

Esteganografia

Introdução ao Processamento de Imagem Digital

Percy Maldonado Quispe
p203968@dac.unicamp.br

1 Introdução

Uma técnica conhecida para ocultar mensagens dentro de uma imagem é a esteganografia, que consiste em alterar os bits de cada pixel, seja uma imagem colorida ou não para logo ocultar a mensagem em código binário. Para saber onde esconder o valor do texto verificou-se o plano de bits (8) 1 de uma imagem, onde o bit mais alto ou o mais significativo reflete mais a imagem real, sendo os bits menos representativos os menos visíveis.

Por isso, normalmente os bits alterados para esconder a mensagem são os menos significativos. O propósito do projeto é implementar a técnica de esteganografia em imagens coloridas, alterando um bit de cada banda de cor de cada pixel da imagem para esconder uma mensagem.

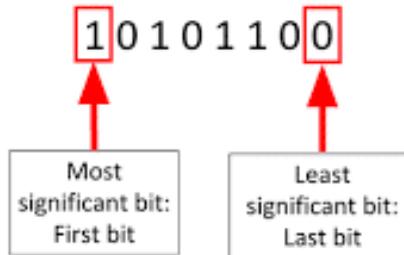


Figure 1: MSB and LSB

2 Características do programa

O programa foi feito na linguagem Python, usando as bibliotecas NumPy e OpenCV (CV2). Os argumentos são passados na execução via argv.



Figure 2: Numpy, Python, OpenCV

Dois programas foram implementados: um para codificar uma mensagem em uma imagem, e outro para decodificar uma mensagem e uma imagem.

Os argumentos do programa de codificação *encode.py* podem ser:

- *in_image*: Arquivo que contém a imagem a ser codificada.
- *in_text*: Nome do arquivo de texto contendo a mensagem.
- *out_image*: Nome para a imagem de saída, codificada.
- *bit_plane*: Plano de bits a ser alterado [0,7].

Os argumentos do programa de decodificação *decode.py*:

- *input_image*: Arquivo que contém a imagem a ser codificada.
- *output_text*: Nome do arquivo de texto de saída, para conter a mensagem decodificada.
- *bit_plane*: Plano de bits que contém a mensagem [0,7].

3 Implementação

Como hemos visto en la sección anterior, se implementaron dos programas para la codificación y decodificación. Na codificação, após a leitura do arquivo-texto, é feita a transformação do texto em uma sequência de códigos ASCII. Isso é feito pela função *frombuffer* da biblioteca NumPy para separação de caracteres, combinado com a função *view* para transformação em inteiros. Esse vetor de inteiros é transformado em vetor de bits (0 ou 1) pela função *unpackbits*. Junto com essa transformação, é adicionado um código que indica final de mensagem, o código 0 do ASCII.

Com a sequência de bits a serem adicionadas à imagem, é feita a codificação a partir da representação unidimensional (flattened) da imagem. São separados os primeiros n pixels, onde n é número de bits da sequência, e a codificação é feita por meio de operações com bits e máscaras. As máscaras dependem do plano de bits a ser alterado, deslocando um bit 1 pelo número do plano (*left_shift*), e invertendo (*invert*) para a máscara correspondente ao bit 0. (Outra ajuda para entender isso, se eu quiser definir um dado bit em uma posição de bit particular, eu posso definir 1 usando $x = x | pos$ ou $x = x \& \neg pos$ para definir 0). Se o elemento da sequência é 1, utiliza a primeira máscara. Senão, utiliza a máscara invertida.

O processo de decodificação começa com a extração do bit no plano onde se salvou a mensagem, para isto primeiro se obtém o valor binário desse plano dos bits da imagem, usando a operação inversa à codificação (*right_shift*) para depois fazer uma operação e com os valores da imagem. Para transformar em códigos ASCII de novo, basta utilizar un *reshape* ou a função *packbits* após separar em conjuntos de 8 bits, e então multiplicar com os valores binários em decimal [128, 64, 32, 16, 8, 4, 2, 1] para obter o valor ASCII dos bits, alem de isso também se valida o caracter de fim de texto que se pôs para ver onde termina a mensagem.

