Insight Toolkit ITK

Prof. Luiz Otavio Murta Jr.
Informática Biomédica
Depto. de Computação e Matemática
(FFCLRP/USP)

Objetivos

- Aprender como usar o Cmake
- Construir ITK
- Ver exemplos que usam ITK

O que é o ITK

- ITK é uma *toolkit*
- Não “faz” nada
- Não se pode “rodar”
- Não existe um itk.exe
- Tipicamente, se usa o ITK em conjunção com outro pacote para dar conta da visualização e da interface gráfica

Então, para quê serve?

- O código ITK é facilmente adicionado a um código C++ existente
- Provê uma variedade de estruturas de dados flexíveis, e modos de processamento e análise para imagens
- Pode-se fazer muito em poucas linhas de códdigo
- Uma vez acostumado, é fácil de usar

Desenvolvimento multiplataforma

- ITK é compilado em muitas combinações de sistemas operacionais e plataformas
- Cada compilador tem seu formato de entrada próprio: Makefiles, workspaces, etc.
- Como coordenar compilações em diferentes plataformas?

A resposta: CMake

- Uma ferramenta transplataforma para gerenciar o processo de compilação
- Simplifica o processo de compilação
  - Auto-configuração
  - Acesso fácil para bibliotecas externas
  - Usado por muitos outros projetos código aberto
- www.cmake.org

O CMake é:

- Gerador de projetos transplataforma
- Frequentemente mais simples que ambientes particular
- Texto como entrada
- Arquivo de projeto como saída
  - Windows: Visual Studio Solution
  - UNIX: Makefile
  - Mac OS X: Makefile ou projeto XCode

Como CMake roda

- Escreva um arquivo CMakeLists.txt descrevendo seu projeto em linguagem CMake
- Rode CMake para gerar um makefile/project/workspace para o seu compilador
- Compile como você normalmente faria

Como CMake roda, cont.

- Não é diferente do processo configure-make que você pode estar acostumado do linux
- Mas… funciona com muitos compiladores
- Os arquivos CMakeLists.txt são facil de fazer controle de revisão

Sintaxe CMakeLists.txt

- Linhas de comentários indicadas por #
- Exemplo simples:

```
#Give this project a name:
PROJECT(cool_demo)
#The command-line executable “demo”
#is built from “demo_code.cxx”
ADD_EXECUTABLE(demo demo_code.cxx)
TARGET_LINK_LIBRARIES(cool_demo ITKCommon)
```

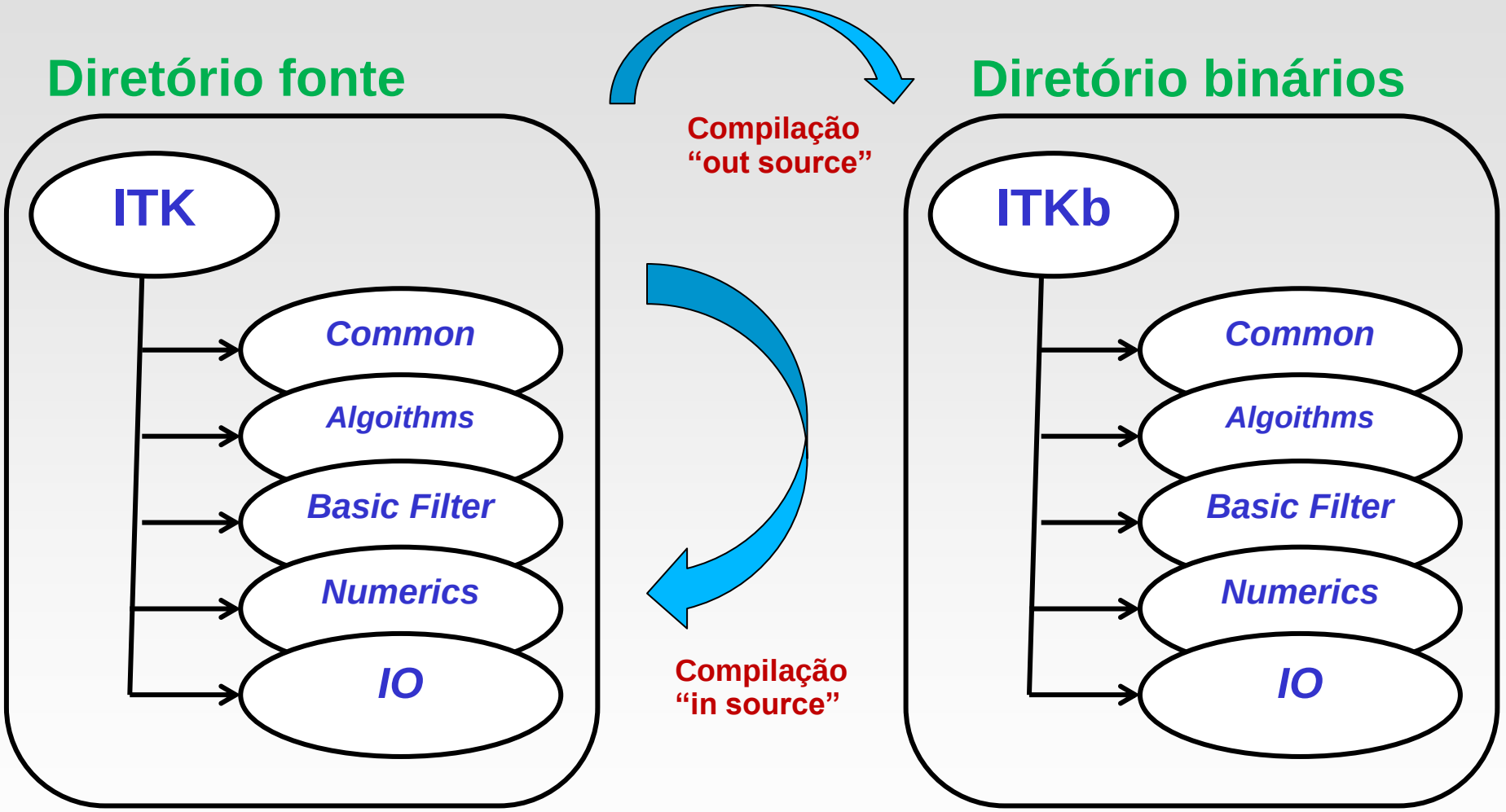
Passo 1 - Instalar Cmake

- Cheque se uma versão recente do CMake já está instalado no seu computador.
- Se não, …
- Baixe e instale uma distribuição do CMake de:
- http://www.cmake.org/

Passo 2 – Instalar o ITK

- Cheque se uma versão recente do ITK já está instalada em seu computador.
- Se não, …
- Baixe a última versão do InsightToolkit:
- http://www.itk.org/ITK/resources/software.html
- Descompacte, por exemplo, InsightToolkit-5.1.1.zip para o seu diretório de trabalho dos arquivos fonte deste tipo

Compilação “in source” vs. “out source”



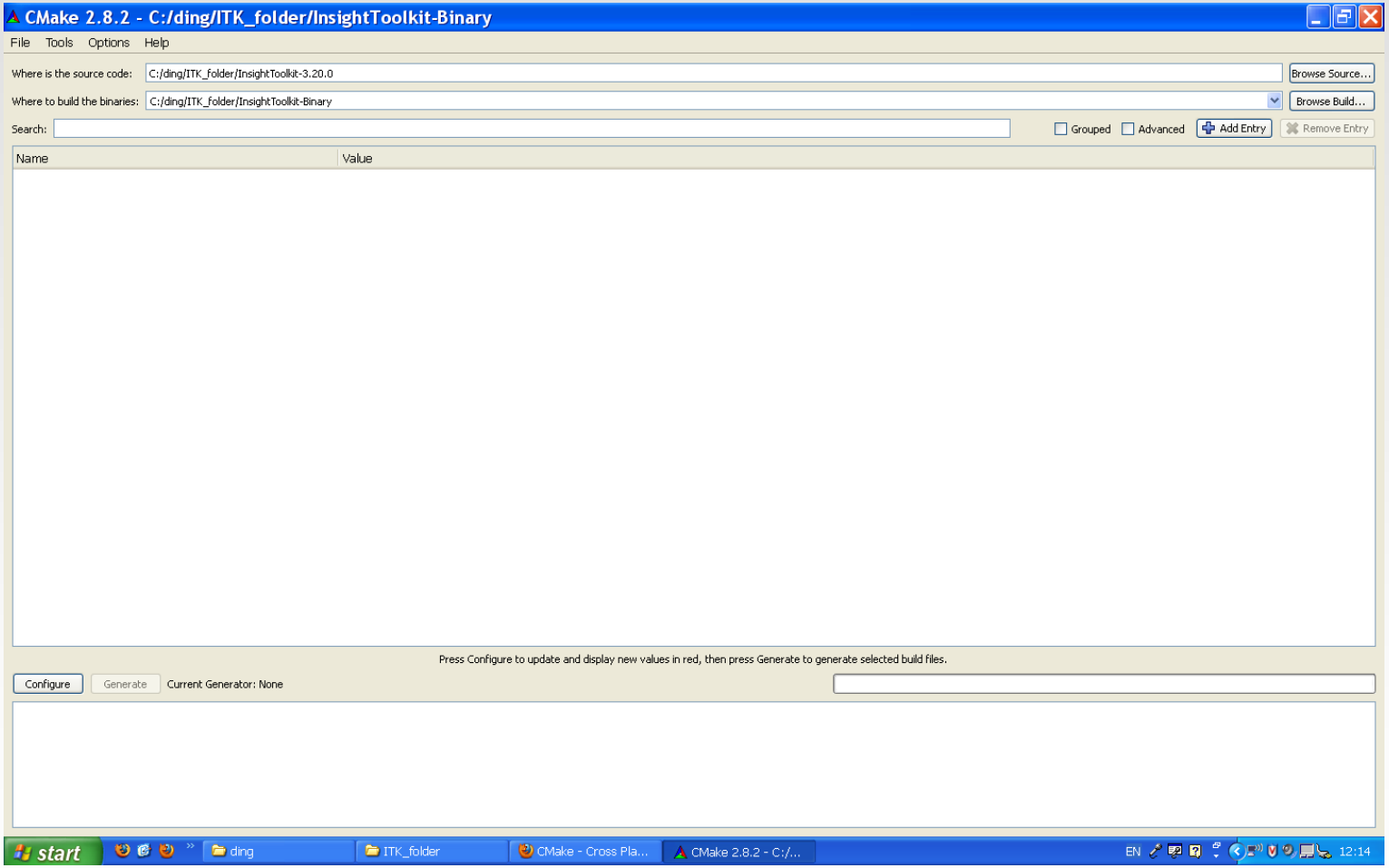
Porquê usar dois diretórios?

- Mantem os seus códigos fonte e binário separados
- Minimiza os danos que você pode fazer
- ITK é encontrado no diretório InsightToolkit-5.1.1
- Sugerimos que você compile em um novo diretório chamado InsightBin

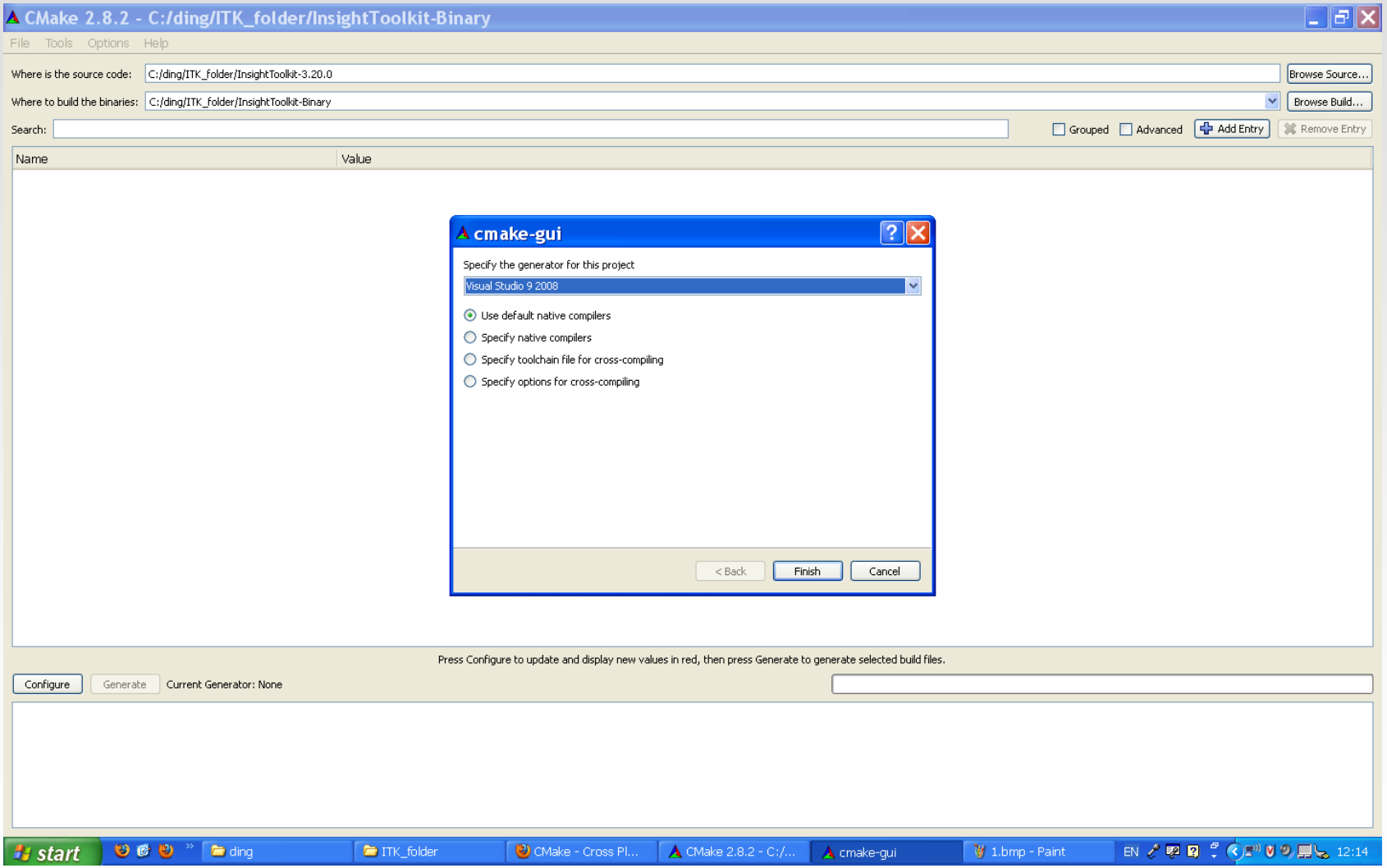
Configure – começo fácil

- Rode Cmake
- Selecione o diretório fonte (SOURCE)
- Selecione o diretório binário (BINARY)

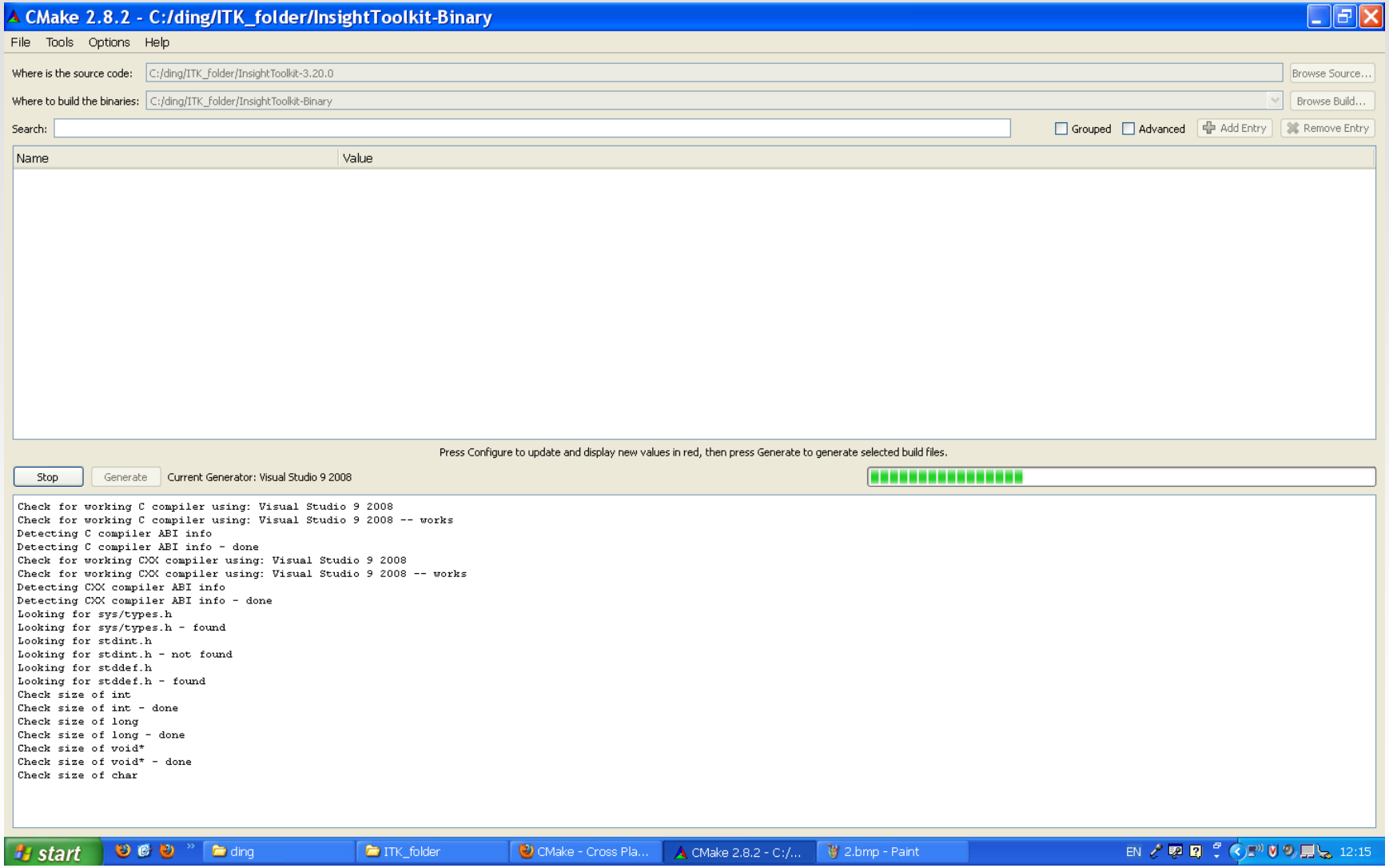
Configure, cont.



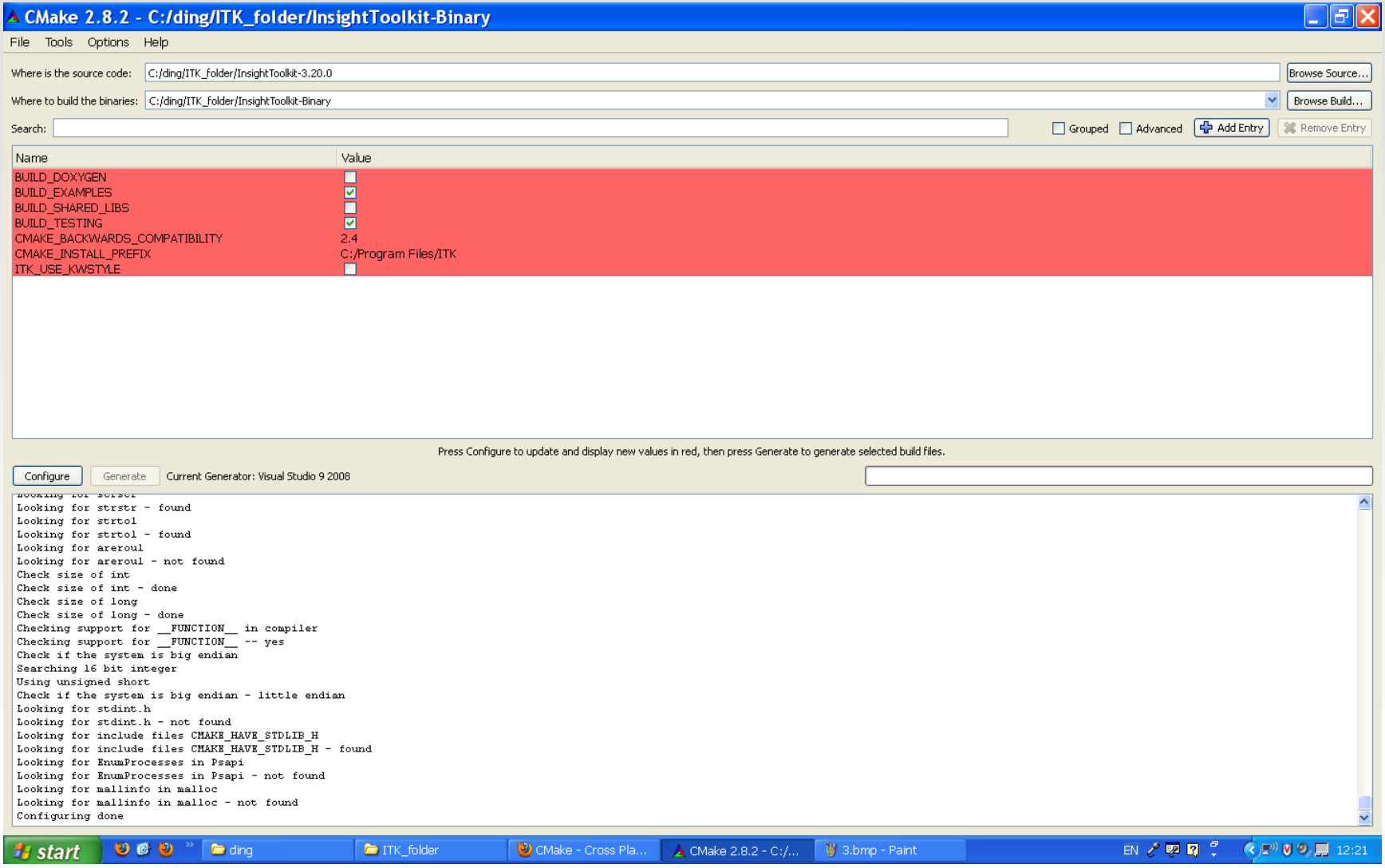
Configure - começo fácil, cont.



Configure - começo fácil, cont.



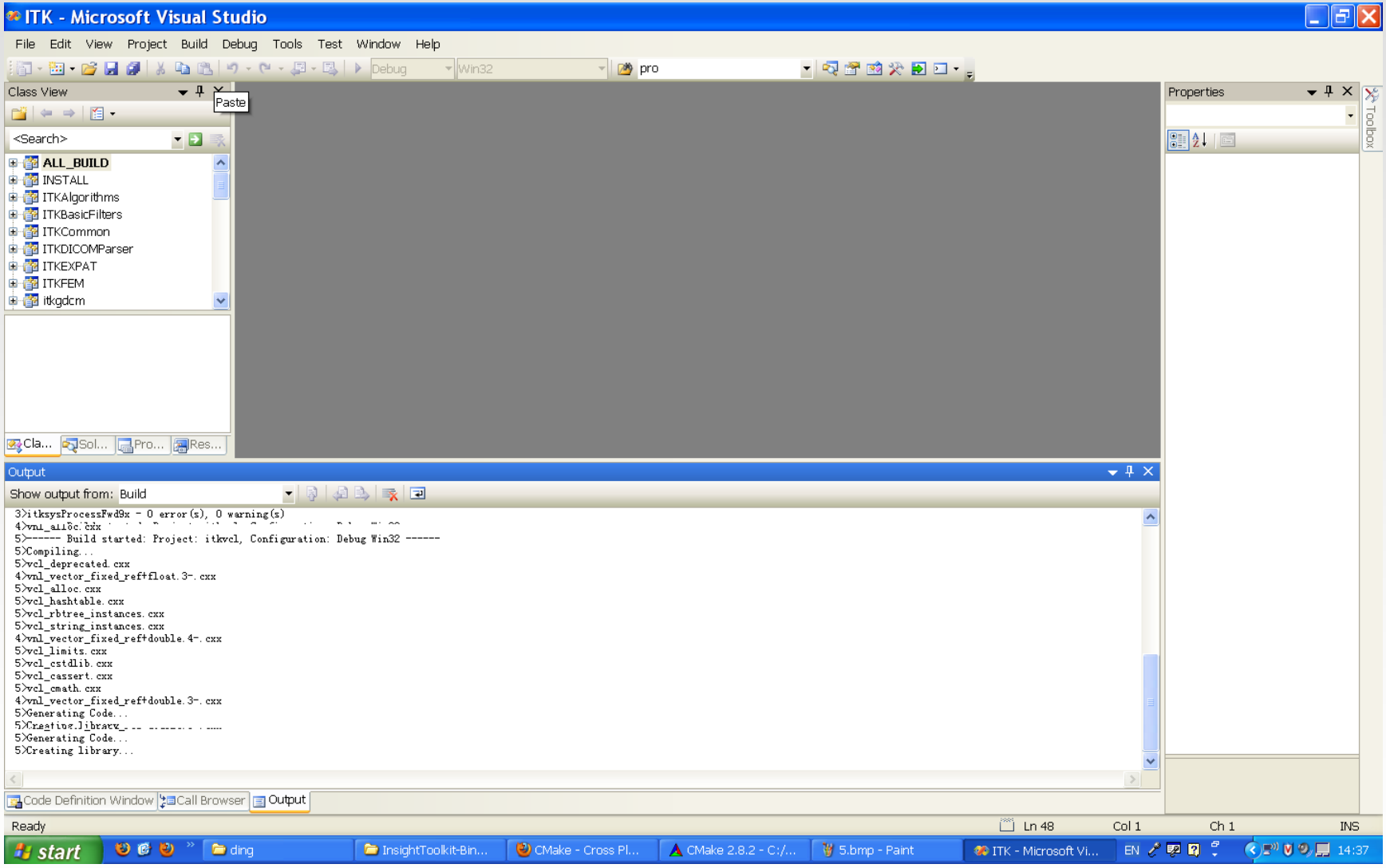
Configure, cont.



Configure, cont.

- Disabilite BUILD_EXAMPLES
- Disabilite BUILD_TESTS
- Disabilite BUILD_SHARED_LIBS

Configure - Easy Start, cont.



Configurando e gerando

- Todas as vezes que você muda uma opção, você precisará confirgurar (“configure”) de novo.
- Se a opção “generate” (“OK” no Windows) não estiver presente, você precisará rodar “configure” novamente.
- Se alguma opção estiver em vermelho, você precisará reconfigurar

Compile o ITK

- Abra o arquivo ITK Visual Studio Solution no diretório de binários
- Selecione Build→Build Solution
- Levará provavelmente algo em torno de 20 – 40 minutos, mas o tempo pode variar

Verifique a compilação

- As bibliotecas estarão em:
- ITK_BINARY / bin / { Debug, Release}

Compilando com gcc

- A ordem das operações será a mesma
- Diferenças:
  - Rode o executável ccmake, o qual usa GUI, as opções são idênticas
  - Rode make no lugar do Visual Studio
- Pense no CMake como substituição ao passo "./configure" que você deve estar acostumado

Complilando com gcc cont.

- Comece no diretório *contendo* InsightToolkit-4.9.0
- mkdir InsightBin
- cd InsightBin
- ccmake ../InsightToolkit-5.1.1
- Edite as opções CMake
- Reconfigure se necessário
- make

E agora?

- Neste ponto, você deve ter duas coisas:
- Um diretório contendo uma versão do código fonte (por.ex. ~/InsightToolkit-5.1.1/)
- Um diretório contendo as bibliotecas ITK compiladas (e.g. ~/InsightBin)
- Como já mencionado, você não tem nenhum executável

Compilando um aplicativo

- ITK vem com um aplicativo simples que você pode compilar para testas a biblioteca ITK “fora do fonte” (ou seja, compilado fora do ITK)
- Pode ser encontrado em:

  InsightToolkit-5.1.1./Examples/Installation

Como compilar o HelloWorld

- Copie & renomeie o *diretório de Instalação* em algum lugar fora do diretório do Insight
- Rode o CMake no *HelloWorld*
- Lembre distinção entre fonte e binário, e use *HelloWorldBin* como seu local de compilação
- O CMake deve achar o ITK automaticamente
  - Senão, edite a opção ITK_DIR

Como compilar HelloWorld, cont.

- Uma vez que o CMake está satisfeito, gere o makefile/project para o seu compilador
- Compile HelloWorld
- Tente ...

Mais exemplos

- Você pode ativar a opção *Examples* no CMake, o qual irá compilar todos os exemplos
- Ou… você pode copiar os exemplos fora, e compilá-lo como você fez com HelloWorld
- Estes exemplos tem ligações no “ITK SoftwareGuide”; leia o capítulo altere o código e veja o que acontece…

Sequência

- Você deve se acostumar com a ideia de:

1. Escrever um código
2. Escrever um arquivo CMakeLists.txt
3. Rodar o CMake
4. Compilar o seu código
5. Debugar, corrigir, e repetir

Uma nota: como usar o ITK com aplicativos existentes

- Seu aplicativo provavelmente não usa Cmake
- Neste cado, você precisa ligar a biblioteca ITK explicitamente e incluir os diretórios fontes apropriados
- Isto não é difícil, mas pode levar algumas tentativas e erros para descobrir tudo o que você precisa

Documentação ITK

- A maioria da documentação do ITK é gerada automaticamente a partir dos comentários no código fonte usando Doxygen
- Familiarize-se com ela navegando na documentação Doxygen online:

  http://www.itk.org/Doxygen/html/index.html

O básico em ITK

- Programação genérica em C++
- Fluxo de dados
- Multi-threading
- Streaming
- Exceções
- Eventos / Observadores
- Tcl, Python and Java wrapping

Programação Genérica

Exemplo: **STL** Standard Template Library

Abstração de **Tipos** e **Comportamentos**

```
std::vector< T >

std::vector< int >
std::vector< double >
std::vector< char * >
std::vector< Point >
std::vector< Image >
```

itk::Image

```
itk::Image< PixelType , Dimension >

itk::Image< char , 2 >
itk::Image< char , 3  >
itk::Image< char , 4 >
itk::Image< float , 2 >
itk::Image< RGB , 3 >
itk::Image< unsigned short , 2 >
itk::Image< itk::Vector<float,2> , 2 >
```

namespaces

Evite coincidência de nomes

```
itk::
itk::Statistics::
itk::fem::
itk::fem::itpack
itk::bio
```

Palavra-chave favorita

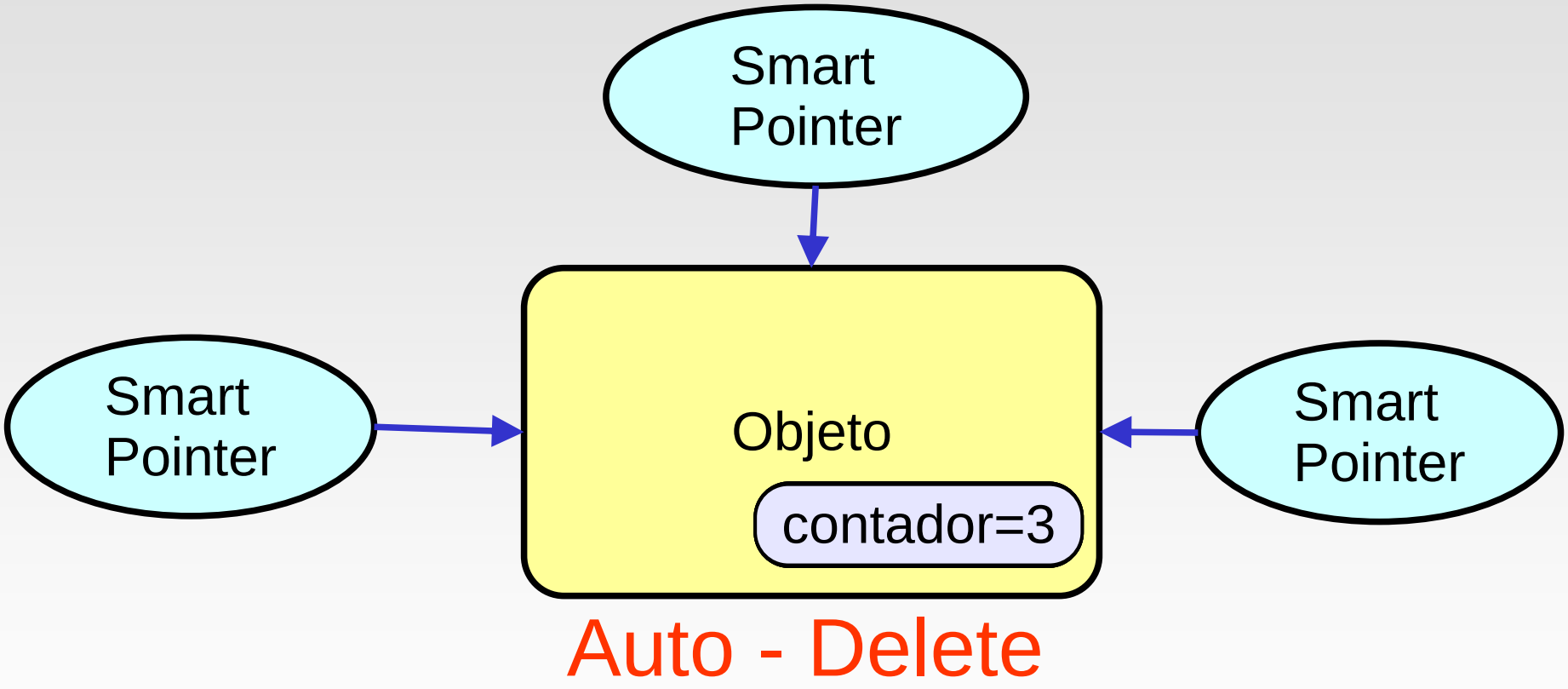
typedef

```
typedef   itk::Image< char , 2 >   ImageType
typedef   itk::ImageFilter< ImageType , ImageType >   FilterType
```

senão...

```
itk::ImageFilter< Image< char , 2 > ,
                  Image< char , 2 > >   FilterType
```

Ponteiros Inteligantes



SmartPointers

```
typedef   itk::Image< char , 2 >   ImageType
typedef   itk::ImageFilter< ImageType , ImageType >   FilterType

    FilterType::Pointer  filter = FilterType::New();

    ImageType::Pointer image = filter->GetOutput();
```

Notação de ponteiro:

```
filter->Update();
```

NÃO PRECISA:

```
filter->Delete();
```

Uso correto de Const

Reconhecer constantes é *Insight.*

Não reconhecer constantes leva ao desastre.

Tao Te Ching, XVI. Lao Tsu

Ponteiros Inteligentes Const

```
typedef   itk::Image< char , 2 >   ImageType
typedef   itk::ImageFilter< ImageType , ImageType >   FilterType

    FilterType::Pointer  filter = FilterType::New();

    ImageType::ConstPointer image = filter->GetOutput();
```

Pode invocar apenas métodos “const”

```
image->GetSpacing ();
```

Erro de compilação para métodos “non-const”

```
image->SetSpacing ( spacing );
```

Criando uma Imagem

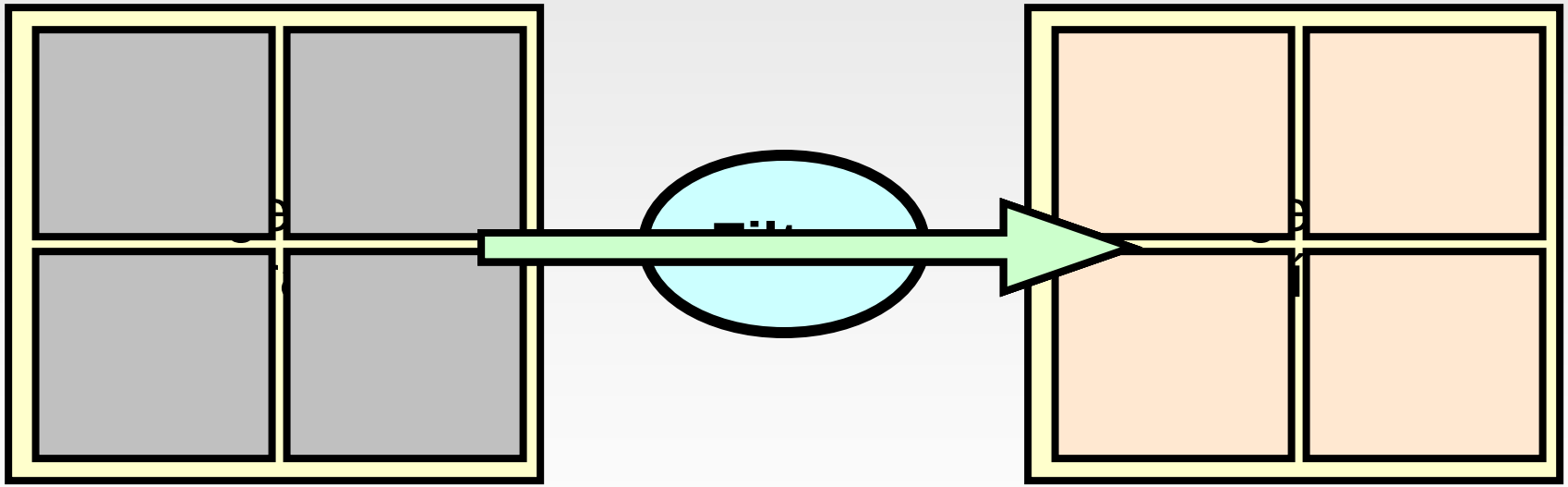
```
typedef   itk::Image< char , 3 >   ImageType

ImageType::Pointer image = ImageType::New();

ImageType::SizeType size;
size[ 0 ] = 512;   // x direction
size[ 1 ] = 512;   // y direction
size[ 2 ] =   50;   // z direction

ImageType::IndexType start;
start[ 0 ] = 0;      // x direction
start[ 1 ] = 0;      // y direction
start[ 2 ] = 0;      // z direction
```

Criando uma Imagem

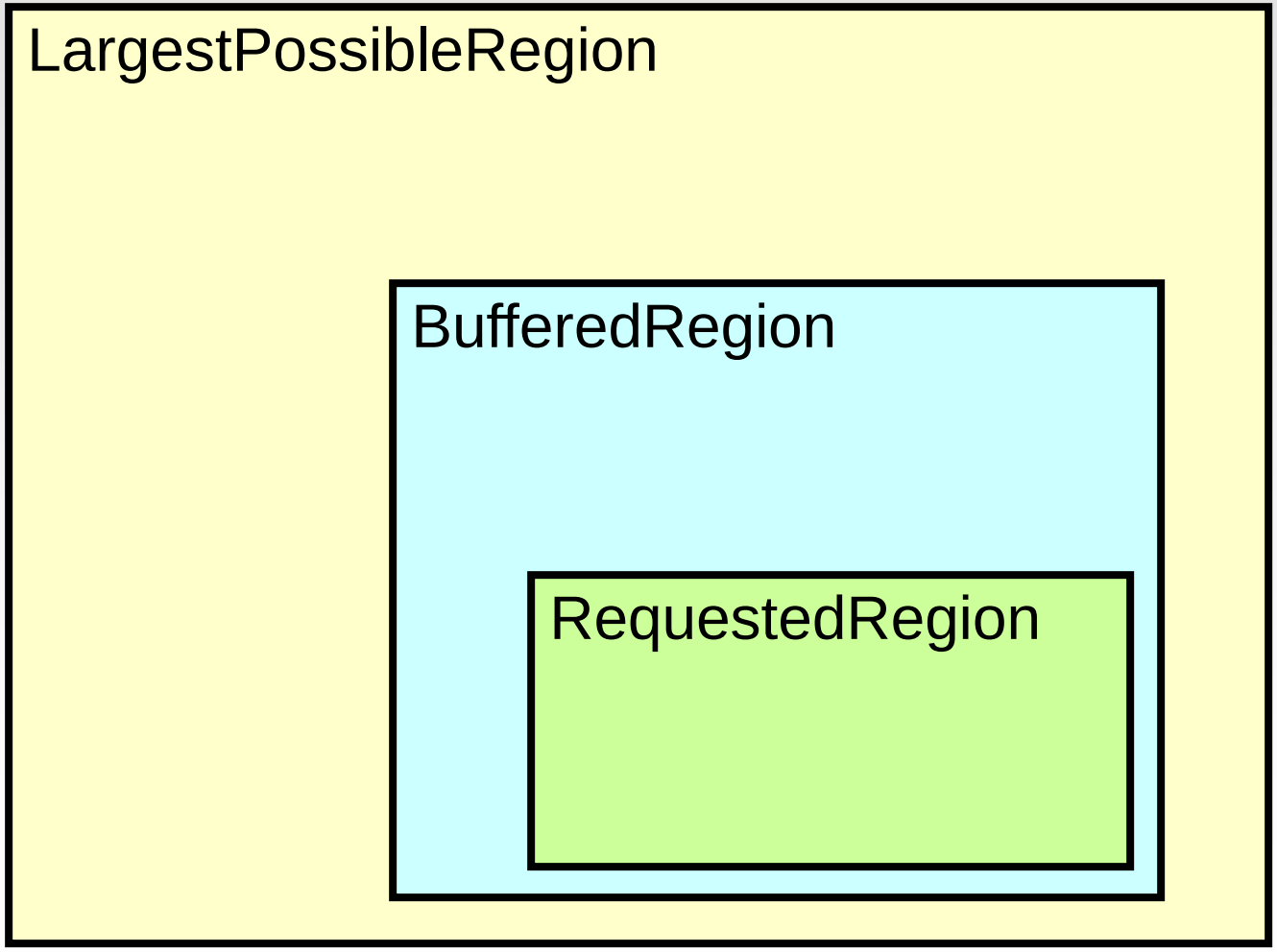
```
ImageType::RegionType region;
region.SetSize( size );
region.SetIndex( start );

image->SetRegions( region );
image->Allocate();
image->FillBuffer( 0 );

ImageType::SpacingType spacing;
spacing[ 0 ] = 0.83;        // x direction
spacing[ 1 ] = 0.83;        // y direction
spacing[ 2 ] = 2.15;        // z direction

image->SetSpacing( spacing );
```

“Streaming”

