Insight Toolkit ITK

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

‘Namespaces’

- **‘Namespaces’ resolve o problema de classes que tem o mesmo nome**
- **E.g., ITK contem uma classe Array**
- **Podemos evitar estes conflitos criando nossos próprios ‘namespace’ dentro do código**

```
namespace itk { code }’
```

‘Namespaces’, cont.

- **Dentro de um dado ‘namespace’, referenciamos a outras no mesmo ‘namespace’ somente pelo nome, e.g. dentro do namespace itk Array significa “use a array do ITK”**
- **Fora do namespace, usamos o prefixo itk::, e.g. itk::Array**
- **Somente código que são parte do toolkit estão dentro do namespace itk**

‘Namespaces’, cont.

- **Notem que o códigodentro do manespace itk namespace devem referir a código fora do namespace explicitamente**
- **E.g. use `std::cout` ao invés de `cout`**

Programação Orientada a Objeto

- **Identifique unidades functionais dentro do seu projeto**
- **Escreva classes para implementar estas unidades functionais**
- **Separe functionalmente o quanto possível para promover reutilização do código**

Membros de uma Classe

- **Classes tem varáveis e métdos membros**
  - Nomes de membros de classes com prefixo “m_”, como “m_VariableName”
- **Membros de Classes são de 3 tipos**
  - Public
  - Private
  - Protected

Membros Public

- **Todos podem accessar o membro**
  - “O resto do mundo”
  - A própria classe
  - Classes filhas
- **Devemos evitar criar variáveis membros públicas para evitar alterações indesejadas**

Membros Private

- **Somente a prória classe pode acessar o membro**
- **Ele não é visível para o resto do mundo**
- **Classes filhas não podem acessar o membro**

Membros Protected

- O meio termo entre public e private
- O “resto do mundo” não pode acessar o membro, mas classes derivadas podem

ITK e membros

- **No ITK, variáveis membros são quase sempre private**
- **Existem métodos de acesso‘public’ que permitem ao resto do mundo acesso via ‘setters’ (set) e ‘getters’ (get) para colocar e pegar valores**
- **Isto certifica que a classe saiba quando o valor de uma variável é modificado**

Porquê fazer desta forma?

- **Considere uma classe filtro – se alguém modifica uma variável no filtro, ele deve reexecutar a si mesmo na próxima vez que o usuário pedir uma saída**
- **Se nada é mudado, não é gasto tempo executando novamente**
- **Métodos de acesso acionam uma "modified flag" para notificar quando as coisas mudarem**

Programação genérica

- **Programação genérica encoraja:**
  - Escrever códigos sem referência a um tipo específico de dados (float, int, etc.)
  - Projetar códigos da maneira mais abstrata possível
- **Porquê?**
  - Troca uma pouco de tempo extra de projeto para ganhar muito em melhorar a reusabilidade do código

Exemplo em Imagems

- **Imagens são usualmente armazenada como uma array de um tipo particular**
  - e.g. `unsigned char[256*256]`
- **É conveniente encapsular esta array em uma classe imagem (um bom projeto POO)**
- **Permitir ao usuário mudar o tamanho da imagem é fácil com arrays dinamicamente alocadas**

Exemplo em imagem, cont.

- **Infelizmente, muda o tipo de dado não é tão fácil**
- **Tipicamente fazemos uma escolha de projeto (ImageAccess) e vivemos com ela**
- **Ou, somos forçados a implementar uma classe double, uma classe float, uma classe int e assim por diante (menos comum, complicado)**

Templates para a salvação

- **Templates provêem um modo de contornar a rigidez do tipo de dados**
- **Se você estiver familiarizado com o conceito de macros, pode pensar templates como um modo melhorado de macros**
- **Com templates, você projeta classe para lidar com tipos arbitrários**

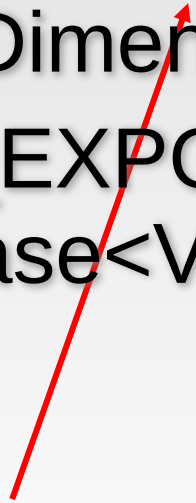
Anatomia de uma classe template

```
template <class TPixel, unsigned int
  VImageDimension=2>

class ITK_EXPORT Image : public
  ImageBase<VImageDimension>
```

