Representação digital de imagens

1 Introdução

Um padrão de representação/compressão é formado por uma série de regras, que definem como os dados serão representados digitalmente. Assim, um padrão não apresenta em si as técnicas para a codificação, mas sim define a organização dos dados. Para ser compatível com o padrão, o codificador deve, portanto, gerar um fluxo de dados no formato definido.

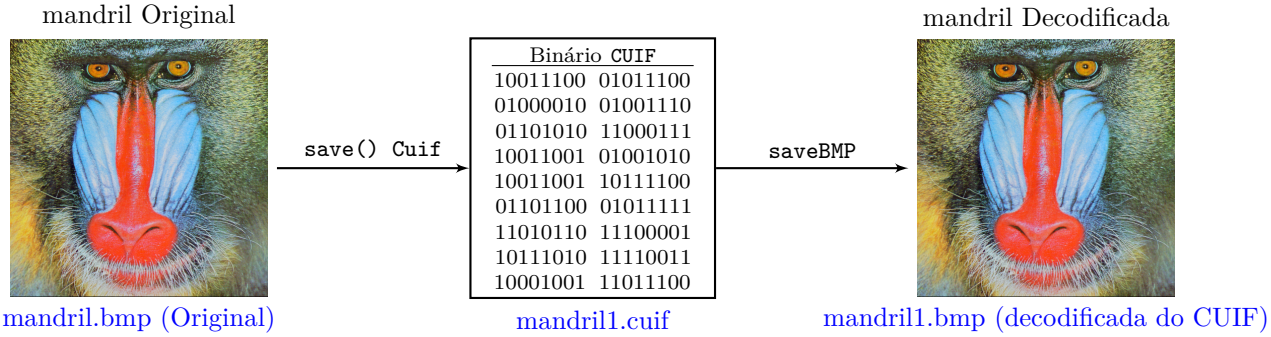
Nesta sequência de aulas práticas sobre imagens, desenvolveremos uma série de padrões de representação/-compressão de imagem digital. Em cada aula prática, iremos incrementar nosso padrão, gerando assim novas versões. Chamaremos nosso padrão inicial de `CUI.1`: `CU`stom `I`mage versão 1; Já seu formato de arquivo será chamado de ***CUstom Image Format*** (ou `CUIF`). Assim, a cada nova versão teremos um novo padrão. Porém o formato de arquivo será o mesmo, independente do padrão.

Para visualizarmos os efeitos da compressão, devemos utilizar algum formato de representação de imagens conhecido, e fornecer meios de converter entre um padrão e outro. Um formato de arquivo comum é o chamado bitmap, ou BMP. A vantagem de usarmos tal formato é sua capacidade de representação de imagens sem compressão e, portanto, sem distorções. Assim, teremos uma baseline para comparação: tanto de taxa de compressão quanto de qualidade.

Nesta aula prática está disponível o arquivo Cuif.py que implementa parcialmente funções para manipulação de arquivos CUIF:

1. `Cuif(img,version,matriculas)`: é o construtor de um objeto Cuif, onde são passadas a imagem (objeto PIL.Image), a versão CUIF a ser utilizada (nesta aula prática será visto as versões 1 e 2), e um array contendo o número de matrícula dos alunos do grupo da aula prática;
2. `printHeader()`: função permitindo imprimir os campos do cabeçalho do objeto CUIF.
3. `show()`: mostra a imagem condificada no objeto Cuif.
4. `save(fimandrilme)`: salva o objeto Cuif em um arquivo condificando a imagem no formato cuif;
5. `open(fimandrilme)`: lê um arquivo codificado com CUIF e cria um objeto Cuif.
6. `saveBMP(fimandrilme)`: Salva o objeto Cuif em formato BMP.

Assim, o fluxo para visualização do efeito da codificação através do padrão `CUI` é o seguinte:



mandril.bmp (Original) mandril1.cuif mandril1.bmp (decodificada do CUIF)

Fonte da imagem: http://sipi.usc.edu/database/database.php?volume=misc&image=12#top

Para isso, devemos primeiramente entender os dois formatos de arquivo.

2 Formato BMP

O formato BMP tem um cabeçalho no início do arquivo. Tal cabeçalho provê as informações necessárias para a interpretação dos dados do arquivo. A Tabela 1 apresenta a descrição do cabeçalho BMP. Por exemplo, o campo *offset* do cabeçalho BMP, nos bytes 10-13, indica a posição onde termina o cabeçalho e começa a parte de dados da imagem.