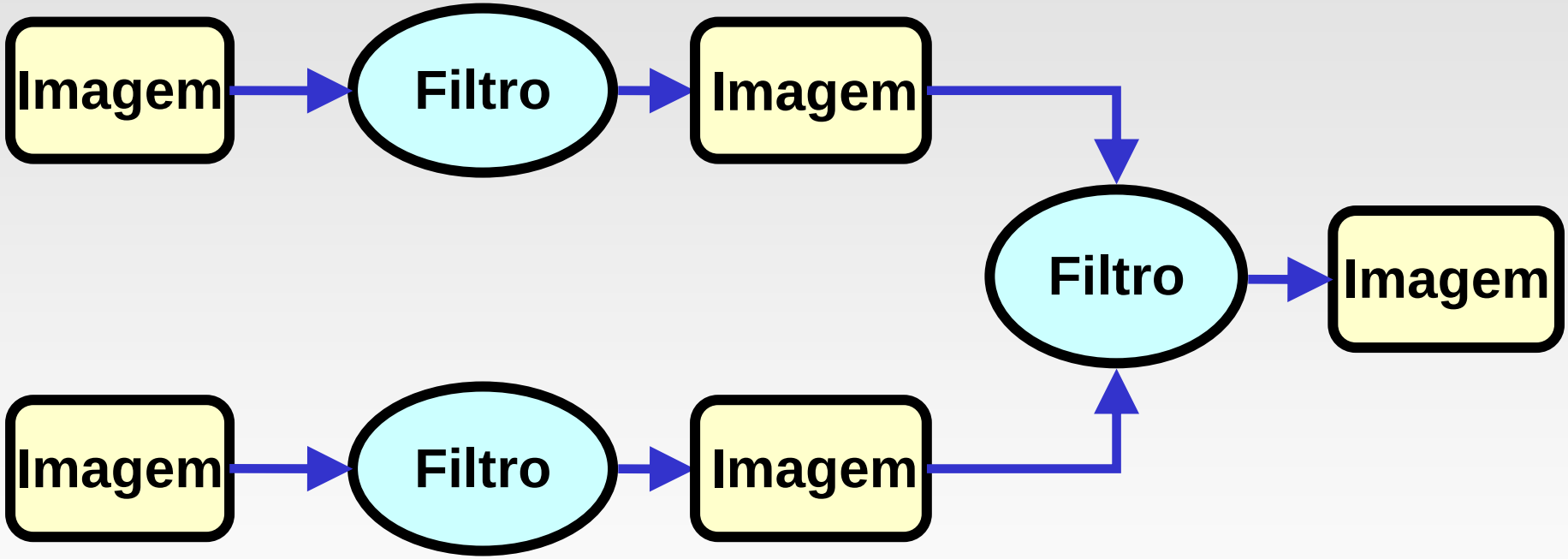
Processando Imagens Grandes



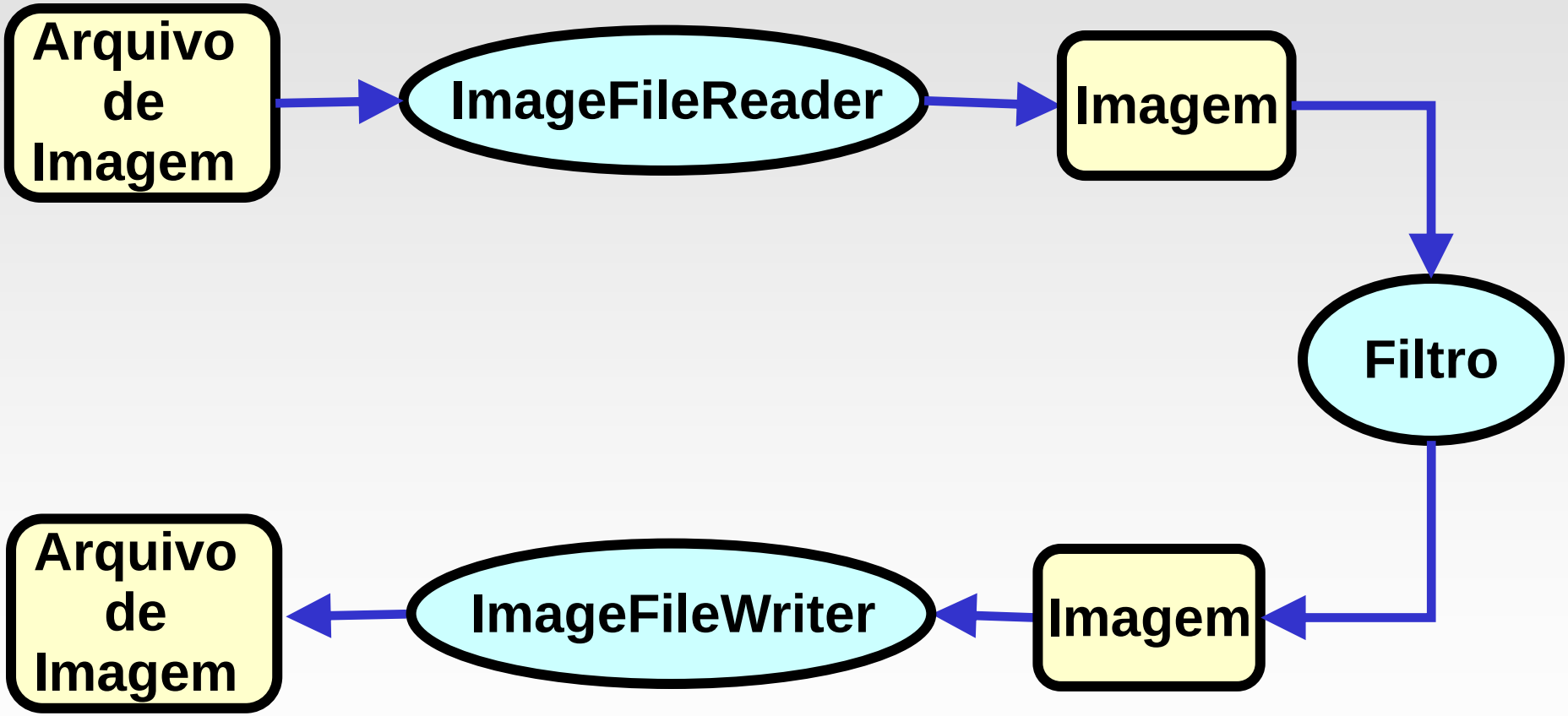
Regiões em Imagens



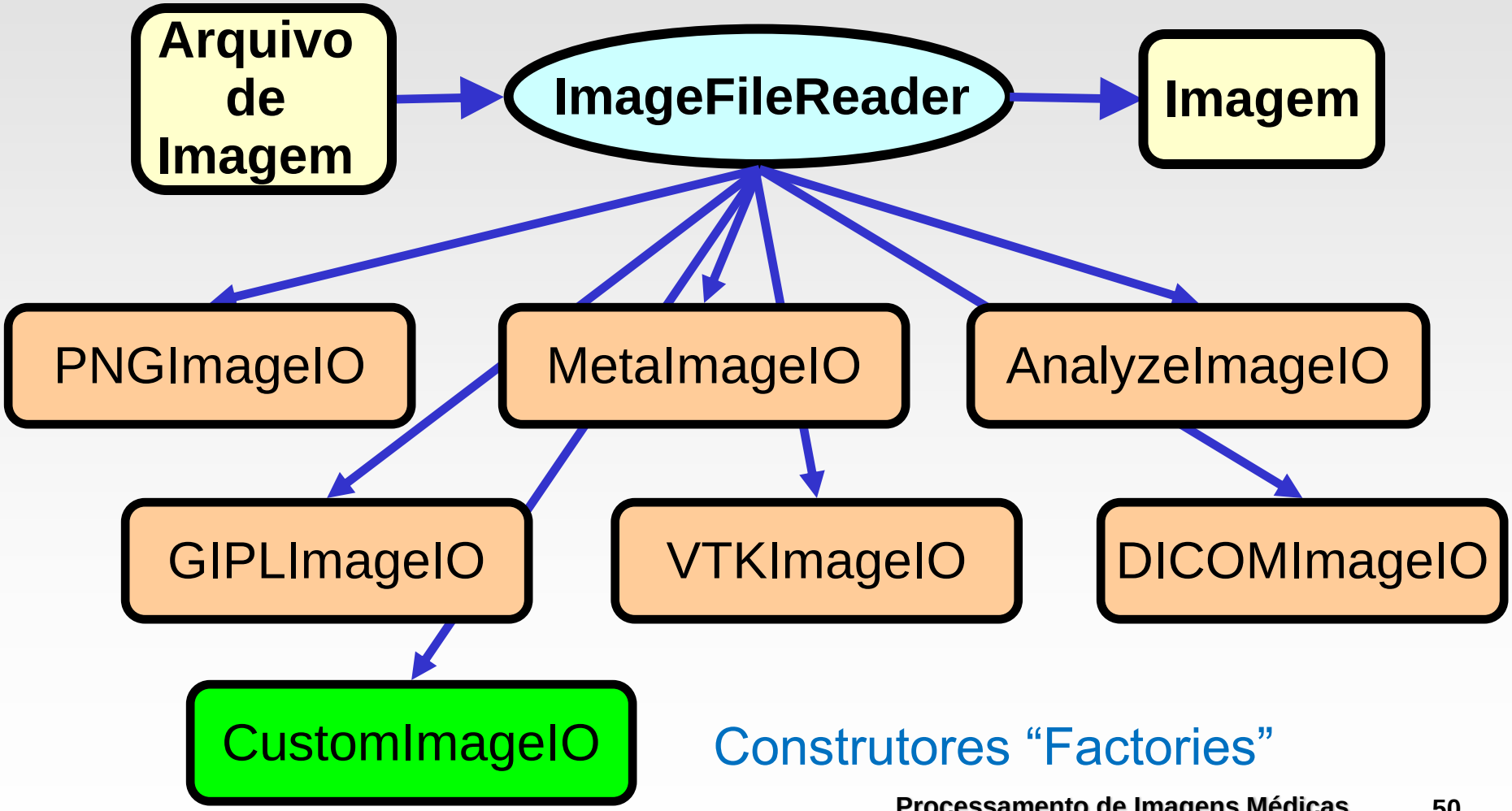
Fluxo de dados



I/O de Imagens Simples



I/O de Imagens Simples



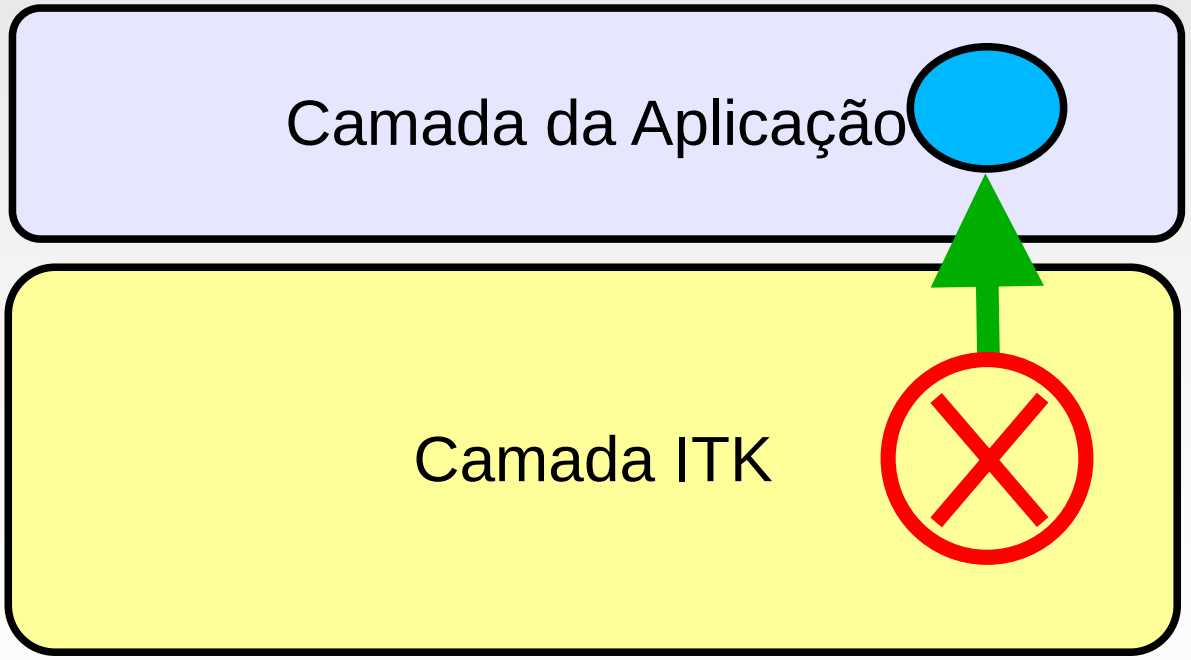
Construtores “Factories”

I/O de Imagens Simples

```
#include “itkImage.h”
#include “itkImageFileReader.h”
#include “itkImageFileWriter.h”

typedef   itk::Image< char , 2 >   ImageType;
typedef   itk::ImageFileReader< ImageType >   ReaderType;
typedef   itk::ImageFileWriter<   ImageType >   WriterType;

ReaderType::Pointer reader = ReaderType::New();
WriterType::Pointer    writer  = WriterType::New();

reader->SetFileName( “inputImage.dcm” );       // DICOM
writer->SetFileName( “outputImage.hdr” );      // Analyze

writer->SetInput(   reader->GetOutput()   );
writer->Update();
```

Exceções

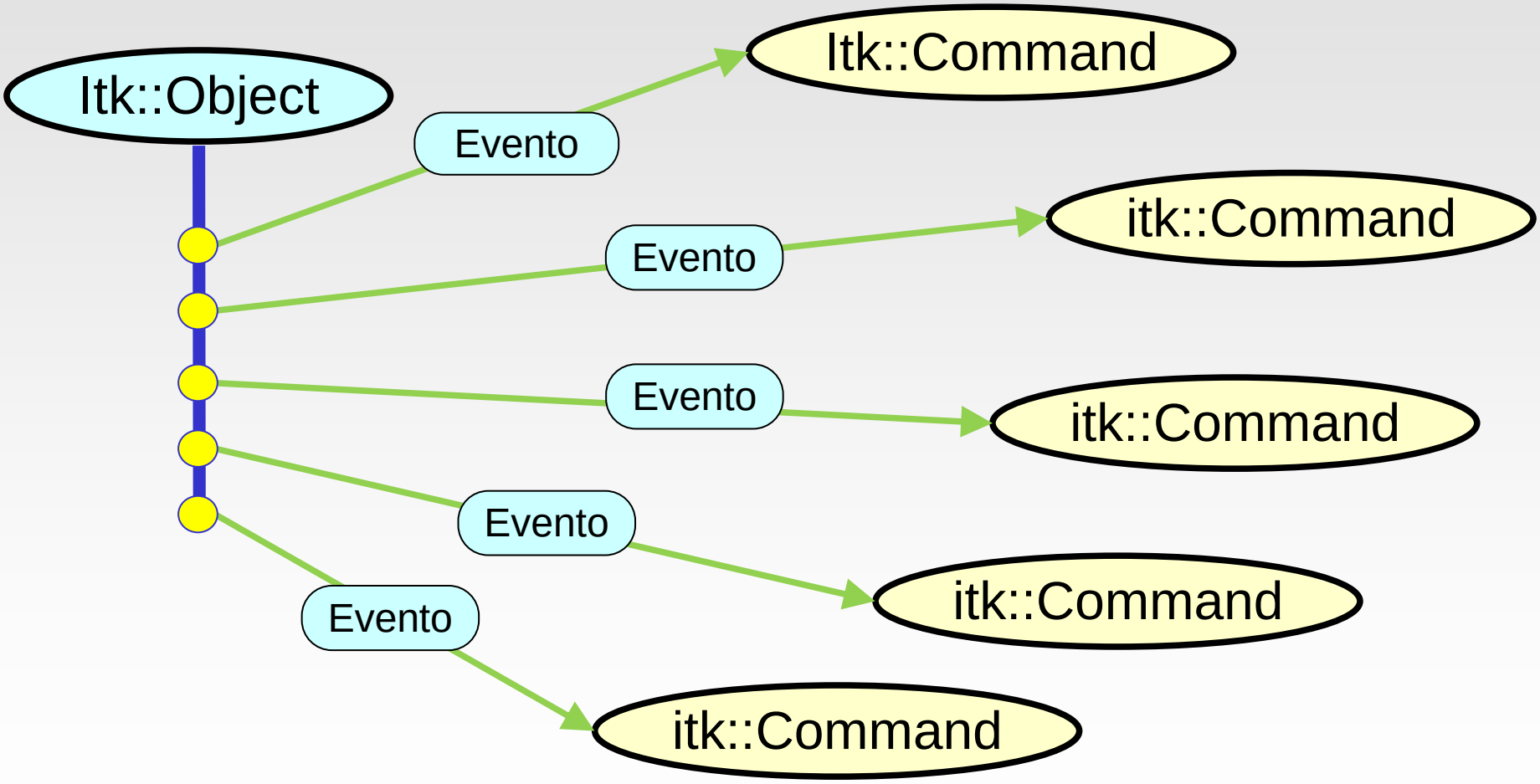
Gerenciamento de Erros



Exceções

```
try
    {
        filter->Update();
    }
catch(  itk::ExceptionObject & exp )
   {
        std::cerr << exp << std::endl;
   }
```

Eventos e Observadores



Eventos e Observadores

Eventos Comuns

AnyEvent()

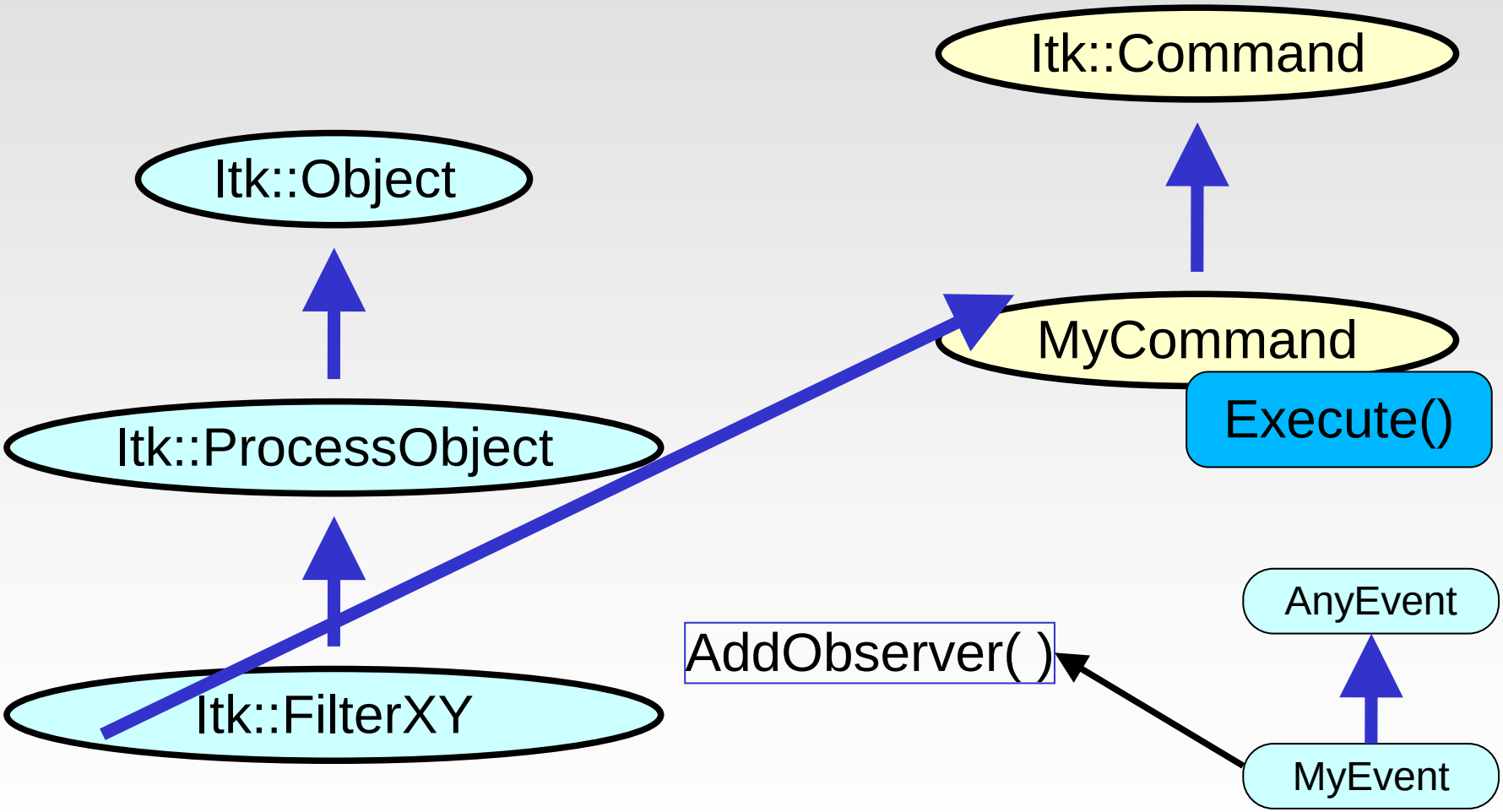
StartEvent()

EndEvent()

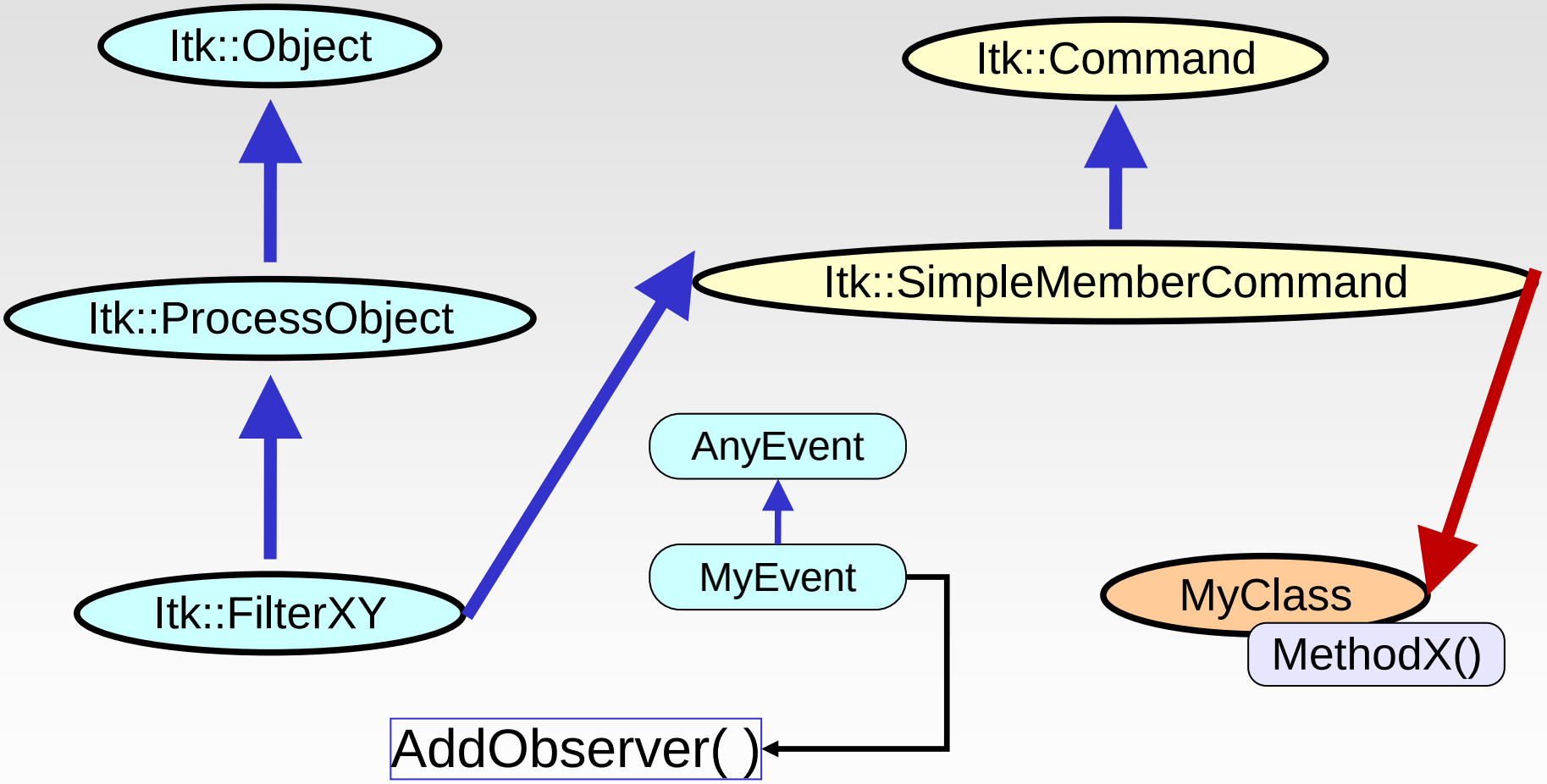
ProgressEvent()

IterationEvent()

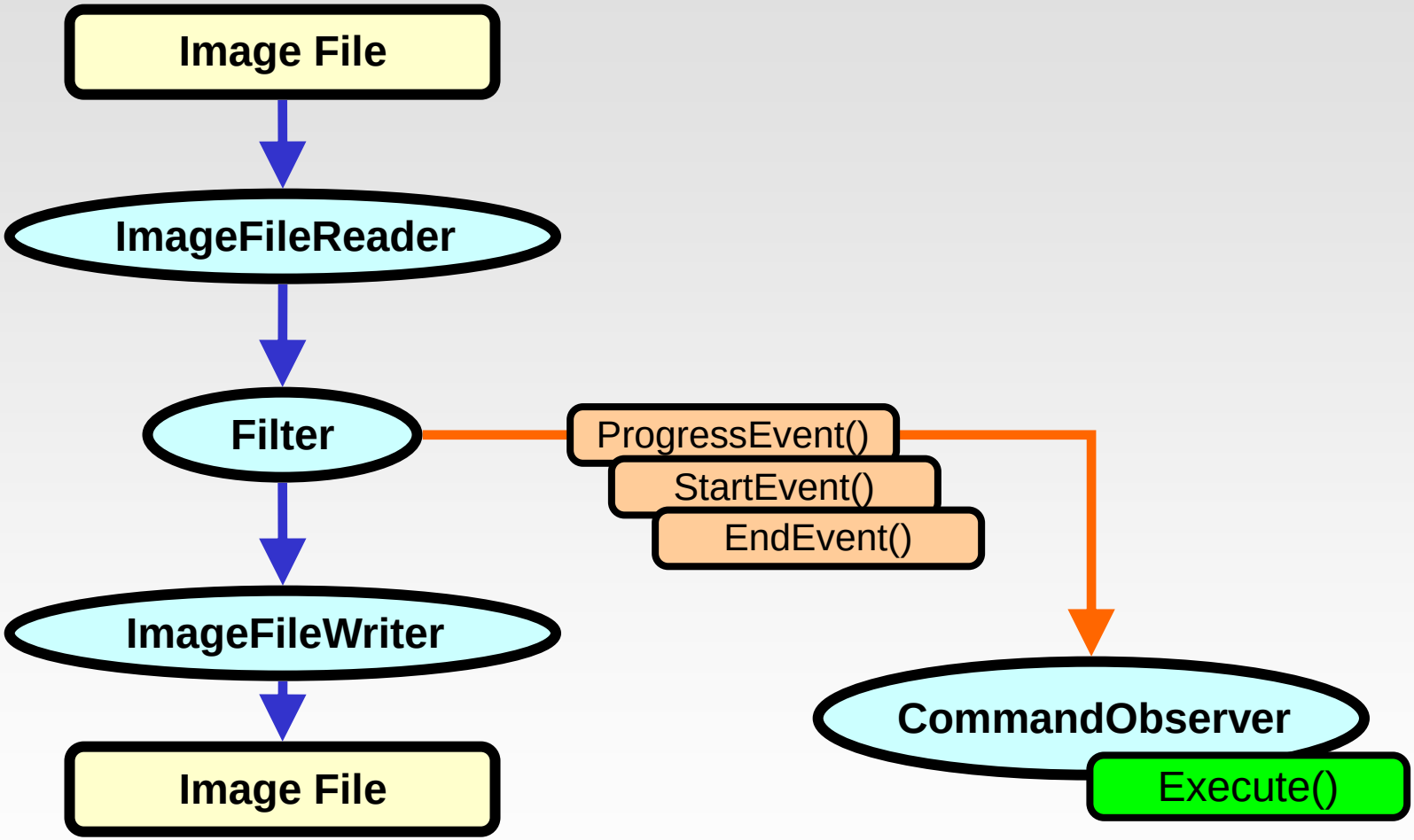
Eventos e Observadores



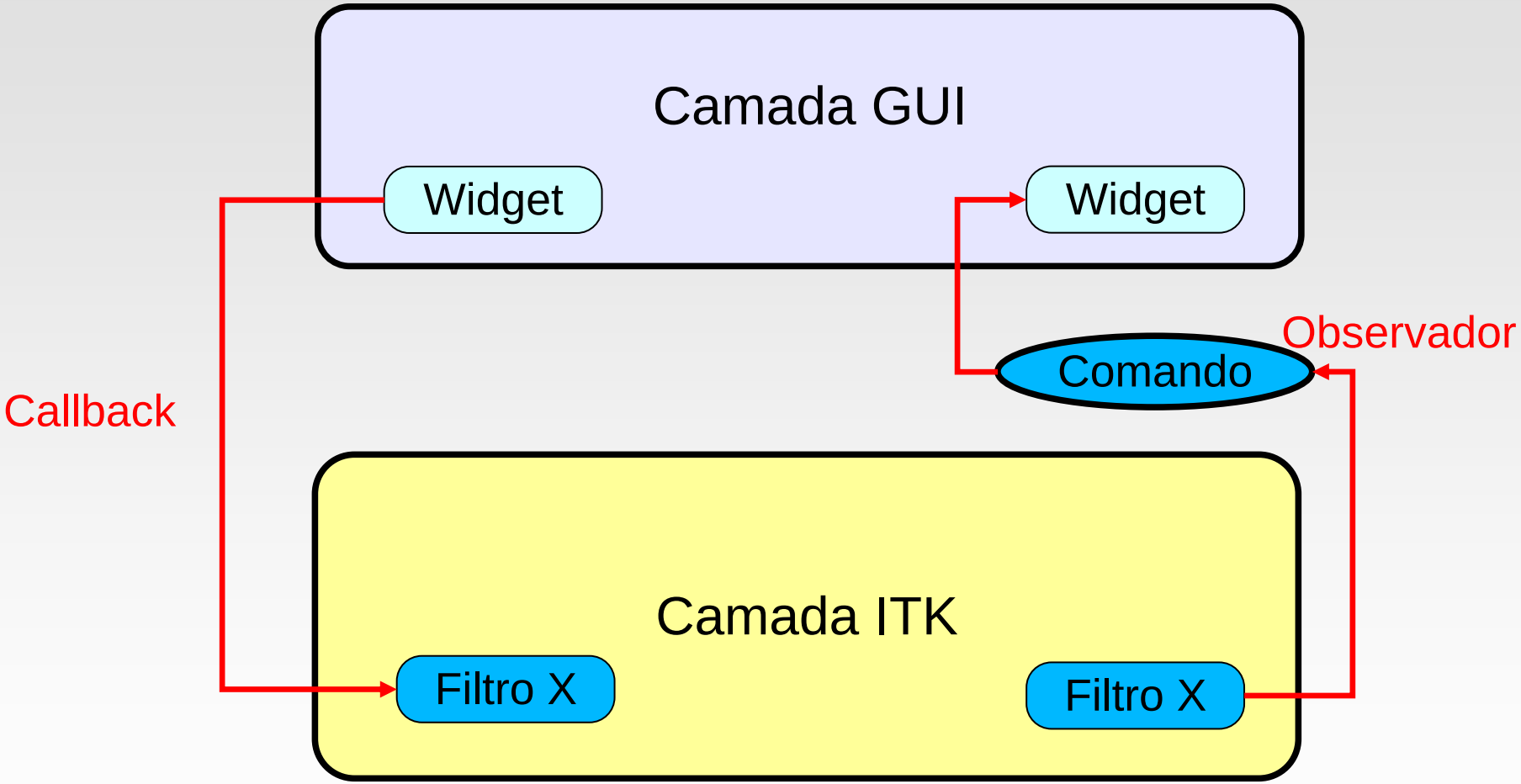
Eventos e Observadores



Eventos e Observadores



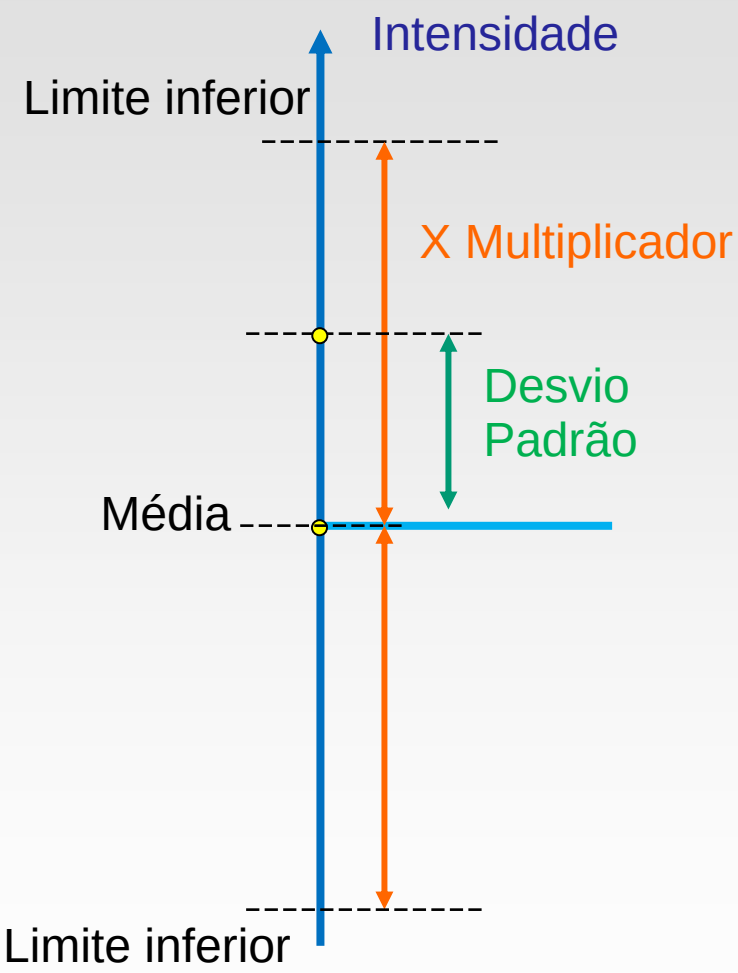
Communicação com GUI

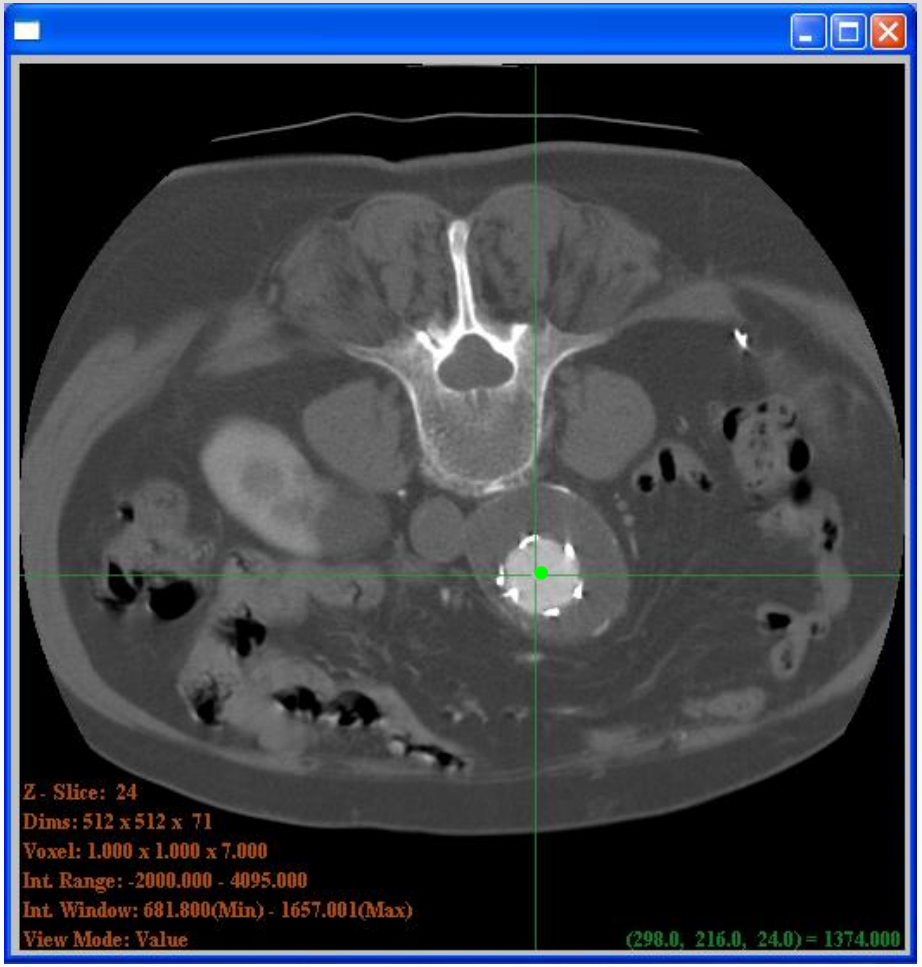


Segmentação

- **Crescimento de Regiões**
  - ConfidenceConnected
  - ConnectedThreshold
  - IsolatedConnected
- **“Watersheds”**
- **“Level Sets”**
  - FastMarching
  - ShapeDetection
  - GeodesicActiveContours
  - ThresholdSegmentation
  - CannySegmentationLevelSet

Conexão de Confiança



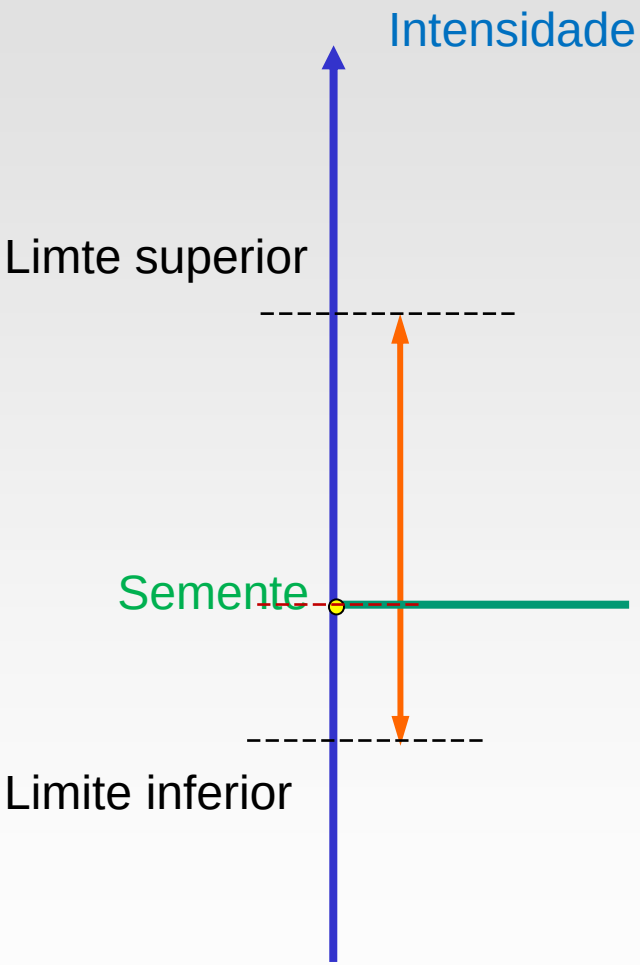


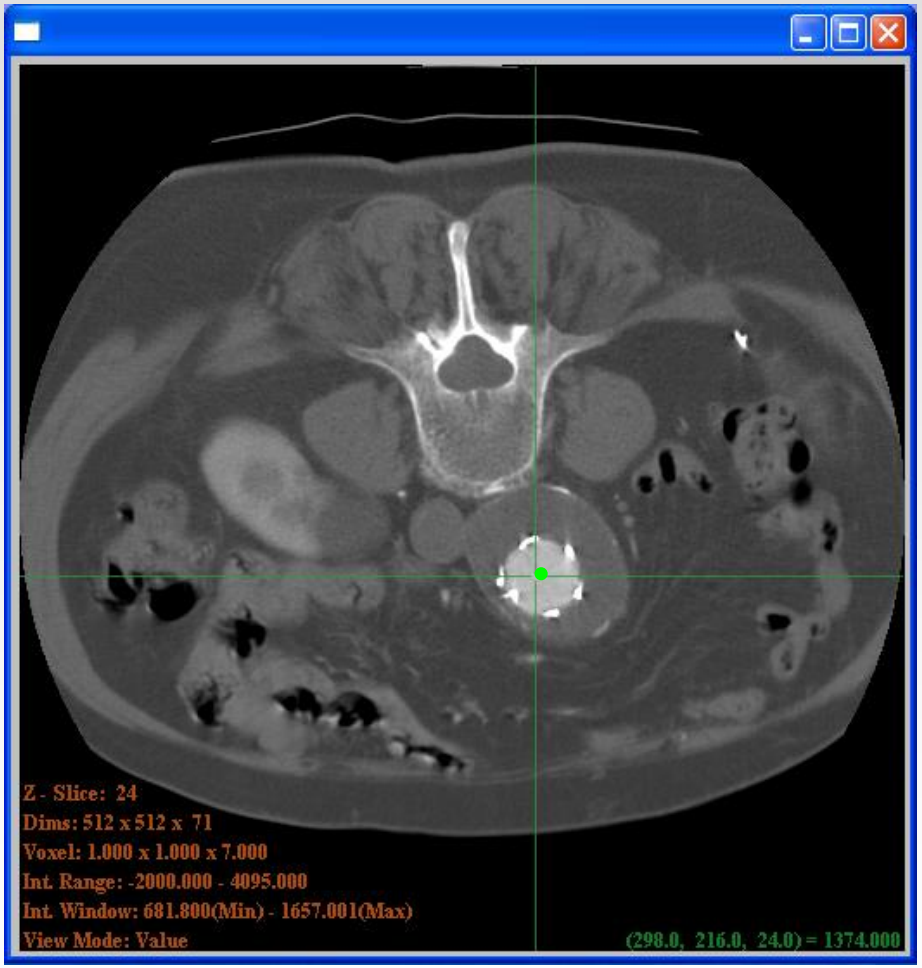
Ponto semente

Conexão de confiança

```
typedef   itk::Image< unsigned char , 2 >   ImageType;
typedef   itk::ConfidenceConnectedImageFilter<
                                ImageType, ImageType >   FilterType;

FilterType::Pointer filter = FilterType::New();
filter->SetMultiplier( 1.5 );
filter->SetNumberOfIterations( 5 );
filter->SetInitialNeighborhoodRadius ( 2 );
filter->SetReplaceValue( 255 );
FilterType::IndexType index;
index[0] = 123;  index[1] = 235;
filter->SetSeed( index );

filter->SetInput(  reader->GetOutput()  );
writer->SetInput(  filter->GetOutput()  );
writer->Update()
```

Limiar conectado



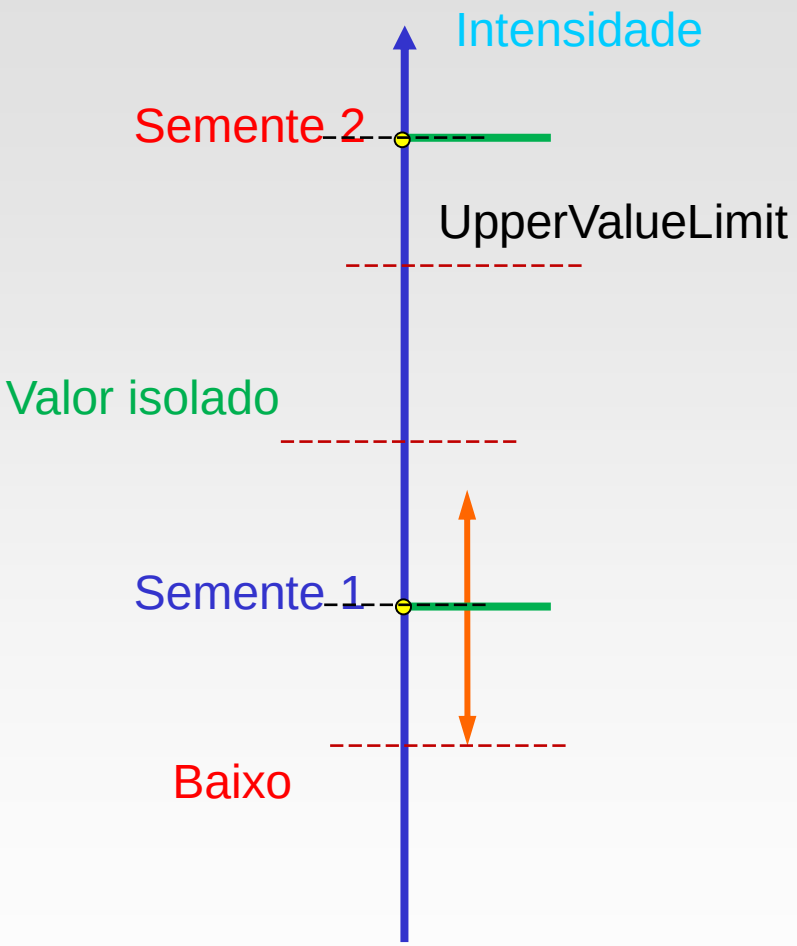


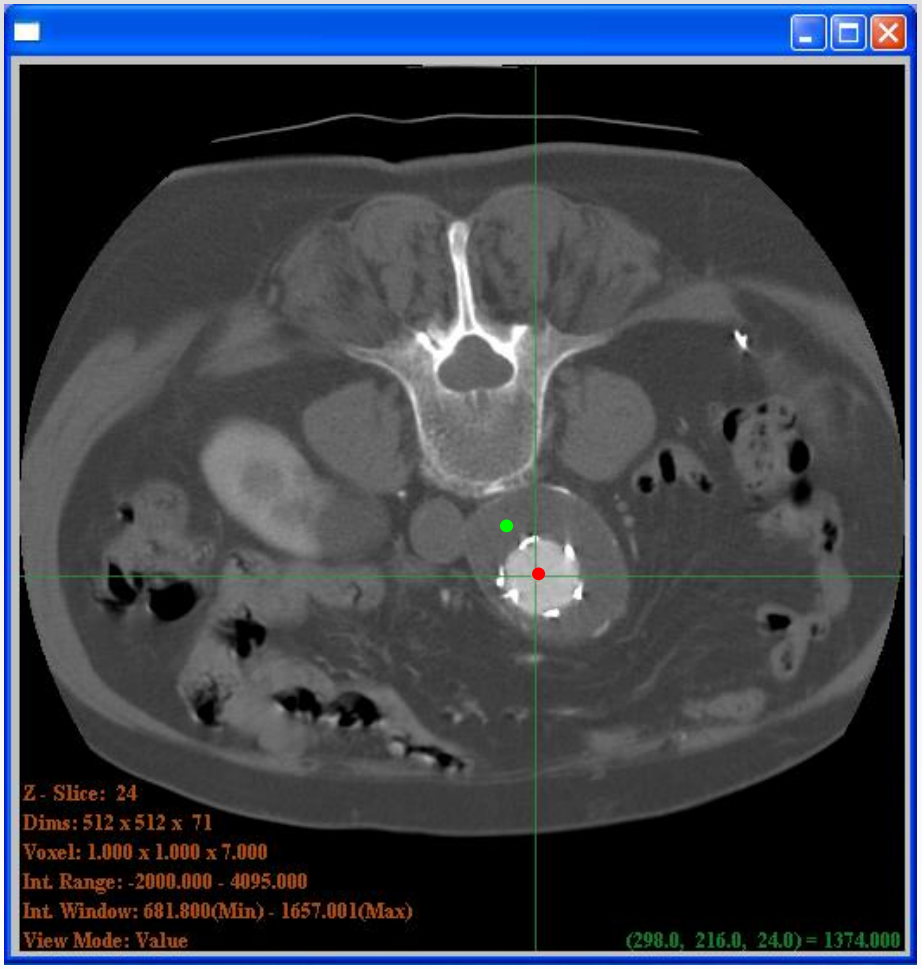
Ponto semente

Limiar conectado

```
typedef   itk::Image< unsigned char , 2 >   ImageType;
typedef   itk::ConnectedThresholdImageFilter<
                                  ImageType, ImageType >   FilterType;

FilterType::Pointer filter = FilterType::New();
filter->SetLower( 155 );
filter->SetUpper( 235 );

filter->SetReplaceValue( 255 );
FilterType::IndexType index;
index[0] = 123;  index[1] = 235;
filter->SetSeed( index );

filter->SetInput(  reader->GetOutput()  );
writer->SetInput(  filter->GetOutput()  );
writer->Update()
```

