MAC 110 – Introdução à Ciência da Computação
Bacharelados em Matemática, Estatística e Matemática Aplicada
IME/USP – Primeiro Semestre de 2009
Terceiro Exercício-Programa
Data de entrega: até 27/junho/2009

Introdução

Quando uma imagem é adquirida por uma câmera ou algum outro sistema de captura, ela normalmente contém variações em intensidade e contraste, conhecidos como "ruídos", que podem inviabilizar o seu uso direto se não forem previamente tratadas. O processo de supressão do "ruído" presente em uma imagem é chamado de **suavização**. Neste exercício-programa você terá a oportunidade de: (1) escrever classe em Java que implementará três estratégias de suavização de imagens e (2) escrever uma classe que permitirá ao usuário carregar, visualizar, suavizar e salvar imagens em arquivos.

Primeira Parte — Suavização

Uma imagem em tons de cinza pode ser definida como uma matriz bidimensional com m linhas e n colunas onde cada célula desta matriz ("pixel") indica uma intensidade na imagem. Para efeitos deste EP, cada intensidade será representada por um valor inteiro i, $0 \leq i < 256$, onde $i = 0$ simboliza a ausência de cor ("cor preta") e $i = 255$ simboliza a cor branca. Valores intermediários representam tons de cinza. No site da disciplina, você encontrará o arquivo `Imagem.java`, que define a classe `Imagem`, que contém como atributo uma matriz bidimensional de tamanho arbitrário, um construtor para sua inicialização, e métodos para descobrir a largura (colunas) e altura (linhas) da imagem.

A sua tarefa inicial é adicionar três métodos à classe `Imagem.java` que permitirão a suavização da imagem, conforme descritos abaixo:

```
void filtroMédio(int tamanho)
```

O filtro médio suaviza uma imagem I através do cálculo da média de todos os valores dos pixels em uma vizinhança local determinada por `tamanho`. Para uma vizinhança 3x3 (`tamanho`=3), o valor do pixel suavizado H na posição (i, j) será:

$$H(i,j) = \frac{1}{9} \sum_{k=-1}^{1} \sum_{l=-1}^{1} I(i+k, j+l) \tag{1}$$

onde $I(i, j)$ representa a intensidade original na posição (i, j). Você deverá calcular os valores suavizados para todos os pixels contidos na imagem. Dependendo da posição e tamanho do filtro, não será possível calcular a média usando toda a vizinhança. Um exemplo disto são os cantos da imagem. Neste caso, você deverá aproveitar a máxima vizinhança possível, respeitando valor de `tamanho`, ajustando o peso apropriadamente na equação acima. Você poderá supor que `tamanho` sempre será ímpar e maior que 1.

```
void filtroMediano(int tamanho)
```

É sabido que o filtro médio acaba borrando descontinuidades em intensidades de uma imagem. Aqui você implementará uma alternativa ao filtro médio, o filtro mediano, que determina o valor do pixel suavizado como sendo o valor mediano em uma vizinhança. Por exemplo, dada uma vizinhança de tamanho 3 e posição (i, j), você deverá ordenar os pixels contidos em uma janela 3x3 centralizada em

(i, j) em ordem crescente de intensidade e selecionar o valor do meio como o valor suavizado $H(i, j)$. Conforme descrito anteriormente, você deverá suavizar todos os pixels da imagem. Novamente, você poderá supor que `tamanho` sempre será ímpar e maior que 1.

```
void filtroGaussiano(double sigma, int tamanho)
```

Já no filtro gaussiano, a ponderação de cada pixel é feita de acordo com a função gaussiana:

$$g(x, y) = \frac{1}{2\pi\sigma^2} \exp(-\frac{x^2 + y^2}{2\sigma^2}) \tag{2}$$

onde σ representa o desvio padrão da distribuição e (x, y) são coordenadas relativas a partir do ponto central de suavização. Você deverá implementar a suavização gaussiana utilizando a seguinte fórmula:

$$H(i, j) = \sum_{k=-1}^{1} \sum_{l=-1}^{1} g(k, l) \cdot I(i + k, j + l) \tag{3}$$

O valor de $\sigma > 0$ representa o grau de suavização aplicado na imagem. Note que quanto maior o valor de σ, maior deverá ser o valor de `tamanho` a fim de não truncar antecipadamente a superfície gaussiana. Mais uma vez, você poderá supor que `tamanho` sempre será ímpar e maior que 1.

