Processamento de Imagens

O processamento digital de imagens usa intensivamente vetores e matrizes. Uma imagem é geralmente representada como uma matriz de pixels, onde cada pixel é descrito por um ou mais valores inteiros que indicam seus níveis de cor ou de luminosidade. Por isso, processar imagens é uma ótima forma de exercitar o uso de matrizes.

O formato PGM

Para facilitar a leitura e escrita dos arquivos de imagem, neste projeto será adotado o formato de imagem Portable Gray Map (PGM), um formato de imagem em **níveis de cinza** bem simples e fácil de ler/escrever. Boa parte dos programas de tratamento de imagens reconhece o formato PGM.

Eis um exemplo de imagem no formato PGM:

feep.pgm

```
P2
# this is a PGM image
24 7
15
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  3  3  3  3  0  0  7  7  7  7  0  0 11 11 11 11  0  0 15 15 15 15  0
0  3  0  0  0  0  0  7  0  0  0  0  0 11  0  0  0  0  0 15  0  0 15  0
0  3  3  3  0  0  0  7  7  7  0  0  0 11 11 11  0  0  0 15 15 15 15  0
0  3  0  0  0  0  0  7  0  0  0  0  0 11  0  0  0  0  0 15  0  0  0  0
0  3  0  0  0  0  0  7  7  7  7  0  0 11 11 11 11  0  0 15  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
```

Existem duas variantes do formato PGM: *PGM plain* (P2) e *PGM raw* (P5); **ambas** devem ser lidas pelos filtros implementados neste projeto. Mais detalhes sobre essas variantes podem ser obtidas nesta página.

Alguns exemplos de imagens no formato PGM para usar no projeto:

- Imagens selecionadas
- Muitos outros exemplos

Filtros de imagem

Este projeto visa construir filtros simples para imagens digitais em níveis de cinza, no formato PGM (P2 e P5) com pixels de 8 bits (1 byte).

Filtro negativo

O filtro negativo consiste em converter cada pixel da imagem em seu complemento. Sendo *max* o valor máximo

para um pixel na imagem, o complemento de um pixel com valor *v* seria *max-v*.

Forma de chamada:

```
pgmnega -i input -o output
```

Exemplo de entrada e de saída:

→

Filtro de rotação simples

O filtro de rotação gira uma imagem em 90° no sentido horário.

Forma de chamada:

```
pgmrotacao -i input -o output
```

Exemplo de entrada e de saída:

→

Filtro de rotação livre

O filtro de rotação gira uma imagem em um ângulo θ (>0) no sentido **horário**, em relação ao seu centro (vide rotação de imagens 2D).

Requisitos:

- O tamanho da imagem de saída deve ser ajustado para **não cortar os cantos** da imagem.
- Os espaços vazios gerados pela rotação devem ser **preenchidas com a cor branca**.
- O ângulo de rotação é informado pela opção `-a`; se não for informado, por default θ = 90º.

Forma de chamada:

```
pgmrotacao -a N -i input -o output
```

Exemplo de entrada e de saída para θ = 30º:

Filtro de limiar

O filtro de limiar (threshold filter) converte uma imagem em tons de cinza em um imagem em preto-e-branco (2 cores). A forma mais simples de fazer isso é comparar o valor de cada pixel com um limiar pré-definido: se o valor do pixel for igual ou superior ao limiar, ele vira branco (*v=max*), caso contrário ele vira preto (*v=0*).

Forma de chamada:

```
pgmlimiar -l N -i input -o output
```

onde *N* é um limiar real entre 0.0 (0% do valor máximo) e 1.0 (100% do valor máximo). Caso o limiar não esteja definido, assume-se que seja 50%.

Exemplo de entrada e de saídas com limiar de 50% e 75%:

Filtro de redução de ruído pela média

O filtro da média é usado para para "limpar" uma imagem, ou seja, reduzir o seu nível de ruído. O algoritmo é básico bem simples: para cada pixel, seu valor deve ser substituído pela média de todos os 9 pixels vizinhos (incluindo ele mesmo).