Isolado Conectado





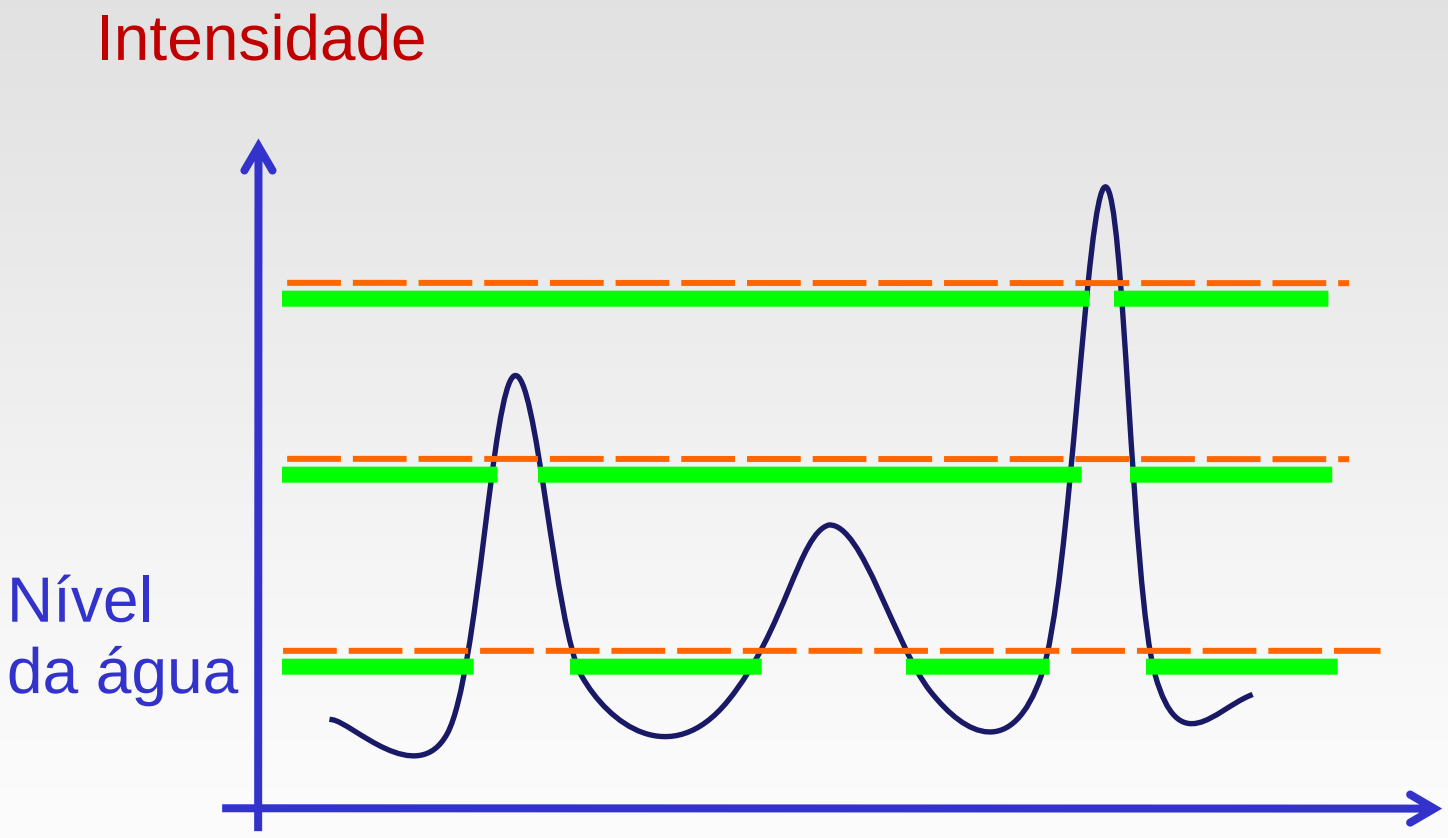
2 pontos semente

Isolado Conectado

```
typedef   itk::Image< unsigned char , 2 >   ImageType;
typedef   itk::IsolatedConnectedImageFilter<
                                  ImageType, ImageType >   FilterType;

FilterType::Pointer filter = FilterType::New();
filter->SetLower( 155 );
filter->SetUpperValueLimit( 235 );

filter->SetReplaceValue( 255 );

filter->SetSeed1( index1 );
filter->SetSeed2( index2 );

filter->SetInput(  reader->GetOutput()  );
writer->SetInput(  filter->GetOutput()  );
writer->Update()
```

Segmentação Watershed

Conceito Watershed



Segmentação Watershed

```
typedef   itk::Image< float , 2 >   ImageType;
typedef   itk::WatershedImageFilter<
                                    ImageType
                                          >   WatershedFilterType;

WatershedFilterType::Pointer filter = WatershedFilterType::New();

filter->SetThreshold( 0.001 );
filter->SetLevel( 0.15 );

filter->SetInput(  reader->GetOutput()  );
filter->Update()
```

Codificando a Saída em Cores

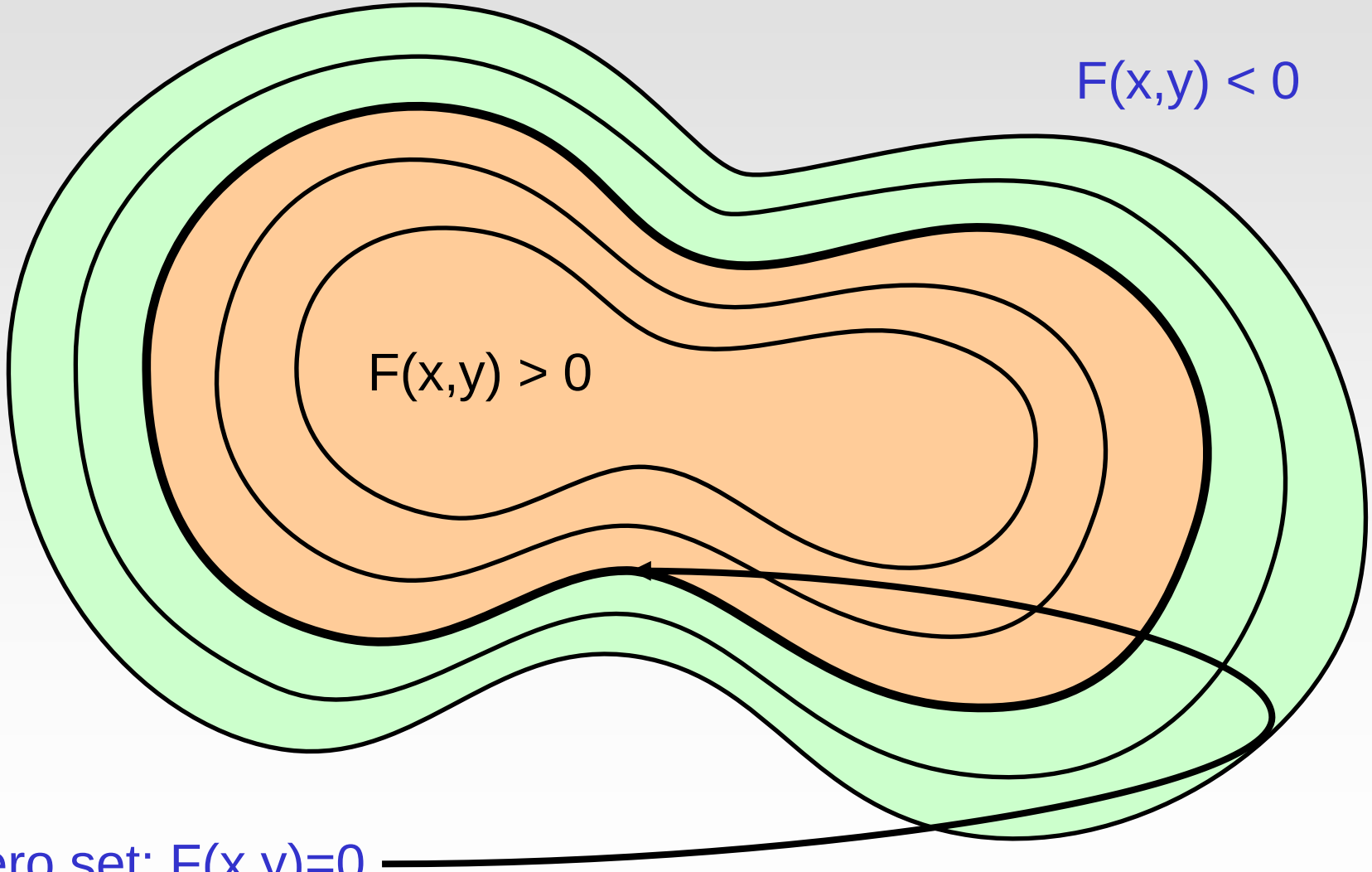
```
typedef   itk::ScalarToRGBPixelFunctor< unsigned long >   FunctorType;
typedef   WatershedFilterType::OutputImageType    LabeledImageType;
typedef   itk::UnaryFunctorImageFilter<         ImageType,
                                                LabeledImageType,
                                                FunctorType
                                                >   ColorFilterType;

ColorFilterType::Pointer colorFilter = ColorFilterType::New();

colorFilter->SetInput(  filter->GetOutput()  );
writer->SetInput(  colorFilter->GetOutput()  );

writer->Update()
```

Criando as Bordas

```
typedef   itk::GradientMagnitudeRecursiveGaussianImageFilter<
                                ImageType, ImageType     >   EdgeFilterType;

EdgeFilterType::Pointer edgeFilter = EdgeFilterType::New();

edgeFilter->SetInput(  reader->GetOutput()  );

edgeFilter->SetSigma(  1.0  );

filter->SetInput(  edgeFilter->GetOutput()  );

writer->Update()
```

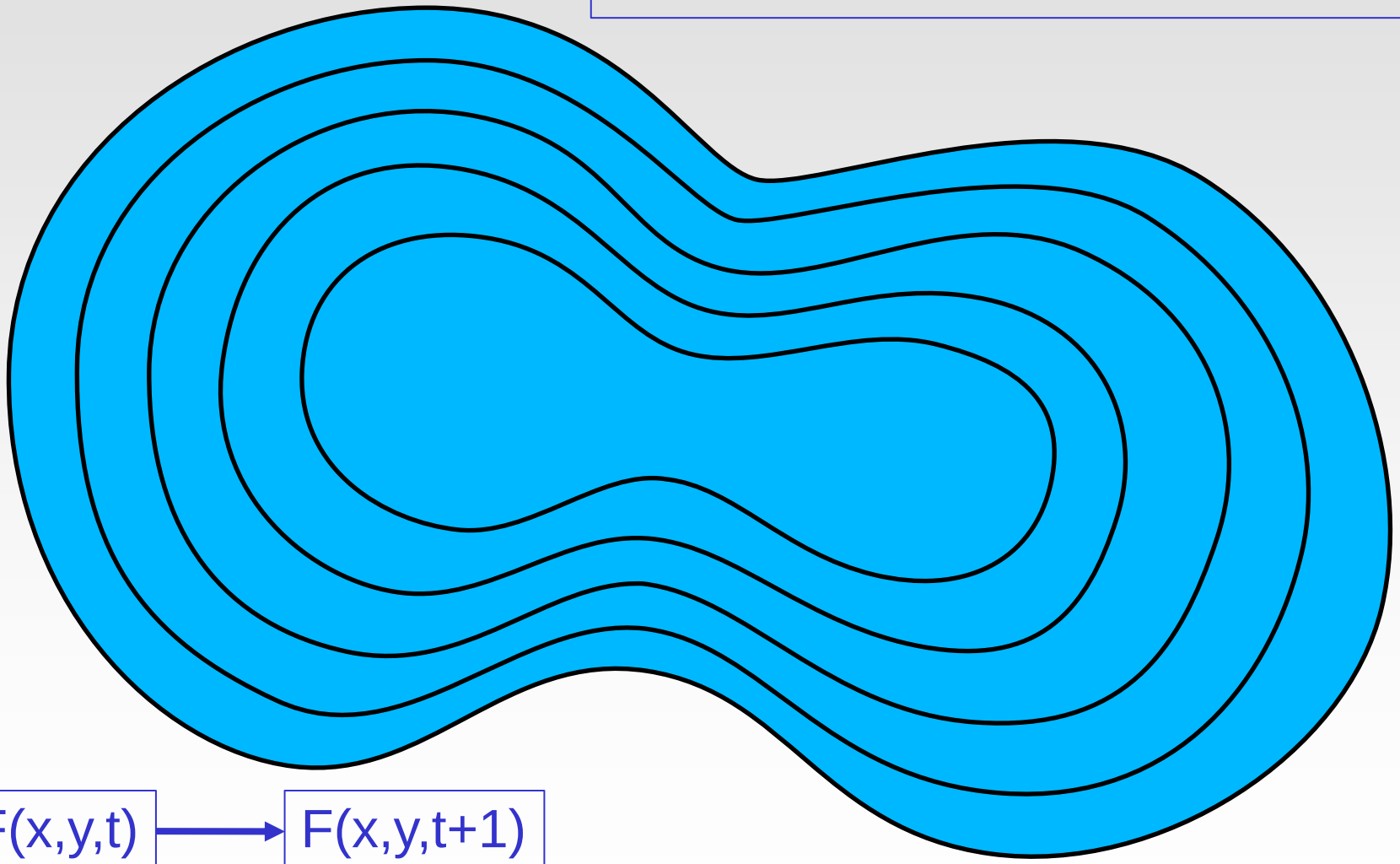
Métodos de Segmentação "Level Set"

Conceito Level set



Evolução Level Set

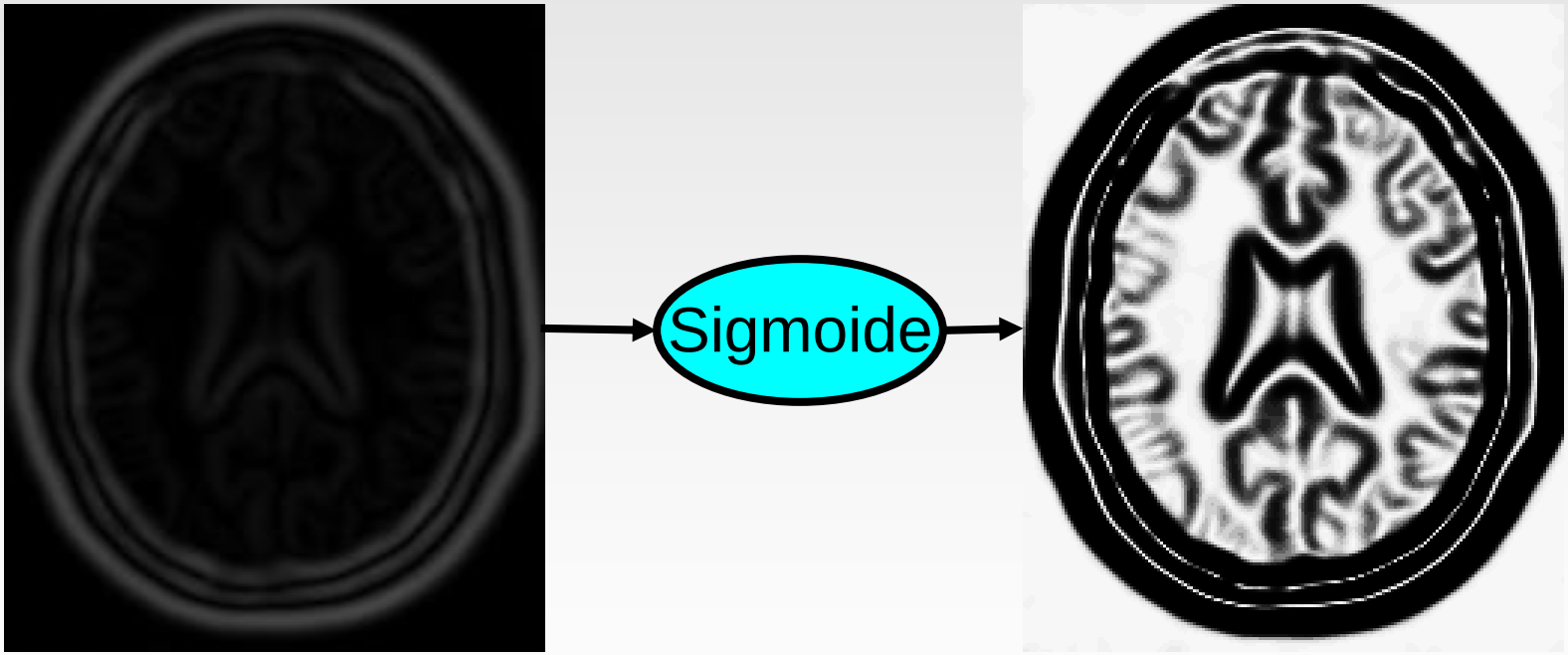
PDE = Automato Celular Restrito



F(x,y,t) → F(x,y,t+1)

Marcha rápida

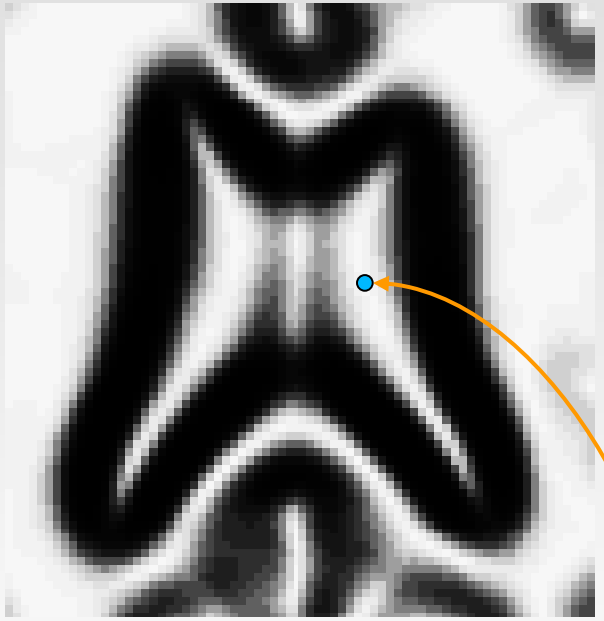
Propagação de Fronte $\Delta x = V \cdot \Delta t$



Magnitude de Gradiente

Imagem de velocidade

Marcha rápida



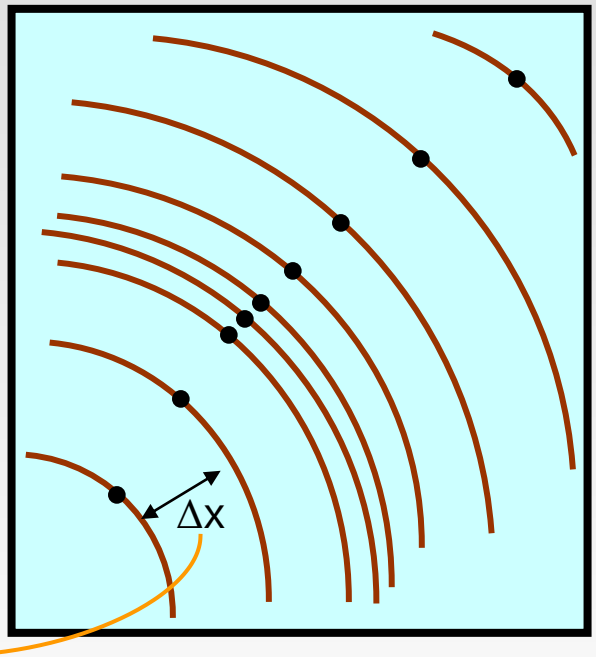


Imagem Velocidade

Mapa Cruzamento Tempo

$\Delta x = V \cdot \Delta t$

Marcha rápida

```
typedef   itk::Image< float , 2 >   ImageType;
typedef   itk::FastMarchingImageFilter<
                                        ImageType,
                                ImageType >   FilterType;

FilterType::Pointer   fastMarching  =  FilterType::New();

fastMarching->SetInput ( speedImage );

fastMarching->SetOutputSize(
          speedImage->GetBufferedRegion().GetSize() );

fastMarching->SetStoppingValue( 100.0 );
```

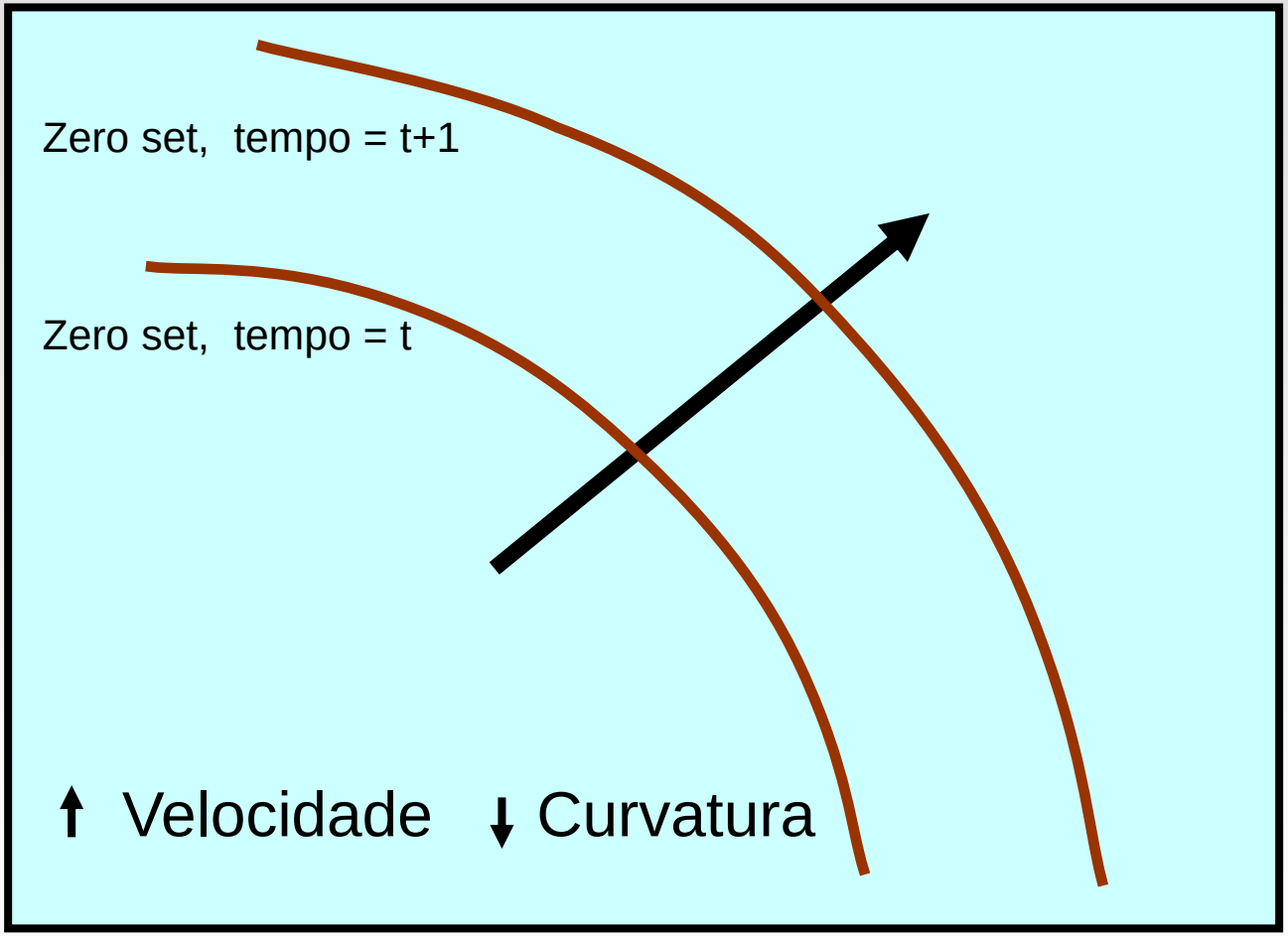
Marcha rápida

```
typedef   FilterType::NodeContainer      NodeContainer;
typedef   FilterType::NodeType           NodeType;

NodeContainer::Pointer   seeds  =  NodeContainer::New();

seeds->Initialize();

NodeType  seed;
seed.SetValue( 0.0 );
seed.SetIndex( index );

seeds->InsertElement( 0, seed );
```

Marcha rápida

```
fastMarching->SetTrialPoints( seeds );

thresholder->SetInput(  fastMarching->GetOutput() );

thresholder->SetLowerThreshold(  0.0 );
thresholder->SetUpperThreshold(  timeThreshold );

thresholder->Update();
```

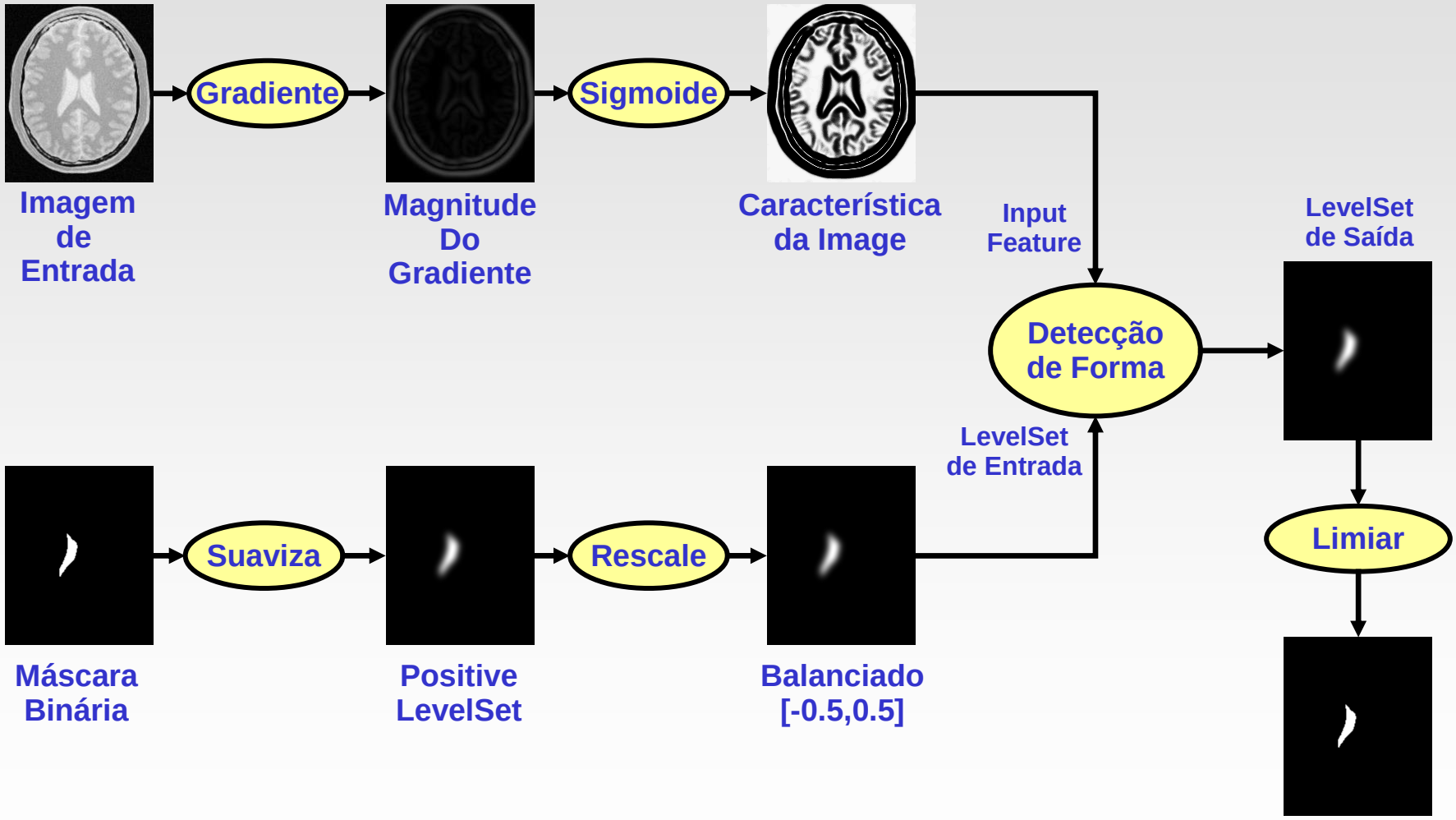
Detecção de Forma



PDE Inclui um termo curvatura

Previne vazamento

Detecção de Forma



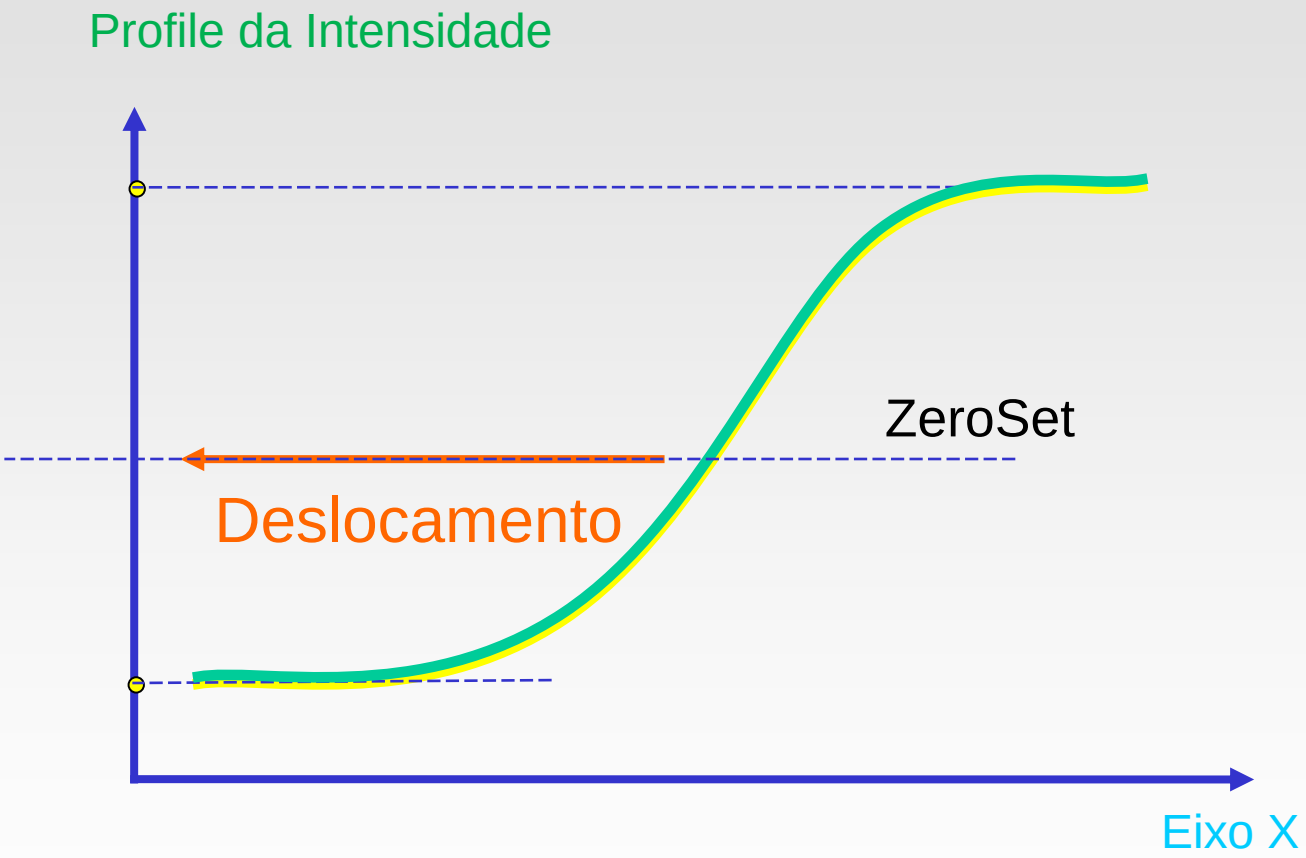
Detecção de Forma

```
typedef   itk::Image< float , 2 >   ImageType;
typedef   itk::ShapeDetectionLevelSetImageFilter<
                                        ImageType,
                                ImageType >   FilterType;

FilterType::Pointer   shapeDetection  =  FilterType::New();

shapeDetection->SetInput( inputLevelSet );
shapeDetection->SetFeatureImage( speedImage );

shapeDetection->SetPropagationScaling( 1.0 );
shapeDetection->SetCurvatureScaling(  0.05 );
```

Detecção de Forma

```
shapeDetection->SetMaximumRMSError( 0.001 );
shapeDetection->SetMaximumIterations(  400 );

shapeDetection->Update();

std::cout << shapeDetection->GetRMSChange() << std::endl;
std::cout << shapeDetection->GetElapsedIterations() << std::endl;

thresholder->SetInput( shapeDetection->GetOutput() );
thresholder->SetLowerThreshold( -1e7 );
thresholder->SetUpperThreshold(   0.0 );
```

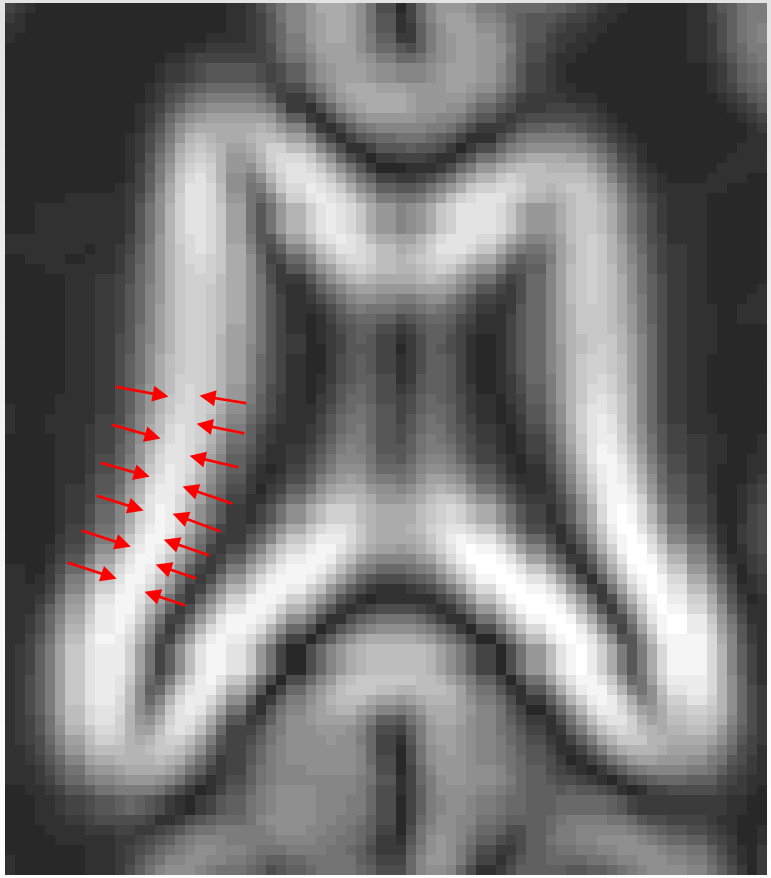
Contorno Ativo Geodésico



Termo adverção

Contorno Ativo Geodésico

Campo Vetorial Computado Internamente



Contorno Ativo Geodésico

```
typedef   itk::Image< float , 2 >   ImageType;
typedef   itk::GeodesicActiveContourLevelSetImageFilter<
                                        ImageType,
                                ImageType >   FilterType;

FilterType::Pointer   geodesicActiveContour  =  FilterType::New();

geodesicActiveContour->SetInput( inputLevelSet );
geodesicActiveContour->SetFeatureImage( speedImage );

geodesicActiveContour->SetPropagationScaling( 1.0 );
geodesicActiveContour->SetCurvatureScaling( 0.05 );
geodesicActiveContour->SetAdvectionScaling( 8.0 );
```

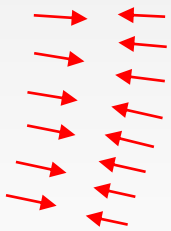
Contorno Ativo Geodésico

```
geodesicActiveContour->SetMaximumRMSError( 0.001 );
geodesicActiveContour->SetMaximumIterations(  400 );

geodesicActiveContour->Update();

std::cout << geodesicActiveContour->GetRMSChange() << std::endl;
std::cout << geodesicActiveContour->GetElapsedIterations() << std::endl;

thresholder->SetInput(  geodesicActiveContour  );

thresholder->SetLowerThreshold( -1e7 );
thresholder->SetUpperThreshold(   0.0 );
```

Limiar Level Set

Termo advecção controlado por um limiar

LevelSet equivalente
de um método de
componentes conectados
dentro de um limiar



mas… com opções
para prevenir vazamenntos

Segmentação por limiar

```
typedef   itk::Image< float , 2 >   ImageType;
typedef   itk::ThresholdSegmentationLevelSetImageFilter<
                                        ImageType,
                                ImageType >   FilterType;

FilterType::Pointer   thresholdSegmentation  =  FilterType::New();

thresholdSegmentation->SetInput( inputLevelSet );
thresholdSegmentation->SetFeatureImage( inputImage );

thresholdSegmentation->SetPropagationScaling( 1.0 );
thresholdSegmentation->SetCurvatureScaling( 5.0 );
thresholdSegmentation->SetAdvectionScaling( 2.0 );
```

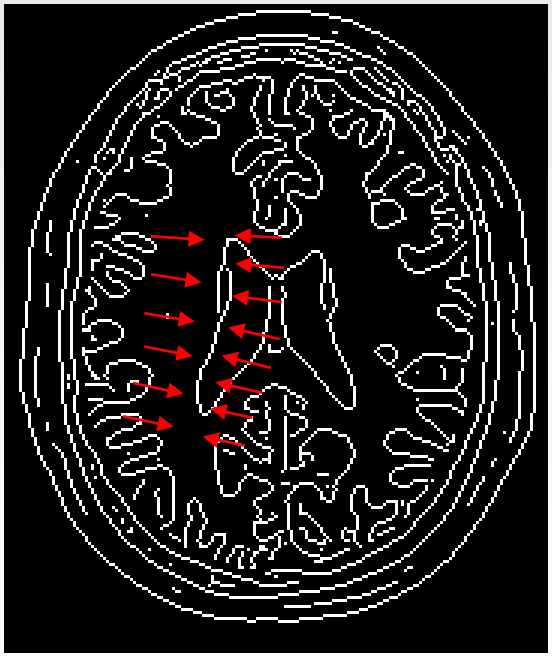
Segmentação por limiar

```
thresholdSegmentation->SetMaximumRMSError( 0.001 );
thresholdSegmentation->SetMaximumIterations(  400 );

thresholdSegmentation->SetLowerThreshold( 210 );
thresholdSegmentation->SetUpperThreshold( 250 );

thresholdSegmentation->SetIsoSurface( 0.0 );  // zero set

thresholdSegmentation->SetUseNegativeFeaturesOn();

thresholdSegmentation->Update();
```

Limiar Level Set

Termo de advecção controlado por bordas

Bordas Canny atraem
o zero set



Segmentação Canny

```
typedef   itk::Image< float , 2 >   ImageType;
typedef   itk::CannySegmentationLevelSetImageFilter<
                                        ImageType,
                                ImageType >   FilterType;

FilterType::Pointer   cannySegmentation  =  FilterType::New();

cannySegmentation->SetInput( inputLevelSet );
cannySegmentation->SetFeatureImage( inputImage );

cannySegmentation->SetPropagationScaling( 0.0 );
cannySegmentation->SetCurvatureScaling( 1.0 );
cannySegmentation->SetAdvectionScaling( 2.0 ); // canny edges
```

Segmentação Canny

```
cannySegmentation->SetMaximumRMSError( 0.01 );
cannySegmentation->SetMaximumIterations(  400 );

cannySegmentation->SetThreshold( 2.0 );
cannySegmentation->SetVariance( 1.0 );

cannySegmentation->SetIsoSurface( 127.0 );  // zero set

cannySegmentation->SetUseNegativeFeaturesOn();

cannySegmentation->Update();
```

Corregistro

- Reamostragem de imagens
- Estruturas de corregistro
- Multimodalidades
- Multirresolução
- Corregistro deformável

Reamostragem de Imagens

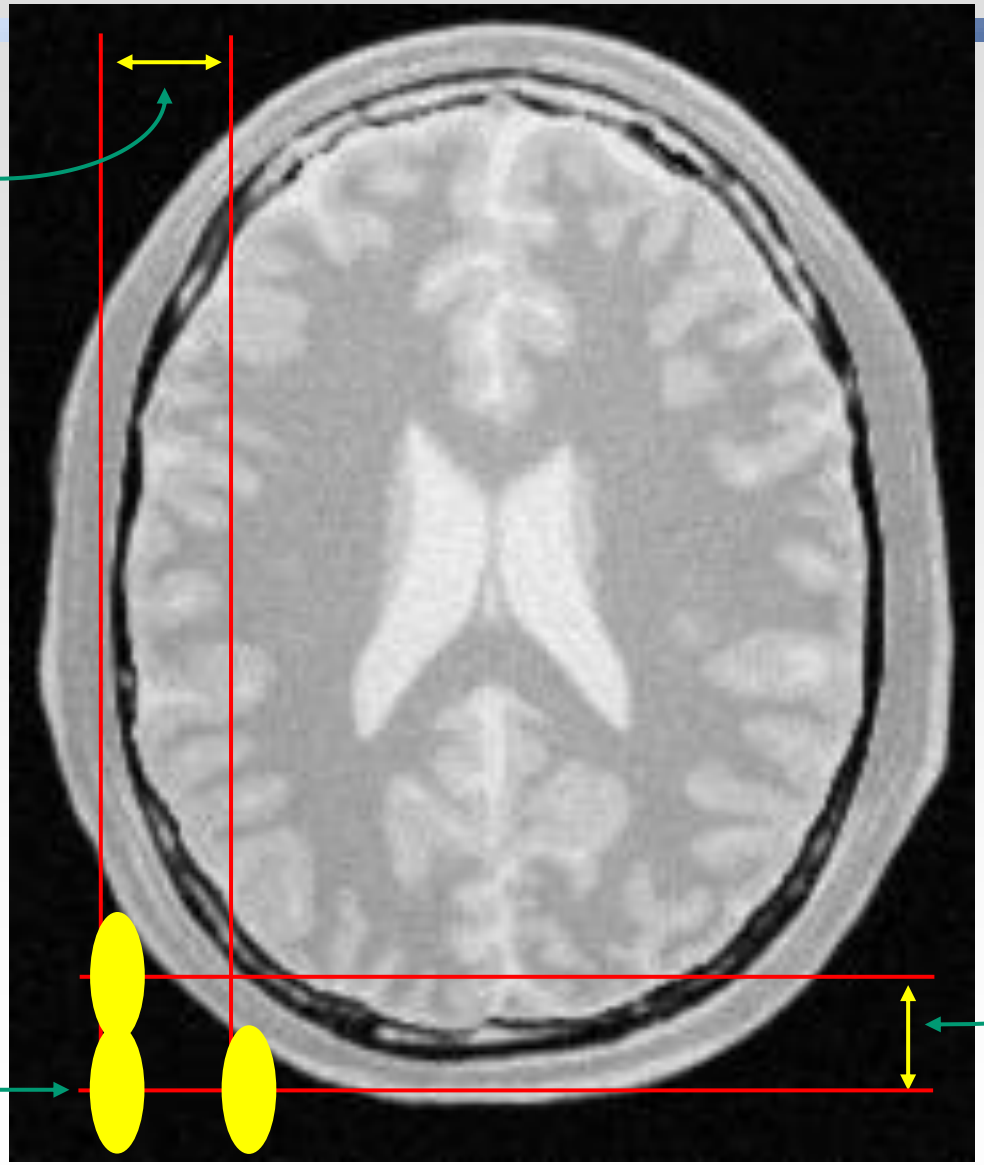
Porquê Reamostragem?

