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 (embora esta possa depender de técnicas específicas). Para ser compatível com o padrão, o codificador deve, portanto, gerar um fluxo de dados no formato definido.

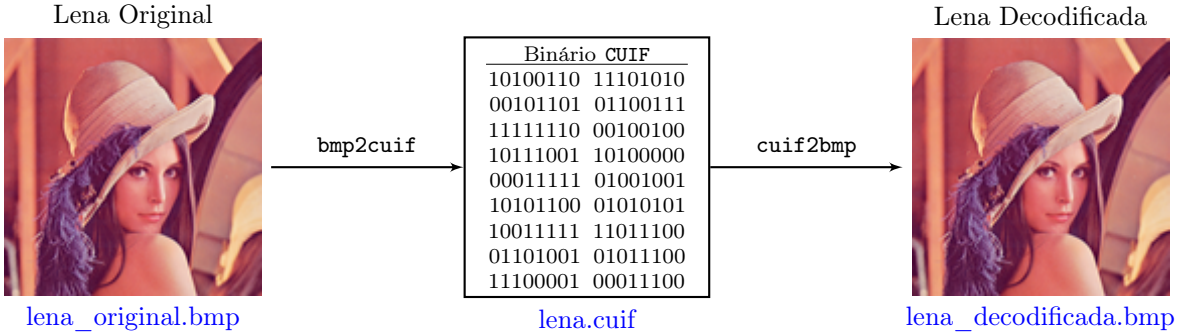
Nesta sequência de aulas práticas sobre imagens (Práticas II até IV), desenvolveremos uma série de padrões de representação/compressão de imagem digital. Em cada aula prática, iremos incrementar nosso padrão, gerando assim novas versões. Chamaremos nosso padrão inicial de `CUI.1`: `CU`stom `I`mage versão 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. 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 *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 = $(\text{FF } 00\ 00)_{16}$ 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	–	–	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 ou aqui;
2. Modifiquem a macro `NUMBER_OF_STUDENTS` (linha 77 do arquivo `cuif.h`) para o número de estudantes no grupo;
3. Atualizem a lista de IDs (`list_of_ids`, linha 86, `bmp2cuif.c`) com seus números de matrícula;
4. Utilizem o `make` para compilar;
   - O `Makefile` gerará dois executáveis, `bmp2cuif` e `cuif2bmp`;
5. Há uma imagem de exemplo na pasta `img`, chamada `lena.bmp`. Converta-a para `CUI.1` usando, a partir da pasta `cui`, o seguinte comando:
   ```
   $ ./dist/bmp2cuif --version 1 img/lena.bmp test/lena.cuif
   ```
6. Façam a conversão inversa usando o comando:
   ```
   $ ./dist/cuif2bmp --verbose test/lena.cuif test/lena.bmp
   ```
7. Verifiquem se os números de matrícula de todos os alunos no grupo foram exibidas no terminal;
   - Caso não sejam exibidas, verifiquem a macro `NUMBER_OF_STUDENTS` e a `list_of_ids`;
8. Abram a imagem `test/lena.bmp` com algum visualizador e respondam:

Questão 1. *(1 ponto)* Qual é o erro na implementação da conversão de BGR para `CUI.1`? **Dica:** a implementação de tal conversão está no arquivo `cuif_v1.c`, função `cuif_v1_new_from_bitmap` (linha 11).

9. Corrijam o erro e respondam as seguintes questões:

Questão 2. *(1 ponto)* 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?

Questão 3. *(1 ponto)* Há perdas nos dados da imagem na conversão `bmp` → `cuif` (`CUI.1`) → `bmp`? Expliquem.

Nosso padrão `CUI.2` é bastante similar ao `CUI.1`. Porém, ao invés de representar os *pixels* em RGB, a representação será em YCbCr, de acordo com a recomendação ITU-R BT.601. O documento descrevendo tal recomendação está disponível em `https://www.itu.int/dms_pubrec/itu-r/rec/bt/R-REC-BT.601-7-201103-I!!PDF-E.pdf`.

10. Baixem a recomendação, consultem[1] as Seções 2.5.1 e 2.5.2 e implementem a conversão RGB → YCbCr em ponto flutuante na função `rgb_to_ycbcr_base_double` do arquivo `color_spaces.c` (linha 31);
11. Testem a corretude da conversão usando a *flag* `--color` no comando `bmp2cuif`. **Dica:** Há, no arquivo `color_spaces_test.c`, uma tabela com os valores esperados para Y, Cb e Cr em ponto flutuante (*Table 2 – Expected signal values after normalization*). Confiram as colunas Y, Cb e Cr (não se atenham à conferir apenas os valores inversos).
12. Gerem um arquivo chamado `lena_cui2.cuif` usando:
    ```
    $ ./dist/bmp2cuif --version 2 img/lena.bmp test/lena_cui2.cuif
    ```

Questão 4. *(1 ponto)* Há perdas nos dados da imagem na conversão RGB → YCbCr → RGB? Como isso pode ser mensurado?

Questão 5. *(1 ponto)* Qual o tamanho teórico[2] (em bytes) dos dados do arquivo `lena_cui2.cuif`? Apresentem os cálculos.

Questão 6. *(1 ponto)* Há alguma compressão entre `CUI.1` e `CUI.2`? Expliquem.

13. Comprimam a pasta `src` em um arquivo `.zip` e enviem ao Moodle da disciplina juntamente com as respostas para as questões;

Atenção: Os 4 pontos restantes serão avaliados através do código fonte entregue.

[1] Vejam também, no material da disciplina, os slides 73 e 84 do Cap. 2 e o slide 87 do Cap. 3.
[2] desconsiderando o sistema de arquivos