4 Resultados

Foram obtidos resultados para 4 imagens coloridas (https://www.ic.unicamp.br/~helio/imagens_coloridas/). As imagens originais podem ser vistas na figura 3.

Para os testes, foi feito com vários textos de diferentes tamanhos (Curtos, Longos, Livros, Código)

4.1 Textos curtos

Quanto à codificação em textos curtos ou pequenos, a mudança na imagem é praticamente imperceptível, isto devido ao pequeno número de bytes utilizados ao realizar a codificação. Assim mesmo, a decodificação foi totalmente um exito, recuperando 100 % da informação guardada sem perda alguma.

- Texto: “O objetivo deste trabalho é implementar um algoritmo de esteganografia em imagens digitais.”, no plano 0.

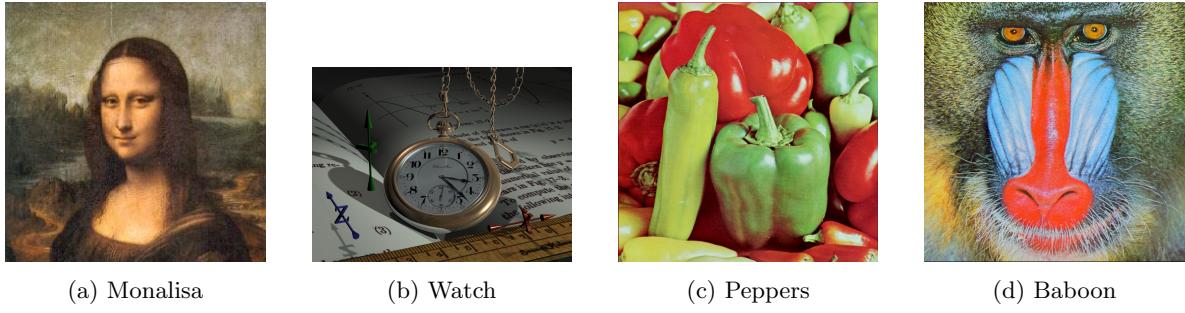


Figure 3: Imagens coloridas

Também se observou na figura 4 que o plano de bits da imagem codificada não mostra alterações visíveis para a pessoa humana, mas pode ser vista com análise mais minuciosa. Para percebermos a presença de um texto na imagem, precisamos testar textos mais longos.

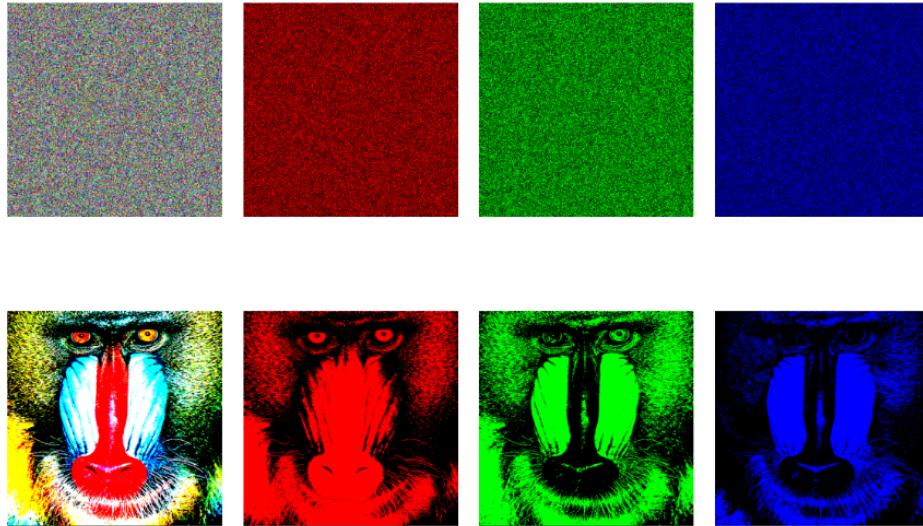


Figure 4: Bandas $[0,7]$, $show_bit_planes(encoded_image, [0, 7])$

4.2 Textos Longos

Com a certeza do funcionamento do programa (seção anterior), foram codificados textos mais longos nas maiores imagens. Na figura 5 foi codificado o 400 primeiros números de Euler no plano 0.