Palavvra chave template, os < >’s abriga os parâmetros da template

Anatomia de uma classe template

```
template <class TPixel, unsigned int
   VImageDimension=2>
class ITK_EXPORT Image : public
   ImageBase<VImageDimension>
```



TPixel é uma classe (de algum tipo)

Anatomia de uma classe template

```
template <class TPixel, unsigned int
   VImageDimension=2>
class ITK_EXPORT Image : public
   ImageBase<VImageDimension>
```

VImageDimension é uma unsigned int, com o valor padrão de 2

Anatomia de uma classe template

```
template <class TPixel, unsigned int
   VImageDimension=2>
class ITK_EXPORT Image : public
   ImageBase<VImageDimension>
```

Image é o nome desta classe

Anatomia de uma classe template

```
template <class TPixel, unsigned int
   VImageDimension=2>
class ITK_EXPORT Image : public
   ImageBase<VImageDimension>
```

Image é derivada de ImageBase de maneira publica

Especialização

- **Quando especificamos todos os parametros template, especializamos completamente a classe**
- **No exemplo anterior, ImageBase<VImageDimension> especializa a classe base especificando seu parâmetro template. Note que o parâmetro VImageDimension é na verdade passado atraves do parametros da template Image**

Derivação de classes template

- **Deve especificar todos os parâmetros da classe base**
- **Os parâmetros template da classe base podem ou não estarem ligados ao parâmetros template da classe derivada**
- **Podemos derivar uma classe não template de uma classe tamplate se quisermos (enrijecendo todos os parâmetros template)**

Instancias de classes Template

- **Para criar uma instancia de uma classe template, devemos especializar completamente a classe**
- **E.g.**

```
itk::Image<int, 3> myImage;
```

Cria uma imagem 3D de inteiros (não exatamente verdade, mas podemos fingir até que vejamos smart pointers)

Alias…

- **C++ na verdade permite especialização *parcial***
- **Por exemplo, escrevemos uma classe Image que deve ser 3D, mas com o tipo de pixel template (ou vice-versa)**
- **Infelizmente nem todos os compiladores suportam isto (VS.net 2003+ supotam, GCCs também)**

Typedefs

- **Uma consequencia de templates é que os nomes de tipos completamente definidos podem ser realemte longos**
- **e.g.**

```
itk::Image<itk::MyObject<3, double>, 3>
```

pode ser um tipo válido

Typedefs cont.

- **Podemos criar tipos definidos pelo progrmanador usando a palavra typedef**

```
typedef itk::Image<int, 3> 3DIntImageType;

3DIntImageType myImage;

3DIntImageType anotherImage;
```

Diversão com typedefs

- **Typedefs podem ser membros globais de classes e serem acessados como tal**

```
typedef itk::Image<double, 3> ImageType;

ImageType::Pointer im = myFilter.GetOutput();
```

- **Nas classes template, membros typedefs são frequentemente definidos em termos de parâmetros template - no problem! Isto na verdade vem a calhar.**

Nomeação de templates e typedefs

- **ITK usa as seguintes convenções:**
  - Parâmetros template são indicados por T (para tipo) ou V (para valor). e.g. TPixel significa “o tipo do pixel” e VImageDimension significa “valor template para parâmetro dimensão da imagem”
  - Tipos definidos são nomeados como FooType. e.g. CharImage5DType

.txx, .cxx, .h

- **ITK usa três extensões padrões para arquivos:**
  - arquivos .h indicam arquivo cabeçalho de classe
  - .cxx indica
    a) código executável (um exemplo, teste, demo, etc.)
    b) uma implementação de classe non-template
  - .txx indica uma implementação de classe template

