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.

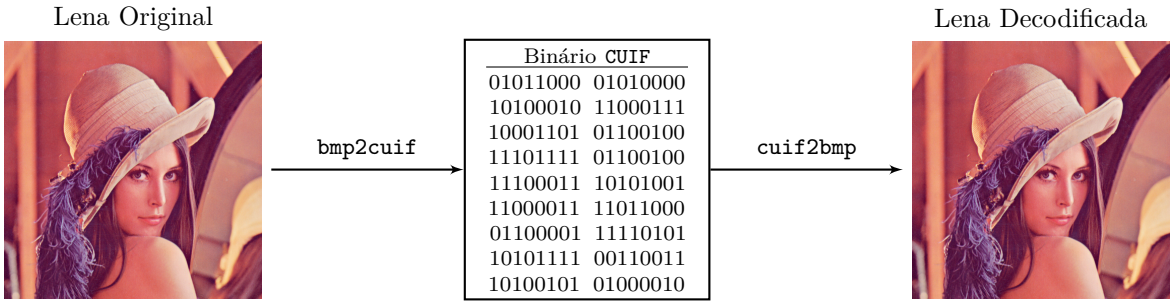
Nesta sequência de aulas práticas sobre imagens, desenvolveremos uma série de padrões de representação/-compressão de imagem digital. Em cada aula prática, iremos incrementar nosso padrão, gerando assim novas versões. Chamaremos nosso padrão inicial de `CUI.1`: `CU`stom `I`mage versão 1; Já seu formato 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.

Desenvolveremos duas ferramentas para conversão:

1. `bmp2cuif`: para conversão de BMP para CUIF;
2. `cuif2bmp`: para conversão de CUIF para BMP;

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



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*, 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*.

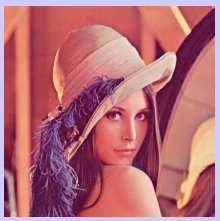
3. No raster (parte de dados do arquivo de imagem), a sequência de linhas da imagem estão invertidas em relação a visualização da imagem. Ou seja, a primeira linha da imagem no arquivo é na realizade a última linha da imagem.

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 `Lena.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

3 CUIF

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

Offset	Tamanho	Descrição
0	2	assinatura (identificador), deve ser 5431_{10}
2	1	versão do padrão `CUI`
3	1	número de estudantes no grupo (NUMBER_OF_STUDENTS)
4	4	largura da imagem (em pixels)
8	4	altura da imagem (em pixels)

Após o cabeçalho, há uma lista de identificadores dos alunos no grupo. Cada identificador (ID) ocupará 4 bytes. Para esta disciplina, será utilizado o número da matrícula de cada aluno como ID. Note que o número de IDs no arquivo deve estar definido corretamente no cabeçalho (NUMBER_OF_STUDENTS).

Após 12 bytes do cabeçalho + 4× NUMBER_OF_STUDENTS bytes, estarão os dados da imagem. O modo como estes dados serão organizados depende da versão do padrão utilizado.

3.1 CUI.1

O padrão `CUI.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 `CUI.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 = (FF 00 00)$_{16}$ green = (00 FF 00)$_{16}$
blue = (00 00 FF)$_{16}$ gray = (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	–	–	5431_{10}		assinatura CUIF
2	–	–	–	1	versão do padrão CUI (CUI.1)
3	–	–	–	1	número de estudantes no grupo
4	2_{10}				largura
8	2_{10}				altura
12	99132042_{10}				matrícula do aluno no grupo
16	–	–	–	FF_{16}	R pixel 0,0
17	–	–	–	00_{16}	R pixel 0,1
18	–	–	–	00_{16}	R pixel 1,0
19	–	–	–	$B7_{16}$	R pixel 1,1
20	–	–	–	00_{16}	G pixel 0,0
21	–	–	–	FF_{16}	G pixel 0,1
22	–	–	–	00_{16}	G pixel 1,0
23	–	–	–	$B7_{16}$	G pixel 1,1
24	–	–	–	00_{16}	B pixel 0,0
25	–	–	–	00_{16}	B pixel 0,1
26	–	–	–	FF_{16}	B pixel 1,0
27	–	–	–	$B7_{16}$	B pixel 1,1

4 Roteiro

1. Crie um projeto java em seu ambiente de desenvolvimento preferido (p.e. Eclipse), e copie neste projeto o código java na pasta src do arquivo cuif1.zip disponível no Moodle. Também coloque o arquivo lena.bmp na pasta do projeto Java.
2. Abra o arquivo lena.bmp disponível no cuif1.zip no editor hexadecimental, por exemplo, disponível em https://hexed.it/ e responda a primeira questão do Relatório.
3. Modifique o valor da variável `numero_de_estudantes` (linha 45 do arquivo `bmp2cuif.java`) para o número de estudantes no grupo;
4. Atualizem o array (`id_estudantes`, linha 46, `bmp2cuif.java`) com seus números de matrícula dos alunos do grupo de alunos;
5. Há uma imagem chamada `lena.bmp`. Converta-a para `CUI.1` usando o comando abaixo, e verifique que o arquivo lena.cuif foi criado.

   ```
   java bmp2cuif -v 1 lena.bmp lena.cuif
   ```

6. Façam a conversão inversa usando o comando abaixo e verifique que o arquivo lenadecodificada.bmp foi criado.

   ```
   java cuif2bmp lena.cuif lenadecodificada.bmp
   ```

7. Verifiquem se os números de matrícula de todos os alunos no grupo foram exibidas no terminal;
8. Abra as imagens `lena.bmp` e `lenadecodificada.bmp` com algum visualizador e verifique que as duas imagens são diferentes.

5 Relatório

Responda as questões e entregue o código modificado:

Questão 1. Abra o arquivo lena.bmp no editor hexadecimal em https://hexed.it/ e indique no relatório: qual é o valor do campo *offset*? Qual é o tamanho de bytes do cabeçalho deste arquivo? Quais são os valores dos componentes de cor (RGB) do primeiro pixel armazenado no arquivo (primeiro píxel da última linha da imagem quando ela for visualizada?

Questão 2. Corrigir é o erro no código para que a imagem decodificada seja igual a imagem original. O código corrigido deve ser entregue. **Dica:** a implementação de tal conversão está no arquivo `Bitmap.java`, método `cuif1toRaster` (linha 219). Também pode analisar o conteúdo do arquivo original BMP e aquele decodificado, por exemplo usando https://hexed.it/

Questão 3. Qual é o tamanho do cabeçalho do arquivo lena.cuif?

Questão 4. Há perdas nos dados da imagem na conversão `bmp` → `cuif` (`CUI.1`) → `bmp`? Expliquem.

Questão 5. Compacte separadamente a imagem lena.bmp e lena.cuif usando zip ou rar. Qual é o arquivo que compactou mais? Em seguida, indique a vantagem de organizar os pixels nesta sequência (primeiro os valores de R, depois de G e finalmente de B) para a compressão baseada em entropia, por exemplo, DPCM? Dica: relembre primeiro o princípio do DPCM e compare a parte de dados de imagem do arquivo lena.bmp e lena.cuif no editor hexadecimal.