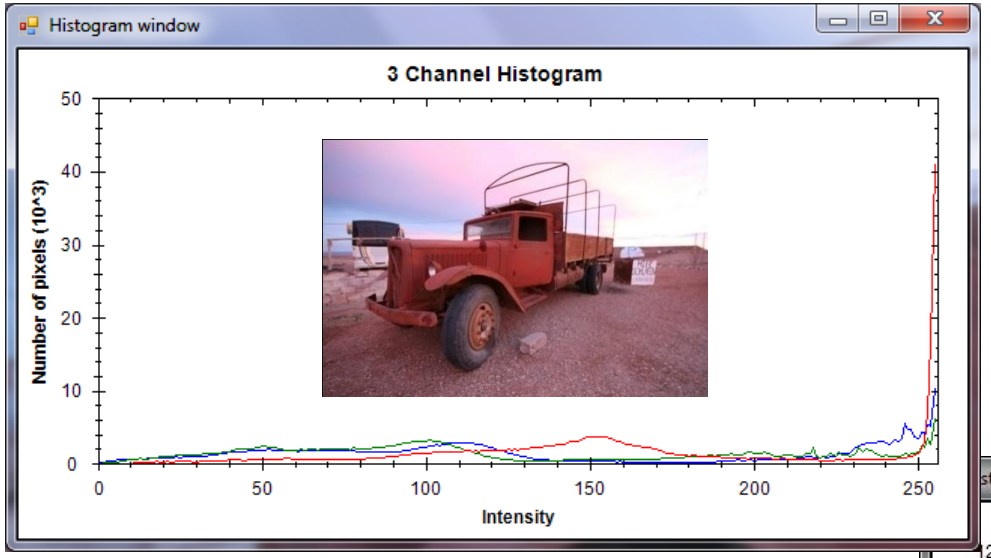


Redução de Entropia

Imagem obtida por diferenciação em x
Pict(x,y)=Pict(x,y)-Pict(x-1,y)

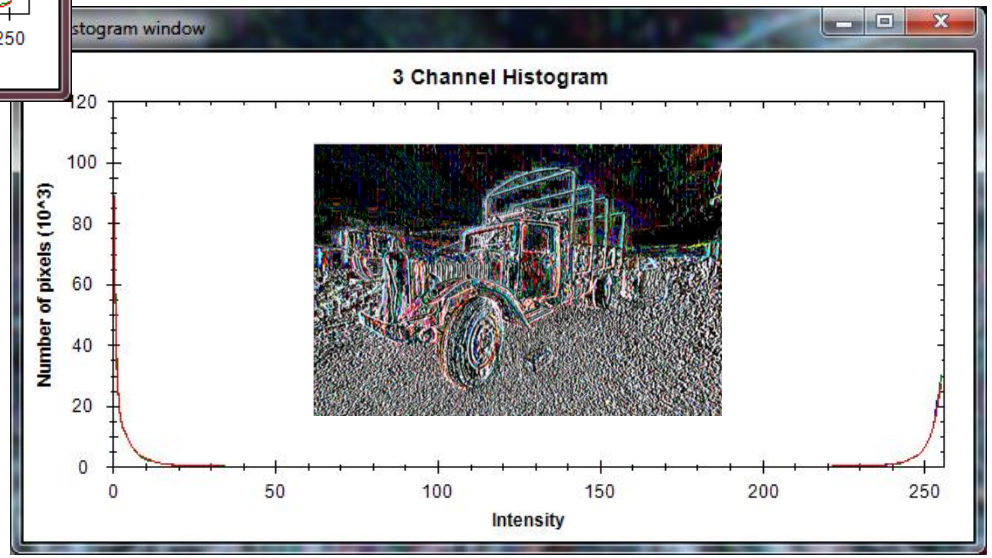
NÃO HÁ PERDA DE INFORMAÇÃO.
A IMAGEM É TOTALMENTE RECUPERÁVEL!

