

Trabalho 4

Tiago Loureiro Chaves (187690)

MC920A - Introdução ao Processamento de Imagem Digital - 2s2019

Resumo

Este projeto teve como objetivo a implementação de um algoritmo de esteganografia para a codificação e decodificação de mensagens em imagens coloridas.

Para isso, os caracteres ASCII da mensagem são convertidos para sua representação binária e então armazena-se cada bit em um plano da imagem, alternando-se entre os canais RGB. Deste modo, cada pixel da imagem original oculta 3 bits de informação, assim, uma imagem com dimensões $M \times N$ pode guardar até $\lfloor \frac{3MN}{8} \rfloor$ caracteres.

Nota-se que imagens com o texto oculto nos dois planos de bits menos significativos são praticamente indistinguíveis das originais, como apresentado na Seção 3.

1 Problema

A esteganografia — do grego, “escrita escondida” — consiste primariamente em ocultar-se a existência de uma mensagem, ao contrário da criptografia, onde o que se esconde é o significado da mensagem.

Neste trabalho, estamos particularmente interessados no armazenamento de arquivos de texto em imagens coloridas, alterando os três canais de cores em um dos 8 planos de bit para embutir o código ASCII binário dos caracteres.

Ao usarmos o plano de bits 0 esse método pode ser classificado como parte da técnica LSB (*Least Significant Bit*), a qual, apesar de ser uma das mais usadas e simples de implementação, é imperceptível à visão humana. Por outro lado, imagens codificadas de tal forma são facilmente identificadas através de análise estatística (apesar da decodificação da mensagem não ser uma tarefa trivial) [1].

O uso de planos de bits mais significativos, entretanto, gera artefatos característicos nas imagens, como veremos na Seção 3.

2 Programa

2.1 Dependências e Organização

O trabalho foi desenvolvido em Python 3.7.4, com as bibliotecas Numpy 1.17.0, Matplotlib 3.1.1 e OpenCV 4.1.0, e é dividido em três arquivos:

- `encode.py` – programa que oculta mensagem de texto na imagem
- `decode.py` – programa que recupera mensagem de texto da imagem
- `utils.py` – funções gerais para processamento de imagens

As imagens utilizadas para teste estão na pasta `i/`, e exemplos de saída encontram-se na pasta `o/` (alguns arquivos texto usados como mensagens estão na pasta `m/`). O código também pode ser visto em github.com/laurelkeys/image-processing.

Para facilitar a visualização isolada dos planos de bit das imagens com texto armazenado, criou-se um Jupyter Notebook ao invés de um script: `bit_planes.ipynb`.

2.2 Execução

Os scripts de codificação e decodificação podem ser executados como:

```
$ python encode.py in_image.png in_message.txt bit_plane out_image.png
```

```
$ python decode.py out_image.png bit_plane out_message.txt
```

Sendo os parâmetros:

- `in_image.png`: imagem em que será embutida a mensagem.
- `out_image.png`: imagem com a mensagem embutida.
- `in_message.txt`: arquivo-texto contendo mensagem a ser oculta.
- `out_message.txt`: arquivo-texto contendo mensagem recuperada.
- `bit_plane`: planos de bits em que a mensagem será embutida (de 0 a 7)¹.

Além disso, caso deseje-se exibir a mensagem que será codificada/decodificada, pode-se utilizar as opções de verbosidade (p.e. `-v`), que podem ser vistas executando-se os scripts com a flag `--help`.

O Notebook para separação dos planos de bits e canais RGB das imagens é executado com `$ jupyter notebook bit_planes.ipynb` (estando na mesma pasta do arquivo).

3 Resultados

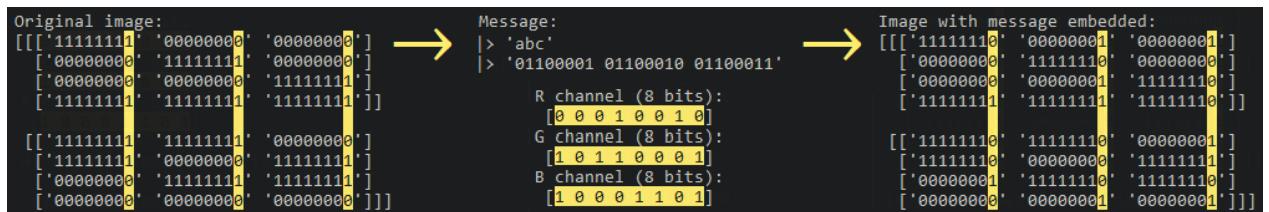


Figura 1: Processo de esteganografia em uma imagem 4×2 , no plano de bits 0²

3.1 Codificação

Ao codificarmos uma mensagem, ignoramos o final do texto caso ele não caiba na imagem, assim como todos os caracteres não ASCII.