Procure implementar estes métodos de uma forma ao mesmo tempo clara (que seja fácil de entender) e eficiente (que seja rápida, i.e., que execute poucas operações). Sinta-se à vontade para utilizar métodos ou funções auxiliares se isso for ajudar na clareza do código. Você não deverá mudar o nome dos métodos acima ou seus parâmetros. Você poderá usar funções auxiliares da biblioteca matemática `java.lang.Math` caso deseje.

Note que tanto o filtro médio quando o gaussiano podem gerar valores fracionários para as intensidades suavizadas. Como a matriz de pixels representada na classe `Imagem` aceita somente valores inteiros como intensidades, você deverá arredondar para cima as intensidades finais (ex: a parte inteira de $H(i, j) + 0.5$). **Atenção:** Se as intensidades resultantes forem maiores que 255, elas deverão ser truncadas para 255.

Só para alunos do BCC: Detecção de Bordas em Imagens

A presença de uma borda em uma imagem, dada uma direção, é obtida pela *segunda derivada* dos valores da imagem na direção dada. Iremos computar as bordas na direção horizontal e na direção vertical.

No nosso caso iremos computar as bordas na direção vertical e na horizontal. A derivada será aproximada pelas diferenças entre duas células adjacentes na direção. Serão necessários dois passos de diferenças. Por exemplo, se estamos calculando as diferenças horizontais de uma matriz A para obter a matriz de diferenças *absolutas* D, então

$$D(i, j) = |A(i + 1, j) - A(i, j)| \tag{4}$$

Note que a cada passo de cálculo das diferenças horizontais, uma coluna da matriz irá desaparecer, que convencionamos ser a última. Similarmente, a cada passo de diferença vertical (não mostrada), uma linha desaparece.

Pede-se que a borda apresentada seja preta, e a parte sem borda seja branca, o que pode necessitar de uma manipulação extra pois o valor 0 indica ausência de cor (preto). A imagem das bordas deve só apresentar valores 0 ou 255, e você deve fixar um valor *limiar* de detecção de borda.

Nota: a detecção de bordas pode ser feita na imagem original ou na imagem suavizada por algum dos métodos de suavização. Verifique qual o método que dá o melhor resultado.

Segunda Parte — Aplicação

Você deverá implementar também uma outra classe que conterá o programa principal dentro do método `main()`. Esta classe deverá permitir que o usuário: (1) carregue uma imagem em tons de cinza, (2) mostre a imagem na tela do computador, (3) escolha a técnica de suavização e a aplique, (4) grave a imagem e (5) termine o programa. Estas cinco operações deverão fazer parte de um menu de opções e o usuário deverá fazer a sua escolha via teclado. Exemplo:

```
Menu
====
1) Ler imagem
2) Visualizar imagem
3) Suavizar com filtro médio
4) Suavizar com filtro mediano
5) Suavizar com filtro gaussiano
6) Bordas verticais
7) Bordas horizontais
8) Gravar imagem
9) Fim

Qual a sua opção?
```

Os itens 6 e 7 são apenas para alunos do BCC.

Leitura e escrita

Para facilitar a sua tarefa (ufa!), já desenvolvemos uma classe (`LeituraEscritaImagem.java`) que efetua a leitura e gravação de imagens em tons de cinza no formato `PGM`[1]. Para tanto, basta especificar o nome do arquivo a ser lido, e para a escrita, o nome do arquivo e uma instância de `Imagem`. Exemplo de utilização:

```
Imagem minhaImagem = LeituraEscritaImagem.leImagem(‘‘Lena.pgm’’);
if (minhaImagem == null)
   System.out.println(‘‘Arquivo não existente ou problemas na leitura’’);
else
   LeituraEscritaImagem.escreveImagem(‘‘Copia_Lena.pgm’’);
```

Você deverá esperar que o usuário entre com o nome do arquivo a ser lido ou escrito via teclado, e efetuar a leitura ou gravação com a classe `LeituraEscritaImagem`. Como você pode perceber, não criamos uma instância da classe `LeituraEscritaImagem` no exemplo acima. Tanto o método `leImagem()` quanto `escreveImagem()` são métodos do tipo `static` e pertencem à classe e não à instância. Usamos este tipo de padrão quando desejamos criar bibliotecas de funções, como `java.lang.Math`.

No site da disciplina estão disponíveis algumas imagens no formato `PGM` para você testar o seu programa.

