Introdução ao Processamento Digital de Imagem (MC920) - Trabalho 2

Heitor Boschirolli - 169477

2018
April

1 Executando o código

O tar HeitorBoschirolli_169477.tar contém os códigos codificar.py e decodificar.py - responsáveis por codificar uma mensagem em uma imagem e decodificar uma mensagem de uma imagem, respectivamente -, quatro imagens para teste, um arquivo texto com uma mensagem para teste e este PDF.

O arquivo codificar.py exige 4 parâmetros para ser executado, o nome da imagem de entrada, o nome do arquivo com o texto de entrada, o plano de bits em que se deseja colocar a mensagem e o nome com que a imagem contendo a mensagem deve ser salva. Para executá-lo, é preciso digitar "python codificar.py" seguido dos parâmetros, na ordem em que foram especificados, e pressionar a tecla enter em um terminal linux com python 2.

O arquivo decodificar.py se comporta de forma similar. Ele exige 3 parâmetros para ser executado, o nome de uma imagem contendo uma mensagem codificada, o plano de bits no qual a mensagem está e o nome do arquivo em que a mensagem obtida a partir da imagem deve ser guardada. Para executá-lo, é preciso dititar "python decodificar.py" seguido dos parâmetros, na ordem em que foram especificados, e pressionar a tecla enter em um terminal linux com python 2.

Caso o número de parâmetros passados como entrada tanto para o arquivo codificar.py quanto para o decodificar.py esteja errado, será exibida uma mensagem indicando a forma correta de executar o programa e este se encerrará.

2 Solução

2.1 Codificação da mensagem

Existem várias formas de se codificar uma mensagem em uma imagem, para a adotada neste trabalho, é preciso converter a mensagem para sua representação binária usando o código ASCII como critério de conversão e, para cada pixel, colocar um bit no canal vermelho, outro no verde e outro no azul (nessa ordem). O primeiro pixel a receber uma parte da mensagem deve ser o do canto superior esquero da imagem e ela deve ser ocupada com os bits da mensagem linha a linha.

A conversão da mensagem para binário foi feita em duas etapas. Primeiramente os caracteres foram convertidos para seu equivalente inteiro de acordo com o código ASCII 8 bits com o uso da função ord(), essa função recebe como parâmetro uma string de tamanho um e devolve sua representação inteira de acordo com o código ASCII 8 bits [1]; em seguida os números inteiros representando os caracteres foram convertidos para sua representação binária com o uso da função format(), essa função recebe como parâmetro um dado e devolve esse dado com uma formatação específica, ao chamar a função como '0:08b'.format(), e passar um número inteiro como parâmetro, ela devolve uma string com a representação binária do número inteiro [1]. Conforme os caracteres eram convertidos para binário, eles iam sendo concatenados em uma string contendo a representação binária da mensagem.

Com a representação binária da mensagem em mãos, para colocá-la na imagem, percorreu-se-a linha por linha, começando no pixel superior esquerdo, e substituiu-se o bit da posição desejada por um dos bits da mensagem; a posição desejada é igual ao plano de bits passado como parâmetro para a função.

Para trocar o valor de um único bit de um número que representa a intensidade de um pixel de um canal da imagem, fez-se uma operação and com um número composto por uns e um zero na posição do bit que se desejava mudar (esse número foi obtido deslocando-se o número 1 para a esquerda até que ele estivesse na posição do bit que

deve ser trocado e negando o resultado), assim, esse bit era zerado. Para colocar o bit desejado na posição certa (que agora contém um zero), fez-se uma operação or do bit que desejava-se trocar com o bit que se queria colocar naquela posição, deslocado para a esquerda de forma a estar na mesma posição do bit que deve ser substituído.

Depois que toda mensagem é colocada na imagem, oito zeros consecutivos são adicionados; isso serve para indicar o decodificador que é preciso parar de ler a mensagem pois esta chegou ao fim.

2.2 Decodificação da mensagem

A decodificação da mensagem é feita de forma similar à codificação. Para decodificar a mensagem, a imagem é percorrida na ordem especificada pelo problema, os bits na posição do plano de bits (que é passado como parâmetro), são adicionados em uma string que, quando toda imagem for percorrida, irá conter a representação binária da mensagem.

Para extrair apenas o bit desejado de um número, realizou-se uma operação de shift right nele de forma que o bit desejado se tornasse o menos significativo, então o resto da divisão por dois passa a indicar o valor do bit desejado. A cada 8 bits da mensagem coletados, um caractere era obtido, primeiro se usava a função int() que recebe um número inteiro e uma base como parâmeto e devolve esse número na base dada [1], assim, o valor decimal do número binário era obtido e entrão, para encontrar o caractere representado pelo número inteiro, foi usada a função chr() que recebe como parâmetro um inteiro e devolve uma string de tamanho um com o caractere que é representado pelo inteiro passado no código ASCII de 8 bits [1].

Ao concatenar todos os caracteres obtidos dessa forma em uma string, a mensagem é reconstruída.

2.3 Exibição de algumas camadas de bits da imagem

Para visualizar a alteração feita na imagem pela inserção de uma mensagem, foram exibidos quatros planos de bits da imagem com a mensagem inserida; o plano zero (correspondente ao plano com os bits menos significativos), o plano um, o plano dois, e o plano sete (correspondente ao plano com os bits mais significativos).

Para extrair um plano de bits qualquer da imagem, foi feito um deslocamento dos valores da imagem para a direita de forma que o bit que se desejava extrair estava na posição do bit menos significativo; em seguida pegou-se o resto da divisão por dois.