Caso a mensagem ocupe menos bytes do que a imagem pode armazenar (i.e. $\lfloor \frac{3MN}{8} \rfloor$ para uma imagem com dimensões $M \times N$), adicionamos o caractere \0 — oito bits 0 seguidos — para demarcar o final da mensagem.



Figura 2: Detalhe 52×32 da imagem `lenna.png`, com o texto Lorem Ipsum ocultado [2]

¹Onde 0 refere-se ao plano menos significativo e 7 ao mais significativo

²Logs da codificação do texto “abc” na imagem `tiny4x2.png` ao utilizar-se a flag `-vvv`

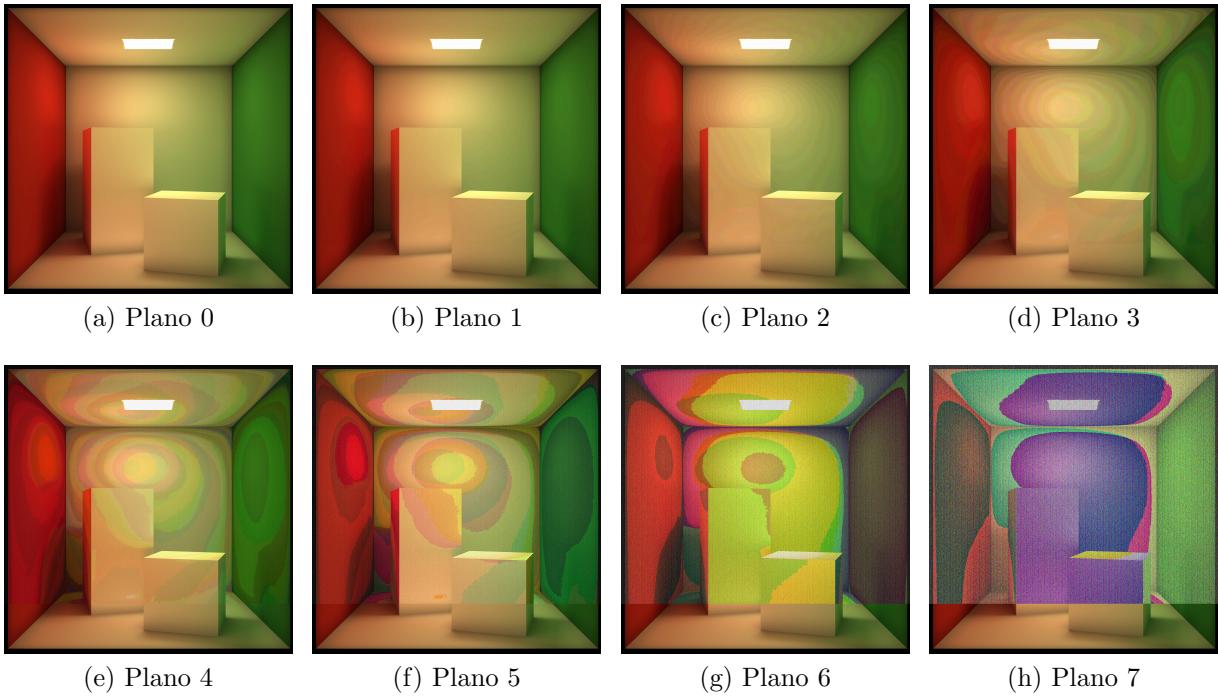


Figura 3: Imagem `cornellbox.png` (1920×1920) com o livro “Crime and Punishment”⁴ ocultado

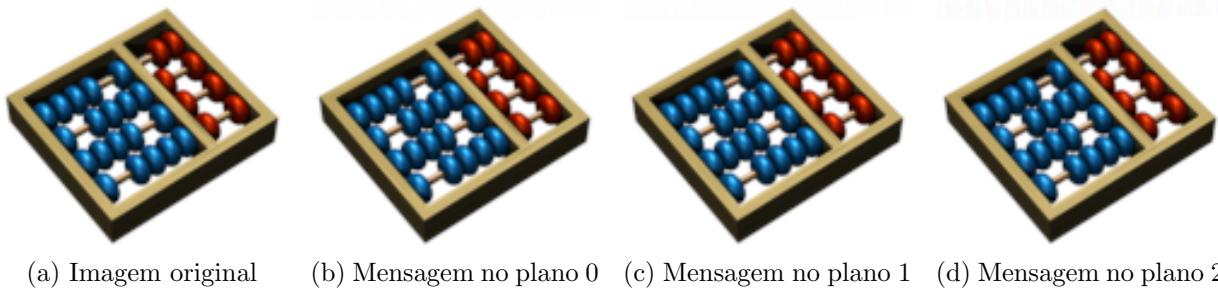


Figura 4: Logo do Instituto de Computação (120×103) com o texto Lorem Ipsum ocultado [2]