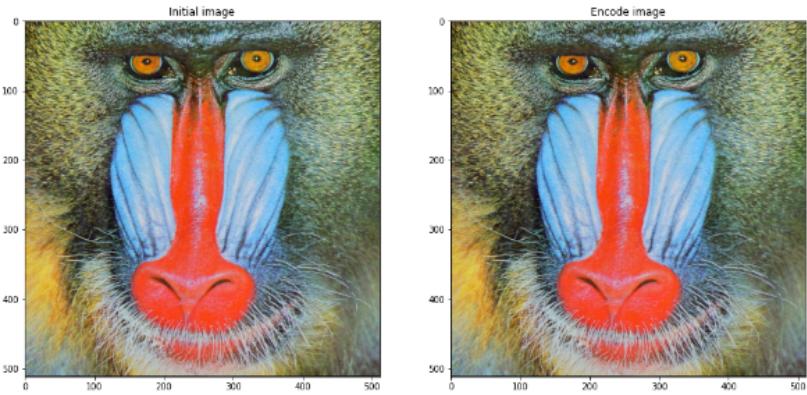


Figure 5: Banda 2

Onde a figura 6 mostra que o plano 0 tem a mensagem codificada, que pode ser percebida ao olho humano.

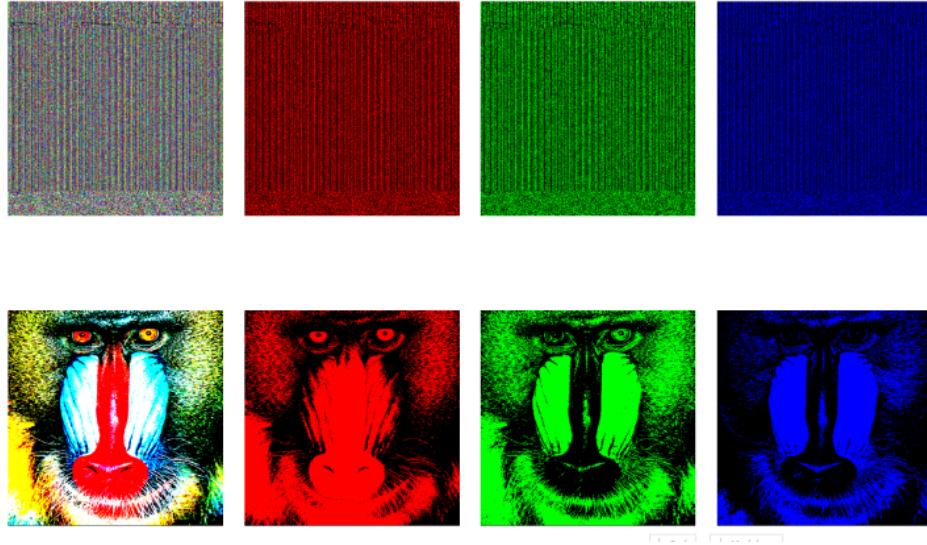


Figure 6: Bandas [0,7], *show_bit_planes(encoded_iimage, [0, 7])*

Também, o mesmo texto foi codificado no segundo plano de bits da imagem, e os resultados podem ser vistos na figura 7. Mesmo com um texto maior, observando a figura não é possível perceber a presença do texto, mesmo estando em um plano de bits mais significativo. Isso mostra o quanto eficaz é o método de ocultar texto na imagem.