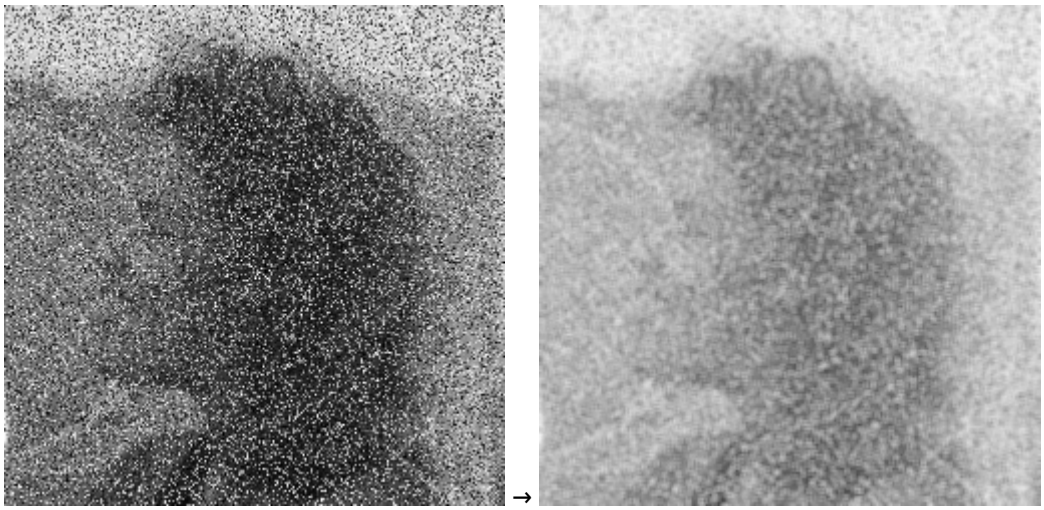
Deve ser tomado cuidado especial ao tratar os pixels nas primeiras e últimas linhas ou colunas, pois eles não têm todos os vizinhos. Nesses casos, devem ser considerados no cálculo somente os vizinhos existentes.

A imagem processada deve ser armazenada em uma segunda matriz, para que não se misturem valores novos e velhos no cálculo dos pixels.

Forma de chamada:

```
pgmmedia -i input -o output
```

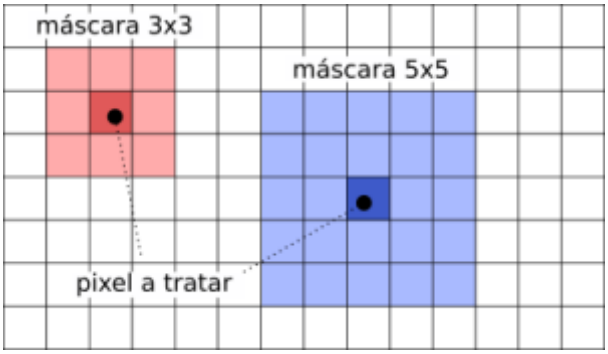
Exemplo de entrada e de saída:



Filtro da mediana

O filtro da mediana reduz o nível de ruído de uma imagem sem prejudicar muito sua nitidez. Este filtro consiste

basicamente em substituir o valor de um pixel pelo valor da mediana de seus pixels vizinhos. O número de vizinhos a considerar é definido por uma **máscara**, ou seja, a matriz de vizinhos que circunda o pixel a tratar (incluindo ele mesmo):



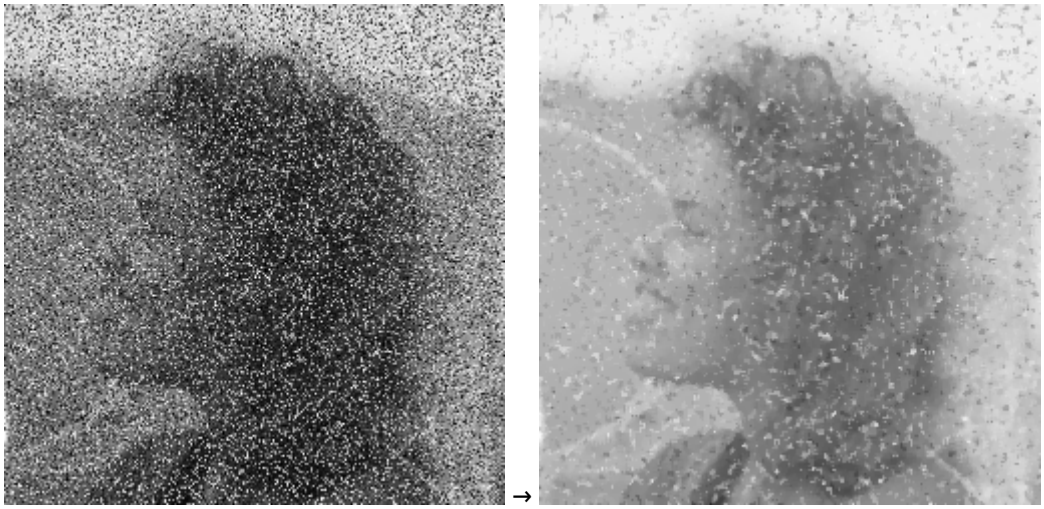
Requisitos:

- O tamanho da máscara é um inteiro positivo ímpar, que pode ser informado na linha de comando (opção `-m`); caso não seja informado, o valor default é 3, para uma máscara de 3×3 pixels.
- Os pixels nas bordas da imagem não têm todos os vizinhos e portanto não devem ser filtrados.
- Para ordenar os valores dos pixels deve ser usada a função `qsort` da biblioteca C (`man qsort`).