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

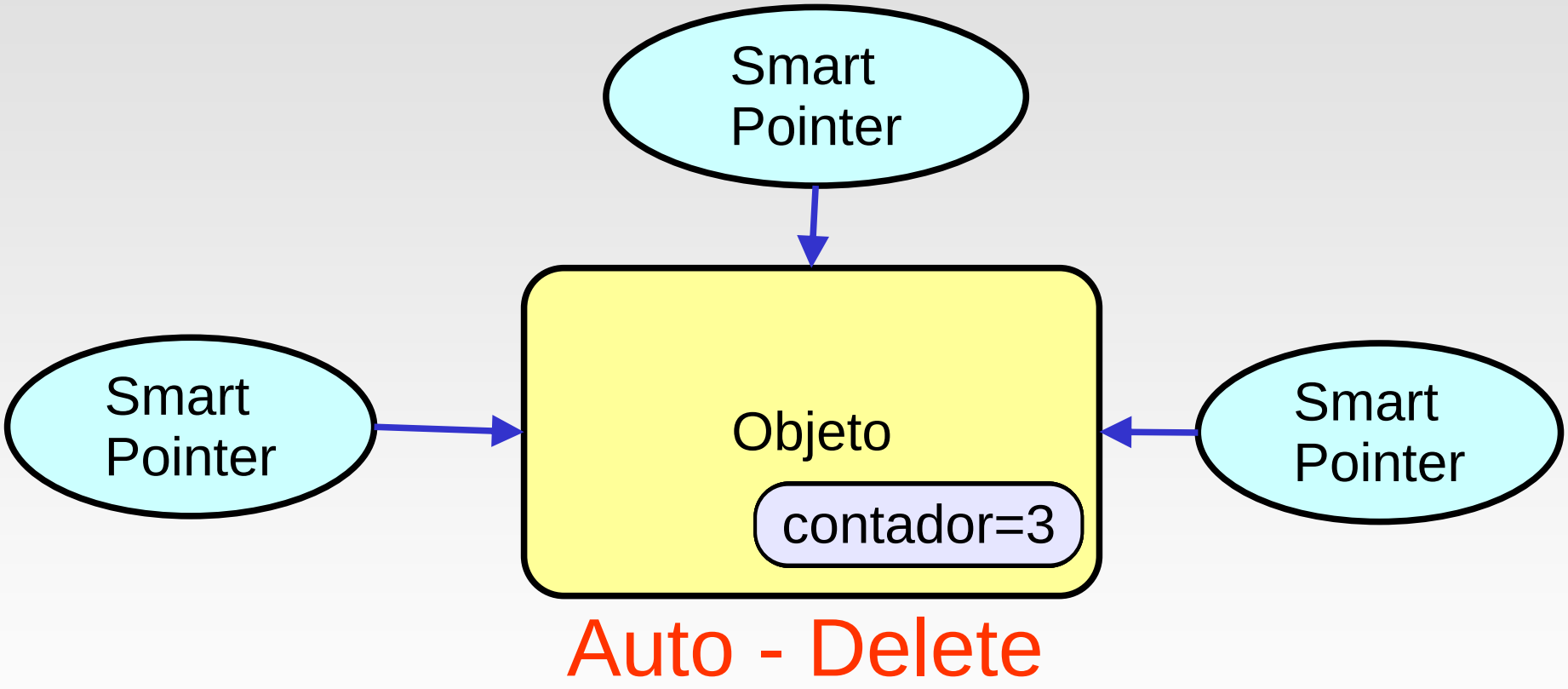
typedef

```
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

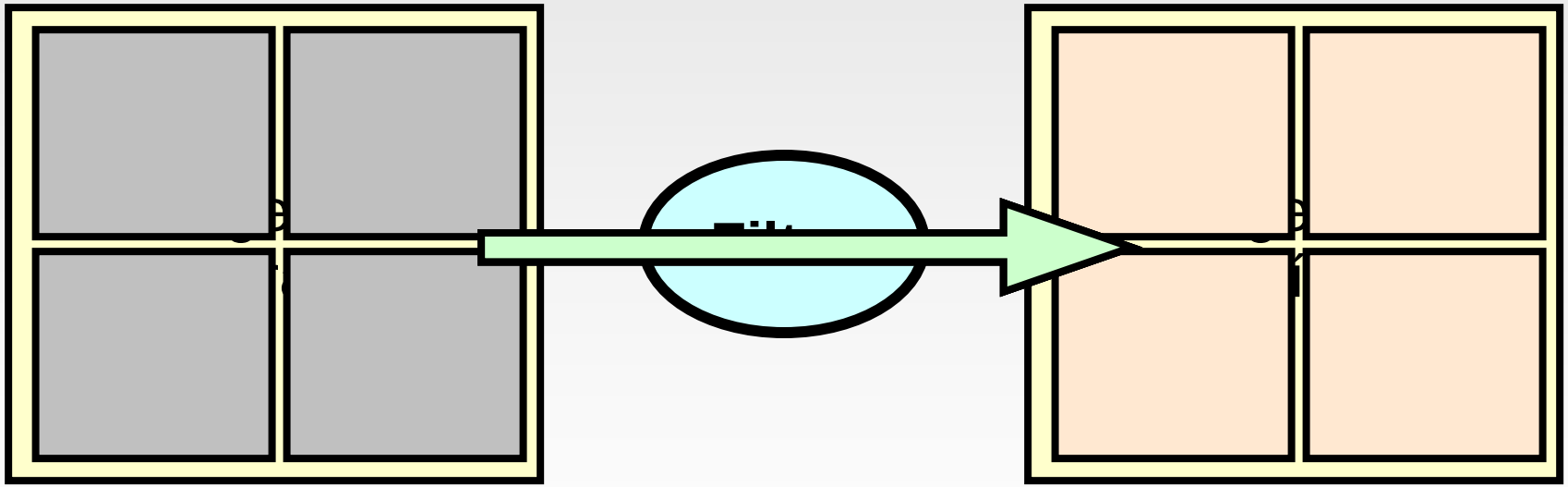
```
typedef   itk::Image< char , 3 >   ImageType