Para manter o projeto simples, trataremos apenas de BMPs de 3 bytes/*pixel* e sem nenhuma compressão. Vamos considerar esta restrição para apresentar o modo como os pixels são codificados no bitmap (bitmap aqui é a matriz de pixels, aqui representada pelos componentes RGB).

Considerando apenas imagens BMP sem compressão, na parte de dados do arquivo BMP, os *pixels* da imagem são codificados em sequência *raster*, que segue da esquerda para a direita e de cima para baixo. No arquivo BMP, os pixels iniciam na posição indicada pelo campo *offset*. Há duas características a serem observadas:

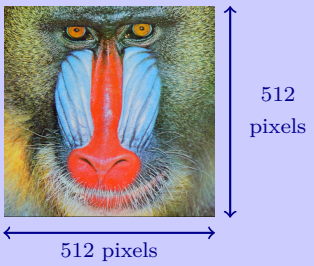
1. Os três canais de cor de cada *pixel* são codificados em sequência, sendo um byte por canal (**R**, **G** e **B**). Porém, ao invés de codificar **R**, **G** e **B**, a ordem adotada no BMP é **B**, **G** e **R**;
2. O comprimento de cada linha da imagem, em bytes, deve ser sempre múltiplo de 4. Caso a imagem não tenha uma resolução vertical múltipla de 4, inserem-se bytes com valor 0 até preencher a linha. Este procedimento é denominado de *padding*.
3. No raster (parte de dados do arquivo de imagem), a sequência de linhas da imagem estão invertidas em relação a visualização da imagem. Ou seja, a primeira linha da imagem no arquivo é na realizade a última linha da imagem.

Tabela 1: Especificação do Cabeçalho BMP

Offset	**Tamanho**	**Descrição**
0	2	assinatura (identificador), deve ser $4D42_{16}$
2	4	tamanho do arquivo BMP em bytes (não é confiável)
6	2	reservado, deve ser 0
8	2	reservado, deve ser 0
10	4	offset, em bytes, até o início dos dados da imagem
14	4	tamanho da estrutura `BITMAPINFOHEADER`, deve ser 40_{10}
18	4	número de *pixels* na horizontal (largura)
22	4	número de *pixels* na vertical (altura)
26	2	número de planos na imagem, deve ser 1
28	2	número de bits por *pixel* (1, 4, 8, ou 24)
30	4	tipo de compressão (0=nenhuma, 1=RLE-8, 2=RLE-4)
34	4	número de bytes da imagem (incluindo *padding*)
38	4	resolução horizontal em *pixels*/m (não é confiável)
42	4	resolução vertical em *pixels*/m (não é confiável)
46	4	número de cores na imagem, ou zero
50	4	número de cores importantes, ou zero

Exemplo de cabeçalho BMP

Vamos usar a imagem `mandril.bmp` como exemplo e criar um cabeçalho de arquivo BMP. Tal imagem tem resolução 512×512 *pixels* e 24 bpp (bits por *pixel*). A imagem BMP descrita não terá nenhum tipo de compressão e não indexará cores.



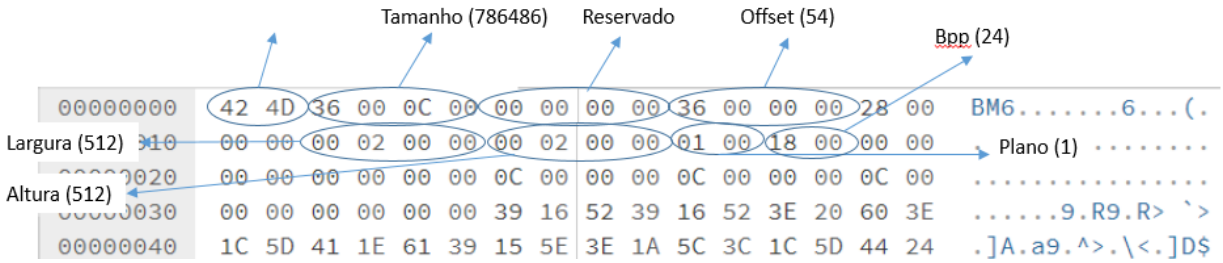
Devemos também calcular o tamanho do arquivo. Os dados ocuparão:

$$\#\text{ bits na imagem} = \text{largura} \times \text{altura} \times \text{bpp}$$

byte	valor		significado
	2	0	
0	–	$4D42_{16}$	assinatura bmp
2	786486_{10}		tamanho do arquivo
6	–	0	reservado
8	–	0	reservado
10	54_{10}		offset para o início do arquivo
14	40_{10}		fixo
18	512_{10}		largura
22	512_{10}		altura
26	–	1	planos (deve ser 1)
28	–	24_{10}	bpp
30	0		sem compressão
34	786432_{10}		número de bytes
38	786432_{10}		*pixels*/m
42	786432_{10}		*pixels*/m
46	0		cores na imagem
50	0		cores importantes

Vejam na Figura 1 como o cabeçalho deve ser interpretado no hexedit.it.