Reamostagem é a Essência
do Corregistro de Imagens
Baseado em Intensidades

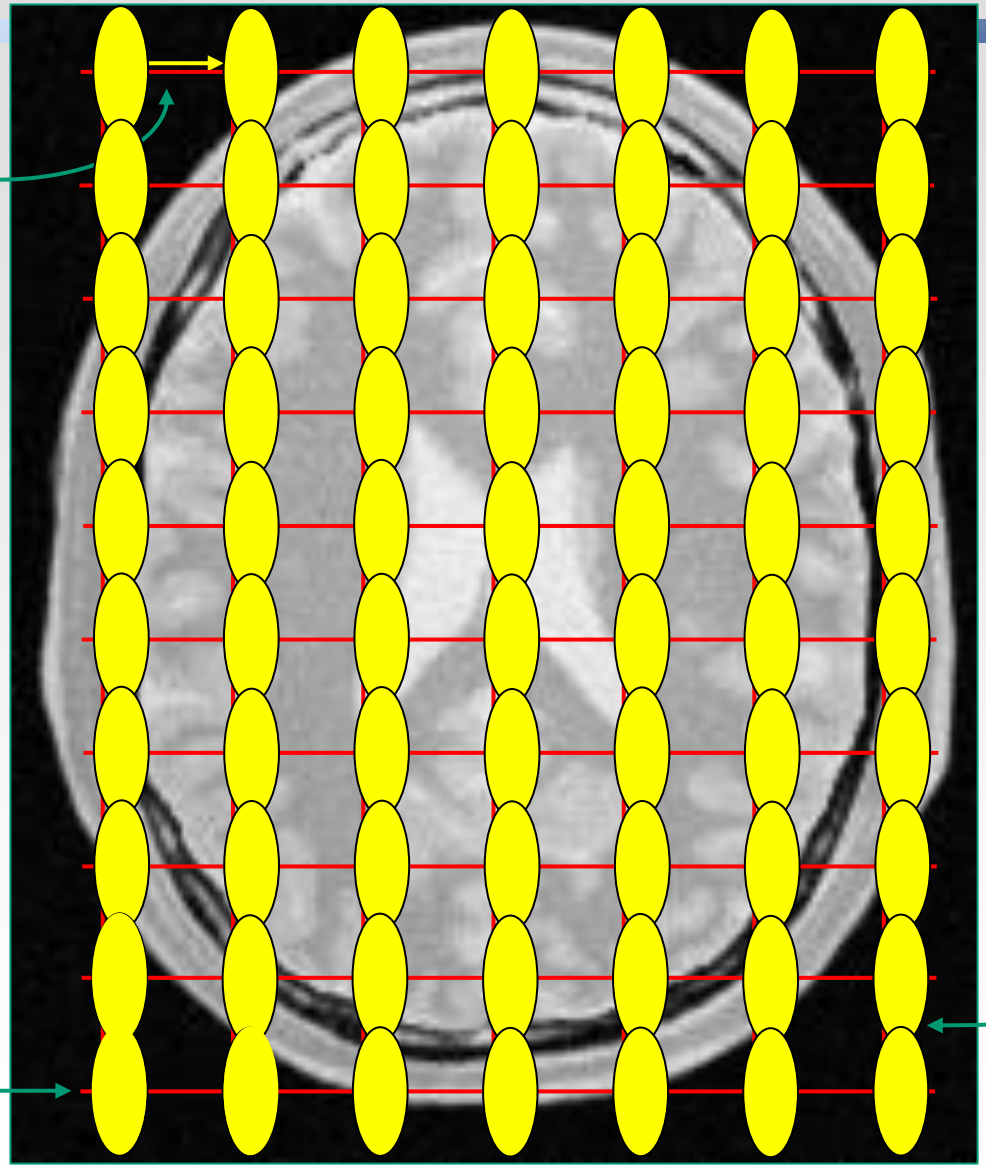
O quê é uma Imagem ?

Uma Imagem é uma
Amostragem de Campo
Contínuo usando
uma Grade Discreta

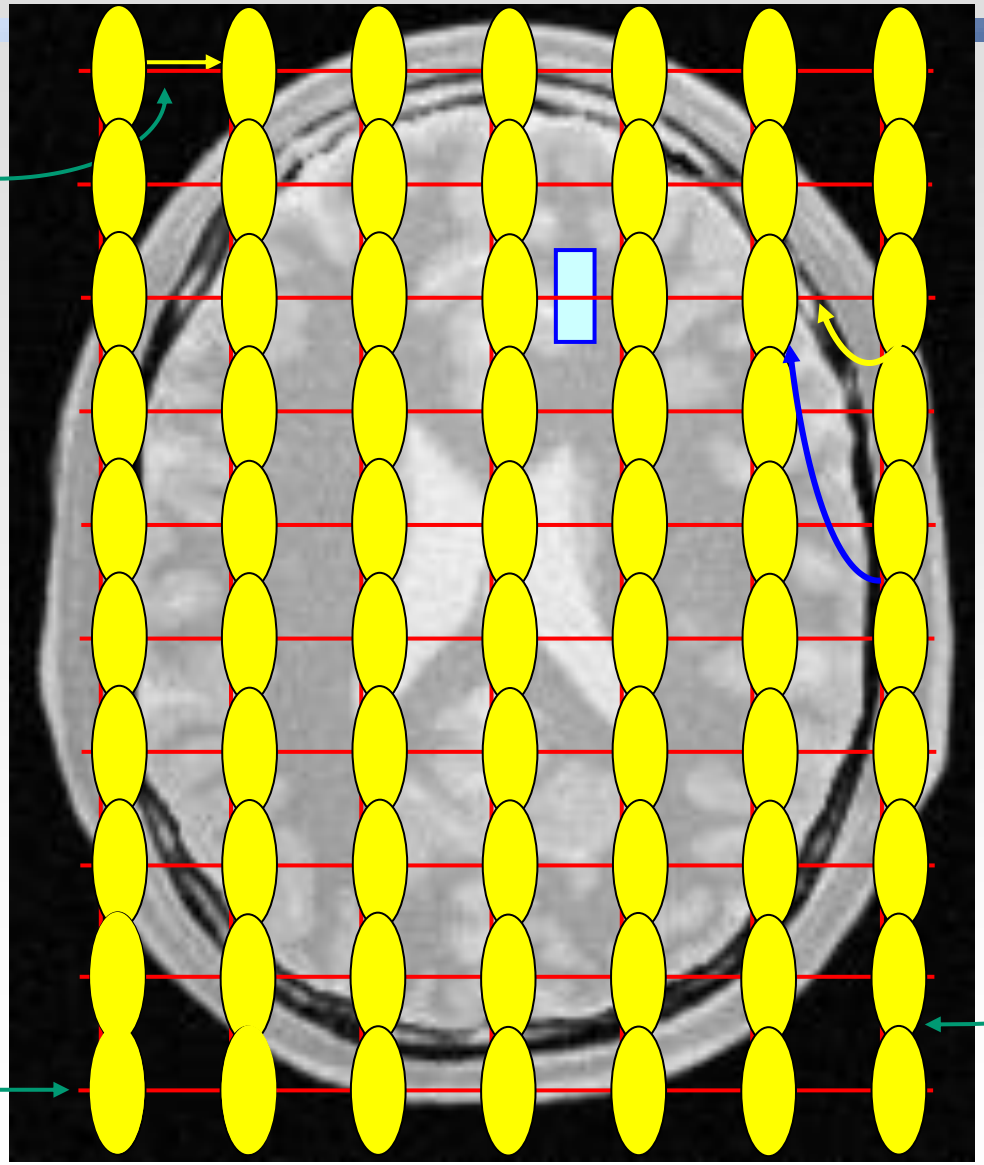
Origem na Imagem & Espaçamento



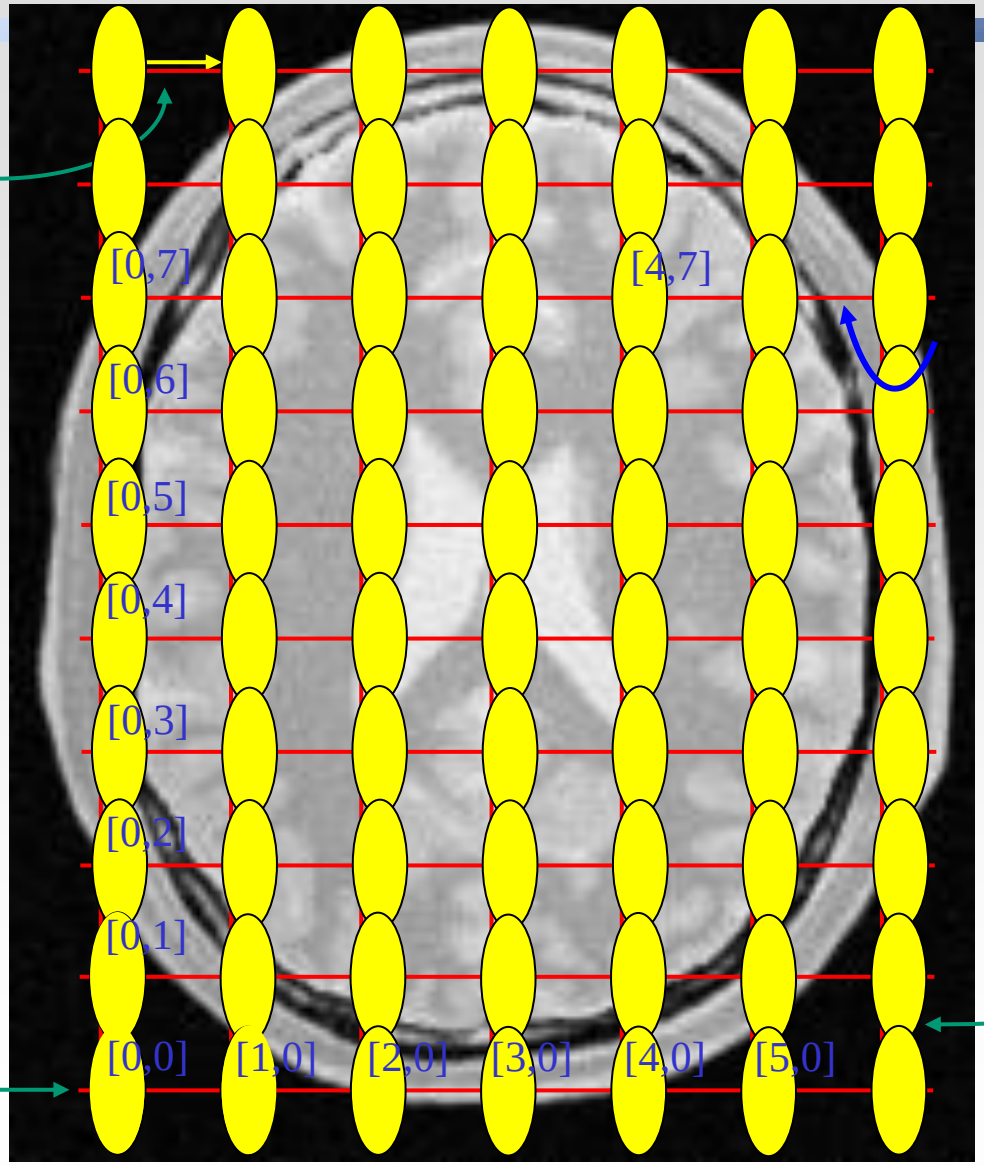
Grade de Amostragem na Imagem



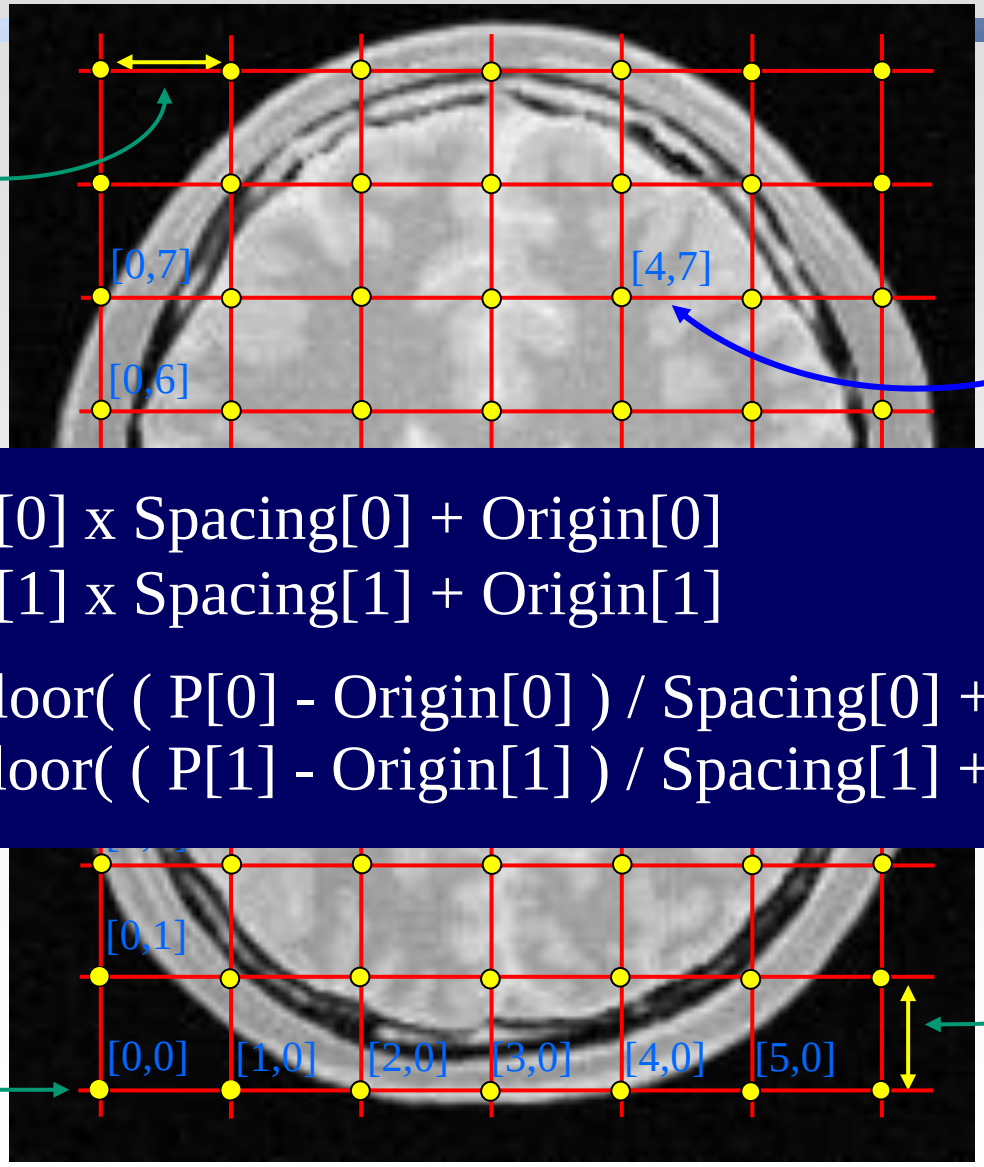
Pixel de Imagem



Índices de Imagens



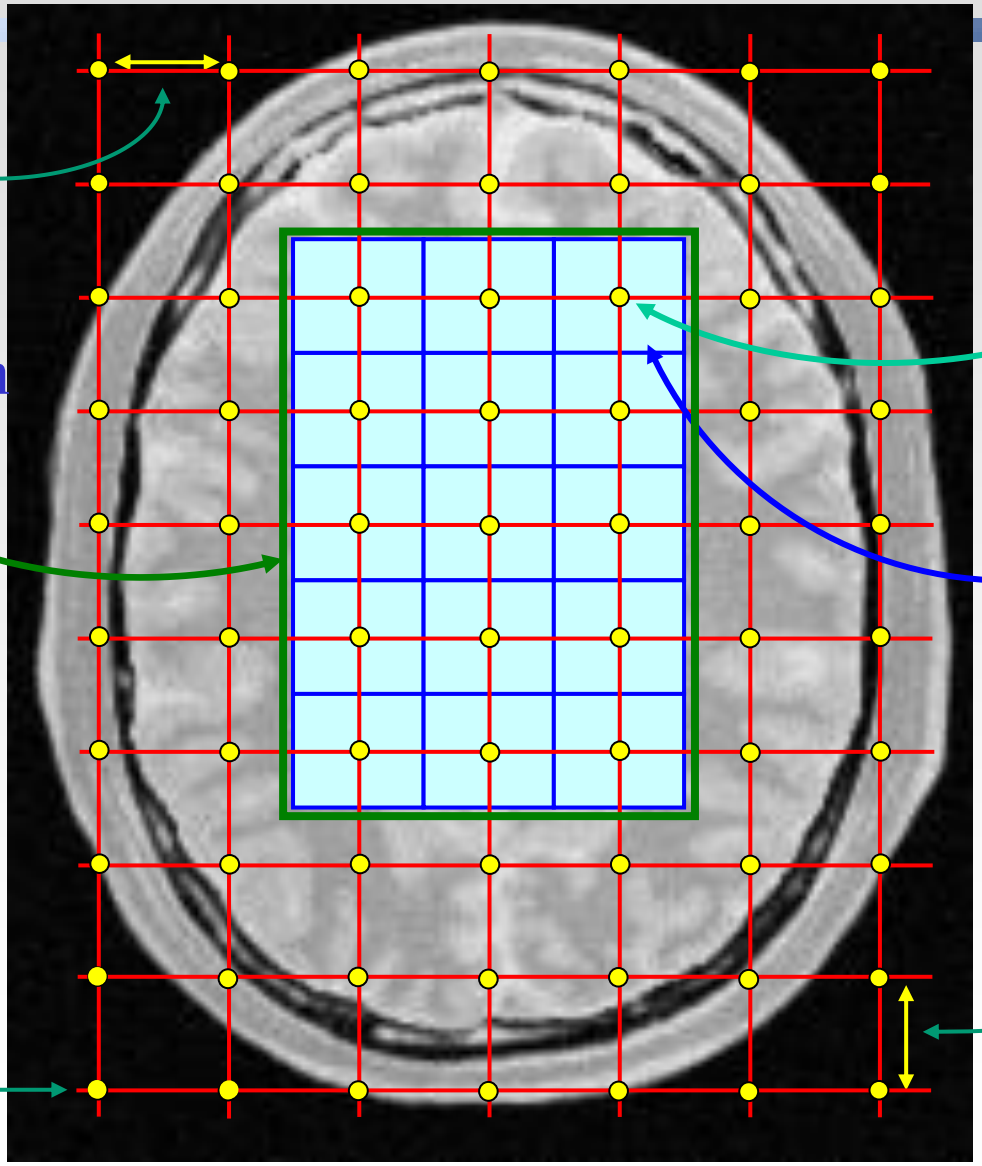
Índice para Coordenadas Físicas



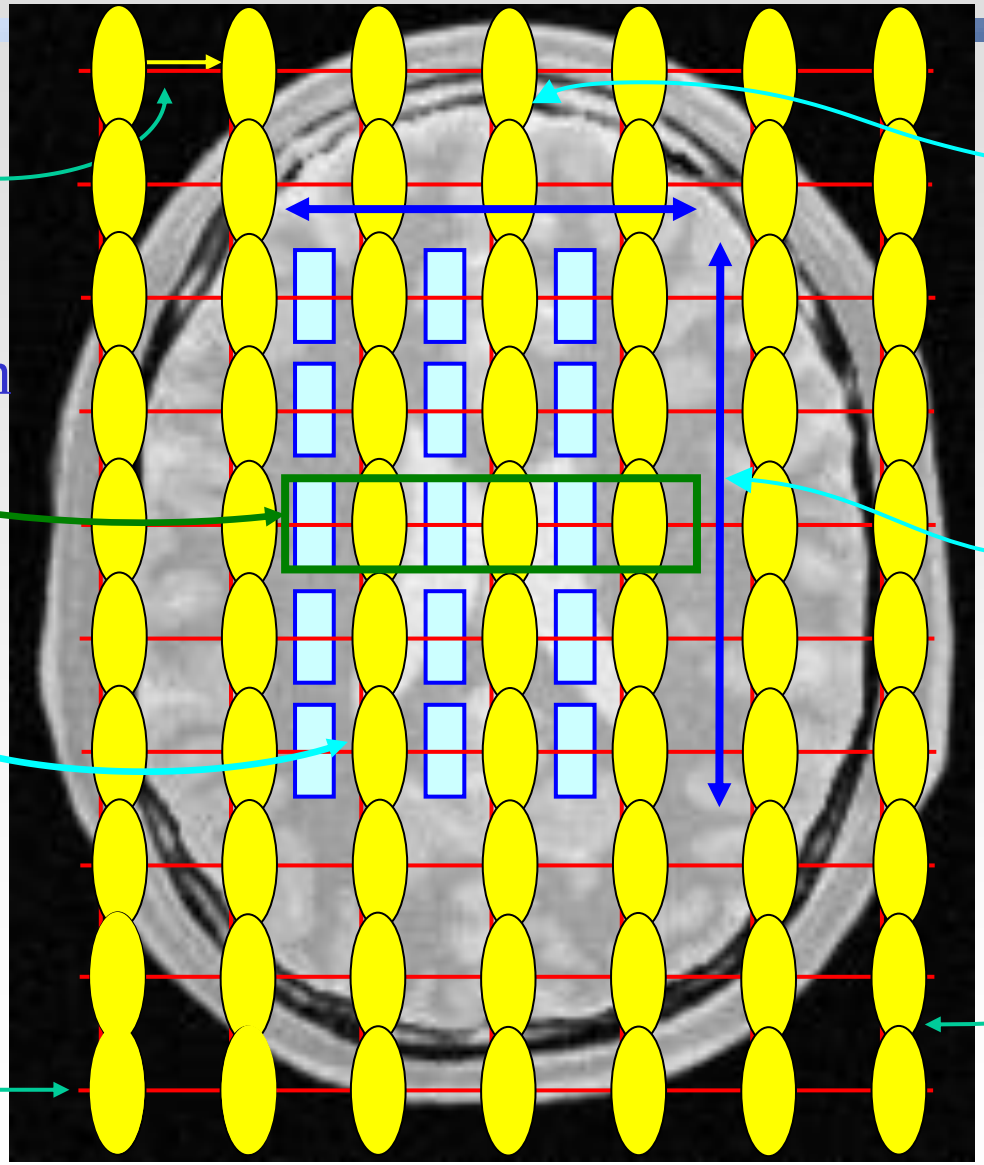
P[0] = Index[0] x Spacing[0] + Origin[0]
P[1] = Index[1] x Spacing[1] + Origin[1]

Index[0] = floor((P[0] - Origin[0]) / Spacing[0] + 0.5)
Index[1] = floor((P[1] - Origin[1]) / Spacing[1] + 0.5)

Região da Imagem



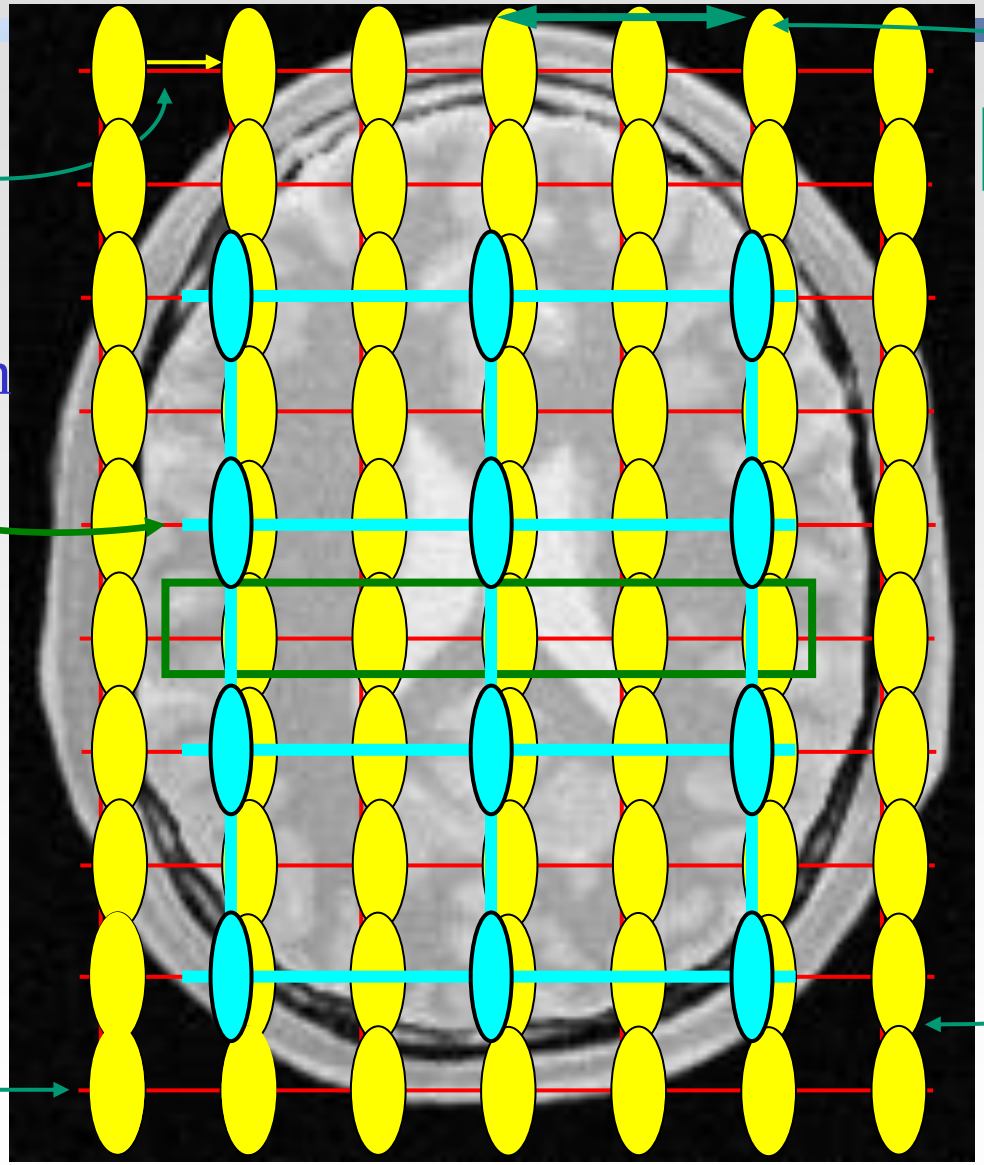
Região da Imagem



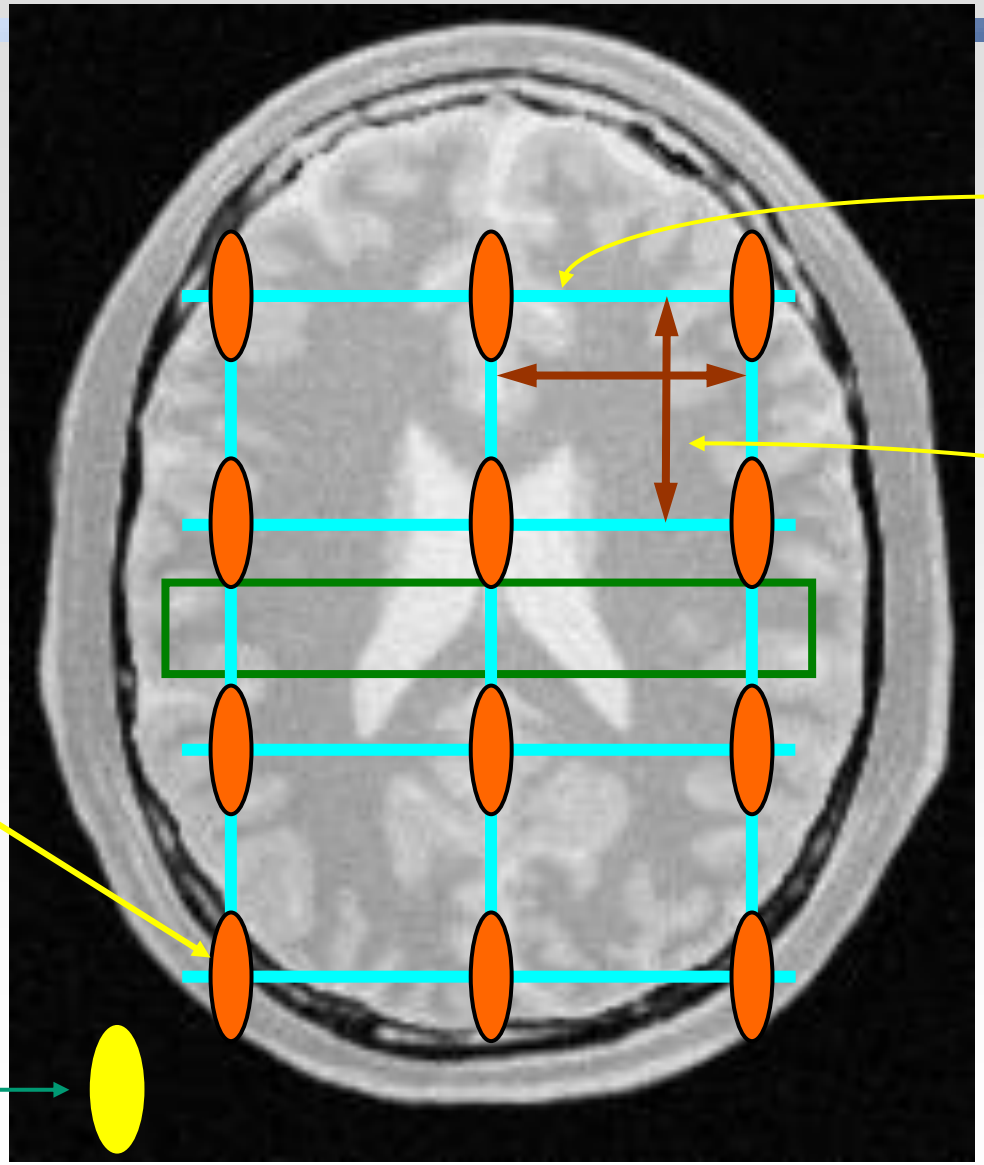
Reamostagem Básica

Reamostragem nos
Casos Triviais

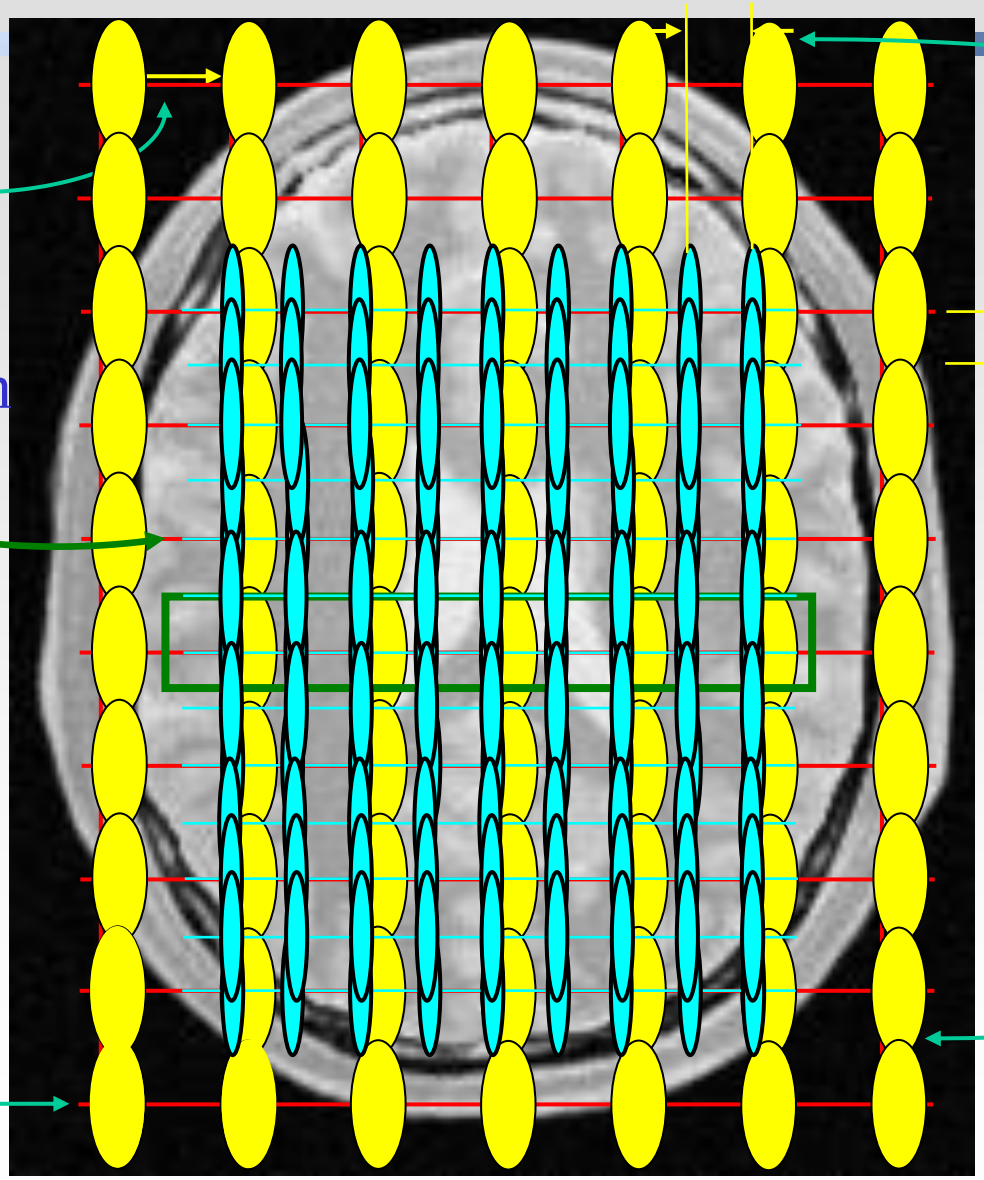
Subamostragem pela metade



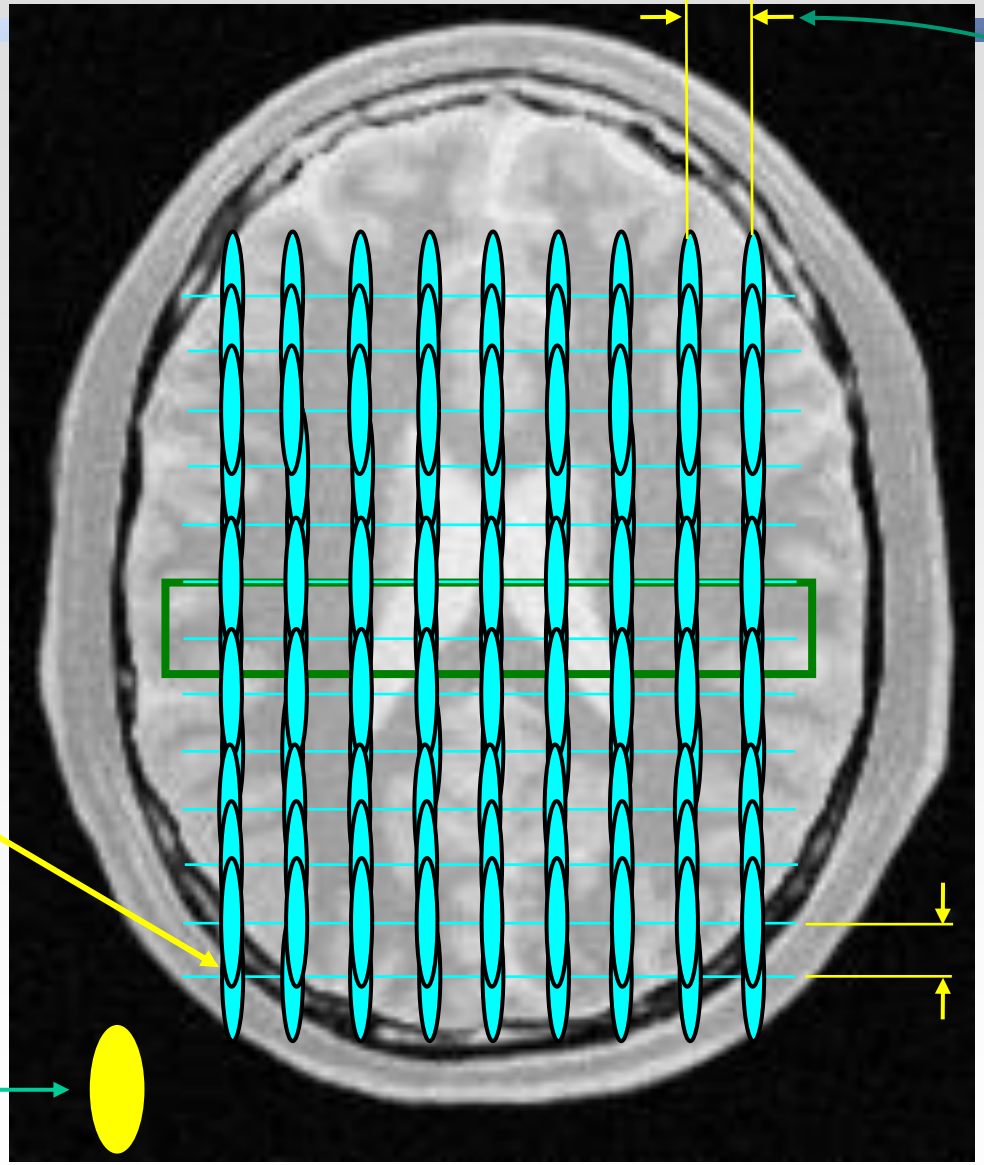
Subamostragem pela metade



Superamostragem pelo dobro



Superamostragem pelo dobro



novo espaçam.
S'x

nova origem
(O'x,O'y)

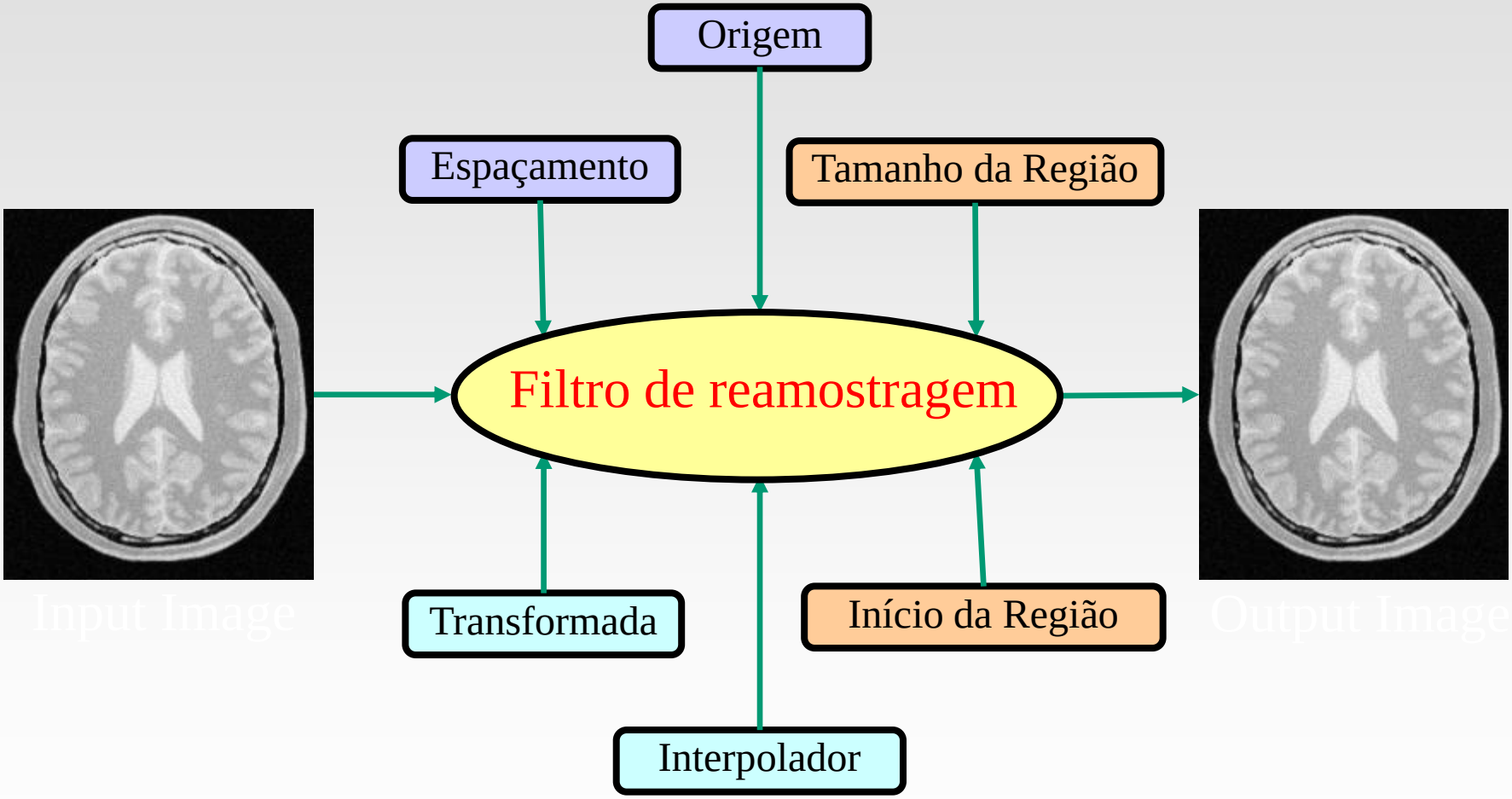
novo espaçam.
S'y

origem (Ox,Oy)

Reamostragem no ITK

itk::ResampleImageFilter

Reamostragem no ITK



Filtro de Reamostragem

```
#include "itkImage.h"
#include "itkResampleImageFilter.h"
#include "itkIdentityTransform.h“
#include "itkLinearInterpolateImageFunction.h"

typedef itk::Image< char, 2 >  ImageType;

ImageType::Pointer inputImage = GetImageSomeHow();

typedef itk::ResampleImageFilter< ImageType >  FilterType;

FilterType::Pointer resampler = FilterType::New();

ImageType::SizeType size;
size[0] = 200;
size[1] = 300;

ImageType::IndexType start;
start[0] = 0;
start[1] = 0;
```

