MAC122 Princípios de Desenvolvimento de Algoritmos
EP no. 1

Prof. Dr. Paulo Miranda

[1]Instituto de Matemática e Estatística (IME)
Universidade de São Paulo (USP)

1. Estrutura dos arquivos de imagens no formato PGM

Uma imagem em tons de cinza $I(x, y)$ (e.g., imagem de ultrasom, fatia tomográfica) e bidimensional é uma matriz com N linhas e M colunas. Nesta matriz cada elemento $I(x, y)$, $x = 0, 1, \ldots, M - 1$ e $y = 0, 1, \ldots, N - 1$, é chamado *pixel* (i.e., uma abreviação de *picture elements*), e o valor de $I(x, y)$ é proporcional ao brilho da imagem neste ponto, de 0 (preto) até o valor máximo (branco), quantizados em uma escala de níveis de cinza (ver Figura 1). Esses valores correspondem a intensidade de luminosidade (ou alguma outra propriedade física relevante) amostrada por um sensor para uma dada região do espaço.

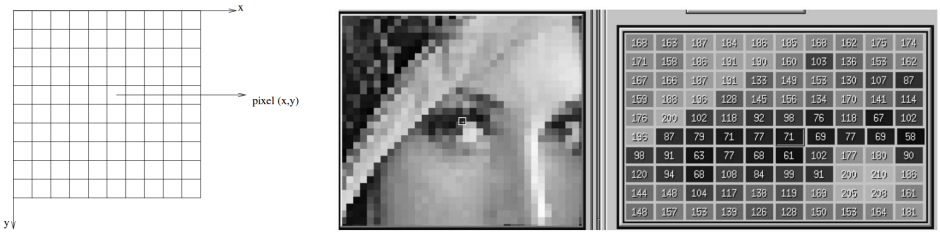


Figura 1: Representação de uma imagem monocromática digital (à esquerda). Ponto indicado sobre o olho de uma imagem da Lenna (centro). Matriz de pixels em uma região de interesse de 10 x 10 pixels em torno do ponto indicado (à direita).

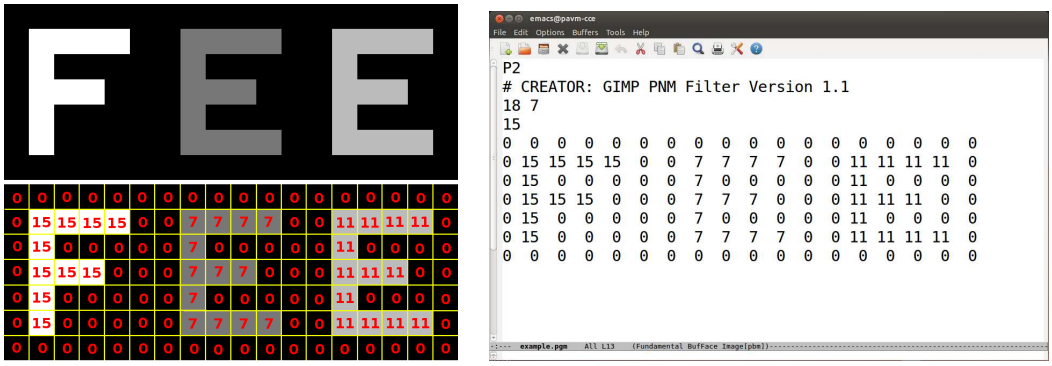
Existem diversos formatos de arquivo para armazenar uma imagem digital em um computador (e.g., JPG, PNG, PCX, BMP). Nesse EP vamos trabalhar com o formato mais simples existente, o formato PGM (*Portable Gray Map*). Esse formato não usa algoritmos de compressão dos dados. O formato PGM do tipo P2 é simplesmente um arquivo texto que pode ser visualizado em um editor de textos. Cada imagem PGM é composta por:

1. Uma primeira linha identificando o tipo de arquivo. Sempre será **P2** no nosso caso.
2. Opcionalmente, uma linha com comentários iniciada com o caracter **'#'**. Por questões de simplicidade, considere que as imagens que iremos trabalhar não possuem esta linha.
3. Uma linha contendo as dimensões da imagem: o número de colunas (largura) e o número de linhas (altura).
4. A próxima linha armazena o valor máximo de intensidade presente na imagem.
5. Na sequência temos os valores dos pixels que são armazenados percorrendo a matriz na ordem da esquerda para a direita e de cima para baixo.