Histograma/entropia comparação

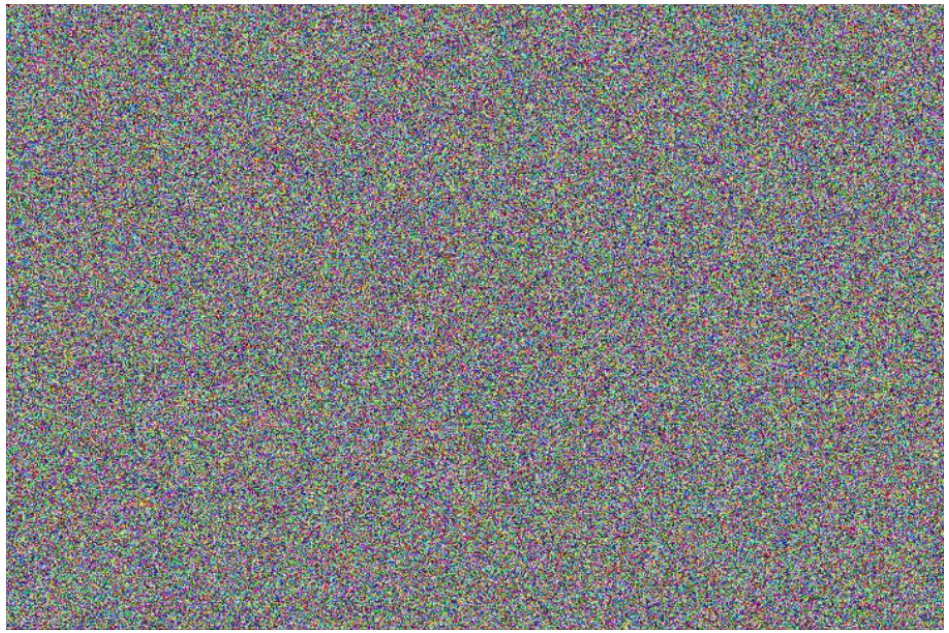


Entropy Red Channel: 6,67
Entropy Green Channel: 7,62
Entropy Blue Channel: 7,34

Entropy Red Channel: 3,84
Entropy Green Channel: 3,77
Entropy Blue Channel: 3,84

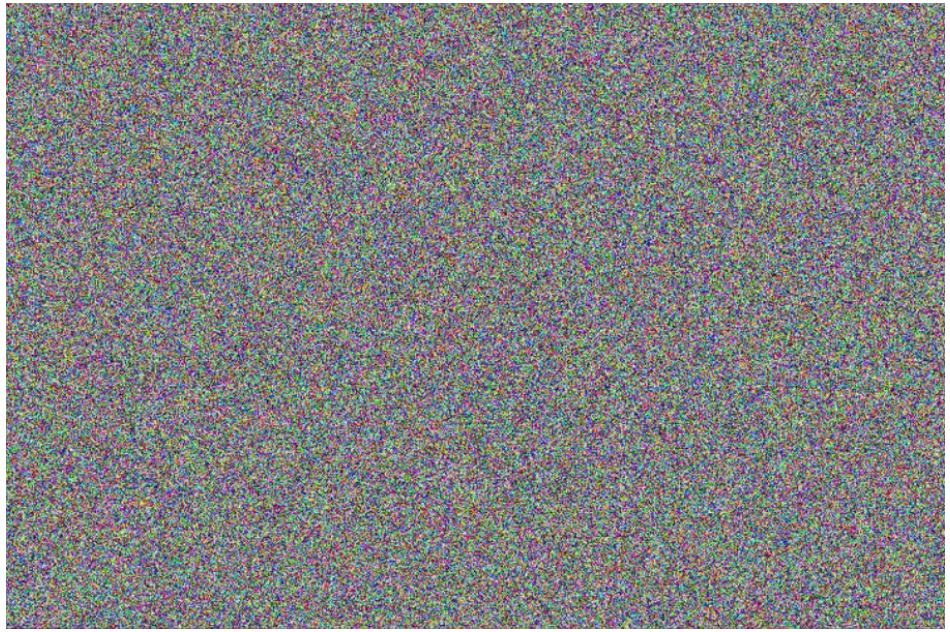


E o que acontece a uma imagem de ruido aleatório?



Entropy Red Channel: 7,95
Entropy Green Channel: 7,95
Entropy Blue Channel: 7,94

Diferenciação



Entropy Red Channel: 7,95
Entropy Green Channel: 7,95
Entropy Blue Channel: 7,95

Não existe vantagem…

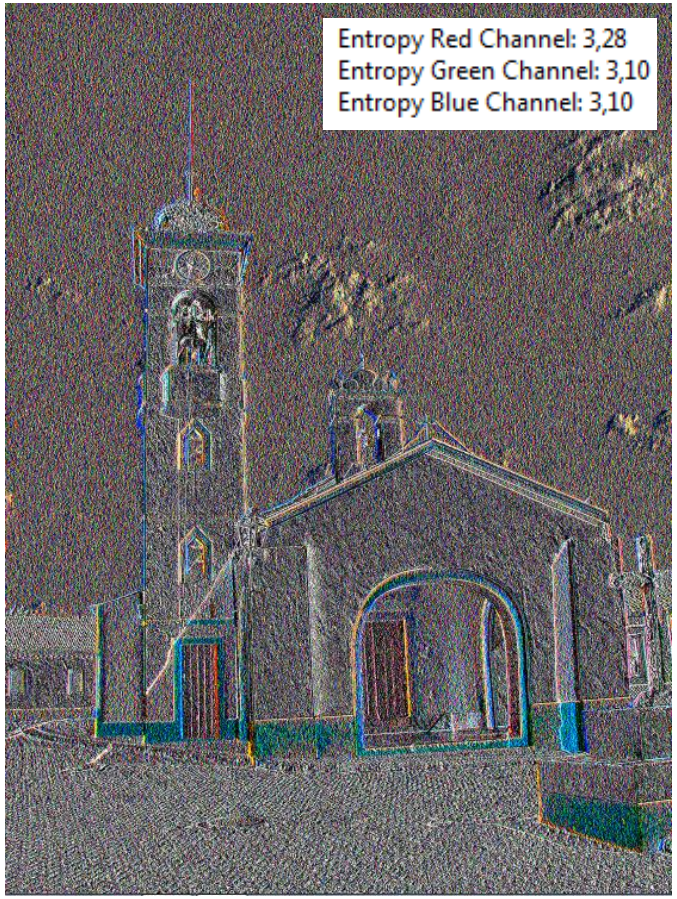
Ainda Entropia...

1704x2272 pixels image
TIFF format

Filtro de Média
de 5x5

Entropia após diferenciação

Based on original image



Based on 5x5 filtered image