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

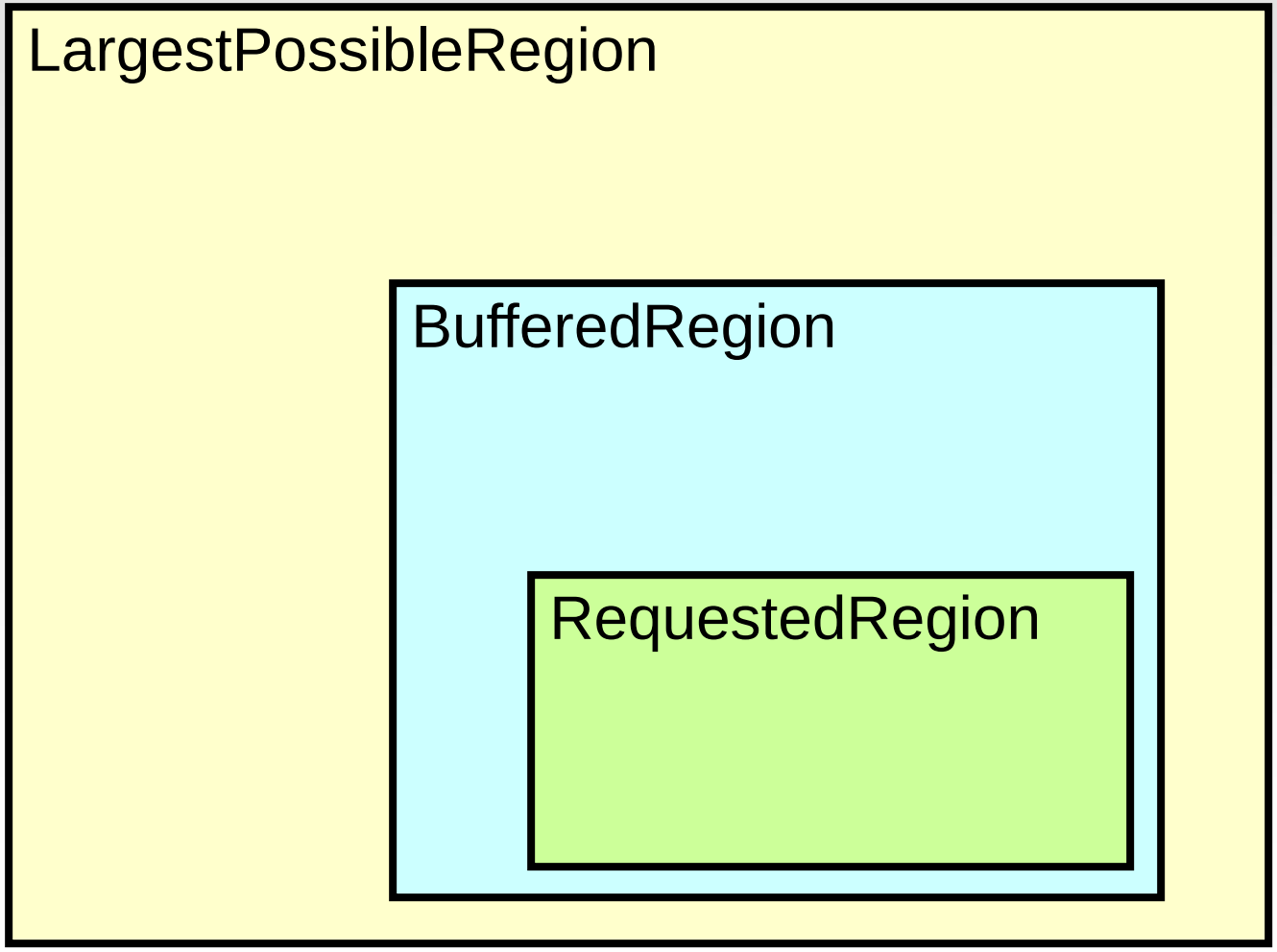
```
ImageType::RegionType region;
region.SetSize( size );