Portanto, um arquivo PGM possui a seguinte estrutura interna:

```
P2
Largura Altura
Valor maximo de intensidade
Valor pixel (0,0)
Valor pixel (1,0)
Valor pixel (2,0)
...
Valor pixel (x,y)
...
Valor pixel (Largura -1, Altura -1)
```

Na Figura 2 temos um exemplo de uma pequena imagem e do seu arquivo texto correspondente no formato PGM do tipo P2. Para mais informações sobre o formato PGM consulte a página: `http://netpbm.sourceforge.net/doc/pgm.html`.



(a) (b)

Figura 2: (a) Exemplo de uma imagem 18 x 7, e (b) seu arquivo correspondente no formato PGM sendo visualizado em um editor de textos (*emacs*).

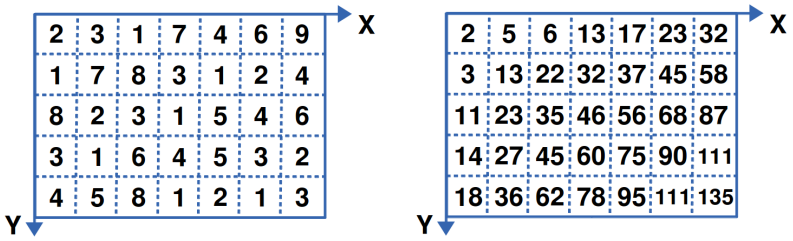
2. Imagem integral

O conceito de imagem integral foi popularizado por Viola & Jones [2] no seu método para localização automática de faces em imagens. Nesse EP vamos explorar o conceito de imagem integral para resolver, de modo eficiente, um problema mais simples.

Uma imagem integral II consiste em uma representação de uma imagem I tal que o valor na posição (x, y) de II contém a soma de todos os pixels acima e à esquerda de (x, y) em I, incluindo inclusive o valor de $I(x, y)$ (Equação 1). Um exemplo de imagem integral é mostrado na Figura 3.

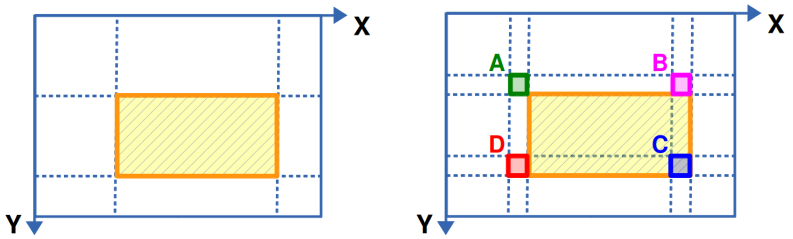
$$II(x, y) = \sum_{x' \leq x,\, y' \leq y} I(x', y') \tag{1}$$

A imagem integral pode ser calculada com uma única passada na imagem original. Tendo-se calculado a imagem integral, é possível encontrar a soma dos elementos de uma área retangular qualquer da imagem original utilizando apenas quatro acessos a elementos da imagem integral, conforme explicado na ilustração da Figura 4, e no exemplo da Figura 5.



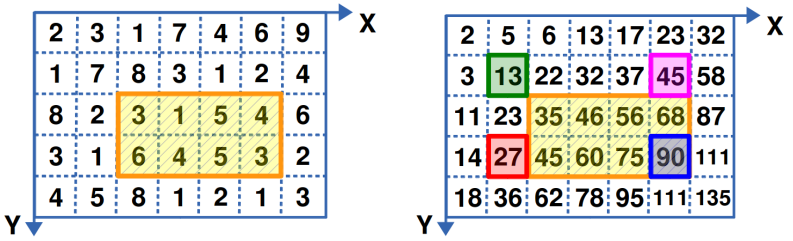
(a) (b)

Figura 3: (a) Representação matricial de uma imagem 5 x 7 em tons de cinza. (b) Sua imagem integral II correspondente calculada pela Equação 1.



(a) (b)

Figura 4: (a) Uma área retangular de uma imagem em tons de cinza. (b) Com quatro acessos a elementos da imagem integral é possível obter o resultado da soma dos valores na região retangular através da expressão $II(C) - II(B) - II(D) + II(A)$.