Filtro de Reamostragem

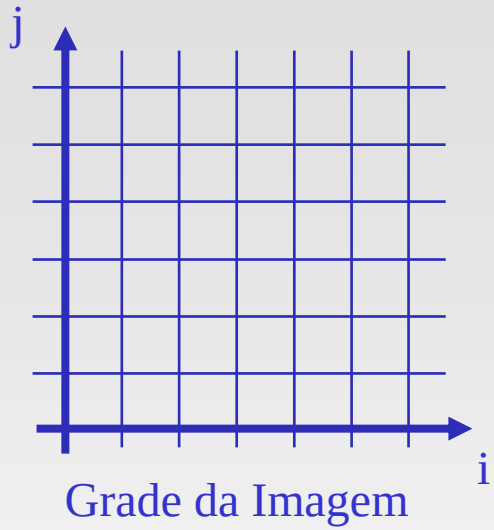
```
ImageType::PointType origin;
origin[0] = 10.0;  // millimeters
origin[1] = 25.5;  // millimeters

ImageType::SpacingType spacing;
spacing[0] = 2.0;  // millimeters
spacing[1] = 1.5;  // millimeters

resampler->SetOutputSpacing( spacing );
resampler->SetOutputOrigin( origin );

resampler->SetSize( size );
resampler->SetOutputStartIndex( start );

resampler->SetDefaultPixelValue( 100 );

resampler->SetInput( inputImage );
```

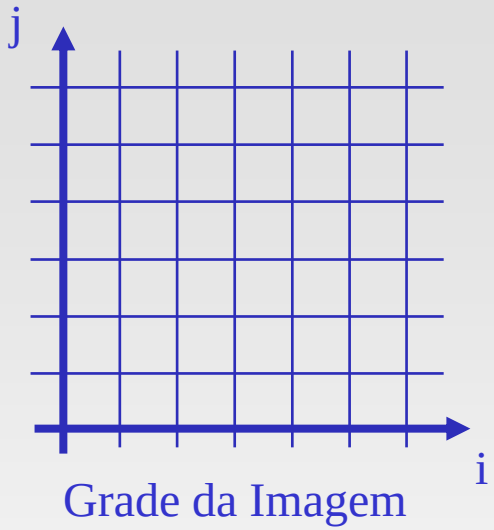
Filtro de Reamostragem

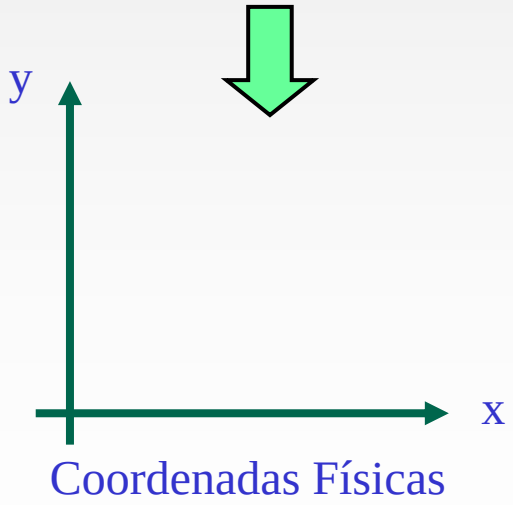
```
typedef itk::LinearInterpolateImageFunction<
                                ImageType,
                                double >     InterpolatorType;

InterpolatorType::Pointer interpolator = InterpolatorType::New();

typedef itk::TranslationTransform< double, 2 >  TransformType;

TransformType::Pointer transform = TransformType::New();

transform->SetIdentity();

resampler->SetInterpolator( interpolator );
resampler->SetTransform( transform );

resampler->Update();

const ImageType * outputImage = resampler->GetOutput();
```

Corregistro Básico

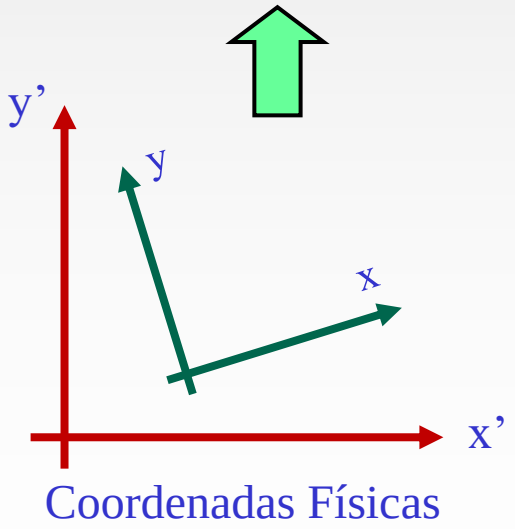
Conversão de Sistemas de Coordenadas







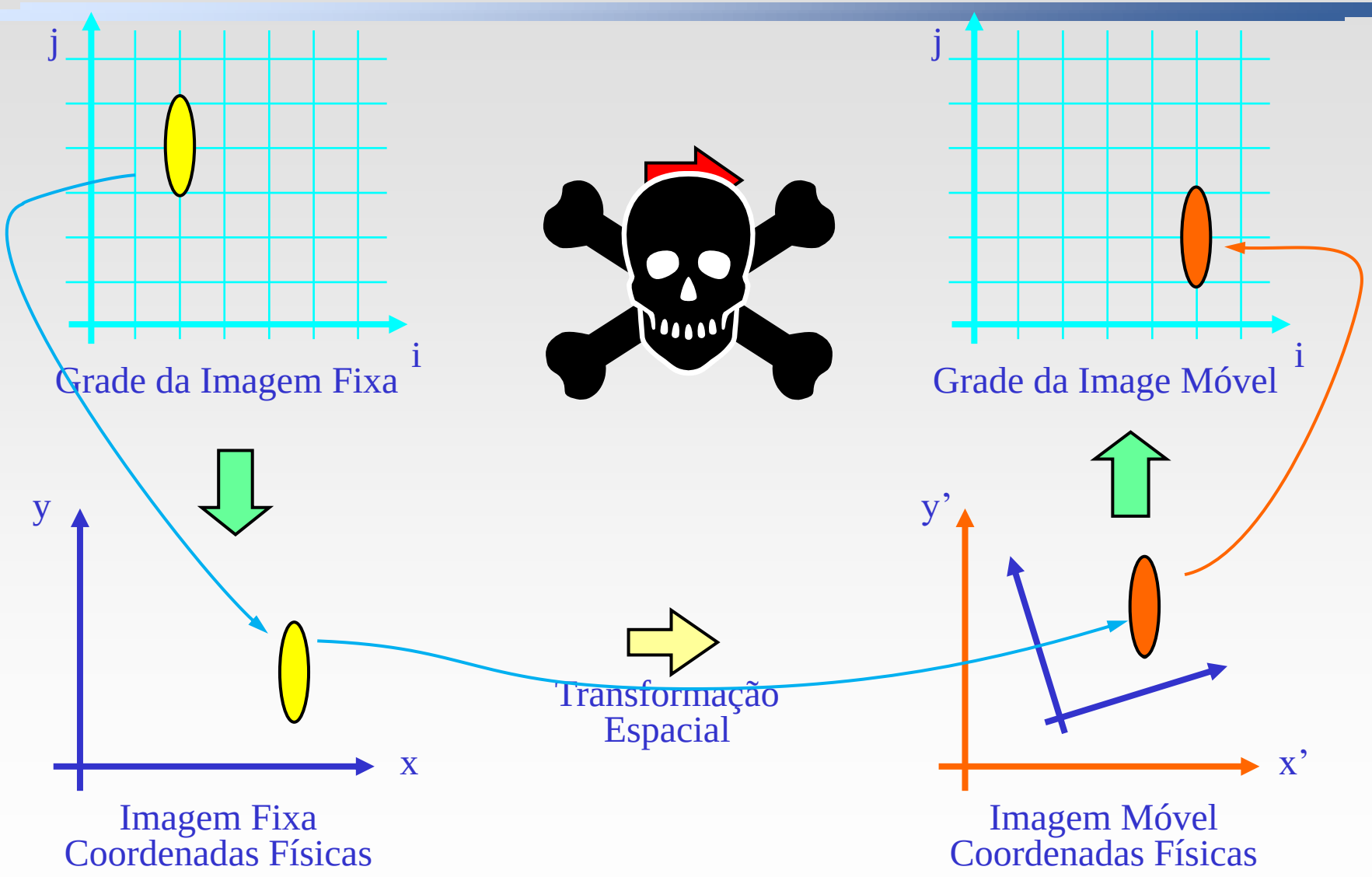
Transformação Espacial



Coisas que não farei...

Não corregistrar imagens no espaço de pixels
Não corregistrar imagens no espaço de pixels
Não corregistrar imagens no espaço de pixels
Não corregistrar imagens no espaço de pixels
Não corregistrar imagens no espaço de pixels
Não corregistrar imagens no esp

Imagem Fixa & Imagem Móvel



Selecionando Imagens Móveis e Fixas

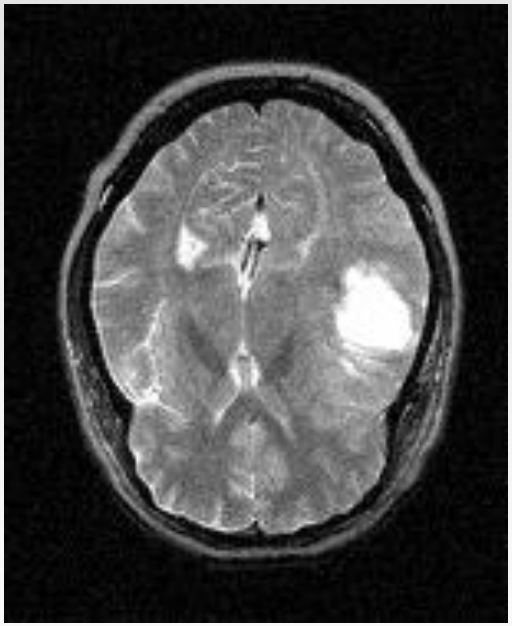
Em princípio, a denominação de
Imagem **Fixa** e Imagem **Móvel**
é arbtrária

Na prática, a Imagem **Móvel**
é aquela que será reamostrada
para o sistema de coordenadas
da Imagem **Fixa**

Pergunta #1

Imagens do mesmo paciente

MRI-T2



256 x 256 pixels

Imagem Móvel ?

Imagem Fixa ?

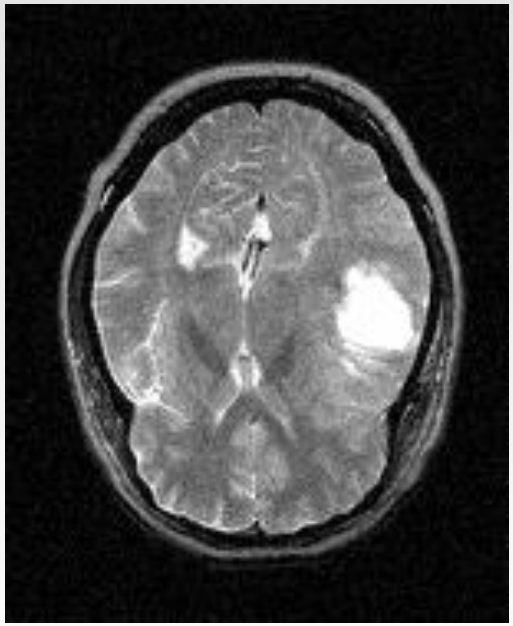
PET

128 x 128 pixels

Pergunta #2

Imagens do mesmo paciente

MRI-T2



256 x 256 pixels

Transformação
de Escala

Qual fator de escala ?

a) 2.0

b) 1.0

c) 0.5

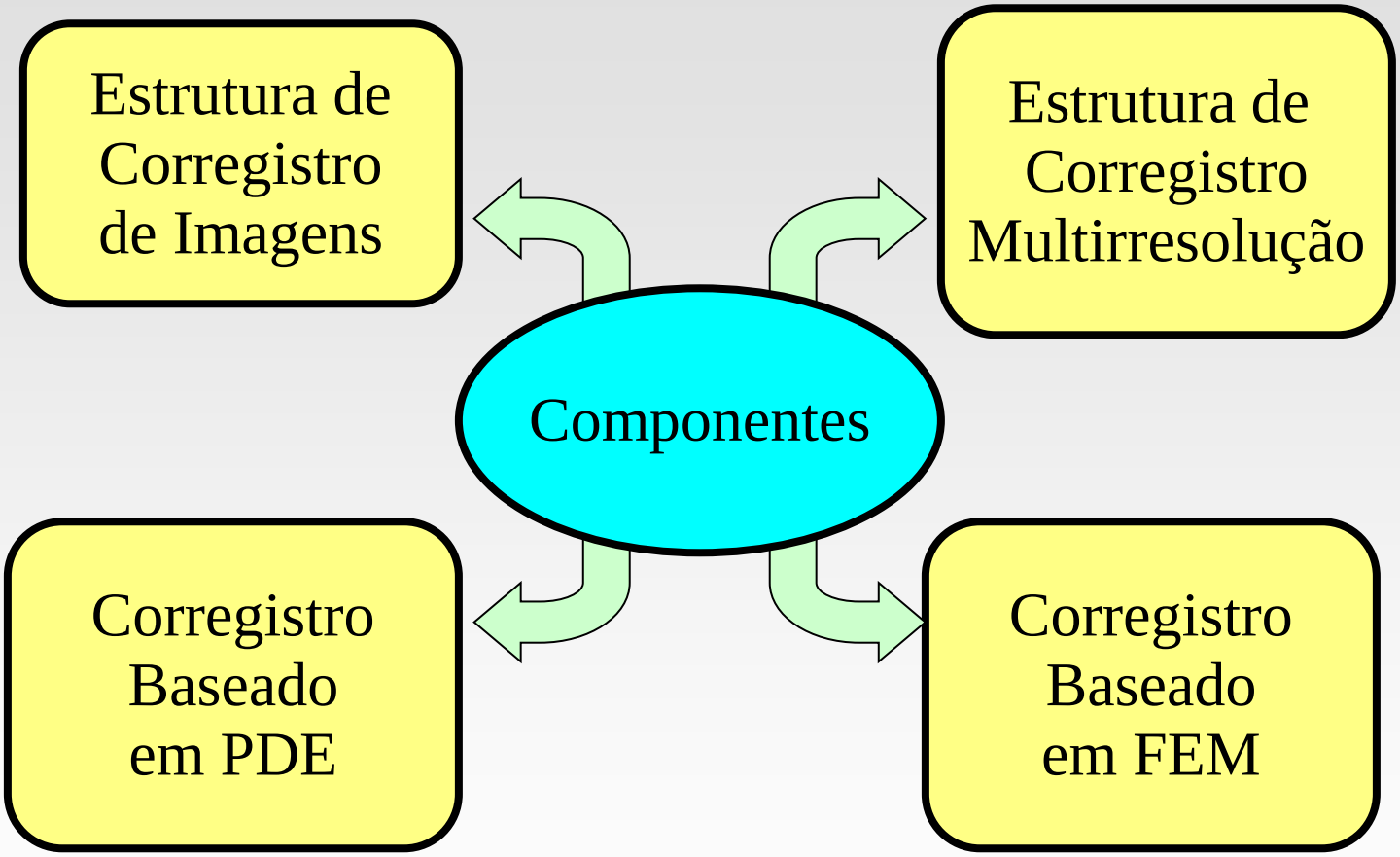
PET

128 x 128 pixels

Coisas que não farei...

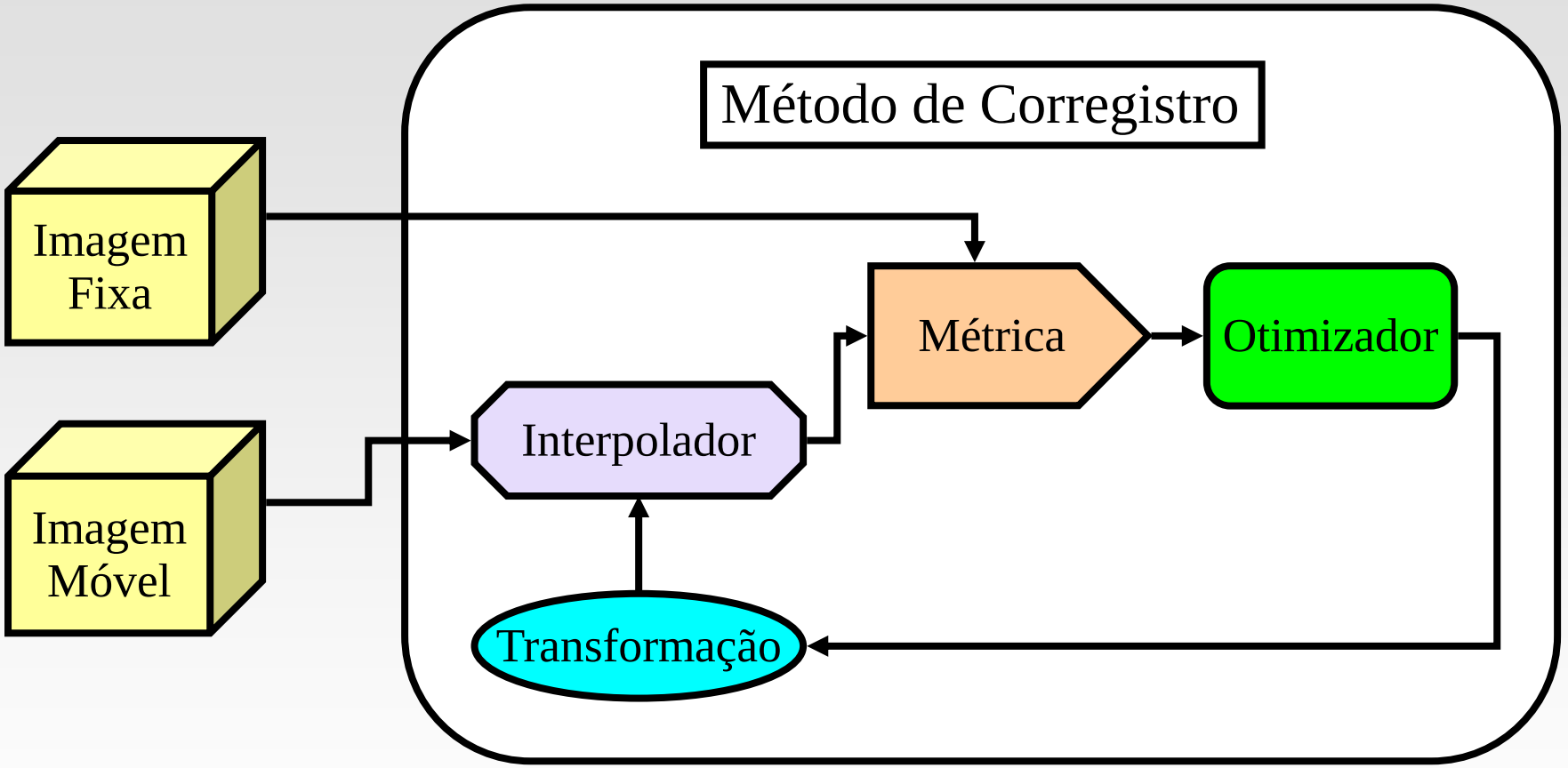
Não corregistrar imagens no espaço de pixels
Não corregistrar imagens no espaço de pixels
Não corregistrar imagens no espaço de pixels
Não corregistrar imagens no espaço de pixels
Não corregistrar imagens no espaço de pixels
Não corregistrar imagens no esp

Corregistro no ITK



Estrutura para Corregistro

Componentes



Métrica de Similaridade

- **Médias Quadráticas**
- **Correlação Normalizada**
- **Diferença Quadrática Média Recíproca**
- **Informação Mutua**
  - Viola-Wells
  - Mattes
  - Baseada em histograma
  - Histograma normalizado

Transformções

- Translação
- Escalonamento
- Rotação
- Rígido3D
- Rígido2D
- Affine
- BSplines

Otimizadores

- Gradiente Descente
- Gradiente Descente com Passos Regulares
- Gradiente Conjugado
- Levenberg-Marquardt
- Algoritmo Evolutivo

Interpoladores

- **Vizinhos Próximos**
- **Linear**
- **BSpline**

Corregistro de Imagens

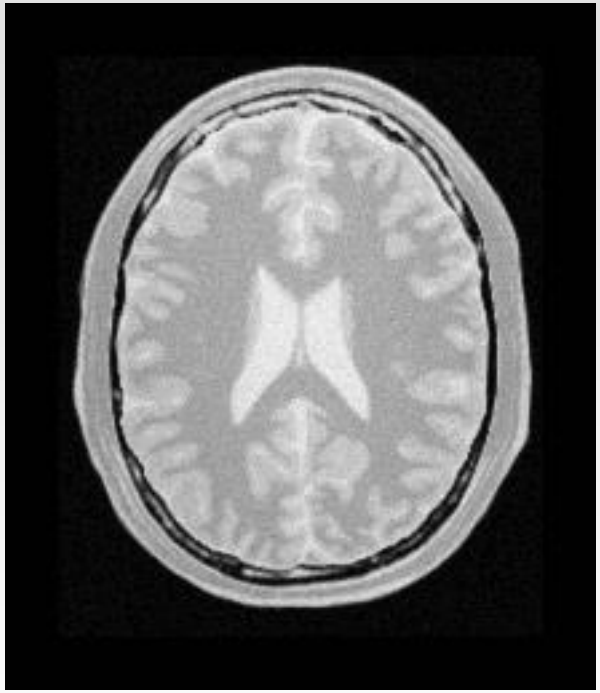


Imagem Fixa

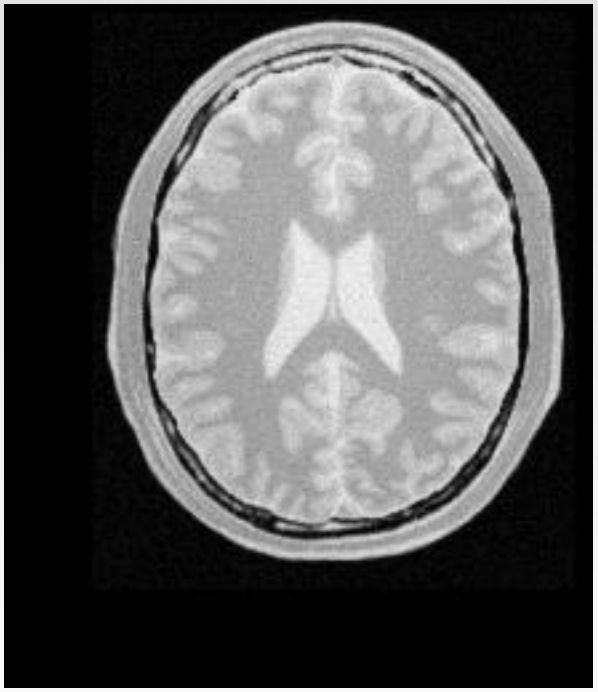


Imagem Móvel

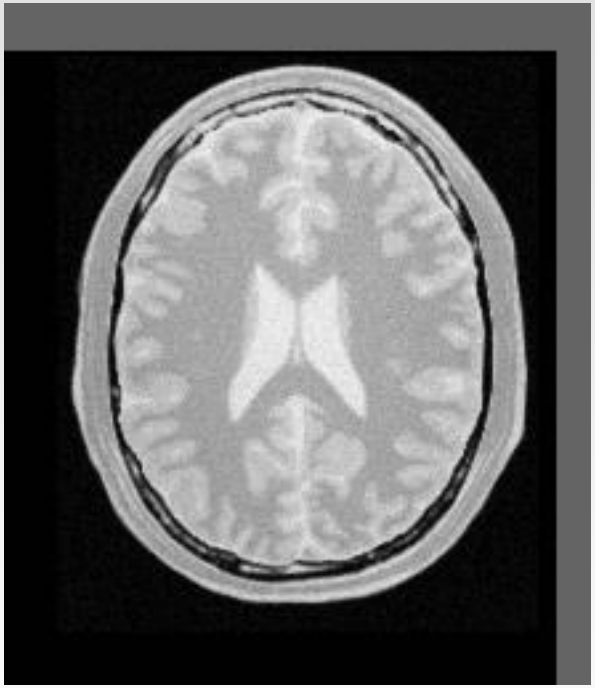


Imagem Móvel Corregistrada

Corregistro de Imagens

```
#include ”itkImageRegistrationMethod.h”
#include ”itkTranslationTransform.h”
#include ”itkMeanSquaresImageToImageMetric.h”
#include ”itkLinearInterpolateImageFunction.h”
#include ”itkRegularStepGradientDescentOptimizer.h”
#include ”itkImage.h”
#include ”itkImageFileReader.h”
#include ”itkImageFileWriter.h”
#include ”itkResampleImageFilter.h”
```

Corregistro de Imagens

```
const unsigned int Dimension = 2;
typedef unsigned char PixelType;

typedef   itk::Image< PixelType , Dimension >                 FixedImageType;
typedef   itk::Image< PixelType , Dimension >                 MovingImageType;
typedef   itk::TranslationTransform< double, Dimension >      TransformType;
typedef   itk::RegularStepGradientDescentOptimizer            OptimizerType;
typedef   itk::LinearInterpolateImageFunction<
                                MovingImageType , double >    InterpolatorType;
typedef   itk::MeanSquaresImageToImageMetric<
                        FixedImageType , MovingImageType >    MetricType;

typedef   itk::ImageRegistrationMethod<
                        FixedImageType , MovingImageType >    RegistrationType;
```

Corregistro de Imagens

```
TransformType::Pointer     transform      = TransformType::New();
OptimizerType::Pointer     optimizer      = OptimizerType::New();
InterpolatorType::Pointer  interpolator   = InterpolatorType::New();
MetricType::Pointer        metric         = MetricType::New();
RegistrationType::Pointer  registrator    = RegistrationType::New();

registrator->SetTransform(    transform    );
registrator->SetOptimizer(    optimizer    );
registrator->SetInterpolator( interpolator );
registrator->SetMetric(       metric       );

registrator->SetFixedImage( fixedImageReader->GetOutput() );
registrator->SetMovingImage( movingImageReader->GetOutput() );
‘
```

Corregistro de Imagens

```
registrator->SetFixedImageRegion(
                fixedImageReader->GetOutput()->GetBufferedRegion() );

typedef   RegistrationType::ParametersType     ParametersType;

transform->SetIdentity();

registrator->SetInitialTransformParameters(
                                        transform->GetParameters() );

optimizer->SetMaximumStepLength( 4.00 );
optimizer->SetMinimumStepLength( 0.01  );
optimizer->SetNumberOfIterations( 100  );

optimizer->MaximizeOff();
```

Corregistro de Imagens

```
try
    {
        registrator->StartRegistration ();
    }
catch( itk::ExceptionObject & excp )
    {
        std::cerr << “Error in registration” << std::endl;
        std::cerr << excp << std::endl;
    }


transform->SetParameters(
                    registrator->GetLastTransformParameters()  );
```

Corregistro de Imagens

```
typedef   itk::ResampleImageFilter<
                    FixedImageType , MovingImageType >     ResamplerType;

ResamplerType ::Pointer   resampler  = ResamplerType::New();


resampler->SetTransform ( transform );
resampler->SetInput( movingImageReader->GetOutput() );

FixedImageType ::Pointer   fixedImage  = fixedImageReader->GetOutput();
resampler->SetOrigin( fixedImage->GetOrigin() );
resampler->SetSpacing( fixedImage->GetSpacing() );
resampler->SetSize(
                  fixedImage->GetLargestPossibleRegion()->GetSize() );

resampler->Update();
```

Monitorando o Corregistro

Observando o Corregistro

```
#include ”itkCommand.h”

class CommandIteration : public itk::Command {
public:
  typedef CommandIteration  Self;
  typedef itk::Command          SuperClass;
  typedef itk::SmartPointer< Self >    Pointer;
  itkNewMacro( Self );

protected:
  CommandIteration() {};

public:
  typedef  itk::RegularStepGradientDescentOptimizer   OptimizerType;
  typedef      const OptimizerType                          * OptimizerPointer;
```

Observando o Corregistro

```
void Execute(  itk::Object * caller, const itk::EventObject & event )
  {
  this-> Execute(  (const itk::Object *) caller,  event  );
  }


void Execute(  const itk::Object * caller, const itk::EventObject & event )
  {
    OptimizerPointer optimizer =
                              dynamic_cast< OptimizerPointer >( caller );

     if( typeid( event  ) == typeid( itk::IterationEvent ) )
       {
       std::cout << optimizer->GetCurrentIteration() << " : ";
       std::cout << optimizer->GetValue() << " : ";
       std::cout << optimizer->GetCurrentPosition() << std::endl;
       }
  }
```

Corregistro de Imagens

```
CommandIteration::Pointer   observer  = CommandIteration::New();

optimizer->AddObserver( itk::IterationEvent(), observer )

try
    {
        registrator->StartRegistration ();
    }
catch( itk::ExceptionObject & excp )
    {
        std::cerr << "Error in registration" << std::endl;
        std::cerr << excp << std::endl;
    }
```

Metricas de Similaridade entre Imagens

Métricas de Similaridade

Quão similar e a

imagem A em relação a imagem B ?

Métricas de Similaridade

A Imagem B

assemelha à Imagem A melhor

que a Imagem C ?

Métricas de Similaridade

Similaridade (A , B) $\gtrless$ Similaridade(A , C)

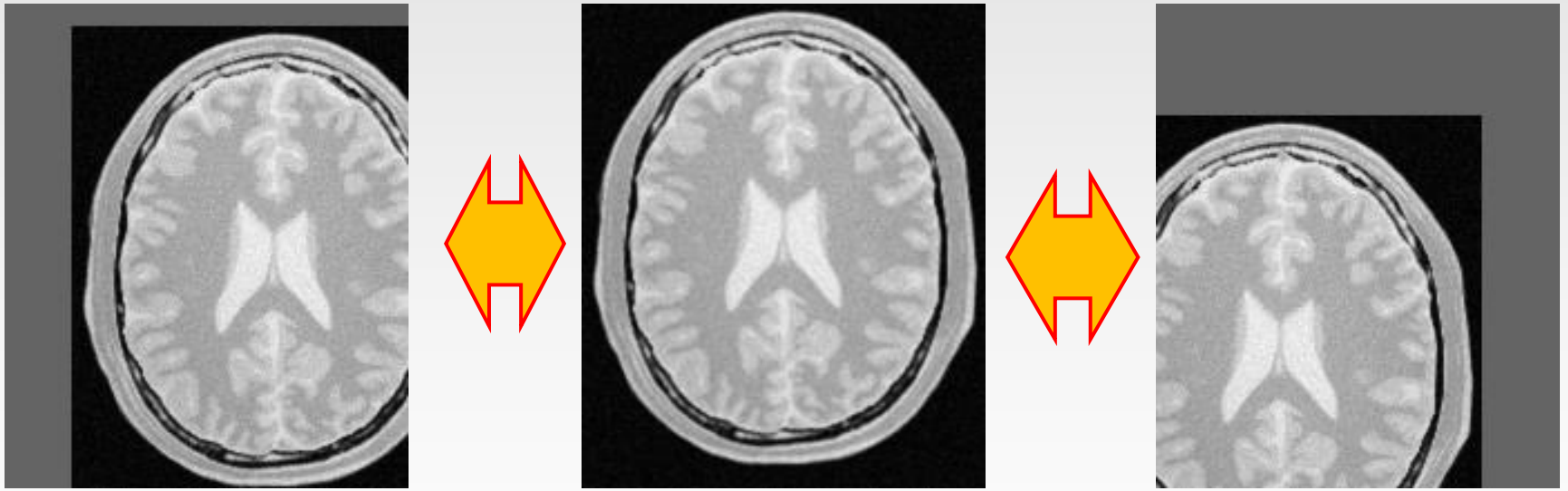


Imagem B Imagem A Imagem C

Métricas de Similaridade

Similaridade(A , B)

Métrica mais Simples

Diferença Quadrática Média

Diferenças Quadráticas Média

Para cada pixel em A

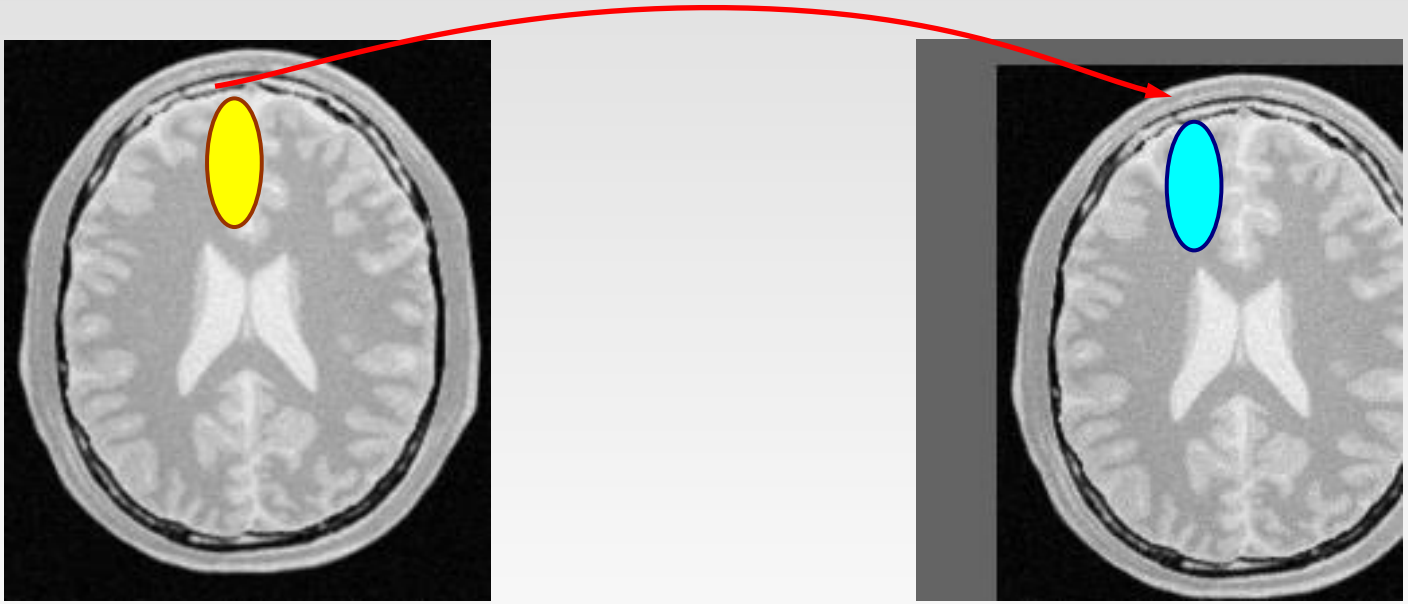


Imagem A

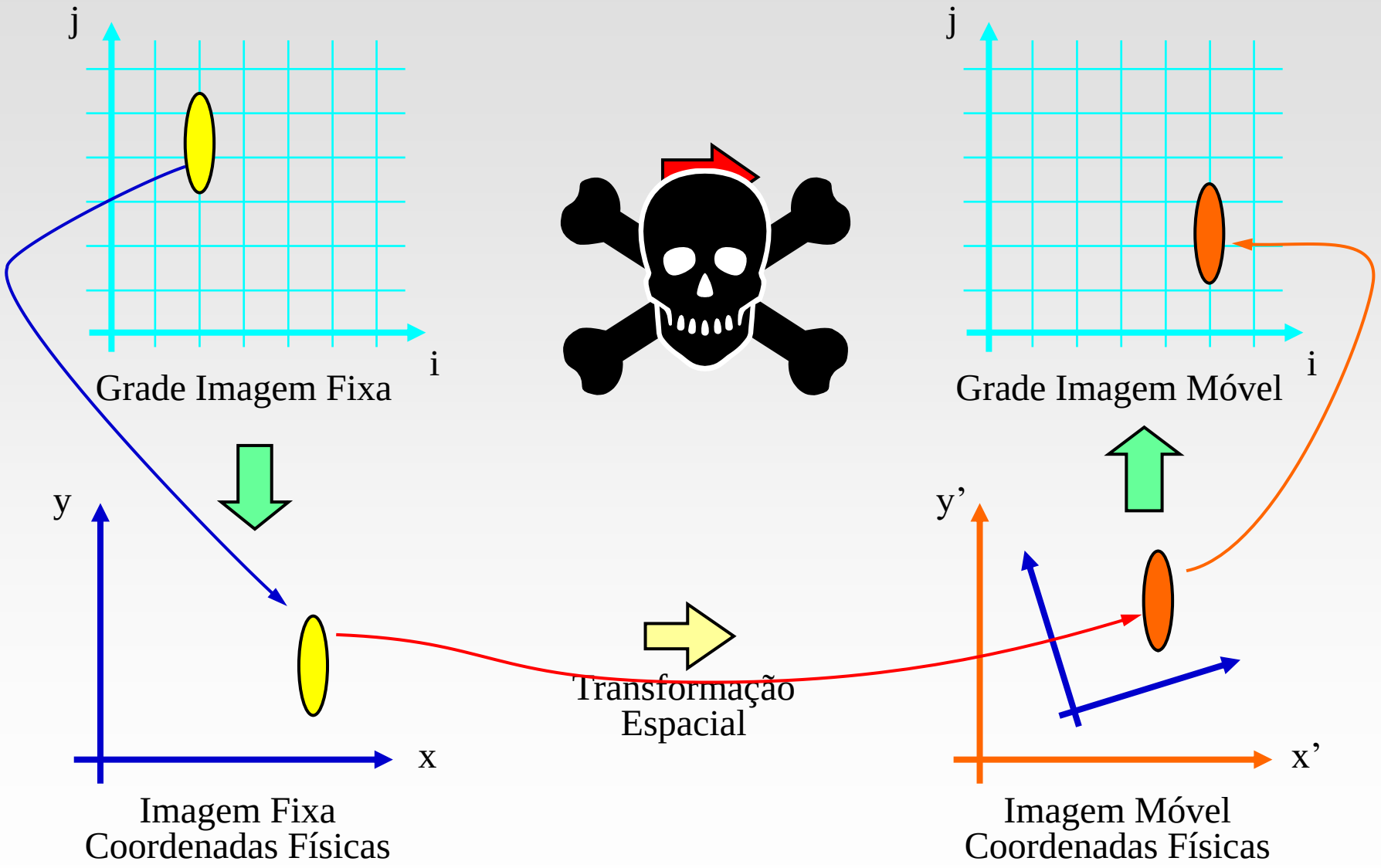
Imagem B

Diferença(índice) = A(índice) – B(índice)

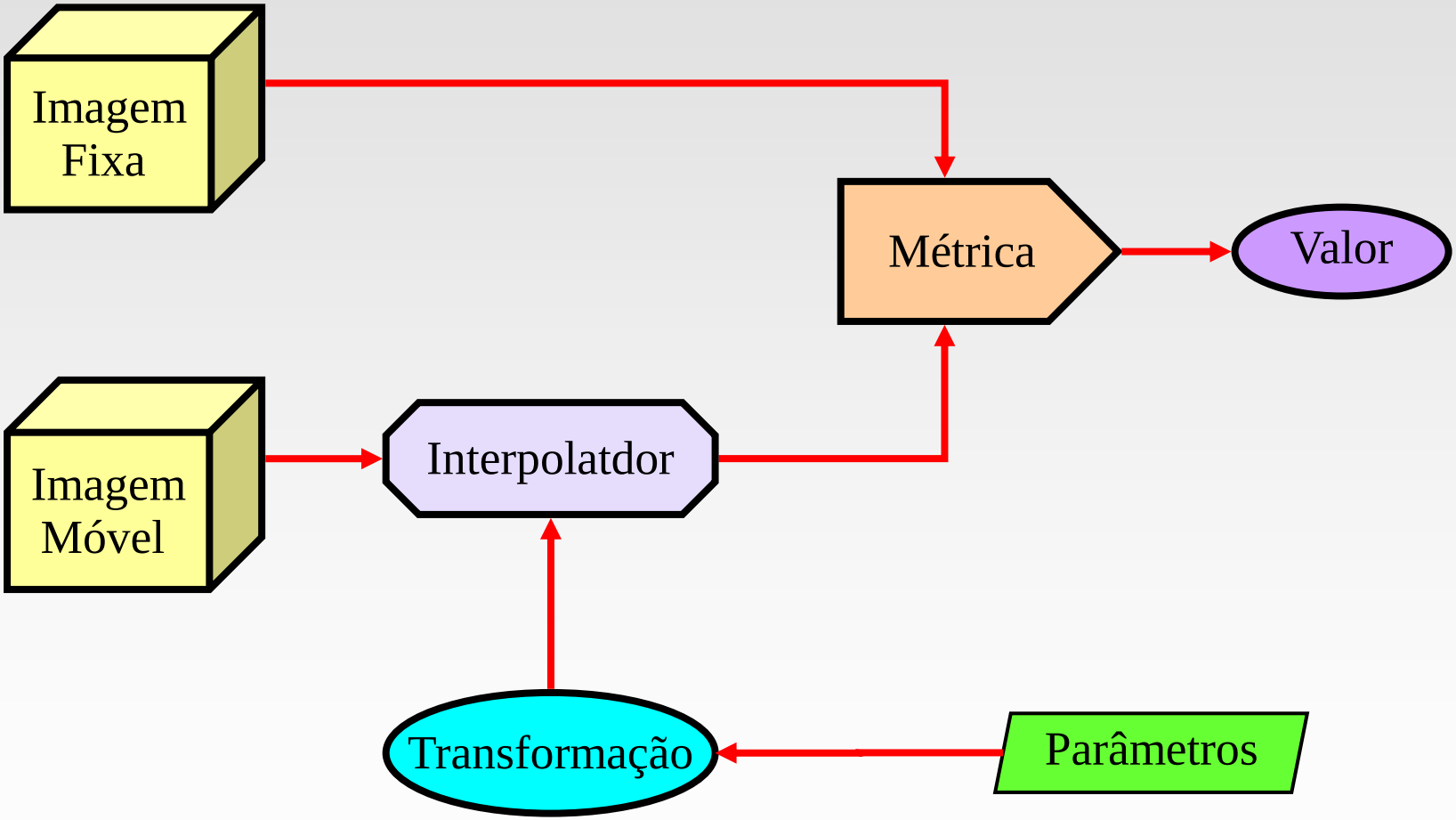
Soma += Diferença(índice) ^{2}

Similaridade(A , B) = Soma / númeroDePixels

Para cada pixel na Imagem Fixa



Métricas de Similaridade



Diferenças Quadráticas Média

```
#include "itkImage.h"
#include "itkMeanSquaresImageToImageMetric.h"
#include "itkLinearInterpolateImageFunction.h"
#include "itkTranslationTransform.h"

typedef itk::Image< char, 2 >  ImageType;

ImageType::ConstPointer fixedImage  = GetFixedImage();
ImageType::ConstPointer movingImage = GetMovingImage();

typedef itk::LinearInterpolateImageFunction<
                                ImageType,
                                double >
                                    InterpolatorType;

InterpolatorType::Pointer interpolator = InterpolatorType::New();


typedef itk::TranslationTransform< double, 2 >  TransformType;

TransformType::Pointer transform = TransformType::New();
```