Figura 1: Exemplo de um cabeçalho real



3 CUIF

De maneira similar ao formato BMP, o CUIF inicia com um cabeçalho apresentado na sequência.

Offset	Tamanho	Descrição
0	4	assinatura (identificador), deve ser "CUIF"
4	1	versão do padrão `CUI`
5	1	número de estudantes no grupo (Nestud)
6	4	largura da imagem (em pixels)
10	4	altura da imagem (em pixels)
14	4×Nestud	Matrículas dos estudantes (4 bytes cada)

Após o cabeçalho inicia a parte de dados da imagem. O modo como estes dados serão organizados depende da versão do padrão CUIF utilizado.

3.1 CUIF.1

O padrão `CUIF.1` é uma representação RGB separada em canais, de maneira similar ao BMP. Porém, diferente do BMP onde cada *pixel* aparece com seus canais BGR, o `CUIF.1` apresenta cada canal R, G e B completos em sequência *raster*. Ou seja, ao invés de codificar *pixel*-a-*pixel*, codifica-se canal-a-canal. Cada *pixel* utilizará 1 byte em cada canal.

Exemplo de `CUI.1`, representado em um arquivo CUIF

Vamos supor uma imagem com 2×2 *pixels*:

red = $(\text{FF } 00\ 00)_{16}$ green = $(00\ \text{FF } 00)_{16}$
blue = $(00\ 00\ \text{FF})_{16}$ gray = $(\text{B7 B7 B7})_{16}$

Para este exemplo, há apenas um estudante no grupo, cuja matrícula é 99132042. Vejamos como fica a organização de um arquivo CUIF para armazenar essa imagem seguindo o padrão `CUI.1`:

byte	valor				significado
	3	2	1	0	
0	–	CUIF			assinatura CUIF
4	–	–	–	1	versão do padrão CUI (CUI.1)
5	–	–	–	1	número de estudantes no grupo
6	2_{10}				largura
10	2_{10}				altura
14	99132042_{10}				matrícula do aluno no grupo
18	–	–	–	FF_{16}	R pixel 0,0
19	–	–	–	00_{16}	R pixel 0,1
20	–	–	–	00_{16}	R pixel 1,0
21	–	–	–	$B7_{16}$	R pixel 1,1
22	–	–	–	00_{16}	G pixel 0,0
23	–	–	–	FF_{16}	G pixel 0,1
24	–	–	–	00_{16}	G pixel 1,0
25	–	–	–	$B7_{16}$	G pixel 1,1
26	–	–	–	00_{16}	B pixel 0,0
27	–	–	–	00_{16}	B pixel 0,1
28	–	–	–	FF_{16}	B pixel 1,0
29	–	–	–	$B7_{16}$	B pixel 1,1

4 Roteiro

1. Baixe os arquivos py e imagem disponibilizaoda no moodle, e abra em seu ambiente de desnvolvimento preferido.
2. Abra o arquivo `lena.bmp` em um editor hexadecimental, por exemplo, como aquele em https://hexed.it/, e responda a primeira questão do Relatório.
3. Abra o arquivo praticaIII.py e inclua na lista matrículas os números de matrículas dos estudantes no grupo;
4. Execute o arquivo praticaIII.py que converte a imagem `lena.bmp` para `CUIF.1`, e verifique que o arquivo lena1.cuif foi criado.
5. Façam a conversão inversa (gerando um arquivo lena.bmp a partir do lena1.cuif) usando os comandos comentados no praticaIII.py.
6. Verifiquem se os números de matrícula de todos os alunos no grupo foram exibidos no console;
7. Abra as imagens `lena.bmp` e `lena1.bmp` com algum visualizador e compare as imagens.

5 Calculando Ruído

A figura abaixo representa um esquema de codificação e decodificação. Na entrada, a mensagem original (Ori) é codificada e posteriormente decodificada, resultando em uma mensagem decodificada (Dec). Em uma codificação com perdas de informações, a mensagem decodificada (Dec) é diferente da mensagem original (Ori). Neste caso, diz-se que o codificador gerou ruído na mensagem original.

