Introdução ao ImageJ

Instituto do Coração (InCor - HC-FMUSP)
Escola Politécnica da USP (EP-USP)

Tópicos

- Sobre o ImageJ
- Representação das imagens no ImageJ
- Exibindo imagens no ImageJ
- Classes úteis do ImageJ
- Regiões de interesse
- Introdução aos *Plug-ins*

Sobre o ImageJ...

- Ferramenta para processamento e análise de imagens (baseado em Java)
- Propósito: Uma aplicação de uso livre capaz de viabilizar a fácil implementação de algortimos de processamento de imagens na forma de *plug-ins*
- Disponível em: http://rsb.info.nih.gov/ij/

Sobre o ImageJ... – cont.

- Manipulação imagens de 8, 16, 24 e 32 bits
- Diferentes formatos: TIFF, GIF, JPEG, BMP, DICOM, raw
- Manipulação de conjuntos de imagens (pilha de imagens)
- *Plug-ins* para abrir filmes (avi)
- Regiões de interesse (ROIs)

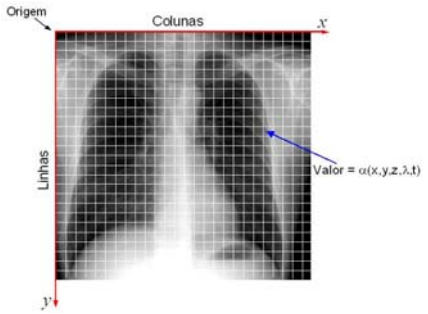
Pontos-chave

- Simplicidade para criar novas funcionalidades (*plug-ins*) e adicioná-las à aplicação
- I/O ficam a cargo do ImageJ
- Fácil acesso aos valores de cada pixel, independentemente do número de bits da imagem.

Interface Gráfica (GUI) ImageJ

Representação da Imagem

Imagem Digital



A Imagem no ImageJ 1/7

- ImageJ possui 5 classes para imagens:
  - 8 bit grayscale (`byte`)
  - 8 bit colour (`byte`)
  - 16 bit grayscale (`short`)
  - RGB colour (`int`)
  - 32 bit image (`float`)
- Suporta também pilhas de imagens (*slices*) de mesmo tamanho

A Imagem no ImageJ 2/7

- ImageJ usa 2 classes para representar e manipular imagens:
  - ***`ImagePlus`***
    - Uma imagem é uma instância de um objeto ***`ImagePlus`***
  - ***`ImageProcessor`***
    - Esta classe encapsula os dados sobre os pixels e possui métodos para acessar diretamente os pixels

A Imagem no ImageJ 3/7

- Os métodos de acesso aos pixels incluem:
  - ***`Object getPixels()`*** – retorna um array de pixel (é necessário fazer o *cast* para o tipo apropriado)
  - ***`int getHeight()`*** – retorna a altura da imagem
  - ***`int getWidth()`*** – retorna largura da imagem

A Imagem no ImageJ 4/7

- Uma subclasse de ***`ImageProcessor`*** é passada ao método `run()` de um *plug-in*:
  - ***`ByteProcessor`*** - 8 bit grayscale
  - ***`ShortProcessor`*** – 16 bit grayscale
  - ***`ColorProcessor`*** – RGB
  - ***`FloatProcessor`*** – 32 bit floating point

A Imagem no ImageJ 5/7

- A representação do pixel usa o tipo de dado `byte` para *grayscale* and imagens coloridas e `short` para imagens 16-bit *grayscale*
  - ***`byte/short`*** são associados às faixas de valores:
    - ***`byte`*** de –128 a 127
    - ***`short`*** de –32768 a 32767
  - Normalmente, para valores *grayscale*, utiliza-se somente valores positivos

A Imagem no ImageJ 6/7

- Para converter um `byte` para um `int`, deve-se eliminar o bit de sinal

```
byte[] pixels=(byte[])
ip.getPixels();

int pixel = 0xff & pixels[j];
```

- Pode-se voltar facilmente ao valor original

```
pixels[j]=(byte) pixel;
```

A Imagem no ImageJ 7/7

- O objeto ***`ColorProcessor`*** retorna os pixels como `int[]`, com os valores RGB numa variável `int`


```
int[]  pixels=(int[]) ip.getPixels();
int red=(0xff0000 & pixels[j]) >> 16;
int green=(0x00ff00 & pixels[j]) >> 8;
int blue=(0x0000ff & pixels[j]);
```