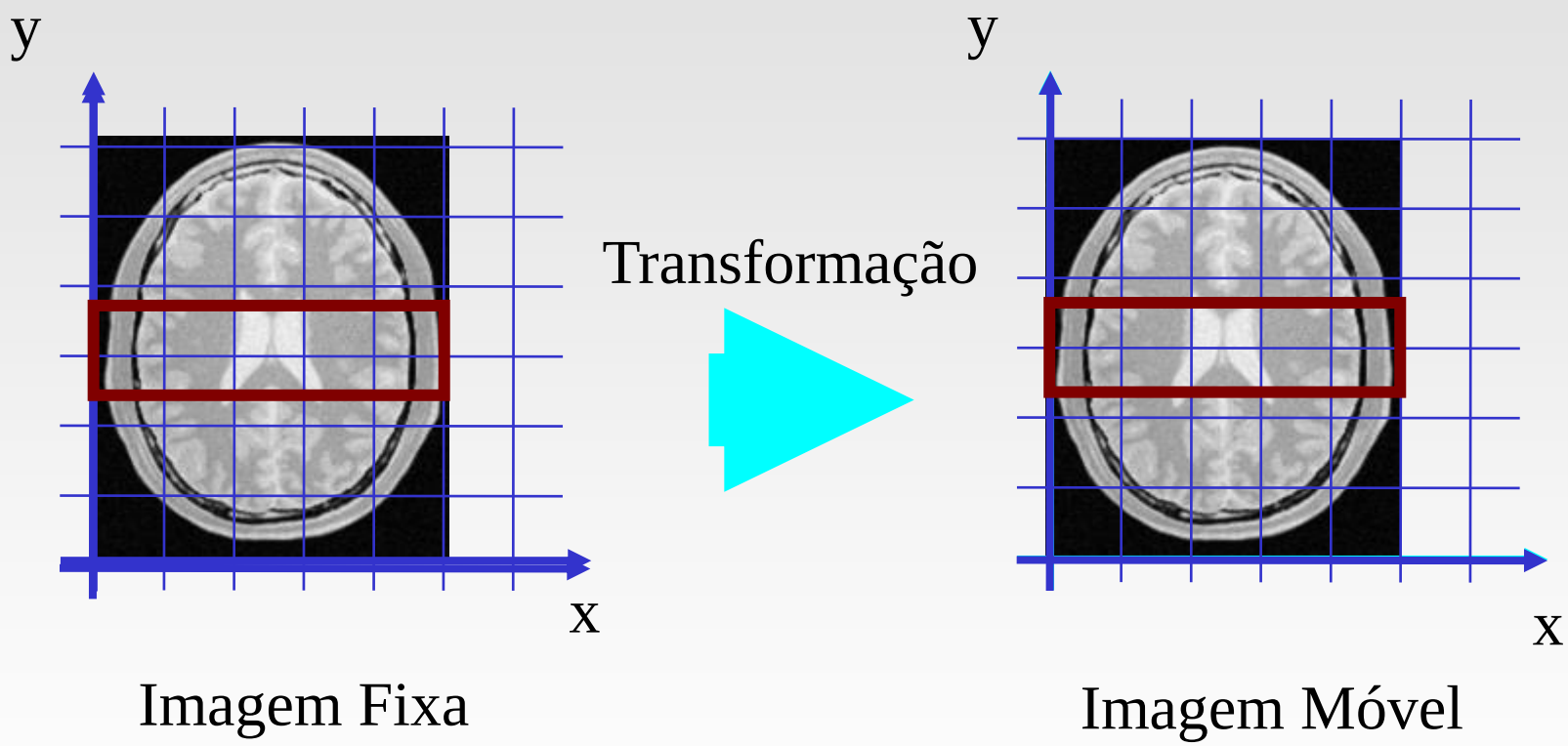
Diferenças Quadráticas Média

```
typedef itk::MeanSquaresImageToImageMetric<
                    ImageType, ImageType >  MetricType;

MetricType::Pointer metric = MetricType::New();


metric->SetInterpolator( interpolator );
metric->SetTransform( transform );

metric->SetFixedImage(  fixedImage  );
metric->SetMovingImage( movingImage );

MetricType::TransformParametersType translation( Dimension );

translation[0] = 12;
translation[1] = 27;

double value = metric->GetValue( translation );
```

Diferenças Quadráticas Média

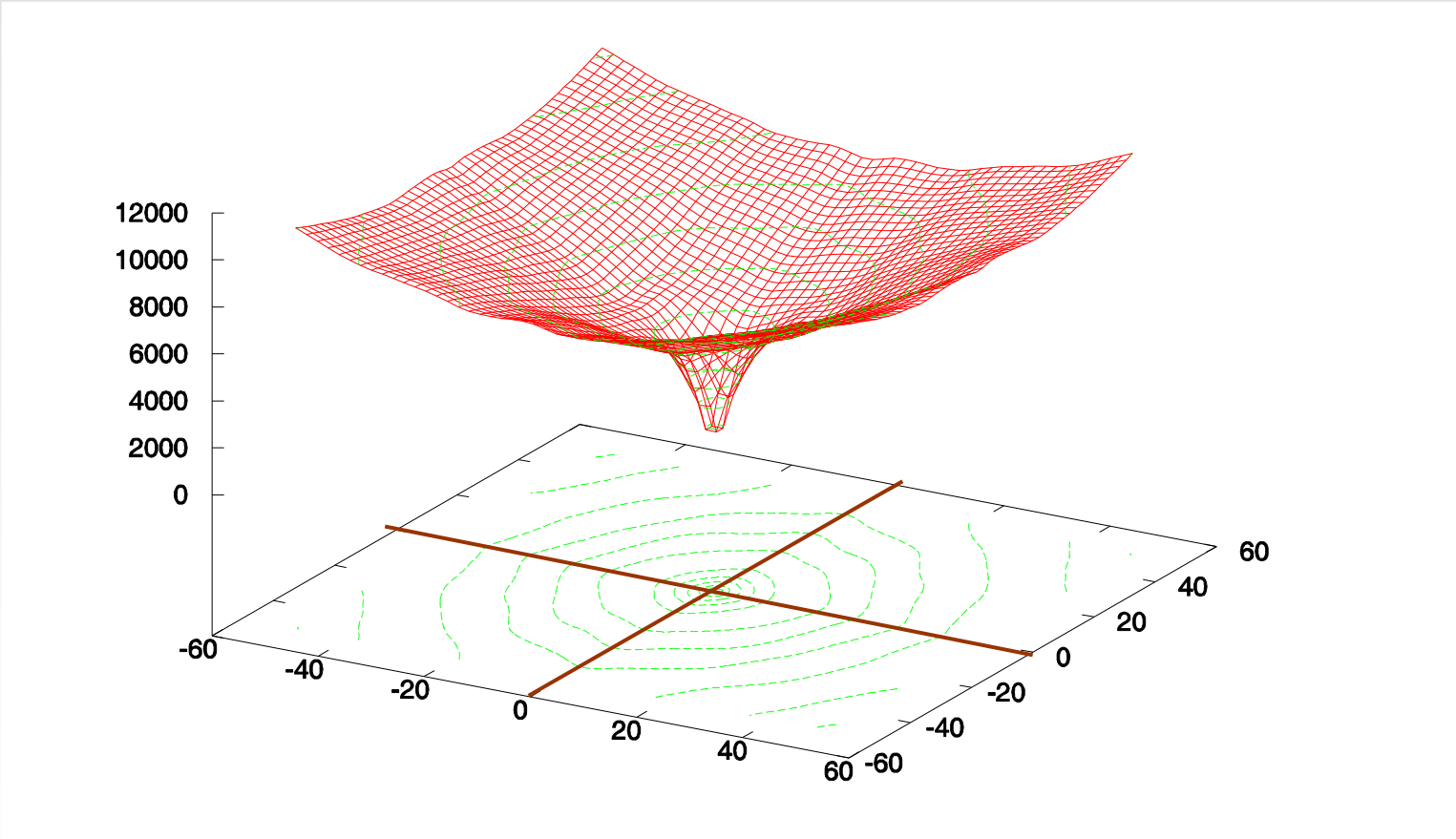
```
MetricType::TransformParametersType translation( Dimension );

double value[21][21];

for( int dx = 0; dx <= 20; dx++)
      {

      for( int dy = 0; dy <= 20; dy++)
                {

                translation[0] = dx;
                translation[1] = dy;

                value[dx][dy] = metric->GetValue( translation );

                }
      }
```

Avaliando muitas comparações



Plotando a Métrica

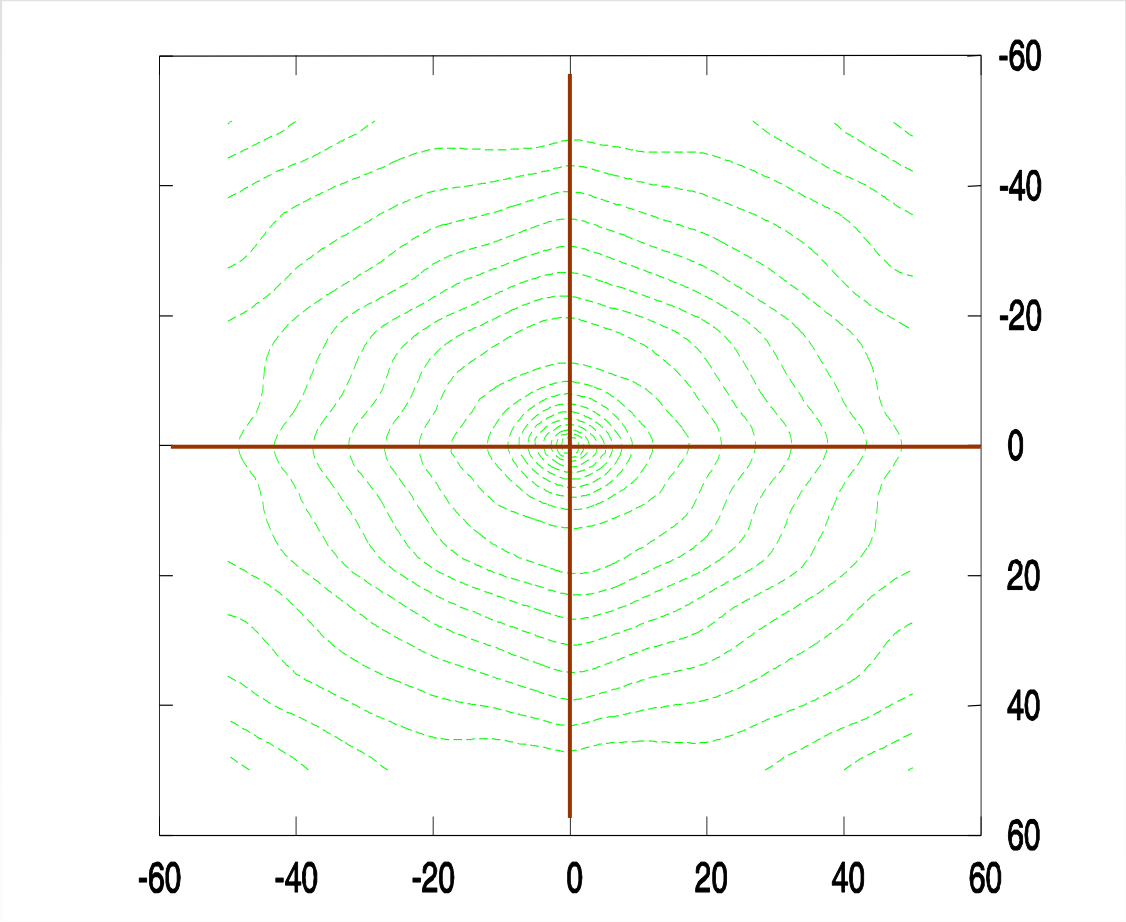
Diferenças Quadráticas Média



Espaço de parâmetros da transformação

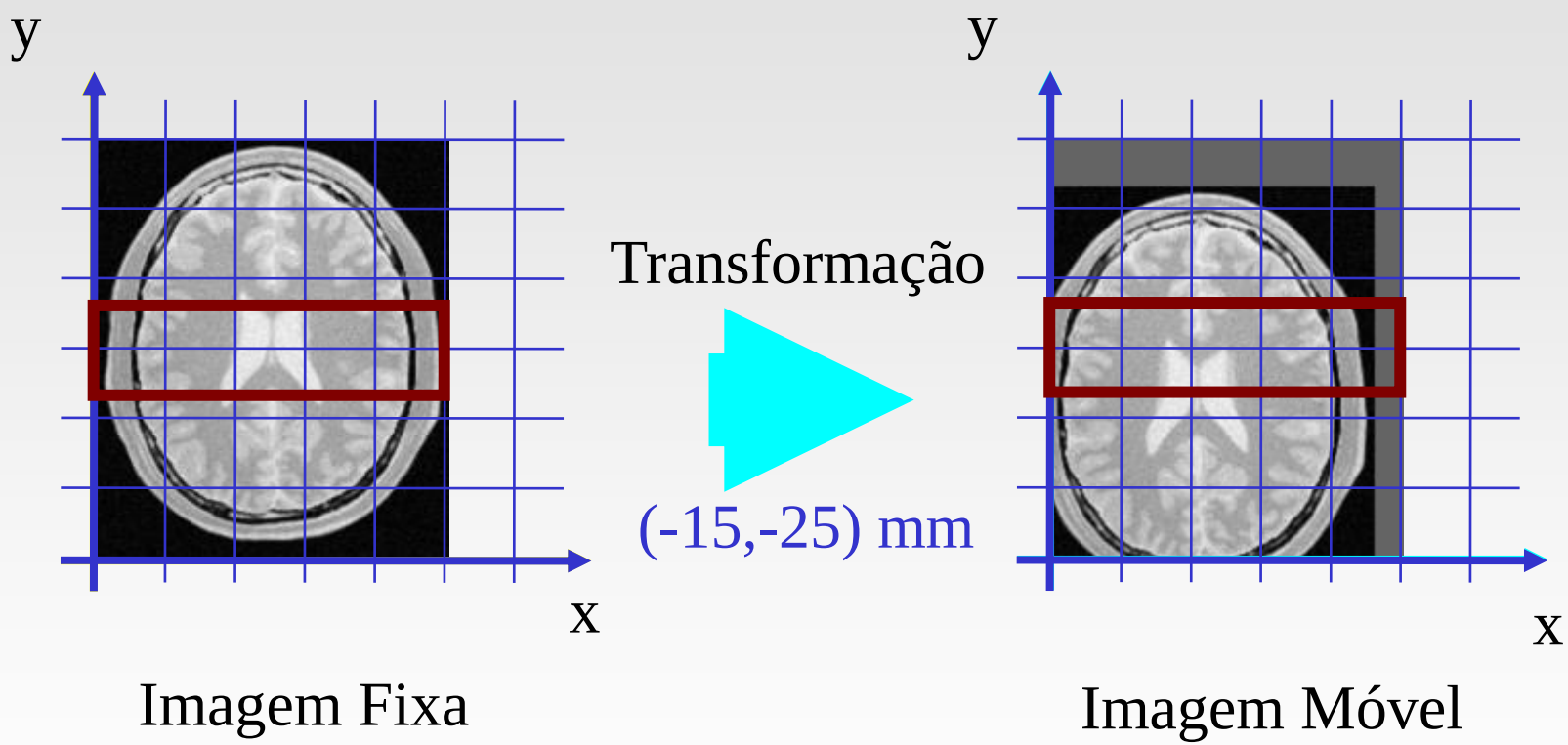
Plotando a Métrica

Diferenças Quadráticas Média



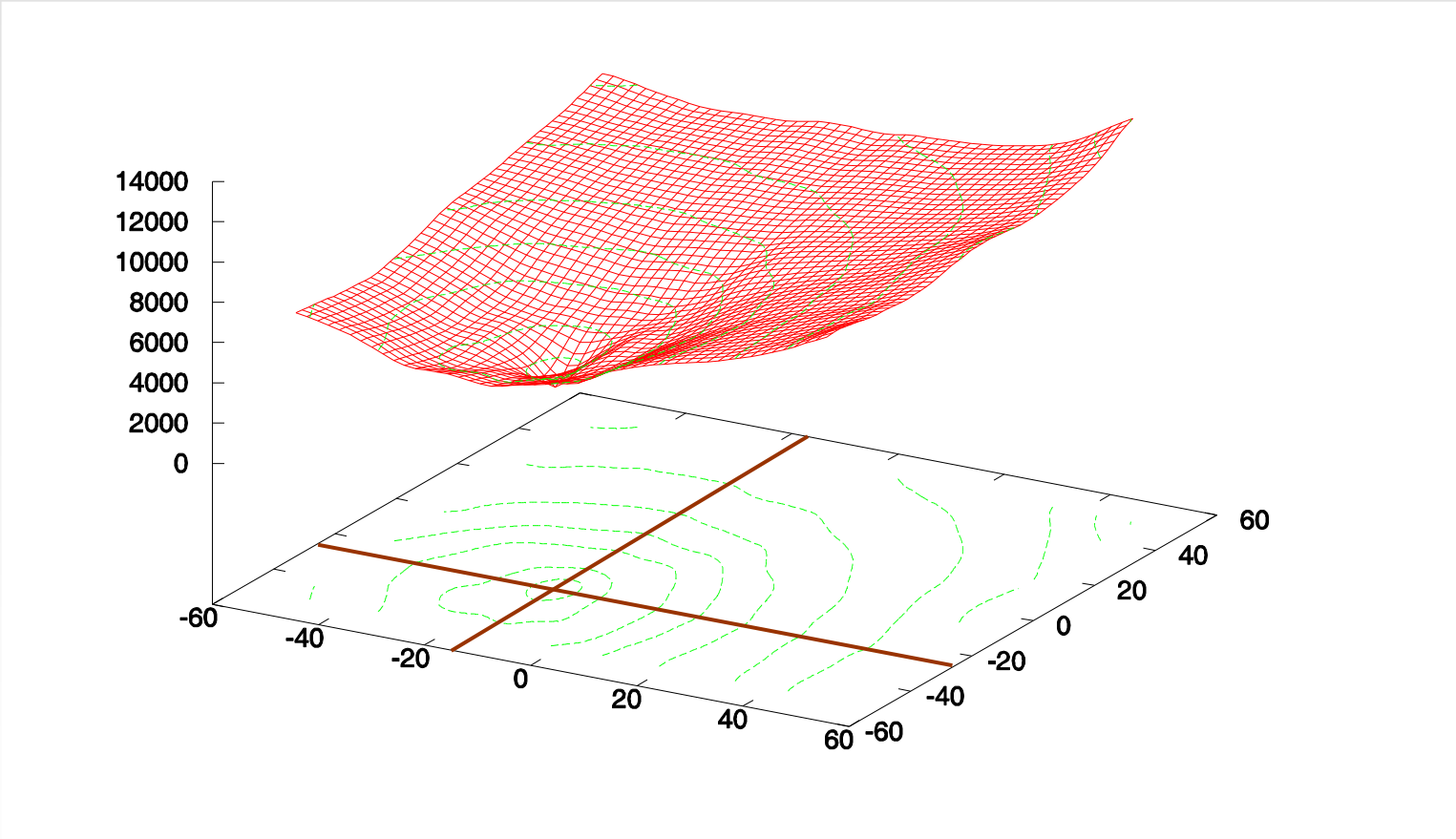
Espaço de parâmetros da transformação

Avaliando muitas comparações



Plotando a Métrica

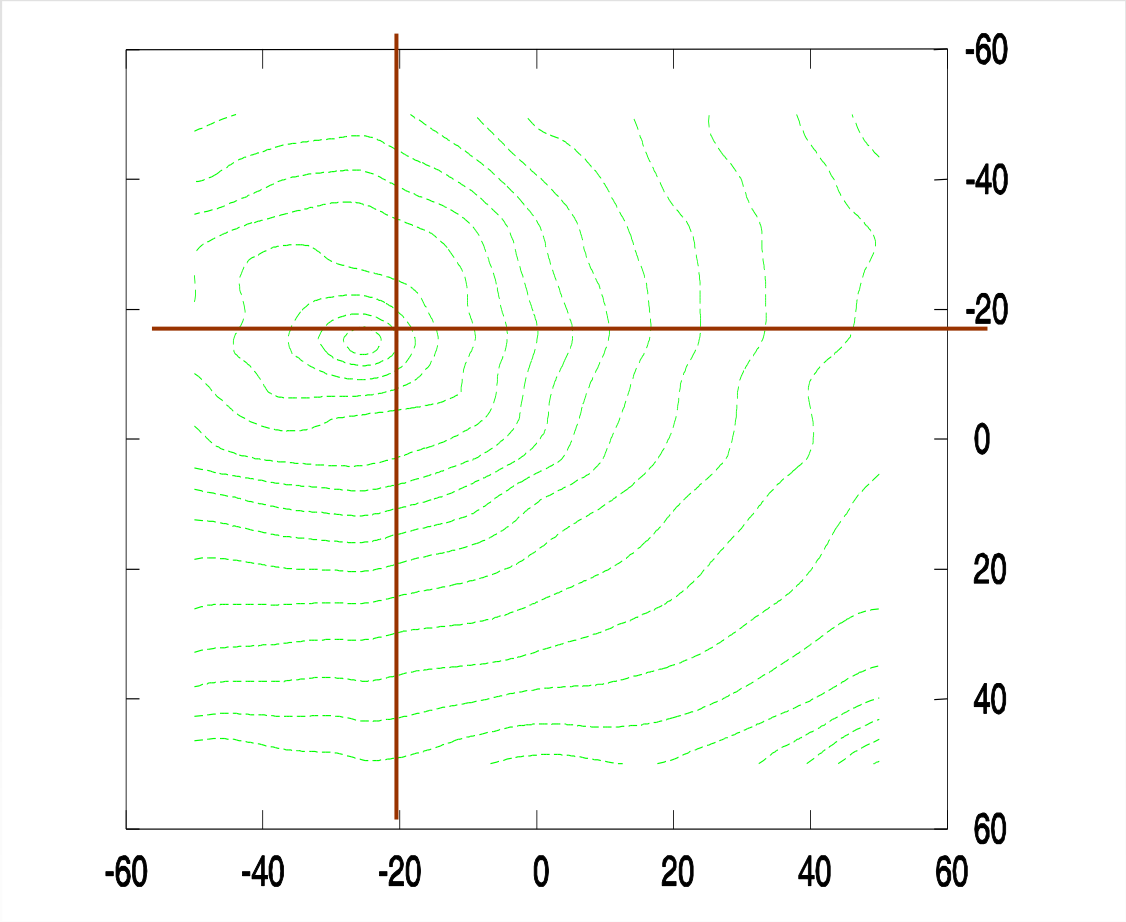
Diferenças Quadráticas Média



Espaço de parâmetros da transformação

Plotando a Métrica

Diferenças Quadráticas Média



Espaço de parâmetros da transformação

O melhor conjunto de parâmetros de transformação

Avaliação do
espaço de parâmetro completo

é equivalente a fazer o
otimização por
busca exaustiva

O melhor conjunto de parâmetros de transformação

Muito seguro

mas

Muito lento

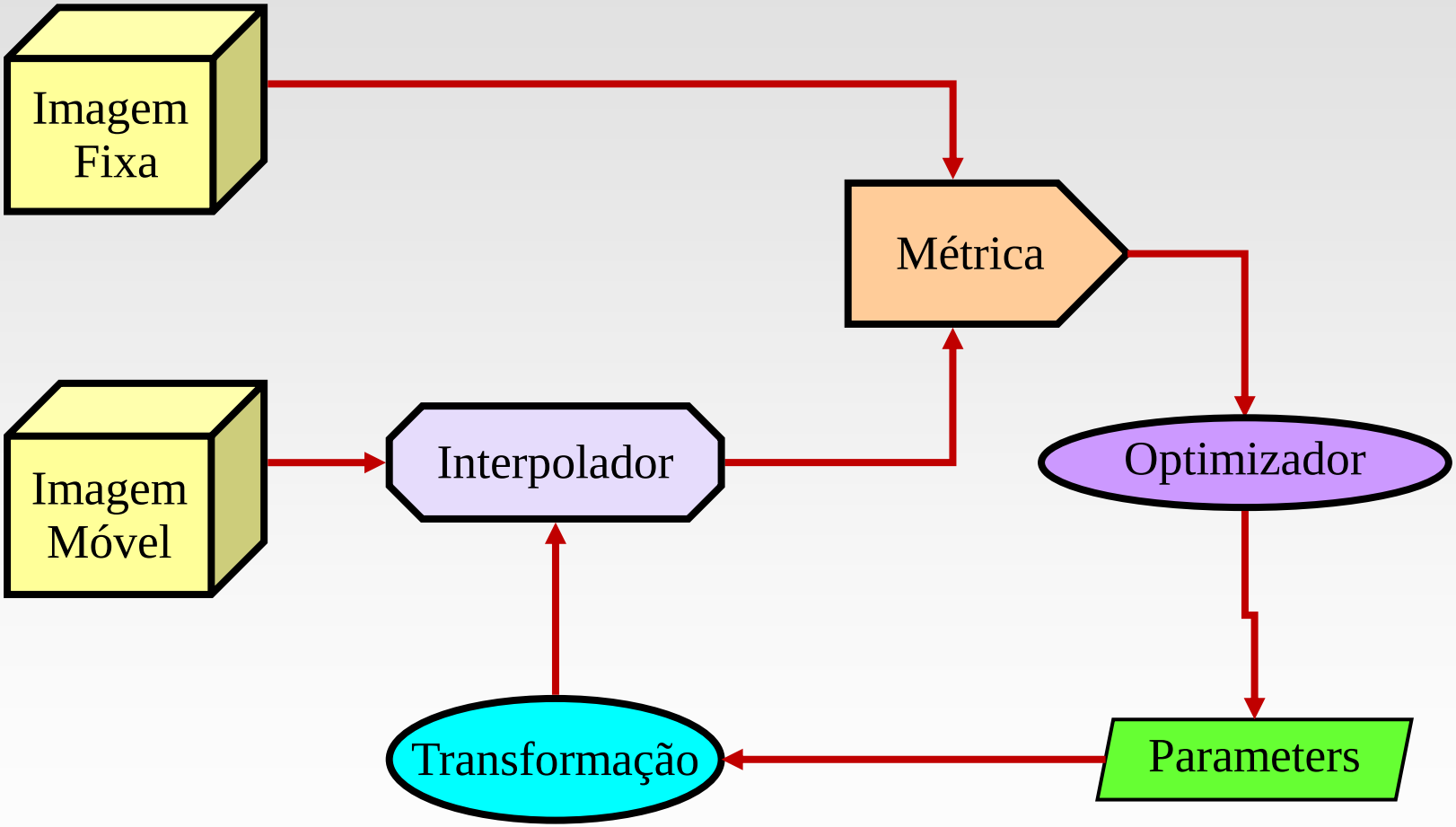
O melhor conjunto de parâmetros de transformação

Melhores Métodos de Optimização

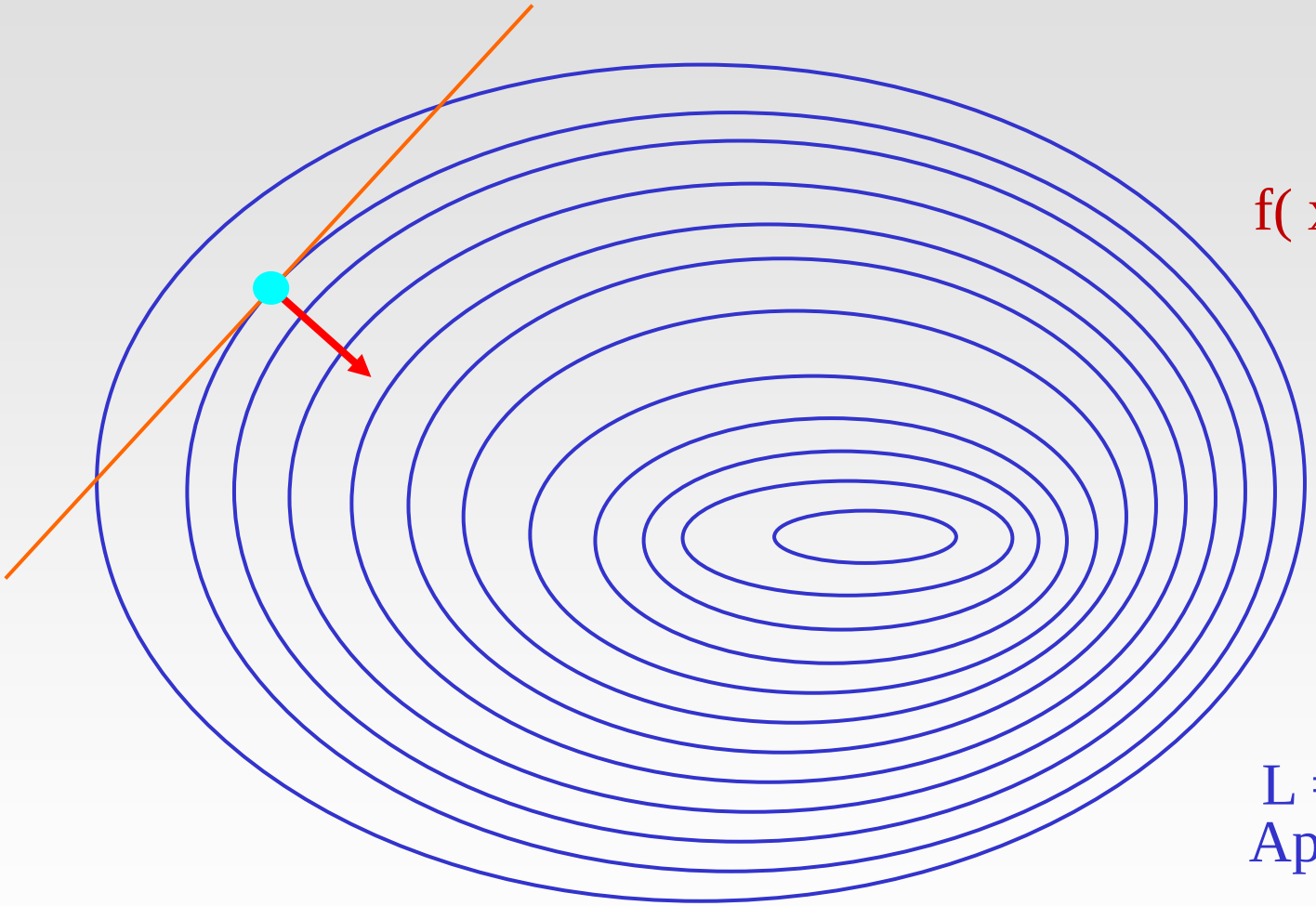
por exemplo

Gradiente Descente

Estrutura de Corregistro de imagens



Optimizador Gradiente Descente



f(x , y)

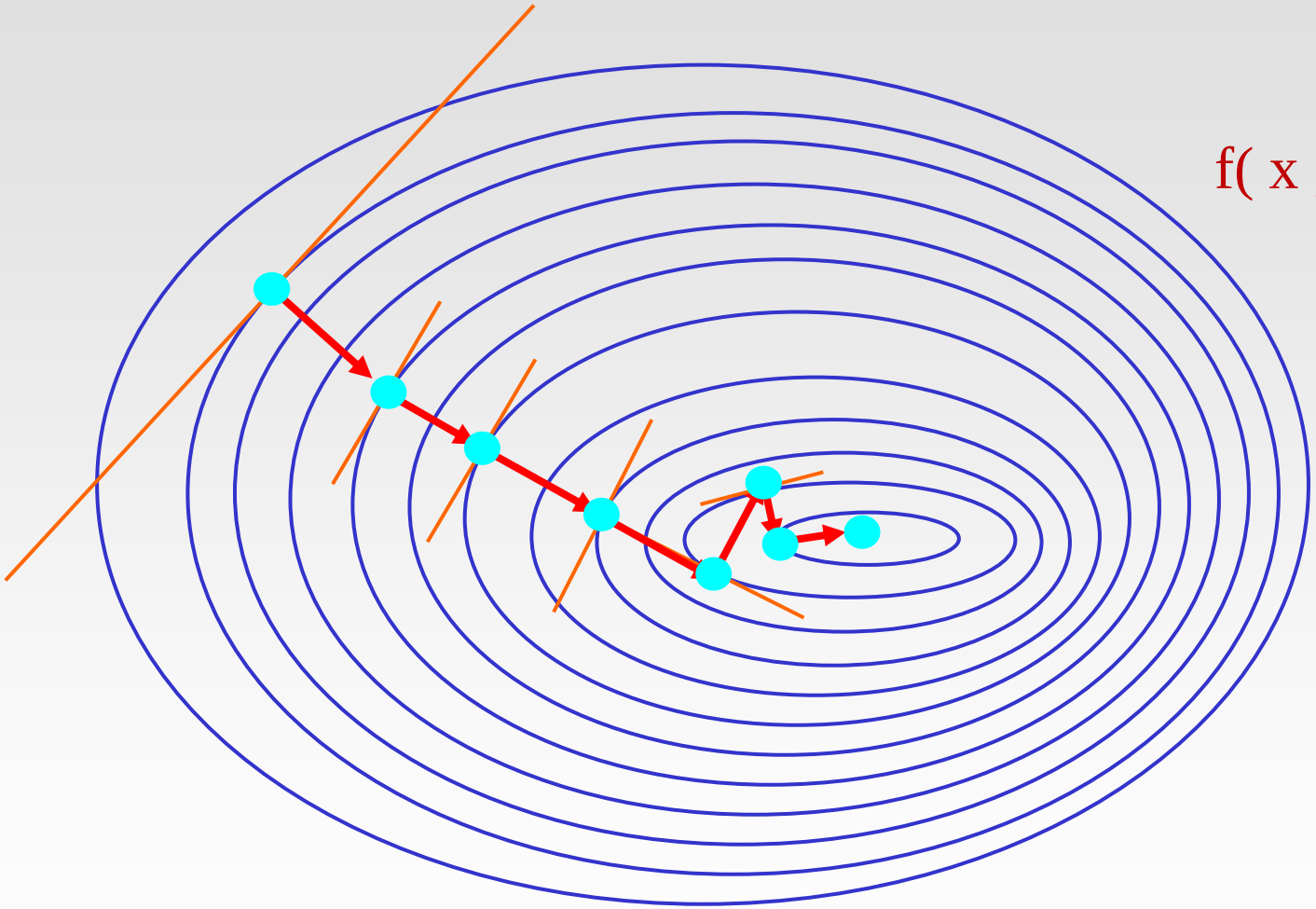
S = Passo

L = Taxa de Aprendizado

G(x , y) ∇f(x , y)

S = L · G(x , y)

Optimizador Gradiente Descente



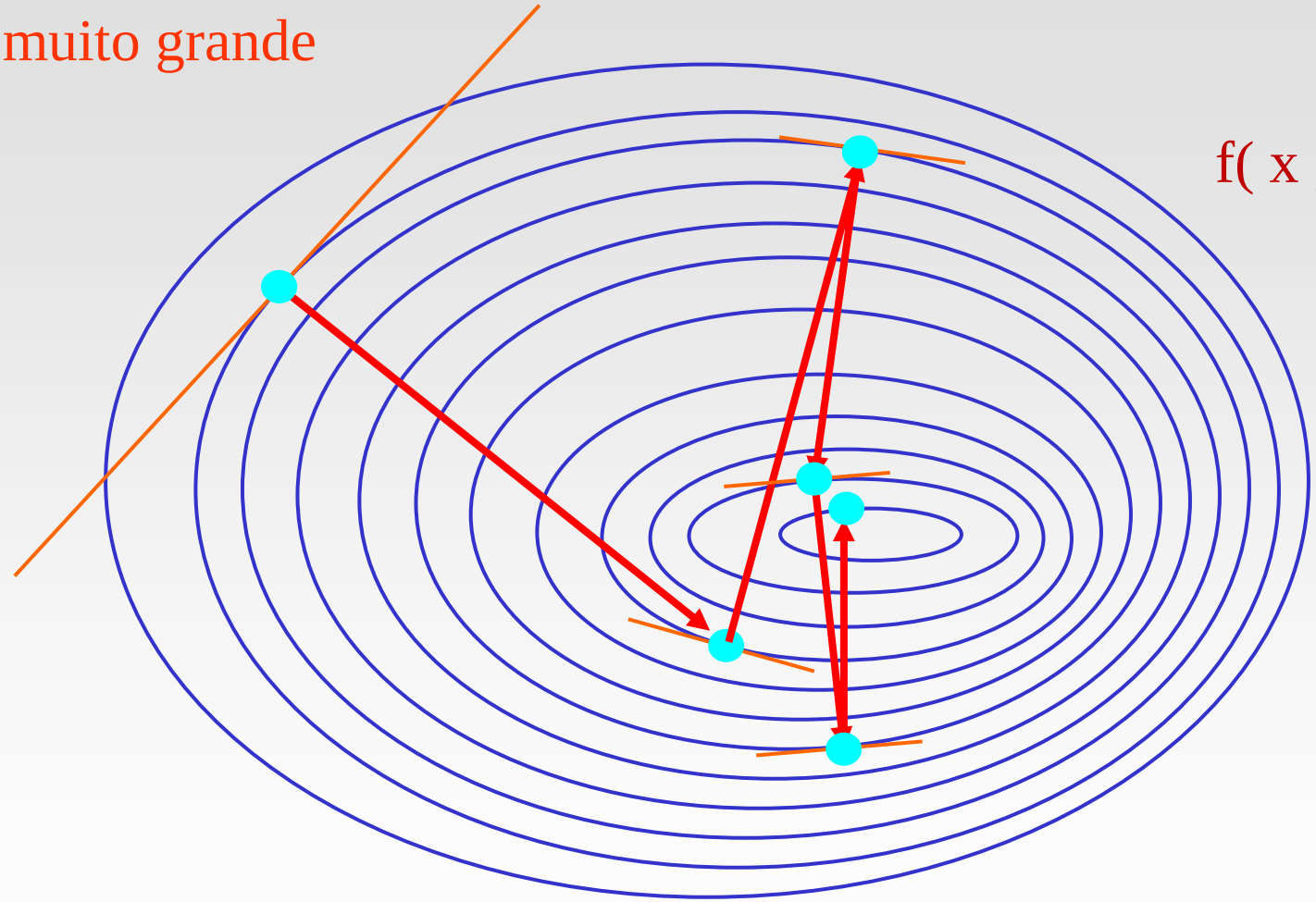
f(x , y)

G(x , y) ∇f(x , y)

S = L · G(x , y)

Optimizador Gradiente Descente

L muito grande



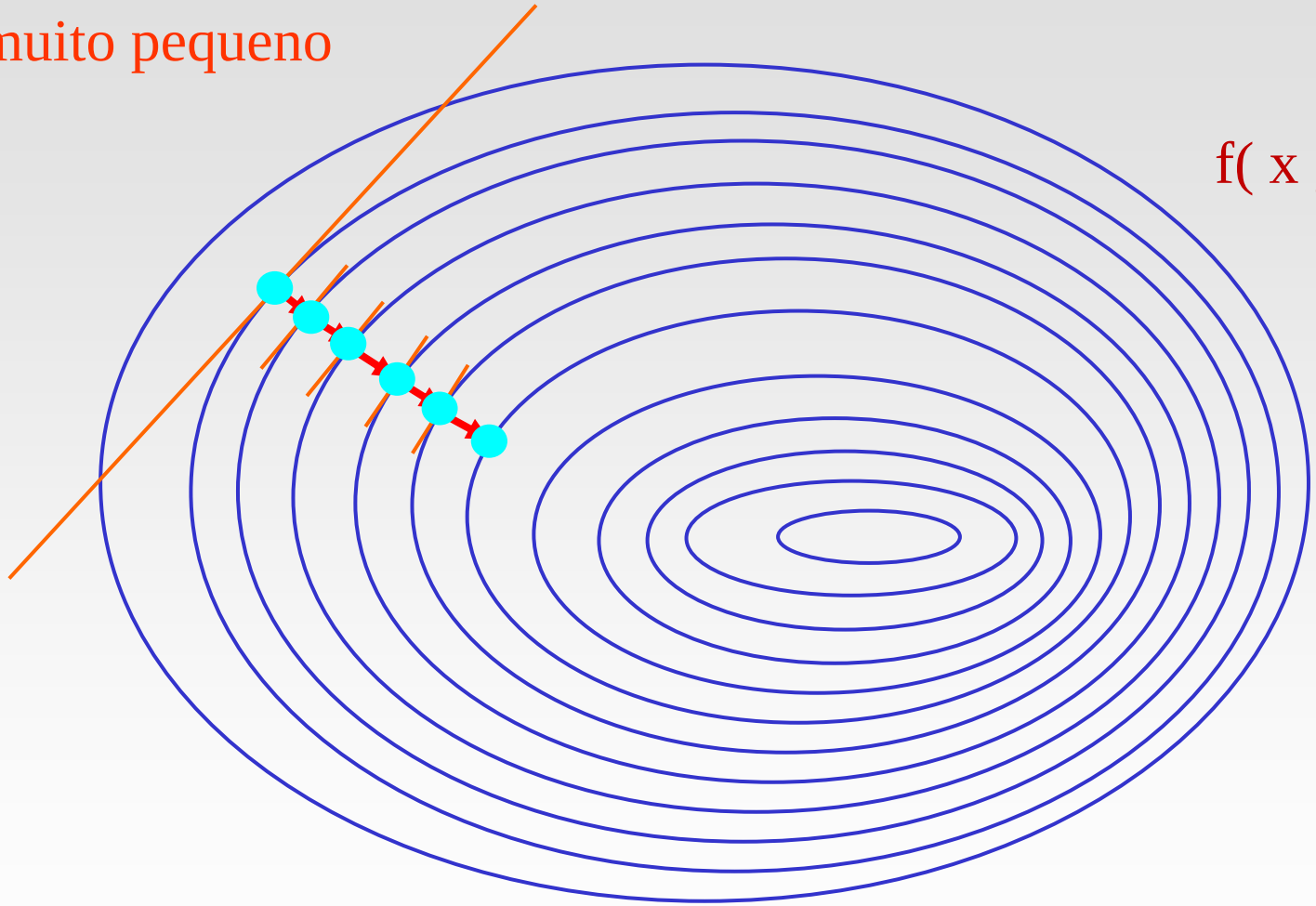
f(x , y)

G(x , y) ∇f(x , y)

S = L · G(x , y)

Optimizador Gradiente Descente

L muito pequeno



f(x , y)

G(x , y) ∇f(x , y)

S = L · G(x , y)

Optimizador Gradiente Descendente

O quê há de errado com este algoritmo ?

