Escondendo uma mensagem em uma imagem (trabalho 1)

Introdução ao Processamento de Imagem Digital

Randerson A. Lemos (103897) 2022-1S

1 Introdução

A esteganografia é a área de conhecimento que se dedica a estudar técnicas de ocultação de informações (por exemplo, mensagens) em imagens sem que estas sofram alterações perceptíveis ao olho humano. A esteganografia, neste trabalho, será aplicada por meio da substituição dos bits menos significativos dos pixels das imagens escolhidas pelos bits da mensagem de interesse que se deseja ocultar. A seguir teremos três seções, a de Solução, a de Resultado e a de Conclusão. Na seção de Solução, detalhes técnicos, de usabilidade e de decisões da solução proposta são fornecidos. Na seção de Resultado, os principais resultados são apresentados. Na seção de Conclusão, há não apenas a apresentação de conclusões, mas também de discussões.

2 Solução

A solução utiliza a linguagem de programação Python e conta com o auxílio do gerenciador de projetos e pacotes Conda. Assumindo que o usuário tenha o Conda instalado em sua máquina, a configuração do projeto pode ser feita pela execução do comando `conda env create -f environment.yml` a partir da pasta **trab1**. Esse comando cria o ambiente de trabalho **mc920-trab1** e instala os seguintes módulos: opencv, numpy, scipy, pandas, matplotlib. Finalizada a configuração do ambiente de trabalho em questão, o usuário deve executar o comando `source source .sh`[1] para carregar as variáveis de ambiente adequadas e, assim, poder usar os programas do projeto dentro do próprio ambiente de trabalho recém configurado.

Dos arquivos presentes na pasta do projeto **mc920-trab1**, destacam-se as pastas **png**, **tex**, **txt** e os programas **codificar.py**, **decodificar.py**, **mostrar_planos.py**. A pasta **png** contém imagens no formato png que podem ser utilizadas como receptáculos da mensagem a ser ocultada. A pasta **txt** contém exemplos de textos que podem ser utilizados no processo de ocultação de suas mensagens. A pasta **tex** contém os arquivos Latex deste relatório. As informações pertinentes dos programas **codificar.py**, **decodificar.py**, **mostrar_planos.py** serão detalhadas a seguir.

2.1 Codificar.py

O programa **codificar.py** é responsável por ocultar uma mensagem de interesse em uma imagem escolhida que deve ser colorida e estar no formato png. Para ser executado, esse programa precisa receber os parâmetros **imagem_entrada**, **texto_entrada**, e **planos_bits**:

- ao parâmetro **imagem_entrada** deve-se fornecer o nome do arquivo da imagem a ser utilizada como 'bau' da mensagem que se deseja esconder;
- ao parâmetro **texto_entrada** deve-se fornecer o nome do arquivo do texto que contém a mensagem que se deseja ocultar;
- ao parâmetro **planos_bits** deve-se fornecer os planos de bits menos significativos dos pixels da imagem escolhida que serão utilizados para registrar a mensagem de interesse. Os valores esperados para esse parâmetro são: 0, ou 1, ou 2, ou combinações desses valores separados por ':'. Quando mais de um plano de bits são passados ao programa pelo parâmetro **planos_bits**, ocorre a ordenação em ordem crescente desses planos de modo que a utilização dos bits dos pixels da imagem se dê sempre do plano de bits menos significativo para o mais significativo.

O texto de entrada que contem a mensagem a ser escondida na imagem escolhida deve apresentar um código de *end of file* para que o processo subsequente de decodificação consiga identificar o final de mensagem. Aqui,

[1] O comando que configura o ambiente de trabalho mc920-trab1 precisa ser executado apenas um vez. Assim sendo, depois que este ambiente está configurado, o usuário precisa apenas executar o comando `source source.sh`

o código escolhido é **!@#FIM#@!**. A combinação dos caracteres desse código foi pensada de modo que uma possível aparição não intencional de tal código ao longo do texto seja significativamente improvável.

Exemplos de como utilizar o programa **codificar.py** utilizando os recursos contidos dentro do próprio projeto são:

```
python3 codificar.py -imagem_entrada=png/watch.png -texto_entrada=txt/texto1.txt -planos_bits=2;
python3 codificar.py -imagem_entrada=png/watch.png -texto_entrada=txt/texto1.txt -planos_bits=1:2;
python3 codificar.py -imagem_entrada=png/watch.png -texto_entrada=txt/texto1.txt -planos_bits=0:1:2.
```

Após executado, o programa **codificar.py** gera uma imagem de saída. Esse imagem nada mais é que a imagem de entrada contendo a mensagem do texto de entrada oculta nos seus bits menos significativos de acordo com os planos de bits passados pelo usuário. Essa imagem de saída é salva automaticamente na pasta **out** que está dentro da pasta do projeto **trab1**. Se a imagem de entrada tem o nome **img_ent** a imagem de saída apresentará o nome **img_entm_planoX**, onde o letra X é um *placeholder* para os números dos planos de bits escolhidos pelo usuário.

2.2 Decodificar.py

2.3 Mostrar_planos.py

O programa **mostrar_planos.py** é responsável por gerar mapas de bits dos três canais (RGB) de uma imagem selecionada. Para funcionar esse programa precisa receber os parâmetros **imagem_entrada** e **planos_bits**:

- ao parâmetro **imagem_entrada** deve-se fornecer o nome do arquivo da imagem que se deseja visualizar os planos de bits;
- ao parâmetro **planos_bits** deve-se fornecer os valores numéricos dos planos de bits que se deseja visualizar. Essa parâmetro aceita valores de 0 (referente ao plano de bits menos significativo) até 7 (referente ao plano de bits mais significativo). Para gerar imagens de múltiplos planos de bits, basta passar os valores dos planos de interesse separados por ':'.

Exemplo de como executar o programa **mostrar_planos.py** utilizando os recursos[2] contidos dentro do próprio projeto são:

```
python3 mostrar_planos.py -imagem_entrada=out/watchm_plano12.png -planos_bits=3,
python3 mostrar_planos.py -imagem_entrada=out/watchm_plano12.png -planos_bits=0:1:2:7.
```

Após executado, o programa **mostrar_planos.py** gera imagens dos planos de bits passados pelo usuário dos canais RGB da imagem selecionada. Essa imagem de saída é salva automaticamente na pasta **out** que está dentro da pasta do projeto **trab1**. Se a imagem de entrada tem o nome **img_entm_planoX**, as imagens de saída apresentarão nome segundo o padrão **img_entm_planoX_plano_bits_Y_Z** em que Y é o número do plano de bits apresentado na imagem e Z é de qual canal (R, G ou B) este planos de bits provém.

[2] É necessário que o comando adequado do programa codificar.py tenha sido executado.