Visualização

Para que possamos observar o resultado da suavização, também disponibilizamos uma classe para efetuar a visualização de objetos do tipo `Imagem`. A classe `VisualizadorImagem` contém o método `mostraImagem()` que abre uma janela e desenha a imagem especificada. Exemplo:

[1] `http://netpbm.sourceforge.net/doc/pgm.html`

```
VisualizadorImagem vis = new VisualizadorImagem();
Imagem minhaImagem = LeituraEscritaImagem.leImagem(‘‘Lena.pgm’’);
if (minhaImagem == null)
   System.out.println(‘‘Arquivo não existente ou problemas na leitura’’);
else
   vis.mostraImagem(minhaImagem, "Lena.pgm");
```

Você deverá usar esta classe para visualizar a imagem carregada em memória. Caso você feche a janela, você precisará criar uma nova instância de `VisualizadorImagem`.

Suavização

Para cada técnica de suavização, você deverá ler via teclado o tamanho da vizinhança, e para o filtro gaussiano, o valor de σ. Uma vez que os parâmetros sejam lidos, você deverá chamar o método correspondente para efetuar a suavização de acordo com a sua implementação da classe `Imagem`. Se você assim desejar, poderá redesenhar a imagem após a filtragem.

Observações finais

Sobre a elaboração:

Este EP pode ser elaborado por equipes de um ou dois alunos, desde que sejam respeitadas as seguintes regras.

- Os alunos devem trabalhar sempre juntos cooperativamente.
- Caso em um grupo exista um aluno com maior facilidade, este deve explicar as decisões tomadas. E o seu par deve participar e se esforçar para entender o desenvolvimento do programa (chamamos isso de *programação em pares*, que é uma excelente prática que vocês devem se esforçar para adotar).
- Mesmo a digitação do EP deve ser feita em grupo, enquanto um digita, o outro fica acompanhando o trabalho.

Sobre a avaliação:

- É sua responsabilidade manter o código do seu EP em sigilo, ou seja, apenas você e seu par devem ter acesso ao código.
- **Não serão toleradas cópias!** Exercícios copiados (com ou sem eventuais disfarces) levarão à reprovação da disciplina e o encaminhamento do caso para a Comissão de Graduação do aluno.
- Exercícios com erros de sintaxe (ou seja, erros de compilação) receberão nota zero.
- É muito importante que seu programa seja elegante, claro e bem indentado, ou seja, digitado de maneira a ressaltar a estrutura de subordinação dos comandos do programa (conforme visto em aula). A qualidade do seu trabalho sob esse ponto de vista influenciará sua nota!
- As informações impressas pelo seu programa na tela devem aparecer da forma mais clara possível. Este aspecto também será levado em consideração no cálculo da sua nota.
- Uma regra básica é a seguinte: do ponto de vista do monitor responsável pela correção dos trabalhos, quanto mais convenientemente apresentado estiver o seu programa, melhor avaliado será seu trabalho.

Sobre a entrega:

- O prazo de entrega é o dia 27/6/2009;
- Entregar apenas um arquivo de nome `Suavizacao.zip` contendo todas as classes de seu exercício, incluindo a sua versão da classe `Imagem`. Se quiser entregar também um arquivo texto com uma explicação sobre o seu programa (opcional), dê a ele o nome `LEIAME`.
- No início de cada arquivo, acrescente um cabeçalho bem informativo, como o seguinte:

```
/*****************************************************************/
/**   MAC 110 - Introdução à Computação                         **/
/**   IME-USP - Primeiro Semestre de 2009                       **/
/**   <turma> - <nome do professor>                             **/
/**                                                             **/
/**   Terceiro Exercício-Programa                               **/
/**   Arquivo: SuavizacaoDeImagens.java                         **/
/**                                                             **/
/**   <nome do(a) aluno(a)>               <número USP>          **/
/**   <nome do(a) aluno(a)>               <número USP>          **/
/**                                                             **/
/**   <data de entrega>                                         **/
/*****************************************************************/
```

  Não é obrigatório que o cabeçalho seja idêntico a esse, apenas que contenha pelo menos as mesmas informações.

- Para a entrega, utilize o Paca. Você pode entregar várias versões de um mesmo EP até o prazo, mas somente a última será armazenada pelo sistema.
- Não serão aceitas submissões por email ou atrasadas. Não deixe para a última hora, pois o sistema pode ficar congestionado e você poderá não conseguir enviar.
- Guarde uma cópia do seu EP pelo menos até o fim do semestre e, novamente, você é responsável por manter o sigilo de seu código-fonte.