Optimizador Gradiente Descendente

S Unidade ? = milimetros

f(x,y) Unidade ? = intensidade

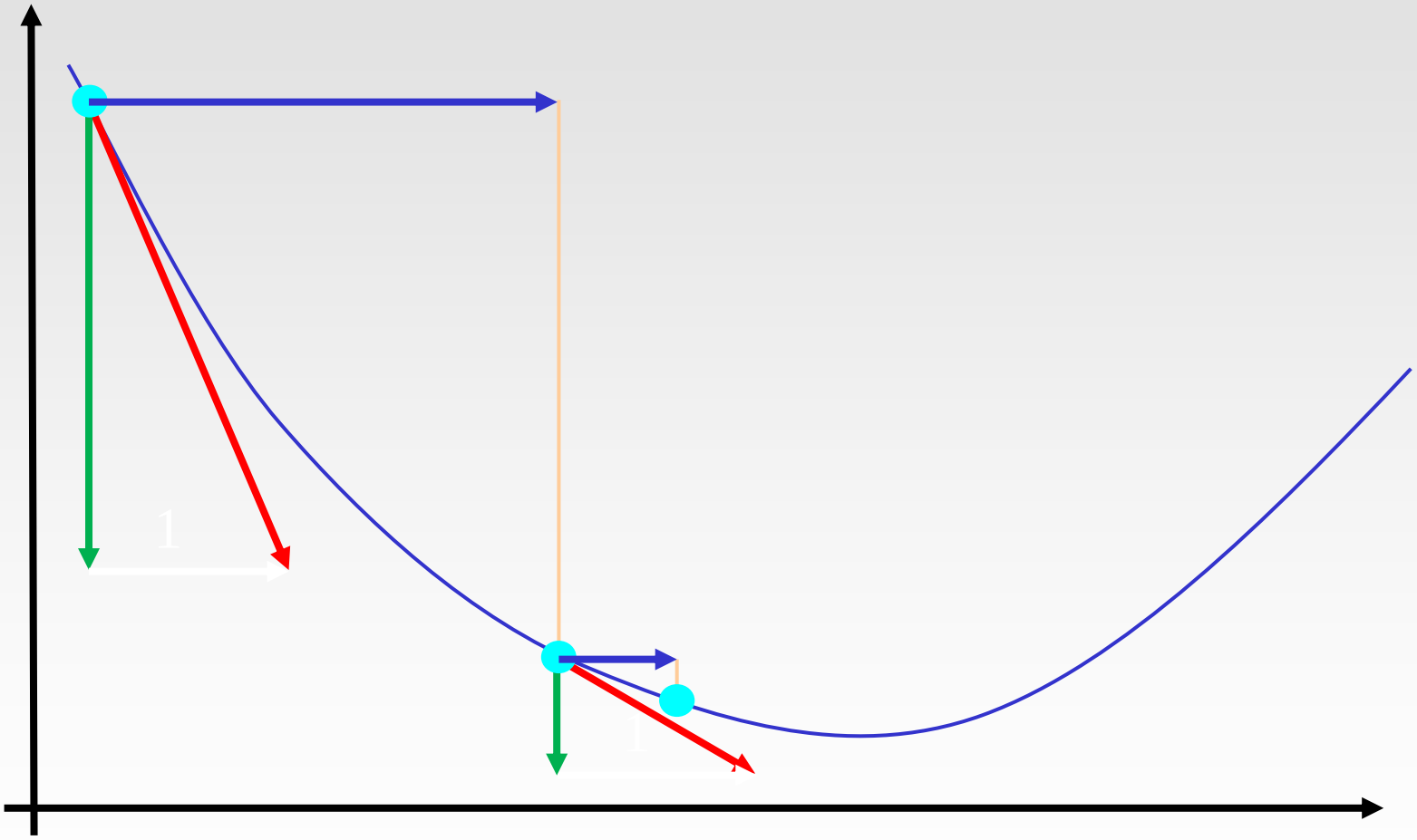
G(x,y) Unidade ? = intensidade / millimetros

$$S = L \cdot G(x , y)$$

L Unidade ? = millimetros^2 / intensidade

Optimizador Gradiente Descendente

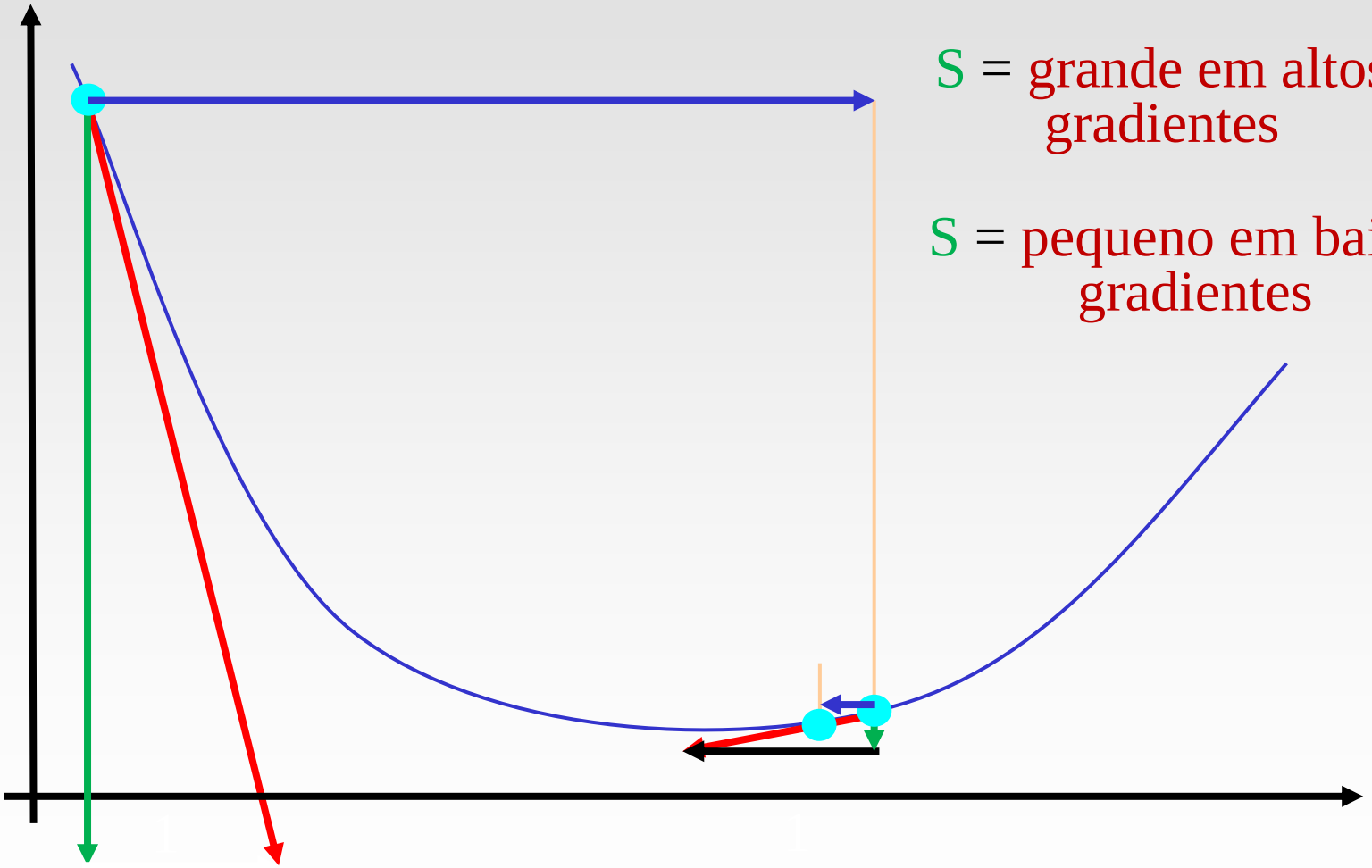
$S = L \cdot G(x)$ $f(x)$



Optimizador Gradiente Descendente

$S = L \cdot G(x)$

$f(x)$



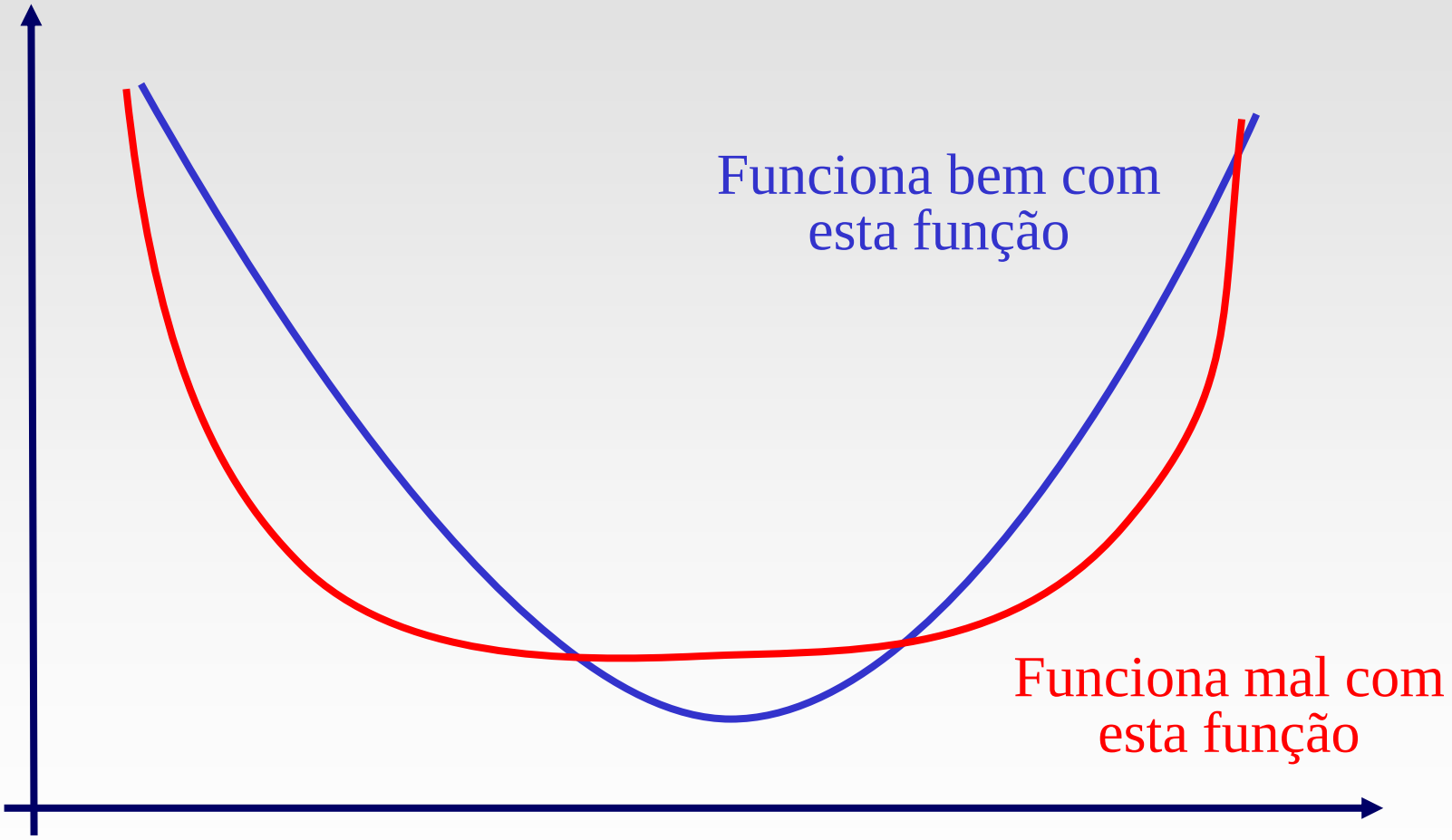
S = grande em altos gradientes

S = pequeno em baixos gradientes

Optimizador Gradiente Descendente

$S = L \cdot G(x)$

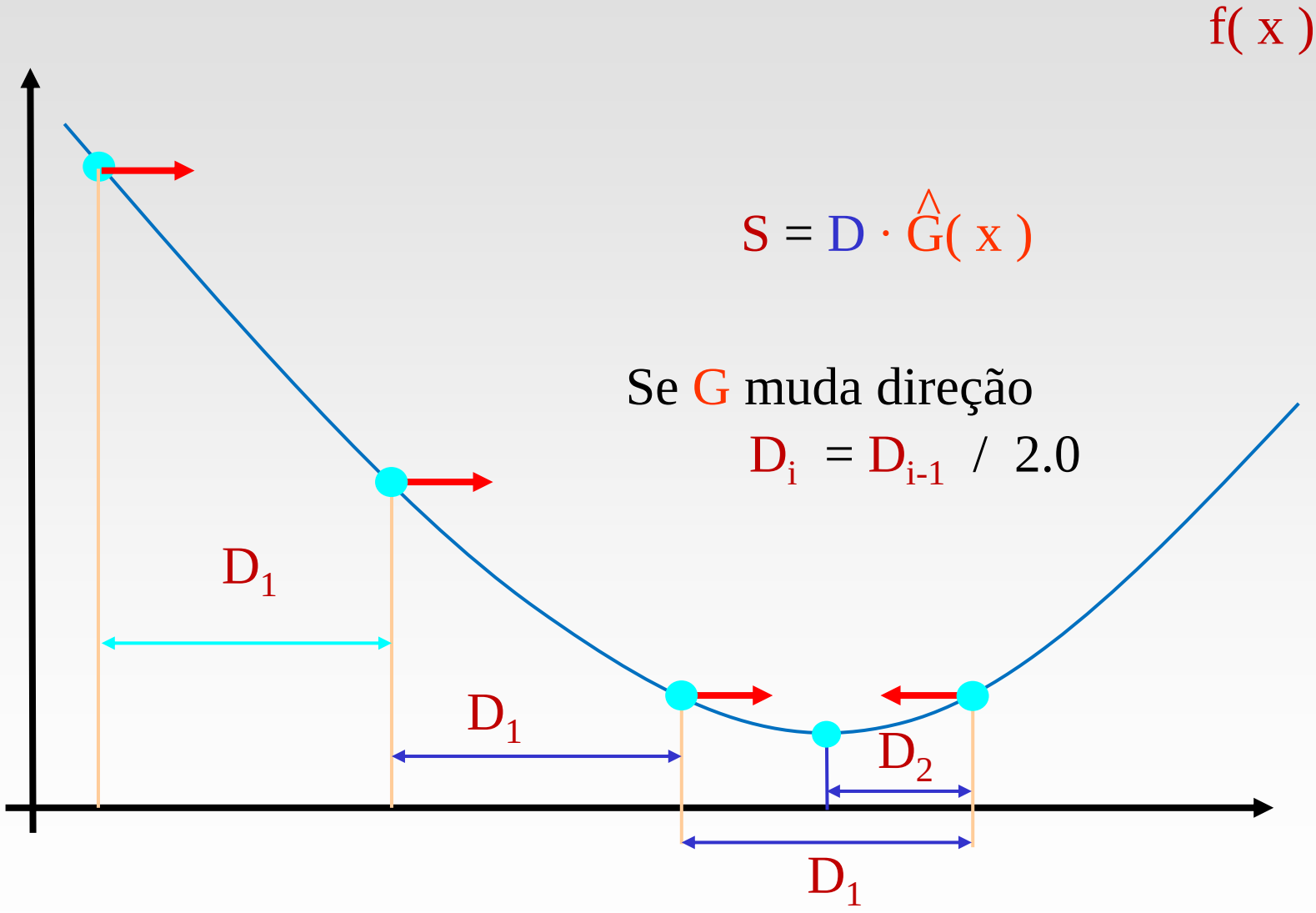
$f(x)$



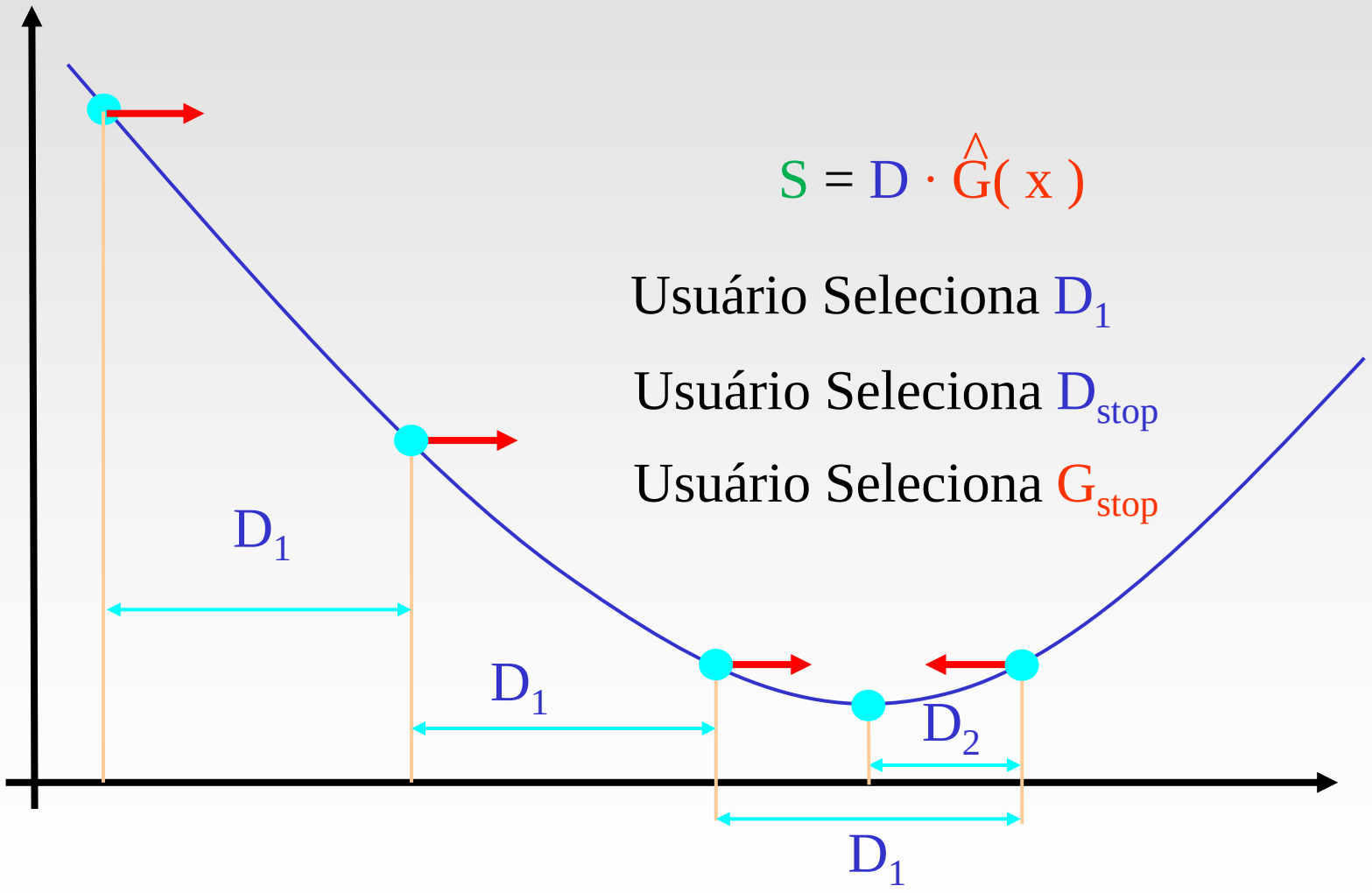
Variante do Gradiente Descendente

Direção Segura !

Gradiente Descendente de Passo Regular



Gradiente Descendente de Passo Regular



Optimizadores são como carro

Cuidado quando estiver dirigindo !

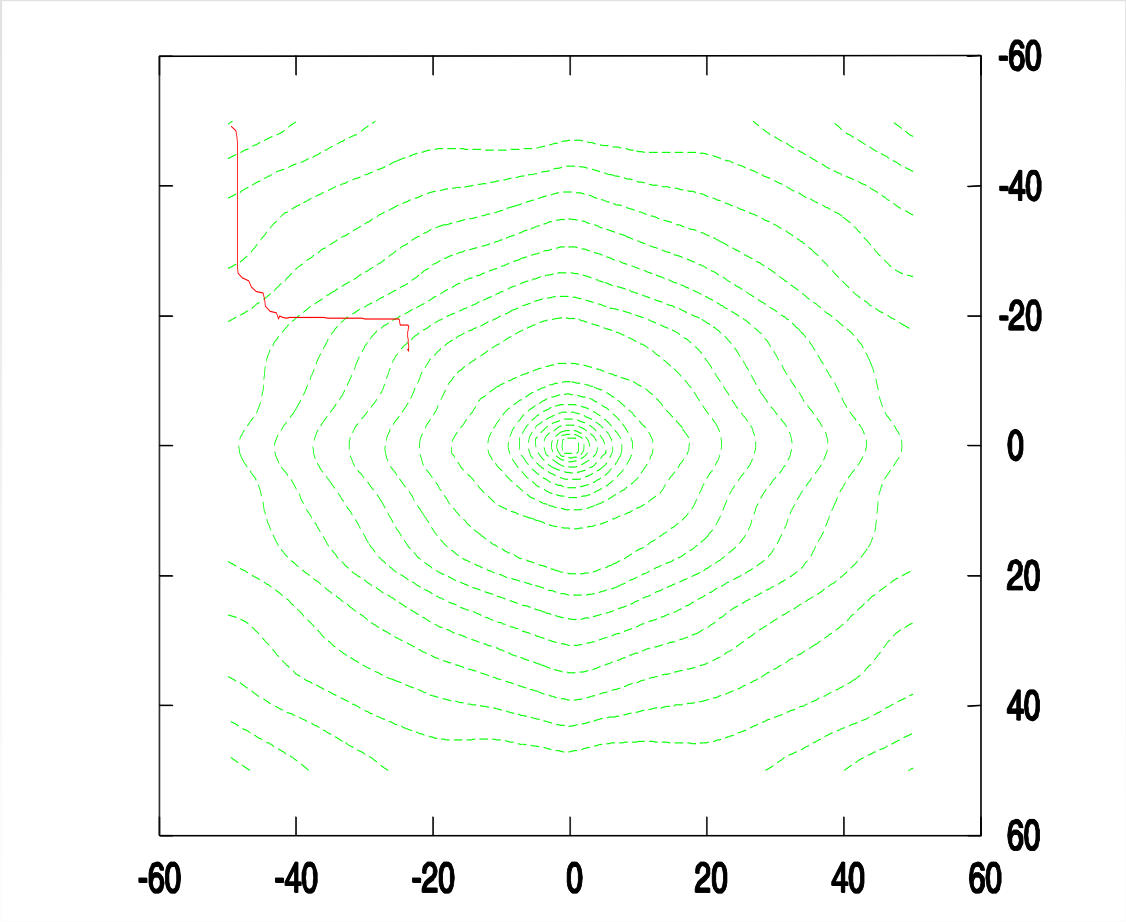
Cuidado com o seu otimizador

Exemplo:

Optimizador corregistrando uma imagem com ela mesma começando com (-15mm, -25mm)

Plotando o trajeto do Otimizador

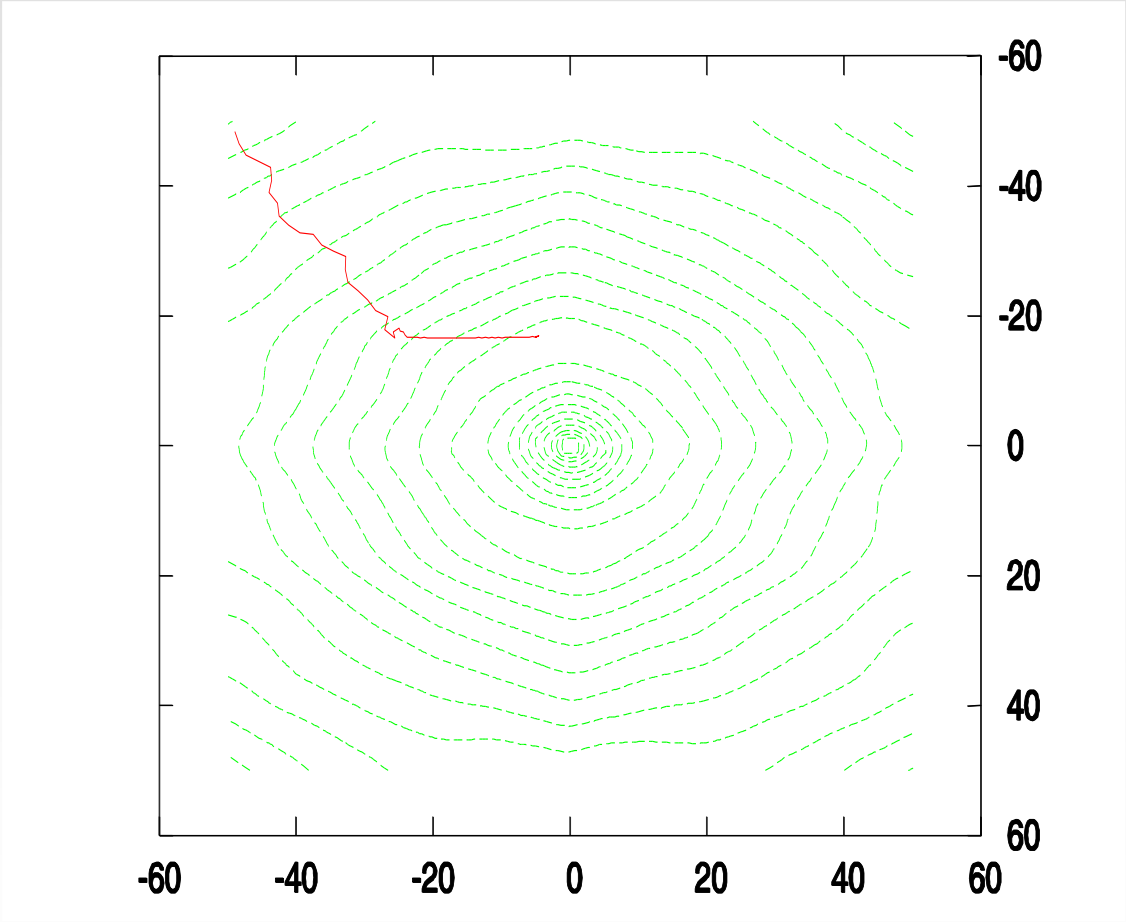
Diferenças Quadráticas Média



Tamanho do passo = 1.0 mm

Plotando o trajeto do Otimizador

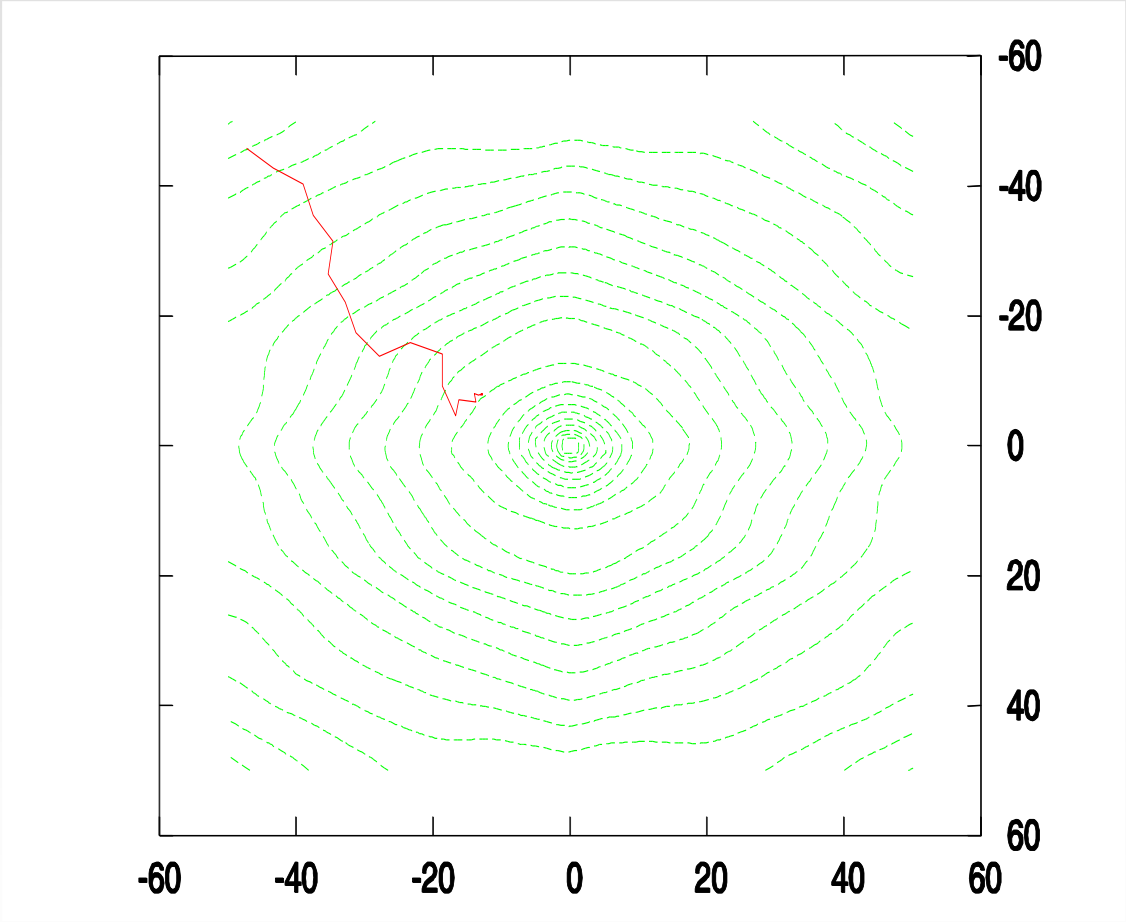
Diferenças Quadráticas Média



Tamanho do passo = 2.0 mm

Plotando o trajeto do Otimizador

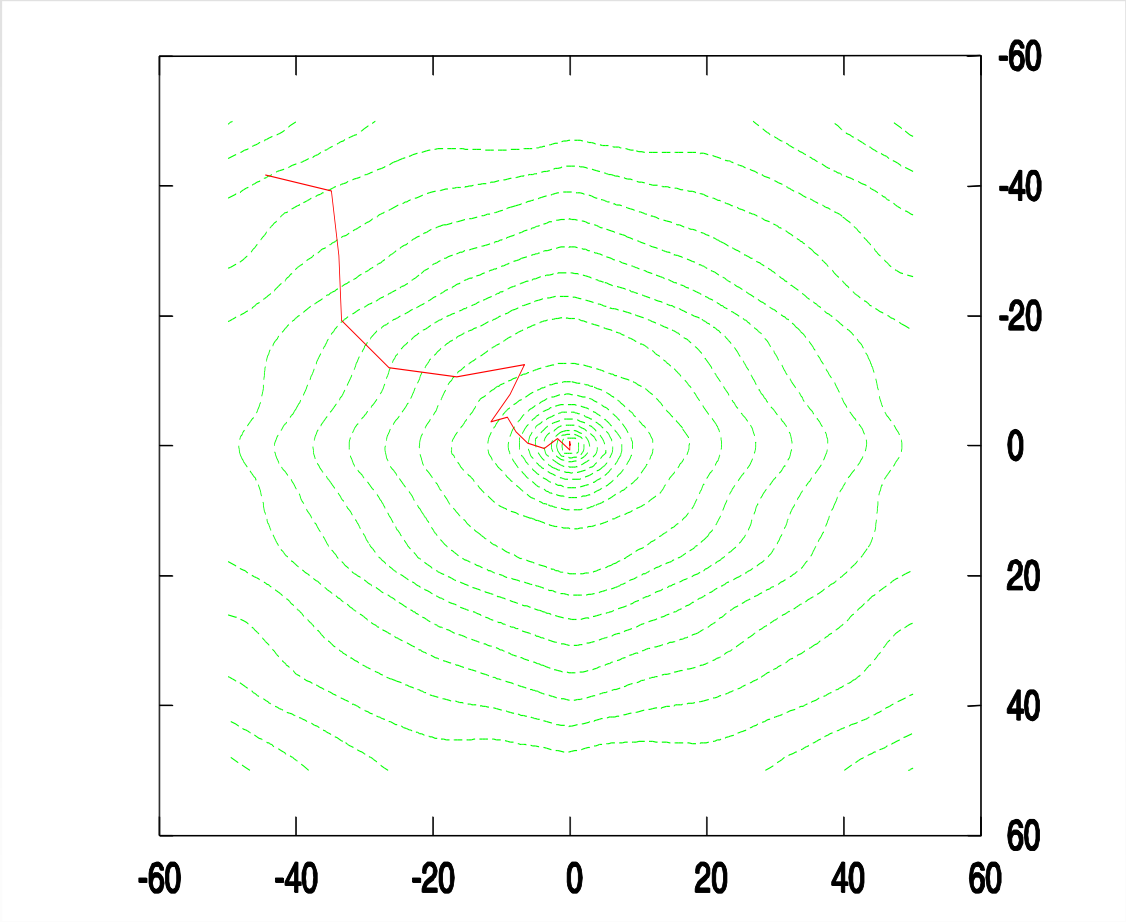
Diferenças Quadráticas Média



Tamanho do passo = 5.0 mm

Plotando o trajeto do Otimizador

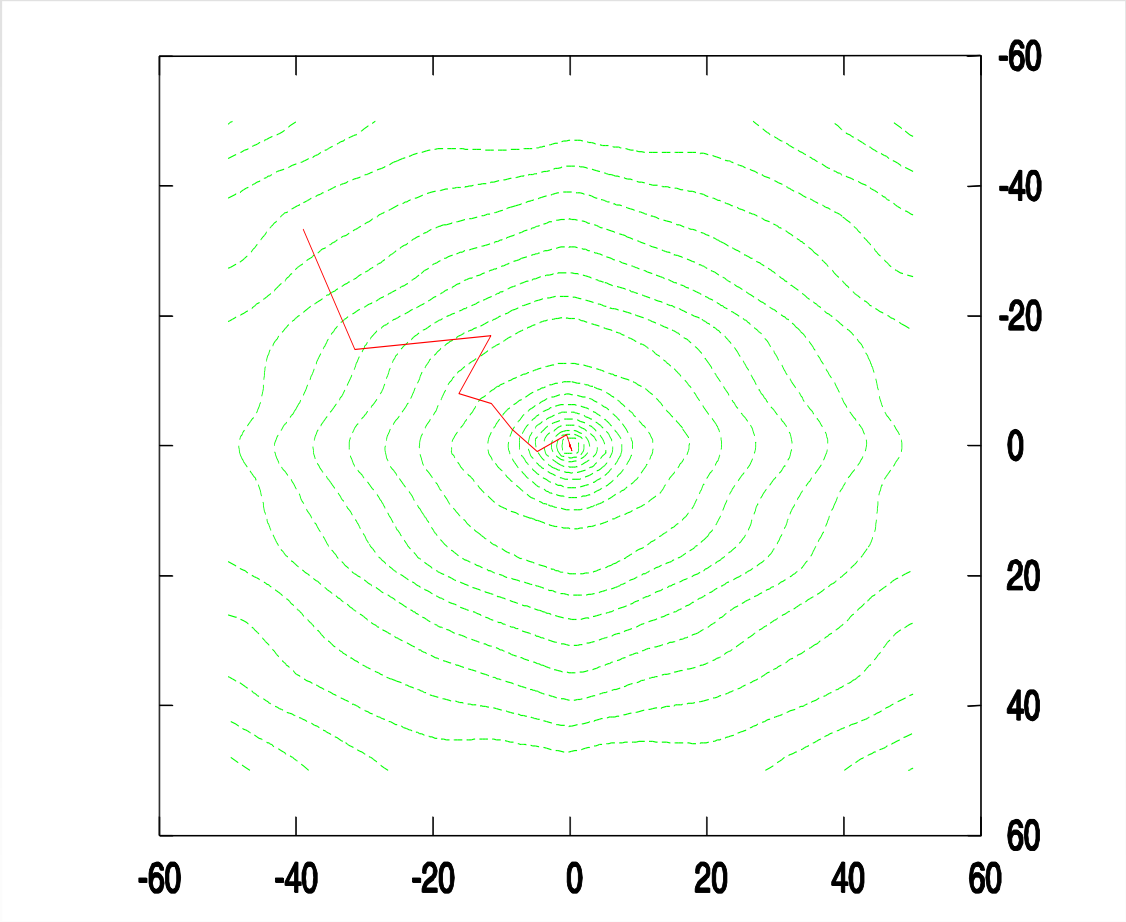
Diferenças Quadráticas Média



Tamanho do passo = 10.0 mm

Plotando o trajeto do Otimizador

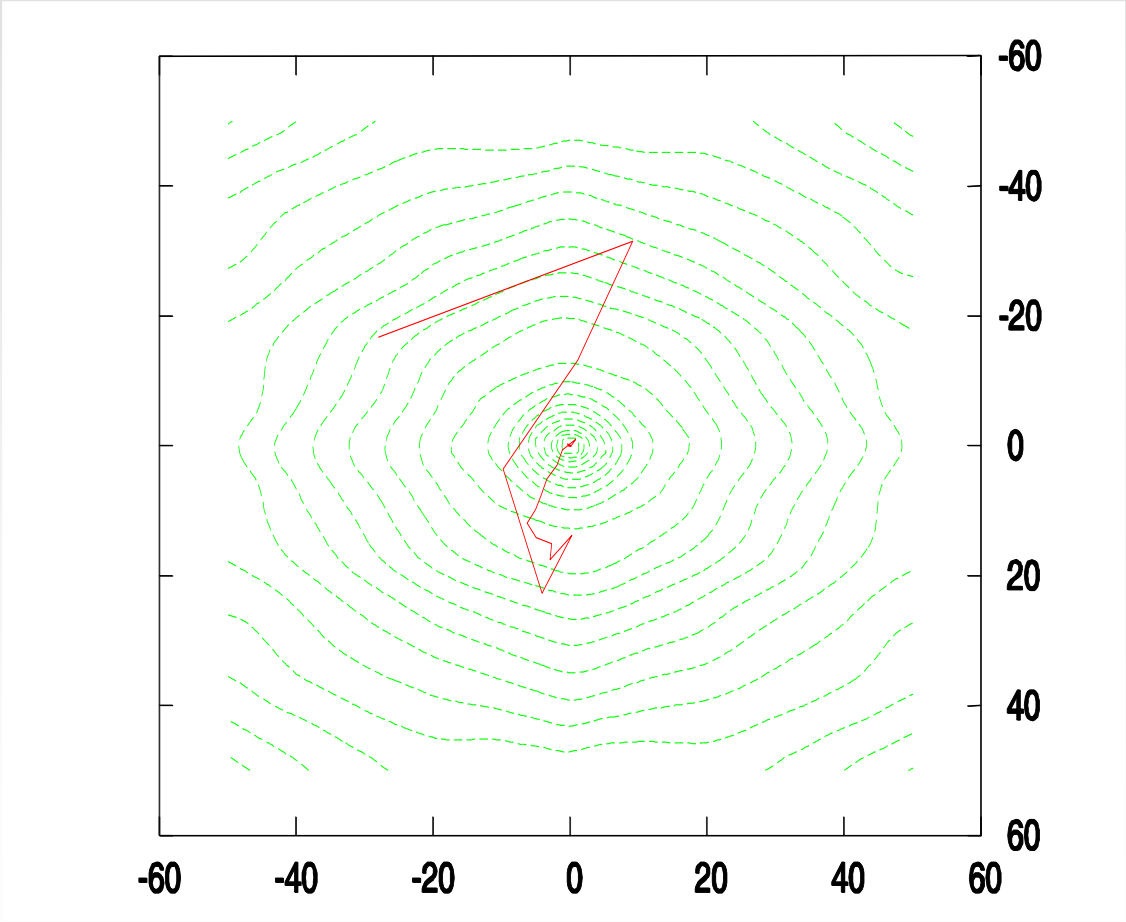
Diferenças Quadráticas Média



Tamanho do passo = 20.0 mm

Plotando o trajeto do Otimizador

Diferenças Quadráticas Média



Tamanho do passo = 40.0 mm

Observe o seu Otimizador

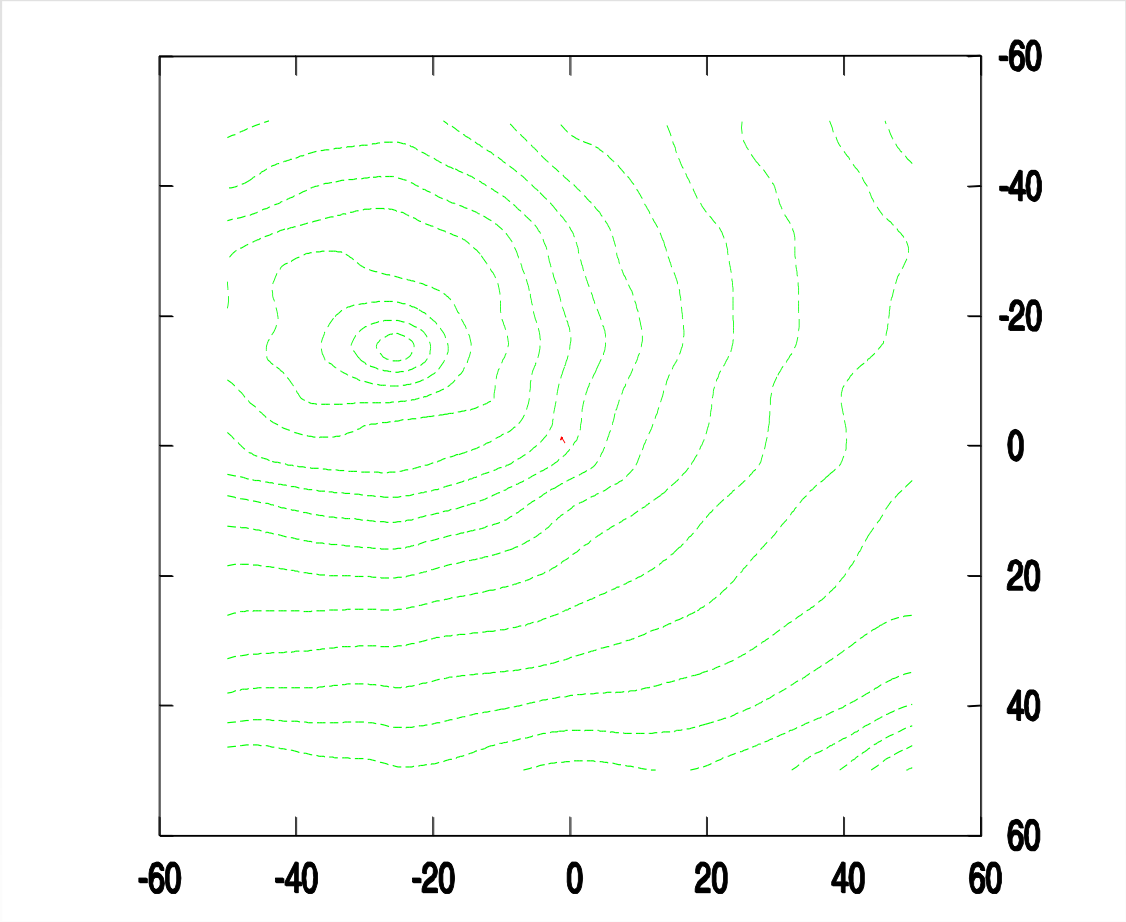
Exemplo:

Otimizador corregistrando uma imagem deslocada de (-15mm, -25mm)

O otimizador começa com (0mm,0mm)

