

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.

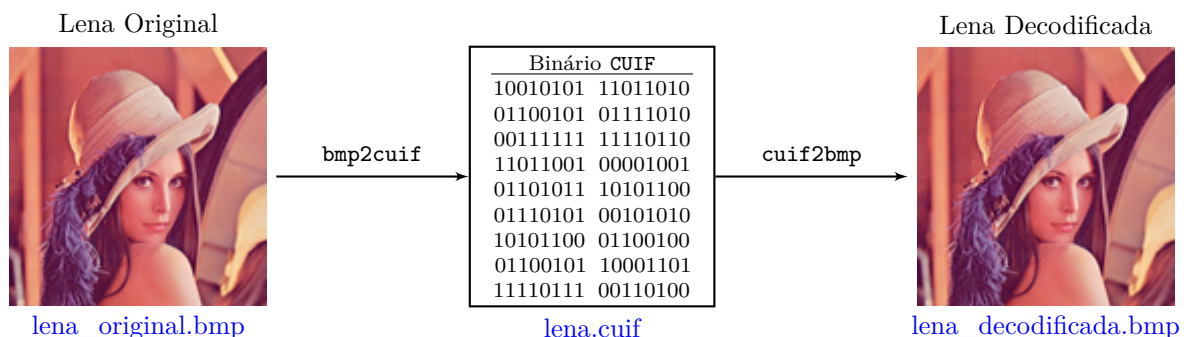
Nesta sequência de aulas práticas sobre imagens, desenvolveremos uma série de padrões de representação/-compressão de imagem digital. Em cada aula prática, iremos incrementar nosso padrão, gerando assim novas versões. Chamaremos nosso padrão inicial de **CUI.1: C**Ustom **I**mage versão 1; Já seu formato de arquivo será chamado de **C**Ustom **I**mage **F**ormat (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. A Tabela 1 apresenta a descrição do cabeçalho **BMP**. Há, nos bytes 10-13 um campo que é preenchido com o *offset* para o início do arquivo. Após esse *offset* inicia de fato o conteúdo da imagem.

Observação: 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. Os *pixels* da imagem são codificados em sequência *raster*, que segue da esquerda para a direita e de cima para baixo. Há duas características a serem observadas:

1. Os três canais de cor de cada *pixel* são codificados em sequência, sendo um byte por canal. Porém, ao invés de codificar **R**, **G** e **B**, a ordem adotada é **B**, **G** e **R**;

2. O comprimento de cada linha da imagem, em bytes, deve ser múltiplo de 4. Caso não o seja, inserem-se bytes com valor 0 até preencher a linha. Este procedimento é denominado de *padding*.

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 <i>pixels</i> na horizontal (largura)
22	4	número de <i>pixels</i> na vertical (altura)
26	2	número de planos na imagem, deve ser 1
28	2	número de bits por <i>pixel</i> (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 <i>padding</i>)
38	4	resolução horizontal em <i>pixels</i> /m (não é confiável)
42	4	resolução vertical em <i>pixels</i> /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		<i>pixels</i> /m
42	786432 ₁₀		<i>pixels</i> /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 + $4 \times \text{NUMBER_OF_STUDENTS}$ 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. Baixem o projeto CUI no Moodle.
2. Modifique o valor da variável `numero_de_estudantes` (linha 45 do arquivo `bmp2cuif.java`) para o número de estudantes no grupo;
3. Atualizem o array (`id_estudantes`, linha 46, `bmp2cuif.java`) com seus números de matrícula dos alunos do grupo de alunos;
4. 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 --version 1 lena.bmp lena.cuif
```
5. Façam a conversão inversa usando o comando abaixo e verifique que o arquivo `lenadecodificada.bmp` foi criado.

```
java cuif2bmp lena.cuif lenadecodificada.bmp
```
6. Verifiquem se os números de matrícula de todos os alunos no grupo foram exibidas no terminal;
7. 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. 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).

Questão 2. Há perdas nos dados da imagem na conversão `bmp` \rightarrow `cuif` (CUI.1) \rightarrow `bmp`? Expliquem.

Questão 3. O que pode ser feito para gerar uma imagem CUI.1 que preserve apenas um dos canais de cor (R, G ou B) do bitmap? Indique o método da classe `Cuif` a modificar, e entregue o método alterado para manter apenas o canal R.