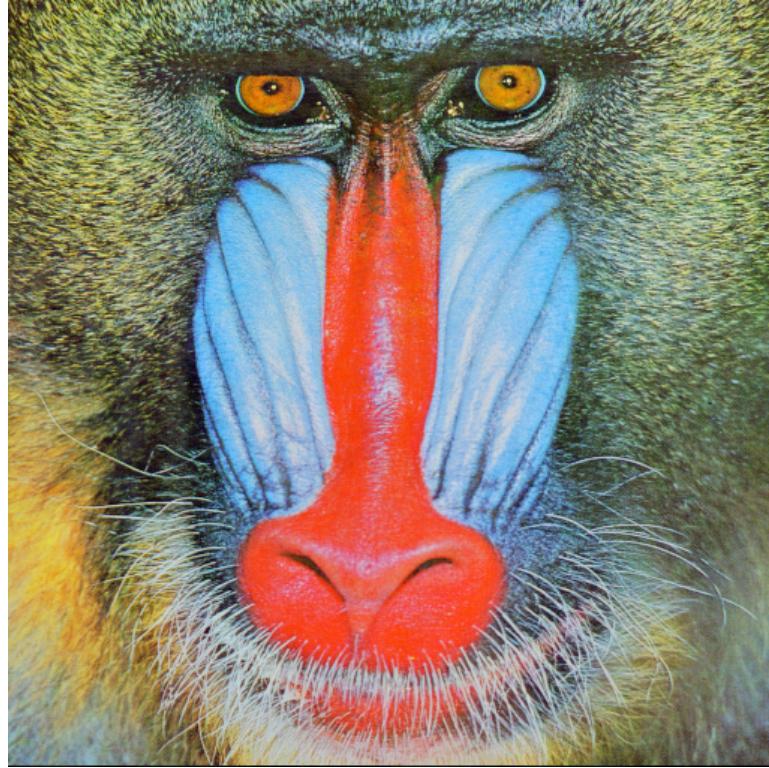


Figure 7: Banda 2

A imagem 8 também mostra que não se tem informação de cor nos bits menos representativos, já que não se mostra a figura da imagem, como se o faz a banda 8.

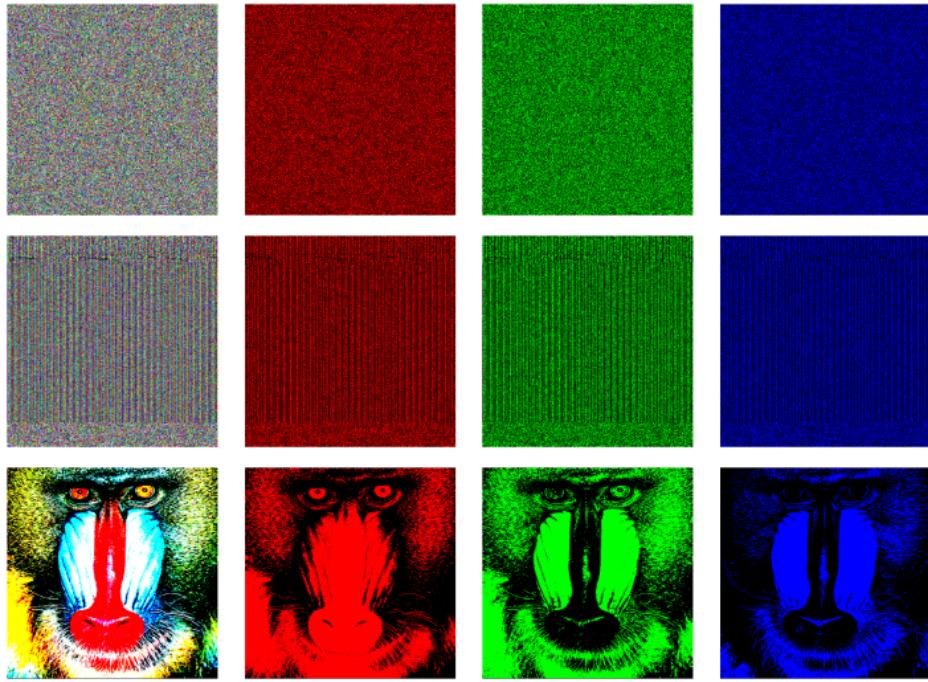


Figure 8: Bandas $[0,1,7]$, $show_bit_planes(encoded_image, [0, 1, 7])$

No entanto, é possível perceber a presença da mensagem olhando os planos de bits de cada cor referentes ao bit 2, onde a mensagem foi inserida. Nesses planos, é perceptível que um padrão diferente surge na parte superior da imagem, referente ao texto. Ou seja, isto demonstra que, em caso interceptado, é possível descobrir facilmente se uma imagem contém texto oculto ou não, e em que plano de bits se situa, apenas analisando os planos de bits.