Plotando o trajeto do Otimizador

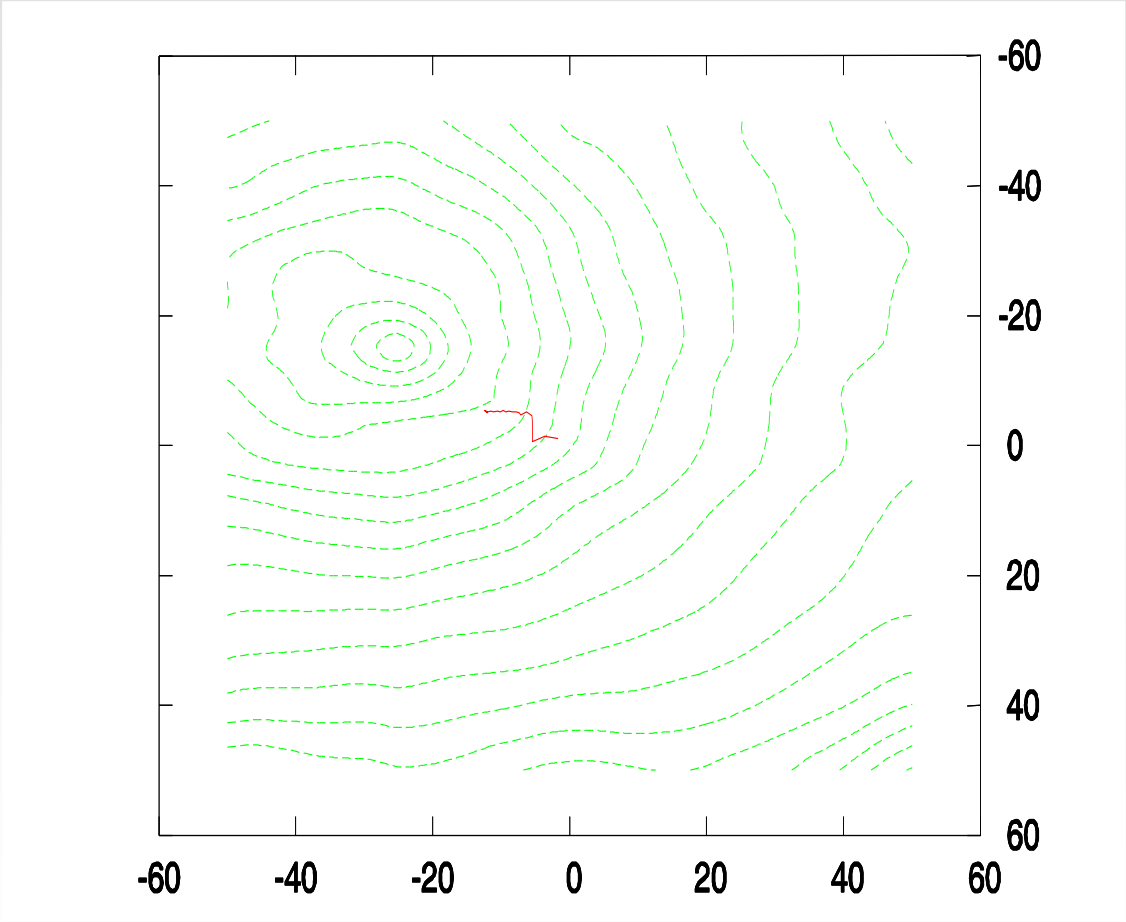
Diferenças Quadráticas Média



Tamanho do passo = 1.0 mm

Plotando o trajeto do Otimizador

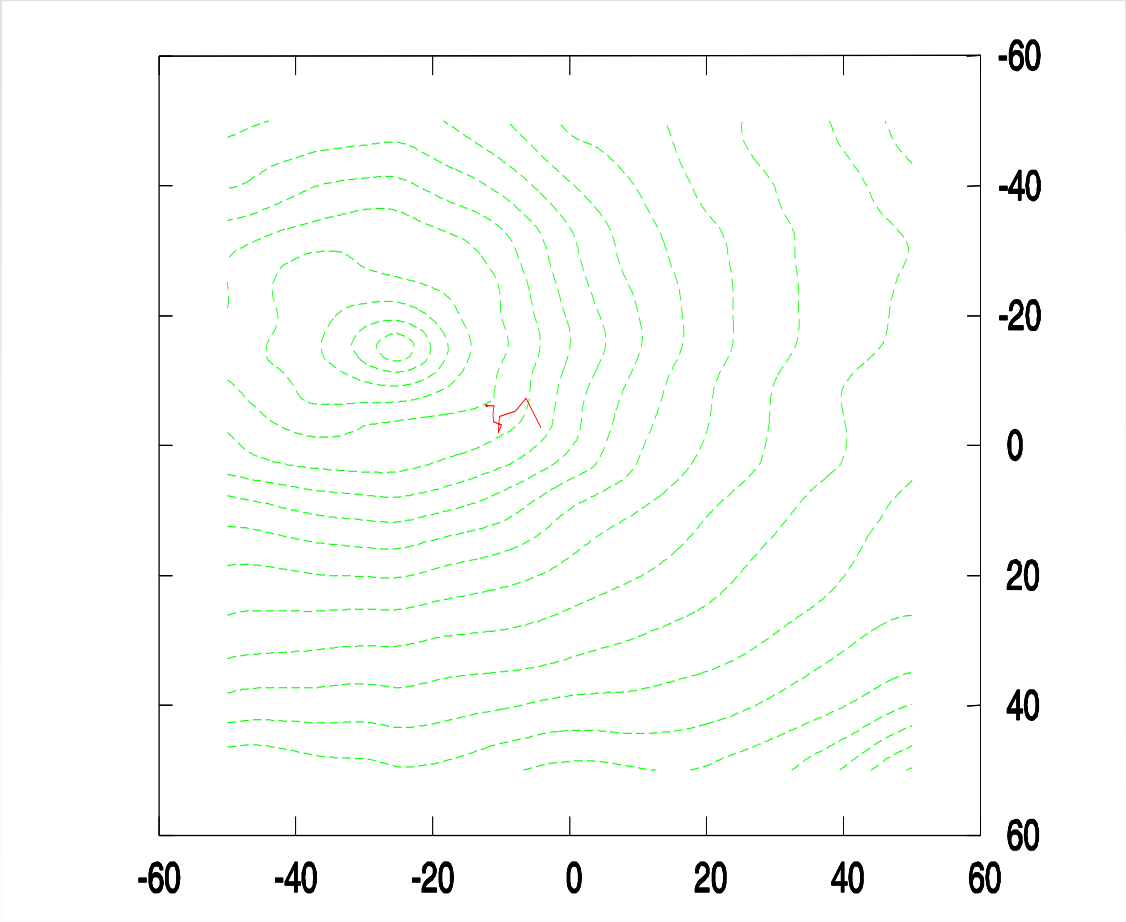
Diferenças Quadráticas Média



Tamanho do passo = 2.0 mm

Plotando o trajeto do Otimizador

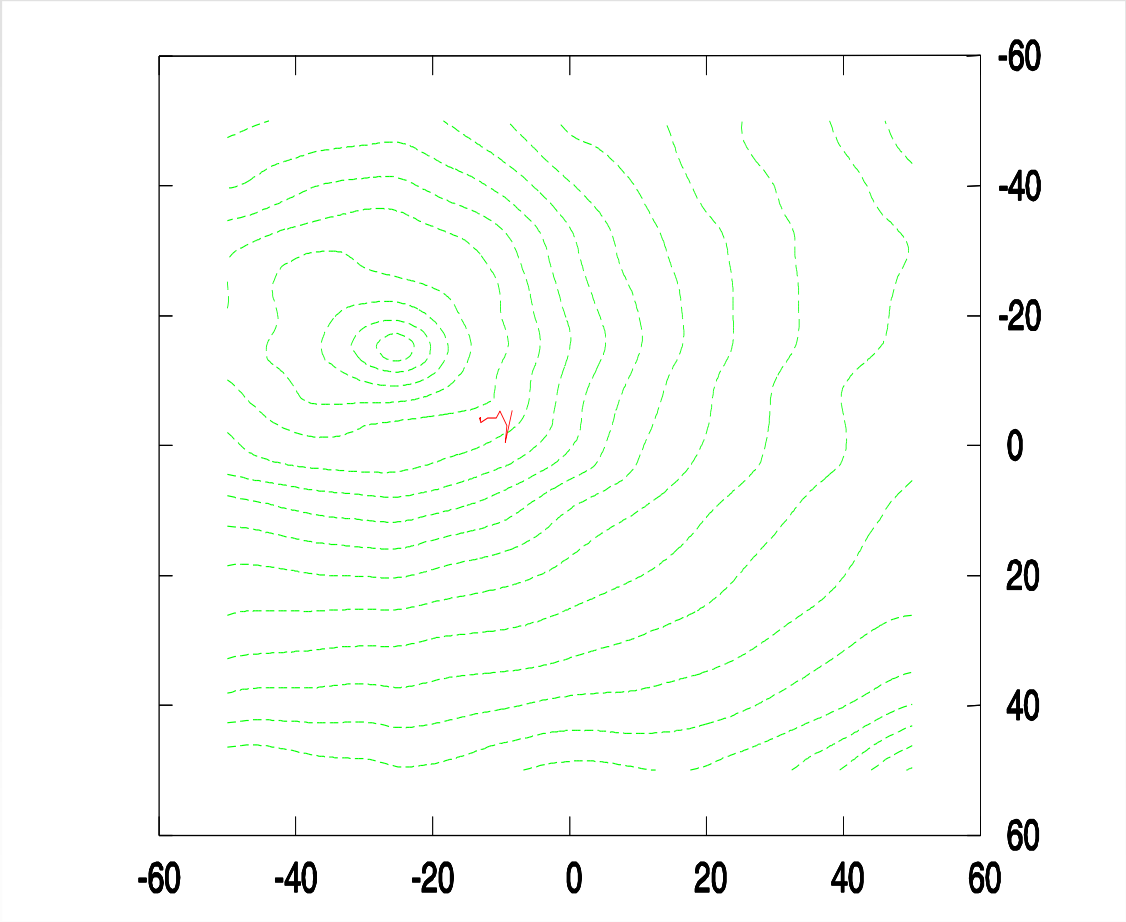
Diferenças Quadráticas Média



Tamanho do passo = 5.0 mm

Plotando o trajeto do Otimizador

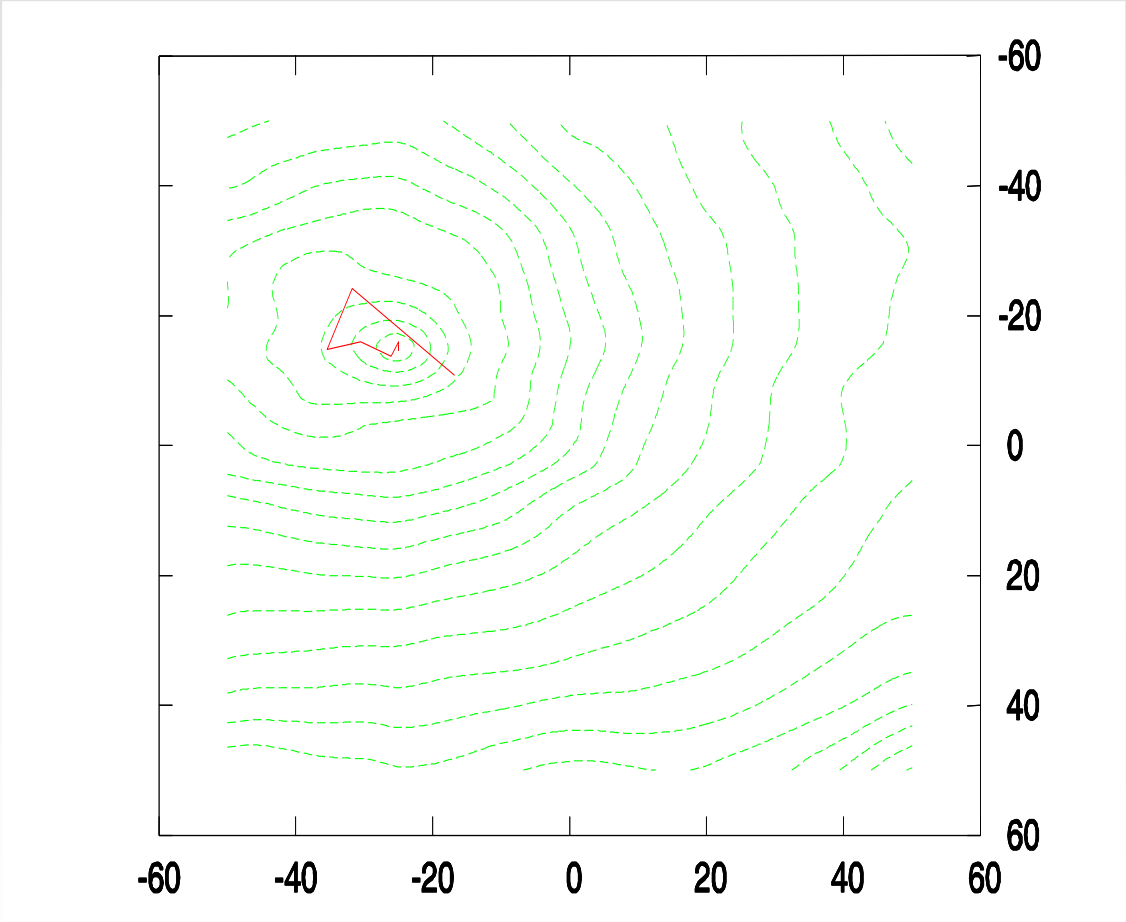
Diferenças Quadráticas Média



Tamanho do passo = 10.0 mm

Plotando o trajeto do Otimizador

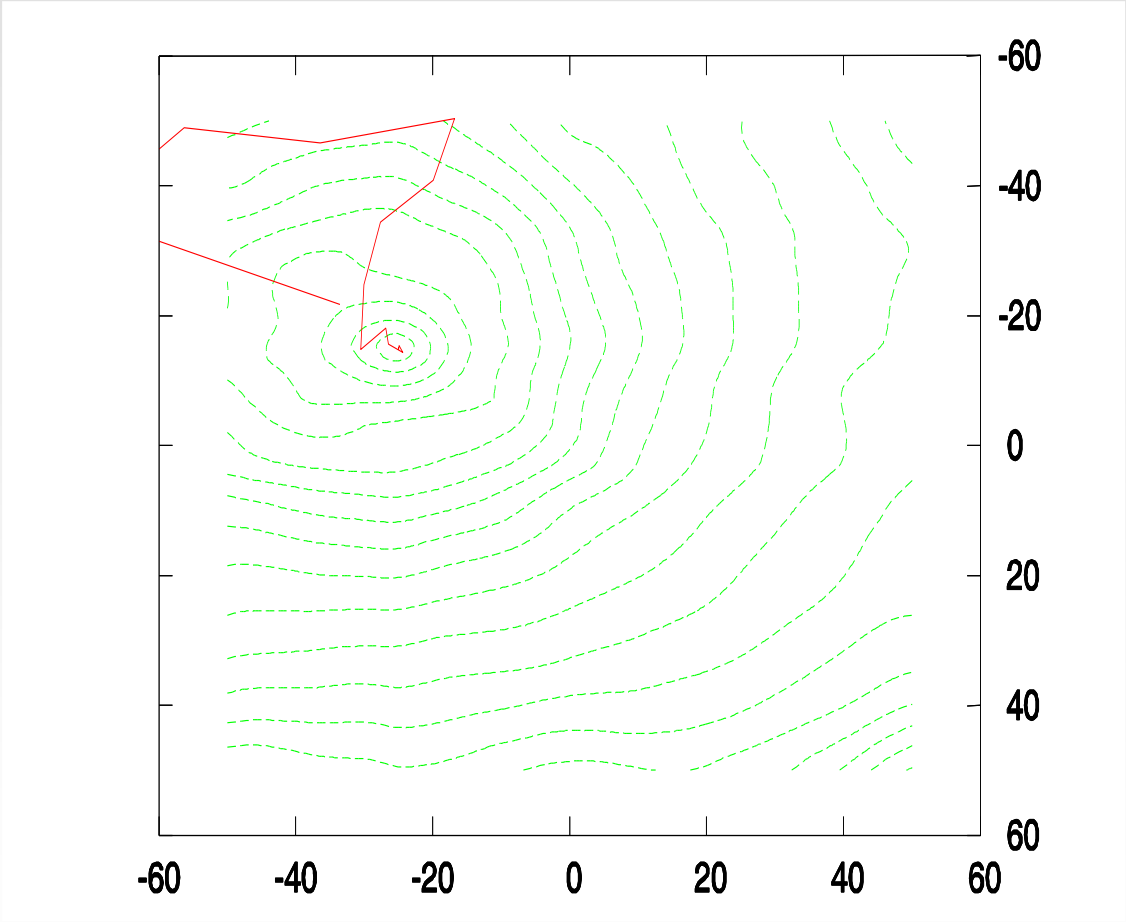
Diferenças Quadráticas Média



Tamanho do passo = 20.0 mm

Plotando o trajeto do Otimizador

Diferenças Quadráticas Média



Tamanho do passo = 40.0 mm

Multimodalidade

Corregistro Multimodal

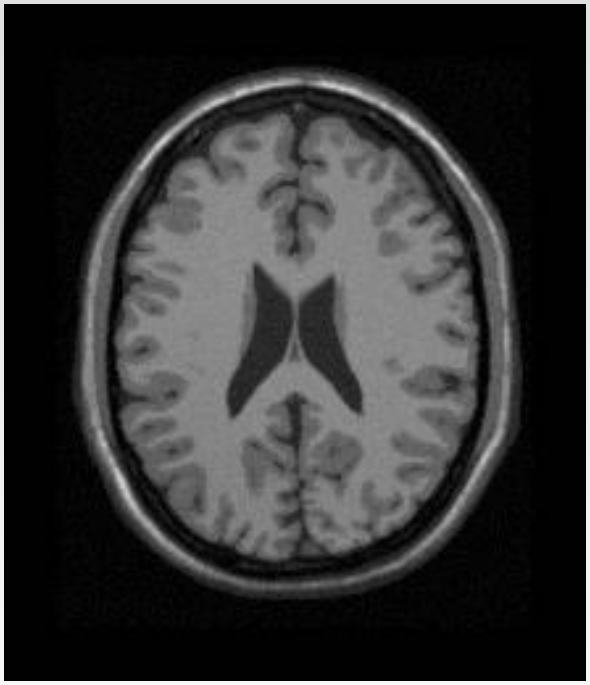


Imagem Fixa

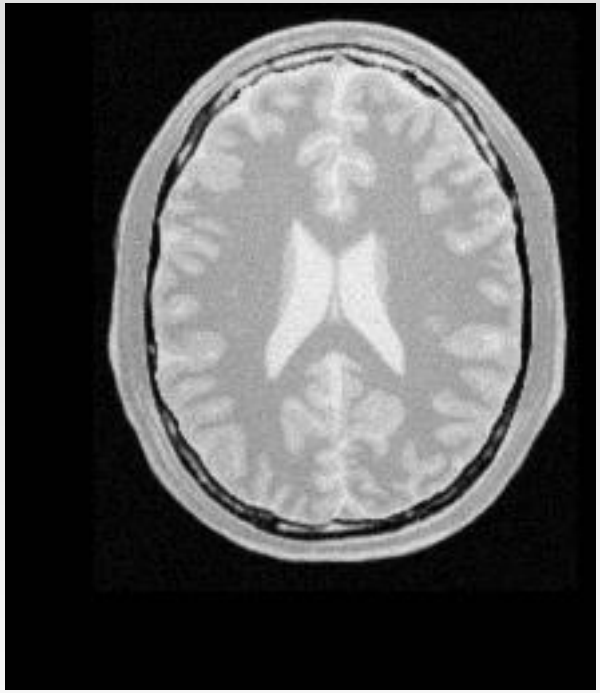


Imagem Móvel

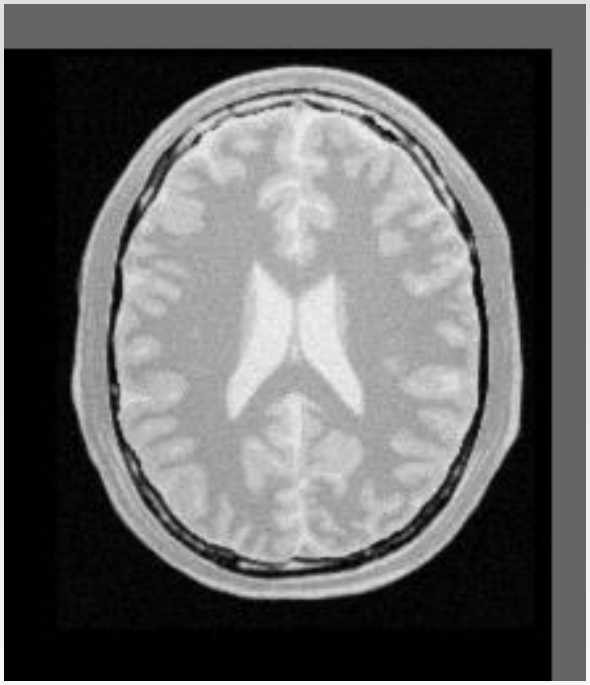
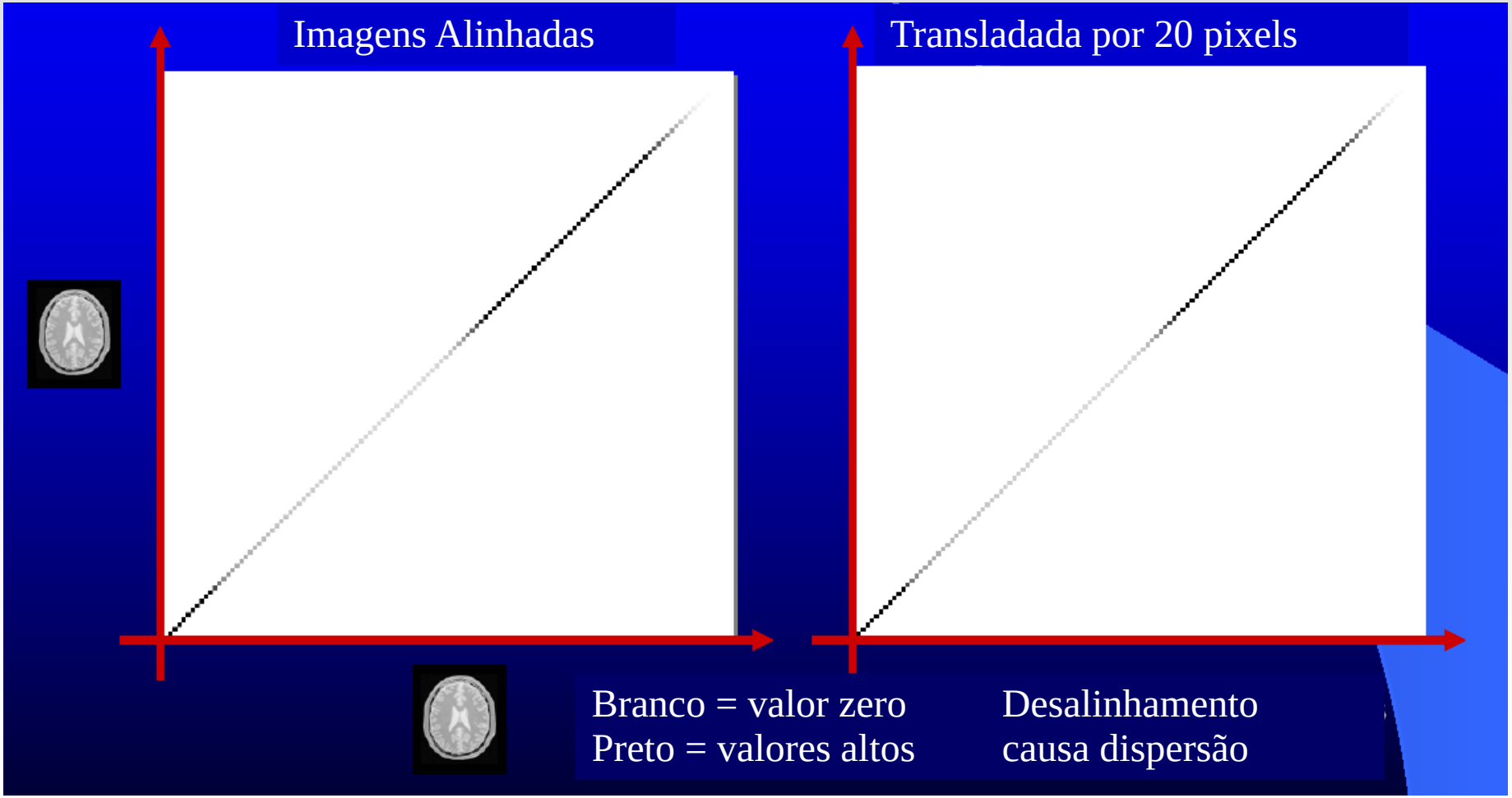
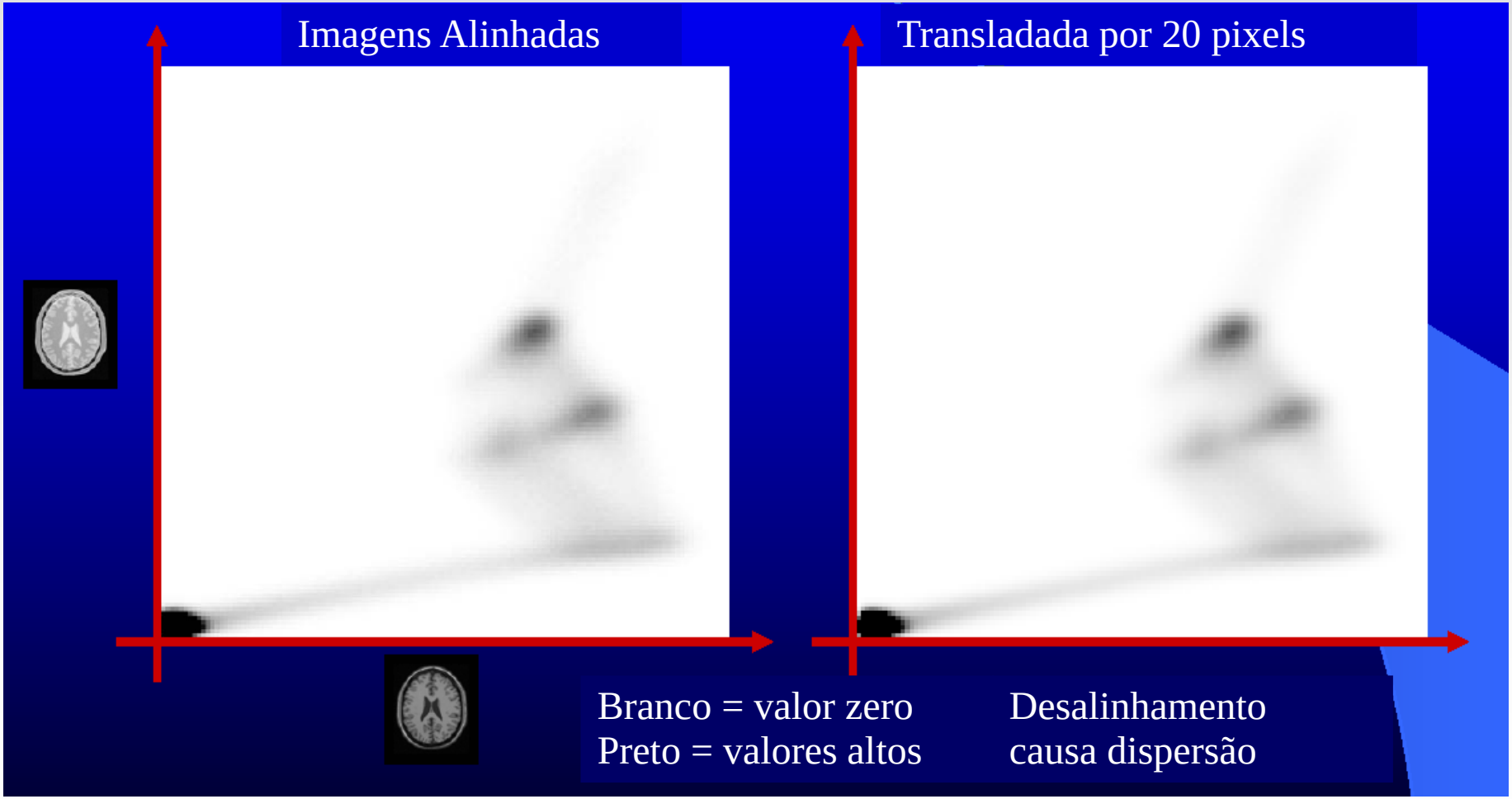


Imagem Móvel Corregistrada

Informação Mutua



Informação Mutua



Corregistro de imagens

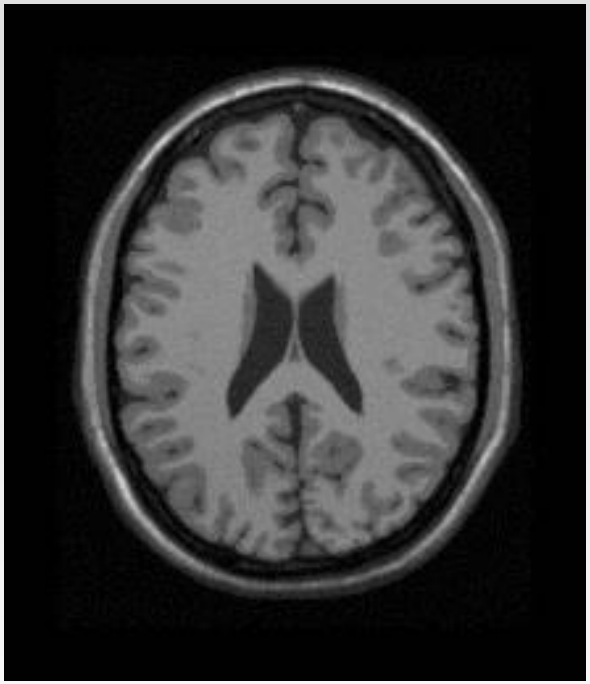


Imagem Fixa

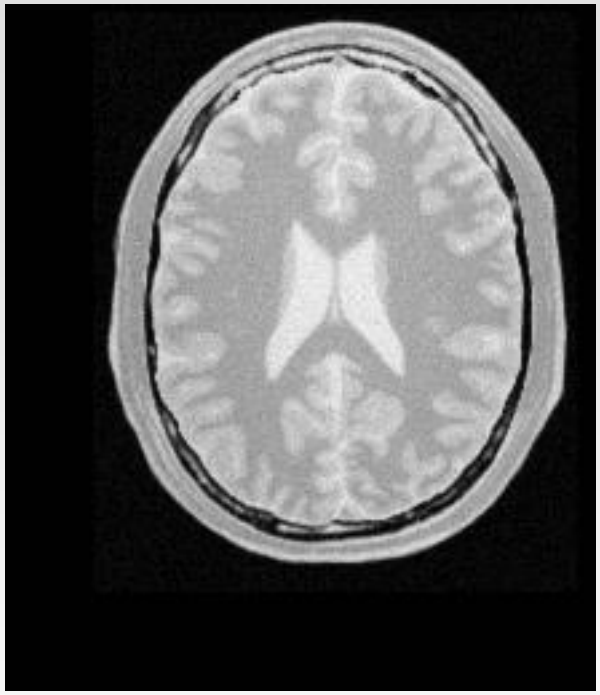


Imagem Móvel

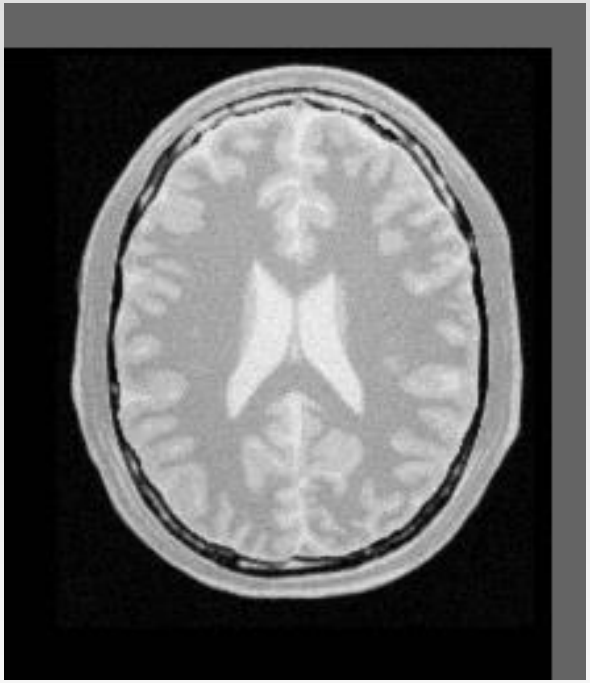


Imagem Móvel Corregistrada

Corregistro de imagens

```
#include ”itkImageRegistrationMethod.h”
#include ”itkTranslationTransform.h”
#include ”itkMattesMutualInformationImageToImageMetric.h”
#include ”itkLinearInterpolateImageFunction.h”
#include ”itkRegularStepGradientDescentOptimizer.h”
#include ”itkImage.h”
#include ”itkImageFileReader.h”
#include ”itkImageFileWriter.h”
#include ”itkResampleImageFilter.h”
```

Corregistro de imagens

```
const unsigned int Dimension = 2;
typedef unsigned char PixelType;

typedef   itk::Image< PixelType , Dimension >                  FixedImageType;
typedef   itk::Image< PixelType , Dimension >                  MovingImageType;
typedef   itk::TranslationTransform< double, Dimension >       TransformType;
typedef   itk::RegularStepGradientDescentOptimizer             OptimizerType;
typedef   itk::LinearInterpolateImageFunction<
                                MovingImageType , double >     InterpolatorType;
typedef   itk::MattesMutualInformationImageToImageMetric<
                FixedImageType , MovingImageType >             MetricType;

typedef   itk::ImageRegistrationMethod<
                FixedImageType , MovingImageType >             RegistrationType;
```

Corregistro de imagens

```
TransformType::Pointer     transform      = TransformType::New();
OptimizerType::Pointer     optimizer      = OptimizerType::New();
InterpolatorType::Pointer  interpolator   = InterpolatorType::New();
MetricType::Pointer        metric         = MetricType::New();
RegistrationType::Pointer  registrator    = RegistrationType::New();

registrator->SetTransform(    transform    );
registrator->SetOptimizer(    optimizer    );
registrator->SetInterpolator( interpolator );
registrator->SetMetric(       metric       );

registrator->SetFixedImage( fixedImageReader->GetOutput() );
registrator->SetMovingImage( movingImageReader->GetOutput() );
```

Corregistro de imagens

```
registrator->SetFixedImageRegion(
                fixedImageReader->GetOutput()->GetBufferedRegion() );

typedef   RegistrationType::ParametersType     ParametersType;

transform->SetIdentity();

registrator->SetInitialTransformParameters(
                                        transform->GetParameters() );

optimizer->SetMaximumStepLength( 4.00 );
optimizer->SetMinimumStepLength( 0.05  );
optimizer->SetNumberOfIterations( 200 );

optimizer->MaximizeOff();
```

Corregistro de imagens

```
metric->SetNumberOfHistogramBins( 20 );                    // Metric specific
metric->SetNumberOfSpatialSamples( 10000 );                // Metric specific

try
    {
        registrator->StartRegistration ();
    }
catch( itk::ExceptionObject & excp )
    {
        std::cerr << "Error in registration" << std::endl;
        std::cerr << excp << std::endl;
    }


transform->SetParameters(
                    registrator->GetLastTransformParameters()  );
```

Corregistro de imagens

```
typedef   itk::ResampleImageFilter<
                    FixedImageType , MovingImageType >     ResamplerType;

ResamplerType ::Pointer   resampler  = ResamplerType::New();


resampler->SetTransform ( transform );
resampler->SetInput( movingImageReader->GetOutput() );

FixedImageType ::Pointer   fixedImage  = fixedImageReader->GetOutput();
resampler->SetOrigin( fixedImage->GetOrigin() );
resampler->SetSpacing( fixedImage->GetSpacing() );
resampler->SetSize(
                    fixedImage->GetLargestPossibleRegion()->GetSize() );

resampler->Update();
```

Rotação – Translação
Escalonamento de Parâmetros

Corregistro de imagens

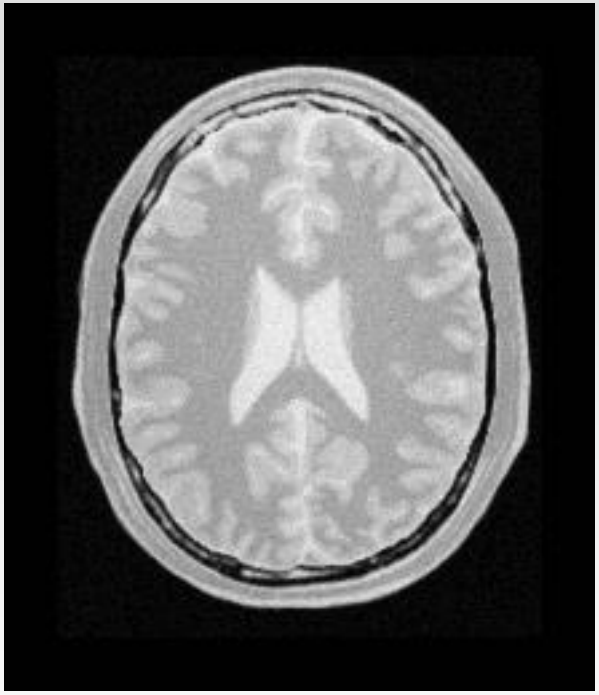


Imagem Fixa

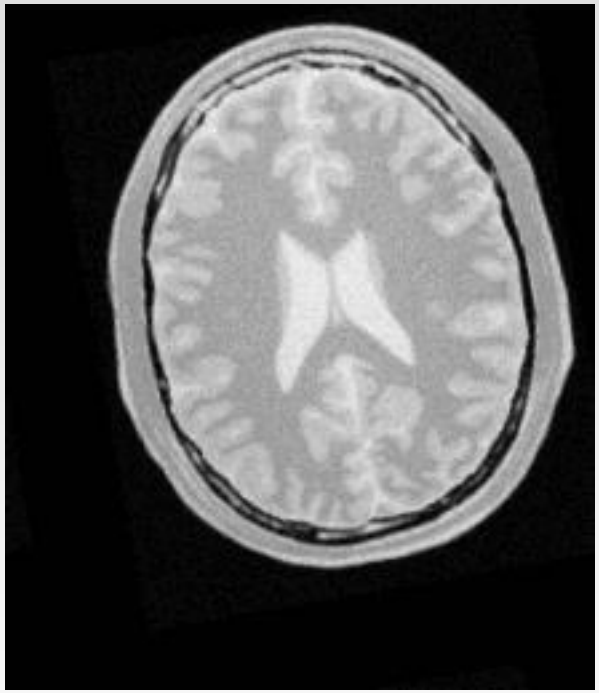


Imagem Móvel

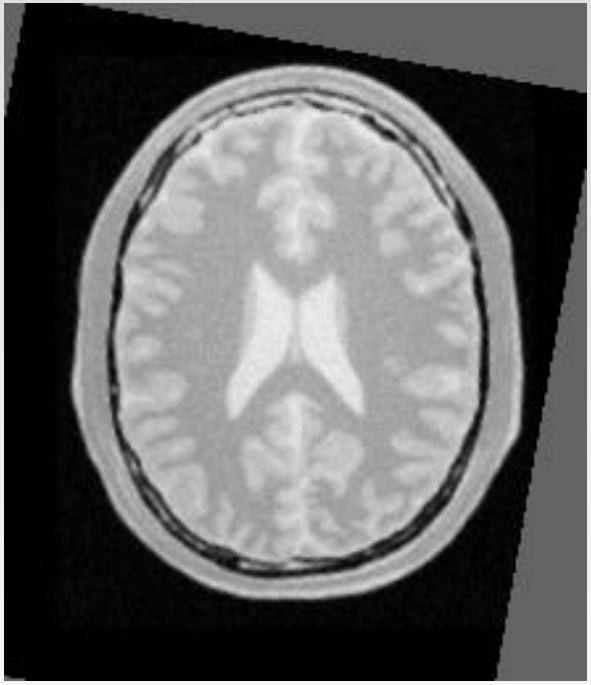


Imagem Móvel Corregistrada

Corregistro de imagens

```
#include ”itkImageRegistrationMethod.h”
#include ”itkCenteredRigid2DTransform.h”
#include ”itkMeanSquaresImageToImageMetric.h”
#include ”itkLinearInterpolateImageFunction.h”
#include ”itkRegularStepGradientDescentOptimizer.h”
#include ”itkCenteredTransformInitializer.h”
#include ”itkImage.h”
#include ”itkImageFileReader.h”
#include ”itkImageFileWriter.h”
#include ”itkResampleImageFilter.h”
```

Corregistro de imagens

```
const unsigned int Dimension = 2;
typedef unsigned char PixelType;

typedef   itk::Image< PixelType , Dimension >          FixedImageType;
typedef   itk::Image< PixelType , Dimension >          MovingImageType;

typedef   itk::CenteredRigid2DTransform< double >      TransformType;

typedef   itk:: CenteredTransformInitializer<
                                    TransformType ,
                                    FixedImageType ,
                                    MovingImageType
                                                >      InitializerType;
```

Corregistro de imagens

```
TransformType::Pointer      transform      = TransformType::New();
InitializerType::Pointer    initializer    = InitializerType::New();
OptimizerType::Pointer      optimizer      = OptimizerType::New();
InterpolatorType::Pointer   interpolator   = InterpolatorType::New();
MetricType::Pointer         metric         = MetricType::New();
RegistrationType::Pointer   registrator    = RegistrationType::New();

registrator->SetTransform(    transform    );
registrator->SetOptimizer(    optimizer    );
registrator->SetInterpolator( interpolator );
registrator->SetMetric(       metric       );

registrator->SetFixedImage( fixedImageReader->GetOutput() );
registrator->SetMovingImage( movingImageReader->GetOutput() );
```

Corregistro de imagens

```
registrator->SetFixedImageRegion(
                    fixedImageReader->GetOutput()->GetBufferedRegion() );

initializer->SetTransform ( transform );
initializer->SetFixedImage(    fixedImageReader->GetOutput()     );
initializer->SetMovingImage( movingImageReader->GetOutput() );

initializer->MomentsOn();

initializer->InitializeTransform();

registrator->SetInitialTransformParameters(
                                        transform->GetParameters() );
```

Corregistro de imagens

```
typedef   OptimizerType::ScaleType       OptimizerScalesType;

OptimizerScalesType optimizerScales(
                                    optimizer->SetMaximumStepLength() );

const double translationScale  = 1.0 / 1000.0 ;

optimizerScales[ 0 ] = 1.0;
optimizerScales[ 1 ] = translationScale;
optimizerScales[ 2 ] = translationScale;
optimizerScales[ 3 ] = translationScale;
optimizerScales[ 4 ] = translationScale;

optimizer->SetScales( optimizerScales );
```

Corregistro de imagens

```
try
    {
        registrator->StartRegistration ();
    }
catch( itk::ExceptionObject & excp )
    {
        std::cerr << “Error in registration” << std::endl;
        std::cerr << excp << std::endl;
    }


transform->SetParameters(
                        registrator->GetLastTransformParameters()  );
```

Corregistro de imagens

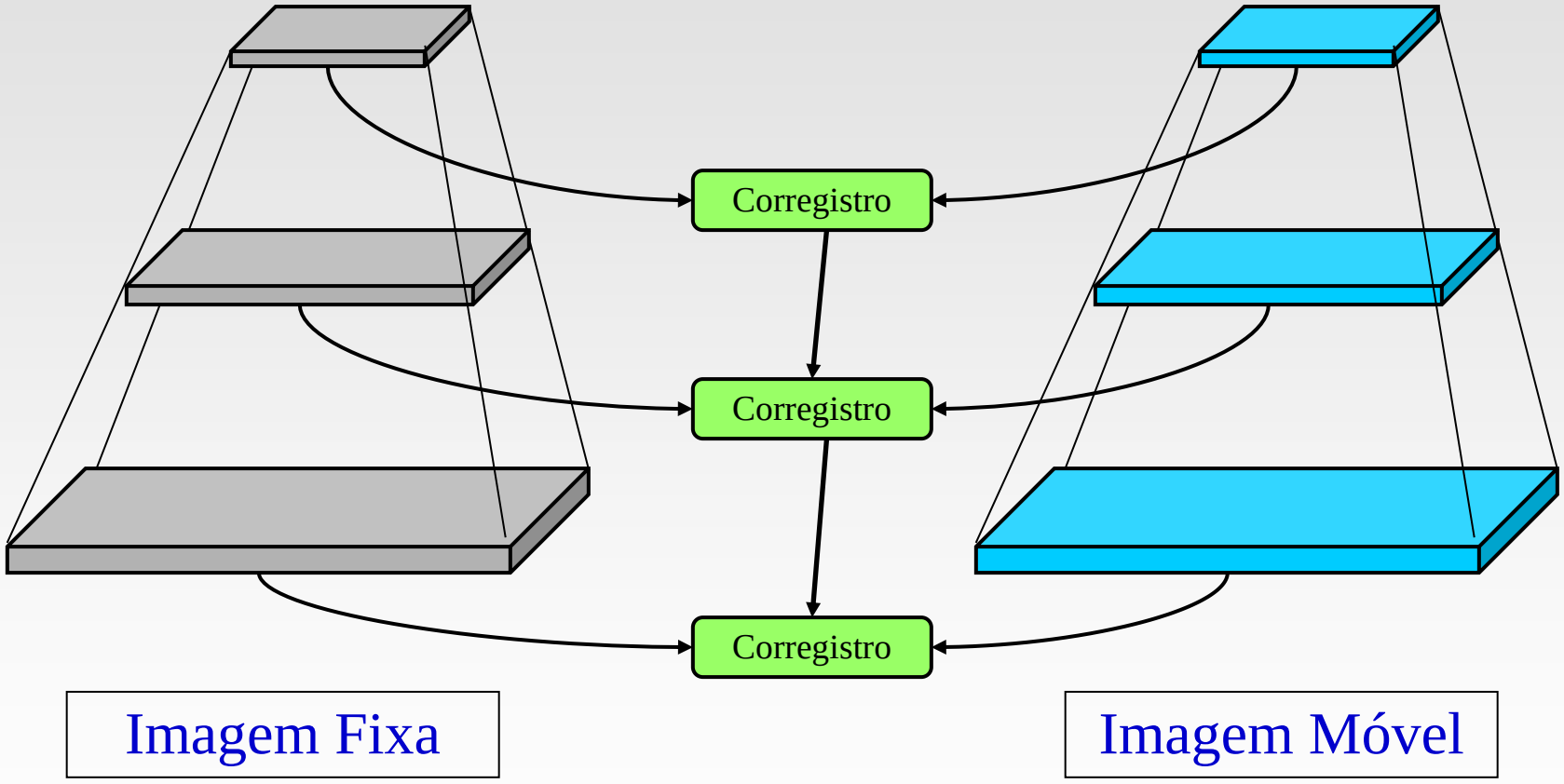
```
typedef   itk::ResampleImageFilter<
                    FixedImageType , MovingImageType >     ResamplerType;

ResamplerType ::Pointer   resampler  = ResamplerType::New();


resampler->SetTransform ( transform );
resampler->SetInput( movingImageReader->GetOutput() );

FixedImageType ::Pointer   fixedImage  = fixedImageReader->GetOutput();

resampler->SetOrigin( fixedImage->GetOrigin() );
resampler->SetSpacing( fixedImage->GetSpacing() );
resampler->SetSize(
                    fixedImage->GetLargestPossibleRegion()->GetSize() );

resampler->Update();
```

Multirresolução

Estrutura de Corregistro Multirresolução

- **Melhora a velocidade;**
- **Acuracia; e**
- **Robustes**

Estrutura de Corregistro Multirresolução



Estrutura de Corregistro Multirresolução

Estrutura flexível que permite mudanças de

- **Parâmetros**
- **Componentes**

entre níveis de resolução