3 Exemplo de entrada e saída

Para o caso em que a imagem de entrada é baboon.png (Figura 1), a mensagem a ser codificada é composta por 20.000 caracteres "A" consecutivos e o plano de bits é o plano zero, a imagem com a mensagem codificada é a out.png (Figura 2). O plano zero de bits pode ser visto na figura 3, é bem fácil perceber a região ocupada pela mensagem. Os planos um e dois de bits podem ser vistos nas figuras 4 e 5 respectivamente e o plano 7 pode ser visto na figura 6.

Além de ser possível identificar a mensagem codificada no plano zero, algumas propriedades interessantes podem ser percebidas. Como os planos zero, um e dois representam os bits menos significativos da imagem, eles possuem uma grande variação nos seus pixels, pois qualquer diferença de intensidade na imagem original resulta em uma grande diferença nos planos dos bits menos significativos. O sétimo plano de bits, por outro lado, é bem similar a imagem original; isso se deve ao fato de que, para que exista uma diferença entre dois pixels no plano 7, é preciso que exista uma diferença grande na imagem (pois o plano sete é o mais significativo).

Figure 1: imagem baboon.png

Figure 2: out.png: imagem da Figura 1 com a mensagem codificada

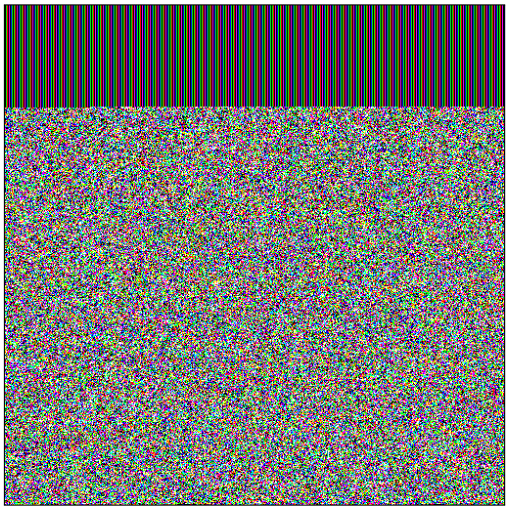

Figure 3: Plano de bits zero da imagem da Figura 2

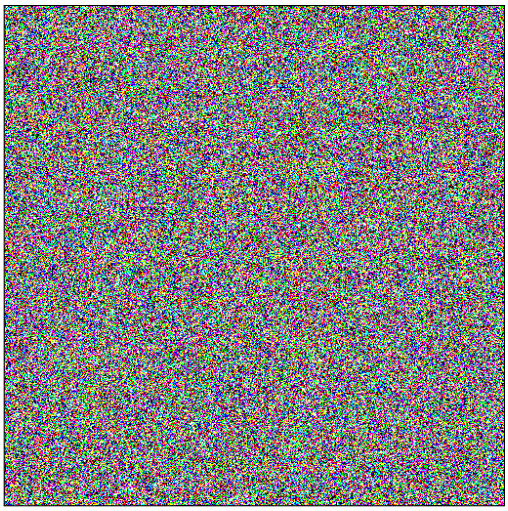

Figure 4: Plano de bits um da imagem da Figura 2

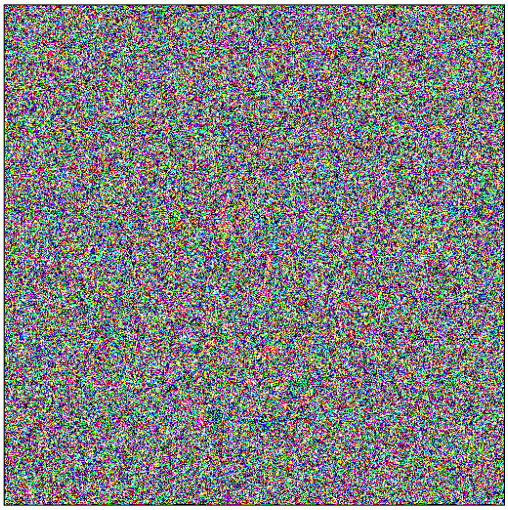

Figure 5: Plano de bits dois da imagem da Figura 2

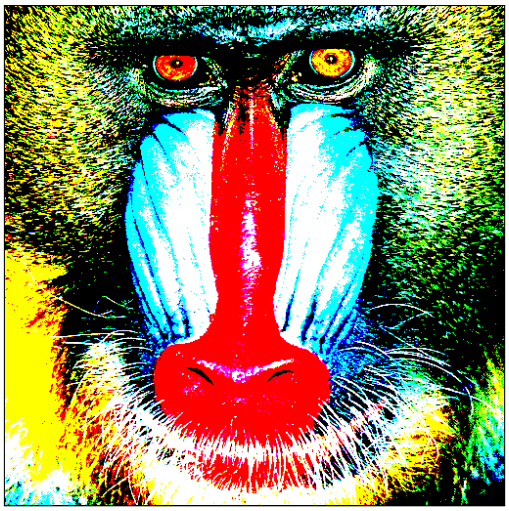

Figure 6: Plano de bits sete da imagem da Figura 2

3.1 Limitações

Para a conversão da mensagem para binário, usa-se o código ASCII de 8 bits, isso restringe o número de caracteres que podem estar presentes na mensagem que se deseja codificar. Se a mensagem contém algum caractere com acento, por exemplo, o programa codificaria a mensagem na imagem até encontrar o primeiro acento, quando isso ocorresse, o programa se encerra indicando que um erro ocorreu na conversão de um caractere para inteiro. Para a troca de mensagens usando a língua portuguesa, isso é uma limitação bastante séria, se as mensagens forem escritas na lingua inglesa, porém, raramente um caractere não presente no código ASCII estará na mensagem, tornando a limitação não muito séria neste caso.

References

[1] The Python Standard Library Documentation - Built-in Functions, `https://docs.python.org/2/library/functions.html#chr`