3.2 Decodificação

A decodificação de um texto escondido em uma imagem se dá pela inversão dos últimos passos do processo ilustrado na Figura 1, com o cuidado de percorrer o texto decodificado da esquerda para a direita em busca de um caractere `\0`, que sinaliza o fim da mensagem codificada. Assim, extraem-se os bits de um determinado plano alternando-se entre os canais de cores, e com eles recuperam-se as letras que compõem a mensagem (pelo código ASCII).

Utilizou-se o comando `diff` [3] para validar o script de decodificação, comparando-se as mensagens originais com as decodificadas.

3.3 Identificação de planos de bit alterados

Vemos pela Figura 4 que o armazenamento de texto nos bits menos significativos resultam em imagens praticamente indistinguíveis das originais a olho nu. Entretanto, ao visualizarmos

⁴1164453 caracteres ASCII do formato *plain text* disponível pelo **Project Gutenberg** (gutenberg.org)

os planos de bit isoladamente, podemos determinar se uma imagem contém uma mensagem oculta em seus pixels, pois o padrão de ruído característico dessa técnica de esteganografia — que vemos nas Figuras 2 e 3 a partir do plano 5 — é facilmente observado.

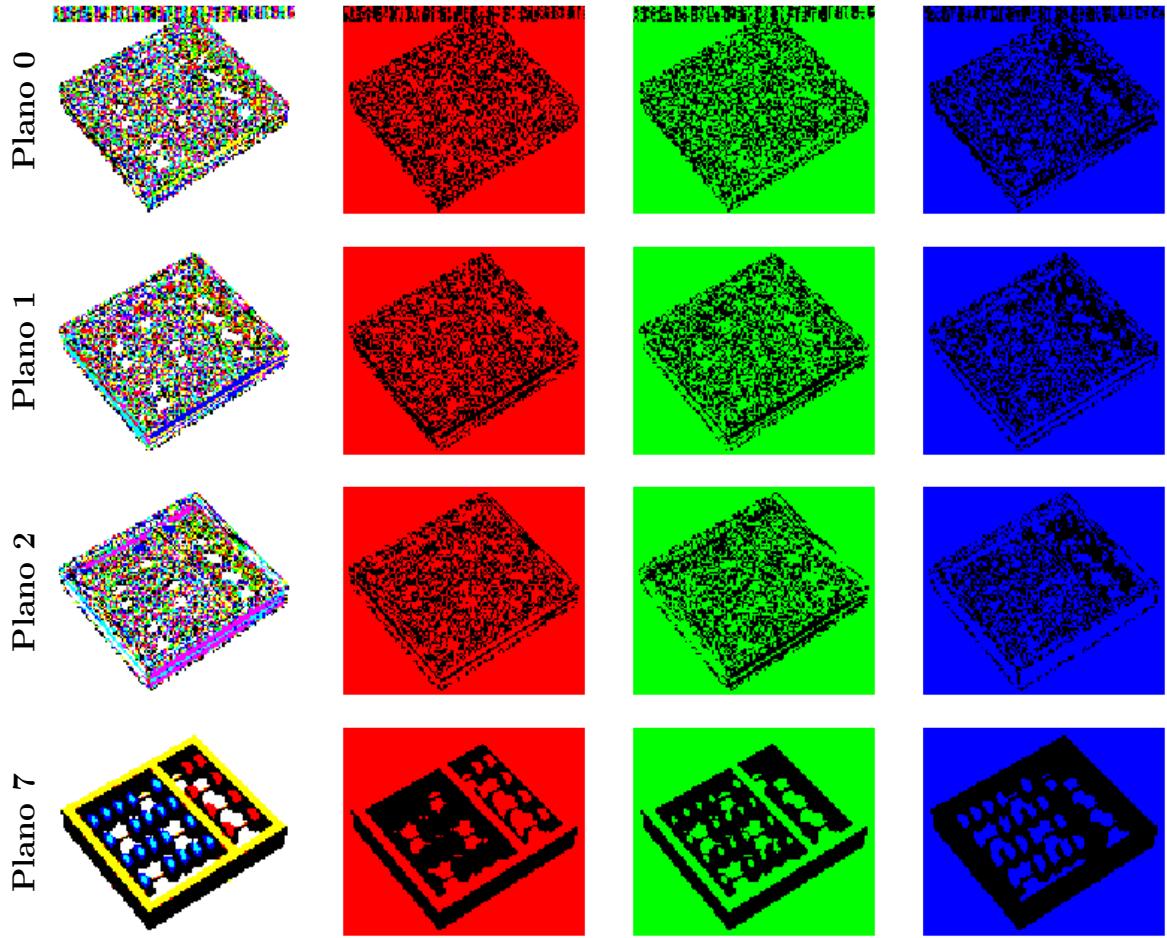


Figura 5: Planos de bits isolados para a imagem `ic.png` com mensagem oculta no plano 0. Os canais RGB são apresentados juntos na primeira coluna, e separados por cor (vermelho, verde e azul) nas outras colunas

4 Análise e Discussão

A partir dos resultados apresentados na Seção 3 percebemos que a identificação de imagens contendo mensagens armazenadas nos planos 0 e 1 é inviável por análise visual. Entretanto, por tratar-se de um método simples, ao separarmos os planos de bit podemos facilmente encontrar imagens com dados ocultados, como vemos na Figura 5.

Além disso, ao alterarmos bits mais significativos, o ruído gerado na imagem é evidente, uma vez que os novos valores de pixels possuem uma diferença maior dos originais. Por exemplo, a diferença máxima que podemos ter em relação ao valor de um pixel na imagem original ao modificarmos o bit menos significativo é 1, e ao modificarmos o segundo bit menos significativo é 2, mas ao substituirmos o bit mais significativo podemos ter uma diferença de 128 (sendo que os valores de intensidade dos pixels vão de 0 a 255) — de forma geral, a diferença máxima é 2^b , onde $b \in [0, 7]$ é o plano de bits.