- Pode-se reconstituir o valor RGB:

```
pixels[j]=((red & 0xff)<<16)+
((green & 0xff)<<8)+(blue & 0xff);
```

Exibindo Imagens no ImageJ

- A classe `ImageWindow` é usada para exibir objetos `ImagePlus`
- Normalmente, não é necessário acessar os métodos da classe `ImageWindow`
  - Estes são chamados automaticamente a partir dos métodos ***`show()`***, ***`draw()`*** e ***`updateAndDraw()`*** de um objeto `ImagePlus`

Classes úteis – `Erro`

- O ImageJ contém uma classe chamada `IJ` que contém um conjunto útil de métodos estáticos:
  - Error messages
    - ***`static void error(String message)`*** – exibe uma mensagem de erro
    - ***`static boolean showMessageWithCancel(String title, String message)`*** – mensagem com a opção de cancelar (por exemplo a execução de *plug-in*)

Classes úteis – `Textos`

- Exibindo texto
  - ***`static void write(String s)`*** - Escreve o texto numa janela – útil para exibir resultados de iterações
- Exibindo texto na barra de status
  - ***`static void showStatus(String s)`***

Classes úteis – `Entrada`

- Entrada de dados
  - ***`static double getNumber(String prompt, double default)`*** – Permite que o usuário digite um valor numérico
  - ***`static String getString(String prompt, String default)`*** – Permite que o usuário digite uma String
- A classe ***`GenericDialog`*** possibilita fazer a entrada de múltiplos valores numa única janela.

Regiões de Interesse – ROI

- Um plug-in nem sempre necessita manipular toda a imagem
- ImageJ supporta ROI que podem ser retangular ou de outras formas.
- Pode-se criar ou obter uma ROI usandos os seguintes métodos de uma classe ***ImageProcessor***:
  - ***void setROI(int x, int y, int w int h)***
  - ***Rectangle getROI()***

Escrevendo *Plug-ins* - 1/5

- Para escrever um plug-in, é necessário devolver uma classe que implementa as interfaces ***PlugIn*** ou ***PlugInFilter***
  - A segunda é mais usada, principalmente quando o filtro exige uma imagem de entrada

```
import ij.*;
import ij.plugin.filter.PlugInFilter;
import ij.process;

class myPlugin implements PlugInFilter
```

Escrevendo *Plug-ins* - 2/5

- Os métodos ***setup()*** and ***run()*** devem oferecer:
  - ***setup()*** configura o *plug-in* para uso

int setup(String arg, ImagePlus imp)

  - ***String arg*** permite a entrada de parâmetros passados ao *plug-in*
  - Parâmetro ***imp*** é manipulado automaticamente pelo ImageJ
    - Correspondente à imagem atualmente ativa

Escrevendo *Plug-ins* - 3/5

- O método ***setup()*** returna uma *flag* representando as capacidades do *plug-in* (por examplo os tipos de imagens suportados). Examplos:
  - `static int DOES_8G`
  - `static int DOES_RGB`
  - `static int DOES_ALL`
  - `static int NO_CHANGES` (o plug in não altera os dados da imagem)
  - `static int ROI_REQUIRED`
  - `etc`

Escrevendo *Plug-ins* - 4/5

- O método ***run()*** é chamado quando o *plug-in* é executado a partir do menu do ImageJ
  - Este contém o código para processar os pixels

Caso não seja necessário entrar com uma imagem

void run(String arg)

Caso seja necessário entrar com uma imagem

void run(ImageProcessor ip)

Escrevendo *Plug-ins* - 5/5

- Uma vez que o programa Java foi escrito e compilado, tem-se o arquivo .class e este deve ser colocado na pasta plug-in que é uma sub-pasta da pasta onde o ImageJ está instalado.
  - Este plug-in aparecerá no menu *plug-ins*

Considerações Finais

- **ImageJ é um ambiente útil e flexível para desenvolvimento de algoritmos de processamento de iamgens**
- **Os algoritmos são facilmente incorporados à aplicação na forma de um *plug-in***
- **ImageJ oferece uma facilidade de acesso aos dados de diferentes tipos de imagens**

Bibliografia

Writing ImageJ PlugIns – A Tutorial
http://mtd.fh-hagenberg.at/depot/imaging/imagej/