Forma de chamada:

```
pgmmediana -m N -i input -o output
```

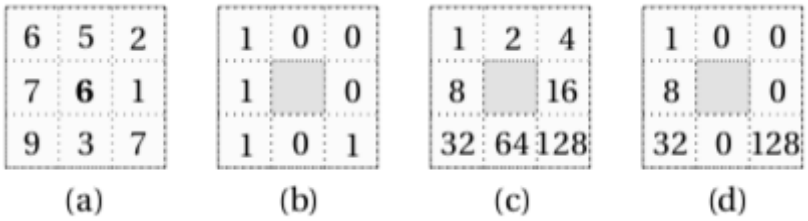
Exemplo de entrada e de saída:



Filtro LBP (Local Binary Patterns)

LBP (*Local Binary Patterns*) ou Padrões Binários Locais, é um descritor de textura muito utilizado em algoritmos de reconhecimento de imagens. O funcionamento desse filtro é simples: para cada pixel de uma imagem com 256 níveis de cinza, seleciona-se uma vizinhança de 8 pixels. O valor LBP é então calculado para o pixel central e armazenado na imagem destino, que possui as mesmas dimensões da imagem original.

Para o cálculo do LBP de um pixel, usa-se uma **máscara** com valores 2^n, com n = {0, ..., 7}, ou seja: 1, 2, 4, 8, 16, 32, 64 e 128. O exemplo abaixo ilustra o cálculo de LBP para o pixel com valor 6:



Considere o pixel central com valor 6 e seus oito vizinhos na imagem (a). O valor do pixel central é utilizado como limiar e todos os pixels com valor de intensidade maior ou igual a ele recebem 1, caso contrário, 0 (b). Esses valores são então multiplicados pela máscara (c), resultando então nos valores apresentados em (d). O valor LBP do pixel 6 é dado pela soma desses valores, ou seja LPB = 1+8+32+128 = 169. Note que essa codificação garante valores LBP entre 0 e 255.

Forma de chamada:

```
pgmlbp -i input -o output
```

Exemplo de entrada e de saída:

Outros filtros (ideias)

- mudança de escala
- detecção de bordas
- melhoria de nitidez
- melhoria de contraste
- ...

Atividade

- Implementar os filtros acima definidos como arquivos e comandos separados. Exemplo: o filtro de negativo deve ser implementado em um arquivo `pgmnega.c` que gera um executável `pgmnega`. **Respeite os nomes dos executáveis** indicados nas descrições acima!
- Os filtros devem aceitar como entrada imagens no formato PGM (P2 e P5, *plain* e *raw*) e devem gerar como saída imagens no mesmo formato da entrada.
- O formato PGM permite definir imagens com pixels de 8 ou 16 bits. Para este trabalho, considere somente imagens com pixels de 8 bits (todas as imagens fornecidas como exemplo têm pixels de 8 bits).
- As rotinas comuns (leitura/escrita de arquivos, tratamento da linha de comando, etc) devem ser implementadas em arquivos separados, cujos cabeçalhos são incluídos nos arquivos de implementação dos filtros.
- Sempre que possível, as informações da imagem necessárias às funções devem ser transferidas como parâmetros (por valor ou por referência, dependendo da situação). Minimizar o uso de variáveis globais.
- Use alocação dinâmica de memória para as imagens, para poder processar as imagens maiores. Só aloque a memória após encontrar o tamanho da imagem.
- Construir um `Makefile` para o projeto:
  - Ao menos os alvos `all` (default), `clean` e `purge`.
  - CFLAGS = `-Wall`
  - Compilar e ligar separadamente (gerar arquivos `.o` intermediários)
- O que deve ser entregue ao professor:
  - arquivos `.c` e `.h`
  - arquivo `Makefile`
  - Por favor, não envie as imagens de teste!

Uso de arquivos

- A opção `-i` indica o nome do arquivo de entrada; se não for informado, deve-se usar a entrada padrão (*stdin*).
- A opção `-o` indica o nome do arquivo de saída; se não for informado, deve-se usar a saída padrão (*stdout*).
- Todas as mensagens de erro devem ser enviadas para a saída de erro (*stderr*).