Há diversas formas para medir o erro gerado pelo codificador. Uma destas métricas é a Média dos Erros Quadráticos (MSE – *Mean Squared Error*). Considerando que tanto Ori quanto Dec tenham tamanho n, cada símbolo pode ser indexado, respectivamente, como ori_i e dec_i, onde $1 \leq i \leq n$. Vamos assumir que os símbolos sejam valores numéricos, assim, a MSE é definida como:

$$\text{MSE}(\text{Ori}, \text{Dec}) = \frac{1}{n}\sum_{i=1}^{n}(\text{ori}_i - \text{dec}_i)^2 \tag{1}$$

Para determinar o MSE entre uma imagem Ori e uma imagem Dec, o tamanho de símbolos é a quantidade de componentes de cor de uma imagem (3 componentes por píxel). Assim, o tamanho n de uma imagem de largura w e altura h é igual a $3{*}h{*}w$.

Outra métrica bastante utilizada, principalmente na compressão de sinais é a chamada Relação Sinal-Ruído de Pico (PSNR – *Peak Signal-to-Noise Ratio*), definida (em dB) como:

$$\text{PSNR}(\text{Ori}, \text{Dec}) = 10 \times log_{10}\left(\frac{(2^b - 1)^2}{\text{MSE}(\text{Ori}, \text{Dec})}\right) \tag{2}$$

Onde b é o número de bits por símbolo. Quanto maior o PSNR melhor é a qualidade da imagem, ou seja, menores são os erros. Notem que quando Ori e Dec são iguais, isto é, quando não há erros, PSNR $= \infty$. Utilizaremos a PSNR para medir os erros causados em nossos padrões `CUI`.

5.1 CUIF.1

O padrão `CUIF.2` é implementa um método de compressão com perdas. Você deverá analisar o código CUIF e explicar a técnica de compressão.

6 Relatório

Entregue via Moodle o relatório contendo as respostas das questões abaixo e o código modificado:

Questão 1. Abra o arquivo lena.bmp no editor hexadecimal em https://hexed.it/ e, analisando o formato do cabeçalho BMP apresentado na Seção 2, indique no relatório: qual é o valor dos campos *offset* e *tamanho do arquivo*? Quais são os valores dos componentes de cor R, G e B do primeiro pixel armazenado no arquivo?

Questão 2. Qual é o tamanho do cabeçalho do arquivo lena1.cuif para seu grupo?

Questão 3. No arquivo praticaIII.py tem um função PSNR incompleta. Implemente esta função de maneira a calcular o PSRN passando como parâmetro a imagem original e uma decodificada. Implemente o cálculo do MAE e PSNR com base nas fórmulas da seção 5 .

Questão 4. Indique o PSNR comparando a imagem original mandril.bmp (original) com a imagem obtida a partir do arquivo CUIF.1 (lena1.bmp). Explique porque do resultado do PSNR para o caso do CUIF.1.

Questão 5. Compacte as imagens lena.bmp e lena1.cuif com zip. Qual a taxa de compressão obtida para os dois arquivos? Qual arquivo compactou mais? Explique porque deste resultado, ou seja, indique a vantagem de organizar os pixels nesta sequência definida pelo CUIF.1 (primeiro os valores de R, depois de G e finalmente de B) para a compressão baseada em RLE ou DPCM? Dica: relembre os princípios da compressão RLE e DPCM e compare a parte de dados de imagem do arquivo lena.bmp e lena1.cuif no editor hexadecimal.

Questão 6. Agora altere o código em PraticaIII.py para que seja gerado o arquivo lena2.cuif, que utiliza a versão CUIF.2 (usar 2 em vez de 1 para indicação da versão) e lena2.bmp. Visualiza as imagens lena1.bmp e lena2.bmp para ver se existem diferenças visíveis. Analise o código que gera o arquivo CUIF.2 (em Cuif.py) e explique o princípio da compressão adotada no CUIF.2

Questão 7. Indique as taxas de compressão obtidas pelos CUIF.1 e CUIF.2 para a imagem lena.bmp? Para este cálculo determine a razão entre um arquivo cuif e a imagem lena.bmp. Qual versão do CUIF compactou mais?

Questão 8. Indique o PSNR comparando a imagem original lena.bmp (original) com a imagem obtida a partir do arquivo CUIF.2 (lena2.bmp). Justifique a resposta do PSNR indicando a fonte do ruído.