Também, foi testado com os primeiros 700 números de Euler, mas em uma imagem maior (*watch.png*). Neste caso, é perceptível na imagem, mesmo em planos menos significativos. Isto se deve a um caráter gráfico da imagem, que tem detalhes nos bits menos significativos, pelo que este plano é representativo dos detalhes na imagem final. Ao alterá-lo, é possível ver o resultado na imagem final.



Figure 9: Encoded imagem - banda 2

Isto é mais perceptível nas partes escuras da imagem. Ao observar os planos de bits, verificou-se que representam diversos detalhes da imagem, ao contrário da imagem do Baboon, que tem algum detalhe mas não tanto para dar lugar a grandes mudanças na imagem se mudar.

Isto se deve à ausência de detalhes da cor que representa na imagem. Então, uma estratégia

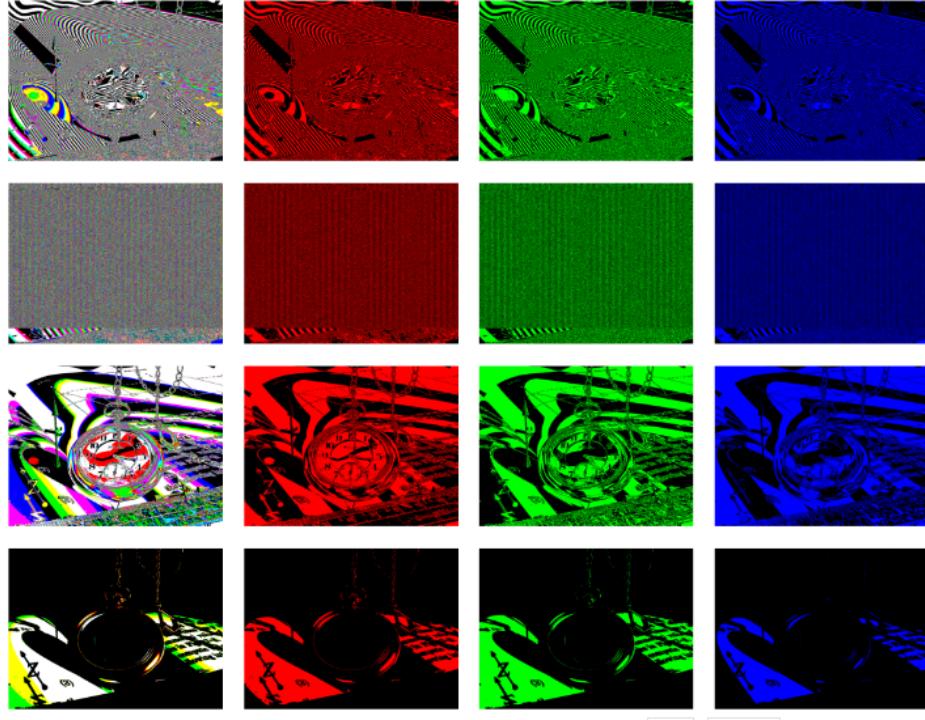


Figure 10: Watch, Bandas [0,1,7], *show_bit_planes(encoded,image,[0, 1, 7])*

para melhor resultado seria detectar, a priori ou em tempo de execução, alguma cor que não é muito utilizada, e colocar o texto só na banda correspondente a ela.

Em todos esses testes, o espaço ocupado pelo texto na imagem também é percebido, e ainda há espaço após a inserção de um texto, que não é um texto pequeno.

5 Conclusão

Os testes realizados foram bem sucedidos em ocultar mensagens em imagens, e em seguida recuperá-las sem perdas. Conclui-se, por conseguinte, que se trata de uma técnica de codificação viável. No entanto, muitos dos modos atuais de envio de imagens envolvem compressão com perdas, e nesses casos a mensagem é provavelmente perdida. Ainda que seja difícil perceber se há uma mensagem oculta na imagem só com a imagem completa, ao separar os planos de bits é fácil perceber se há uma mensagem, e em qual deles está, pois a mudança é visível, principalmente em textos mais longos. Se a imagem tiver muitos detalhes em planos de bits menos significativos, o resultado também pode ser visto na imagem codificada completa. Quanto maiores forem as dimensões da imagem e menor o texto, mais difícil será a visualização por planos de bits.