Essas opções podem ser usadas em qualquer combinação, ou seja:

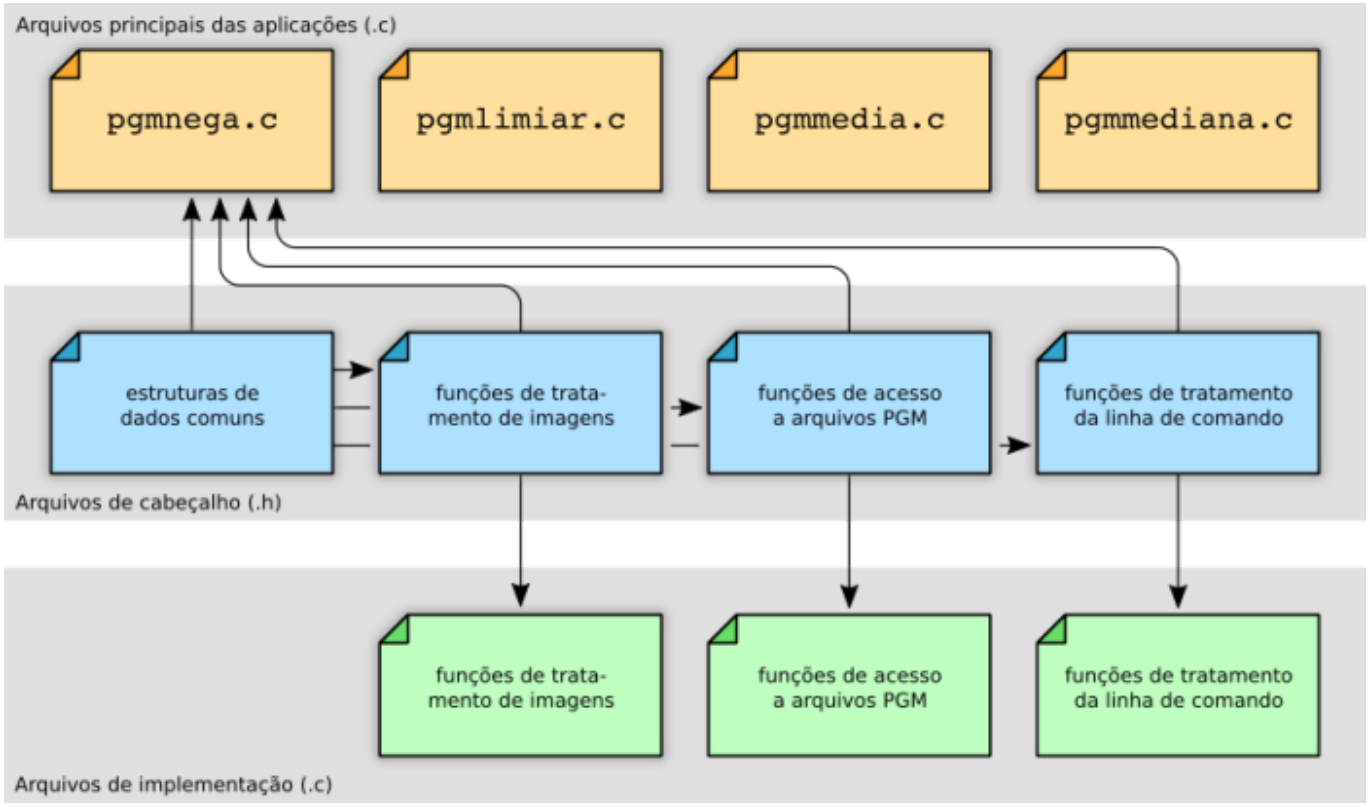
```
// entrada e saída em arquivos
pgmmediana -i input.pgm -o output.pgm
pgmmediana -o output.pgm -i input.pgm

// entrada em arquivo, saída em stdout, vice-versa ou ambos
pgmmediana -i input.pgm > output.pgm
pgmmediana -o output.pgm < input.pgm
pgmmediana < input.pgm > output.pgm

// as opções podem estar em qualquer ordem
pgmmediana -m 5 -i input.pgm -o output.pgm
pgmmediana -i input.pgm -m 5 -o output.pgm
pgmmediana -o output.pgm -i input.pgm -m 5
```

Estrutura do código-fonte

O código-fonte deve ser estruturado em arquivos `.c` e `.h` que agrupem as diversas funcionalidades. A figura abaixo traz uma uma **sugestão de estrutura** para o código-fonte (as setas correspondem a `includes`):



From:
https://wiki.inf.ufpr.br/maziero/ - **Prof. Carlos Maziero**

Permanent link:
https://wiki.inf.ufpr.br/maziero/doku.php?id=c:processamento_de_imagens

Last update: **2023/08/01 19:23**