(a) (b)

Figura 5: (a) Considere a área retangular indicada na imagem. (b) Com quatro acessos a elementos da sua imagem integral é possível obter o resultado da soma de todos valores da região retangular (ou seja, $3 + 1 + 5 + 4 + 6 + 4 + 5 + 3 = 31$) através da expressão $II(C) - II(B) - II(D) + II(A) = 90 - 45 - 27 + 13 = 31$.

3. Estimativa de nível de ruído presente na imagem

Algum grau de ruído sempre está presente em qualquer aparelho eletrônico que transmite ou recebe um "sinal". Para as televisões esse sinal são os dados da transmissão enviados por cabo ou recebidos pela antena da TV; para as câmeras digitais, o sinal é a luz que atinge o sensor da câmera. Quando o ruído se dá de modo independente em relação a posição dos pixels na imagem, o desvio padrão do sinal dos pixels sobre uma região homogênea da imagem gera uma boa estimativa do ruído existente [1]. O desvio padrão é calculado extraindo a raiz quadrada da variância, que é dada por:

$$Var(X) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2 \quad (2)$$

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \quad (3)$$

Uma formulação alternativa e equivalente pode ser obtida por:

$$Var(X) = \frac{1}{n}\left[\sum_{i=1}^{n} x_i^2 - n\bar{x}^2\right] \quad (4)$$

4. Atividade

Faça um programa em C que:

- lê o nome de um arquivo de imagem no formato PGM (e.g., "fig01.pgm"),
- lê o tamanho T do lado de um quadrado/janela (e.g., 50),
- aloca memória para a imagem e carrega a imagem do disco,
- desloca a janela T x T sobre a imagem original, calculando a variância do brilho dos pixels dentro da janela para cada posição (considere apenas as posições com a janela inteiramente contida na imagem, sem transbordar para fora).
- imprime o valor da menor variância encontrada (esse valor corresponde a uma região homogênea nos fornecendo uma estimativa do nível de ruído).

O programa deve considerar três implementações diferentes para o cálculo da variância:

- cálculo através da Equação 2, percorrendo duas vezes todos elementos dentro da janela (a primeira vez para calcular $\bar{x}$ (Eq. 3)).
- cálculo através da Equação 4 percorrendo uma única vez os elementos dentro da janela, utilizando duas variáveis acumuladoras (uma para a soma das intensidades dos pixels, e outra para a soma do quadrado das intensidades).
- cálculo através da Equação 4, porém usando imagens integrais, evitando ter que acessar todos elementos dentro da janela.

No final deve ser entregue (junto com o programa) um pequeno relatório contendo experimentos de avaliação do tempo de execução das diferentes implementações, para diferentes tamanhos de janela (e.g., variando entre T=10 e T=300). Esse relatório deve ser incluído dentro do código fonte do próprio programa na forma de um comentário de bloco. Para realizar medições de tempo, use as funções da biblioteca padrão time.h:

```
#include <time.h>
...

int main(){
  clock_t start, end;
  double cpu_time_used;
  start = clock();

  ... /* Codigo a ser cronometrado */
```

```
end = clock();
cpu_time_used = ((double) (end - start)) / CLOCKS_PER_SEC;
```

No código acima *cpu_time_used* armazena o tempo gasto pela operação em segundos.

Observações:

- Procure utilizar funções sempre que possível para resolver cada subtarefa (e.g., uma função para alocar memória da matriz, uma função para leitura dos dados do arquivo, uma função para calcular a imagem integral de uma imagem fornecida como parâmetro).
- Para evitar problemas com *overflow* [1], sugiro utilizar o tipo **long long** ou **double** para a representação da imagem integral.
- Instruções mais específicas para padronizar e facilitar a correção serão disponibilizadas no fórum do PACA.

Referências

[1] Johannes T. Heverhagen. Noise measurement and estimation in MR imaging experiments. *Radiology*, 245:638–639, Dec 2007. doi: 10.1148/radiol.2453062151.

[2] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages I–511–I–518, 2001.

[1] A condição de *overflow* ocorre quando o valor atribuído a uma variável é maior que o maior valor que o tipo desta variável consegue representar.