Desse modo, tal técnica é apropriada para aplicações como a inclusão de marcas para

verificação de direitos autorais [4], onde a detecção de que imagens possuem dados ocultados não é um grande problema. Por outro lado, quando o objetivo de aplicar-se uma técnica esteganográfica é a divulgação de mensagens sem o conhecimento de sua existência por possíveis interceptadores, esse método é em geral inadequado.

5 Conclusões

Para imagens genéricas, a técnica de esteganografia demonstrada é apropriada, principalmente ao alterarmos apenas dois bits menos significativos, porque assim não é gerado nenhum ruído perceptível (veja as Figuras 2, 3 e 4), e com isso atingimos o objetivo de esconder a existência da mensagem de inspeções visuais.

Em contrapartida, por tratar-se de uma técnica relativamente simples, é possível detectar-se adulterações ao visualizar-se os planos de bits isoladamente para a maioria das imagens, como demonstrado na Figura 5, onde vemos a Figura 4b.

Portanto, quando o objetivo principal é que as mensagens trafeguem de forma despercebida, mas não há preocupação com a descoberta de que elas possuam informações gravadas, ou se a probabilidade de que hajam interceptadores tentando decifrá-las é mínima, esse método é uma boa solução. Caso contrário, esse método ainda assim pode ser usado, porém deve-se cuidadosamente escolher imagens que tenham seus planos de bits menos significativos já “ruidosos”, para que a mensagem se camuflie neles, e, principalmente, deve-se ocultar mensagens pequenas em relação ao tamanho da imagem (que não utilizem mais do que as duas primeiras linhas de pixels, por exemplo), pois isso dificulta a identificação de dados escondidos mesmo com técnicas mais robustas como análises estatísticas.

Referências

- [1] “Esteganografia — Wikipedia, The Free Encyclopedia,” 2019. [Online]. Available: <https://pt.wikipedia.org/wiki/Esteganografia#Imagens> 1
- [2] “Lorem ipsum — Wikipedia, The Free Encyclopedia,” 2019. [Online]. Available: https://en.wikipedia.org/wiki/Lorem_ipsum 2, 3
- [3] “Diff — Wikipedia, The Free Encyclopedia,” 2019. [Online]. Available: <https://en.wikipedia.org/wiki/Diff> 3
- [4] “Marca de água digital — Wikipedia, The Free Encyclopedia,” 2019. [Online]. Available: https://pt.wikipedia.org/wiki/Marca_de_%C3%A1gua_digital 5

Links acessados em outubro de 2019.