region.SetIndex( start );

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”

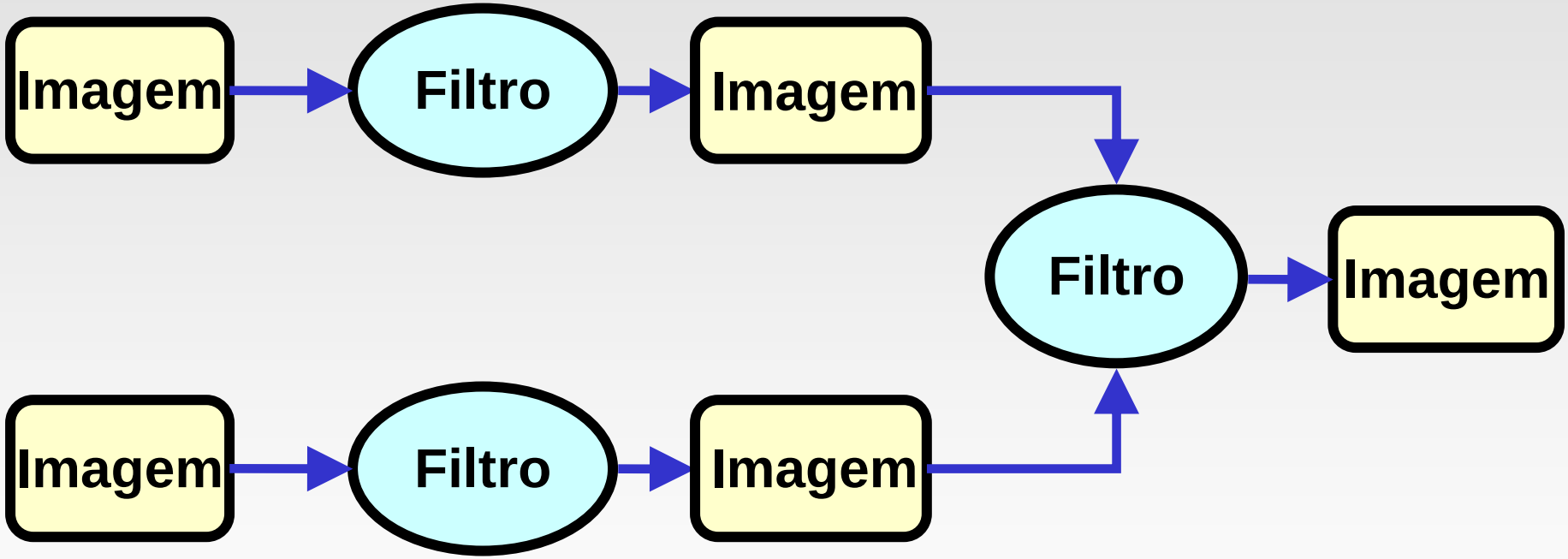
Processando Imagens Grandes



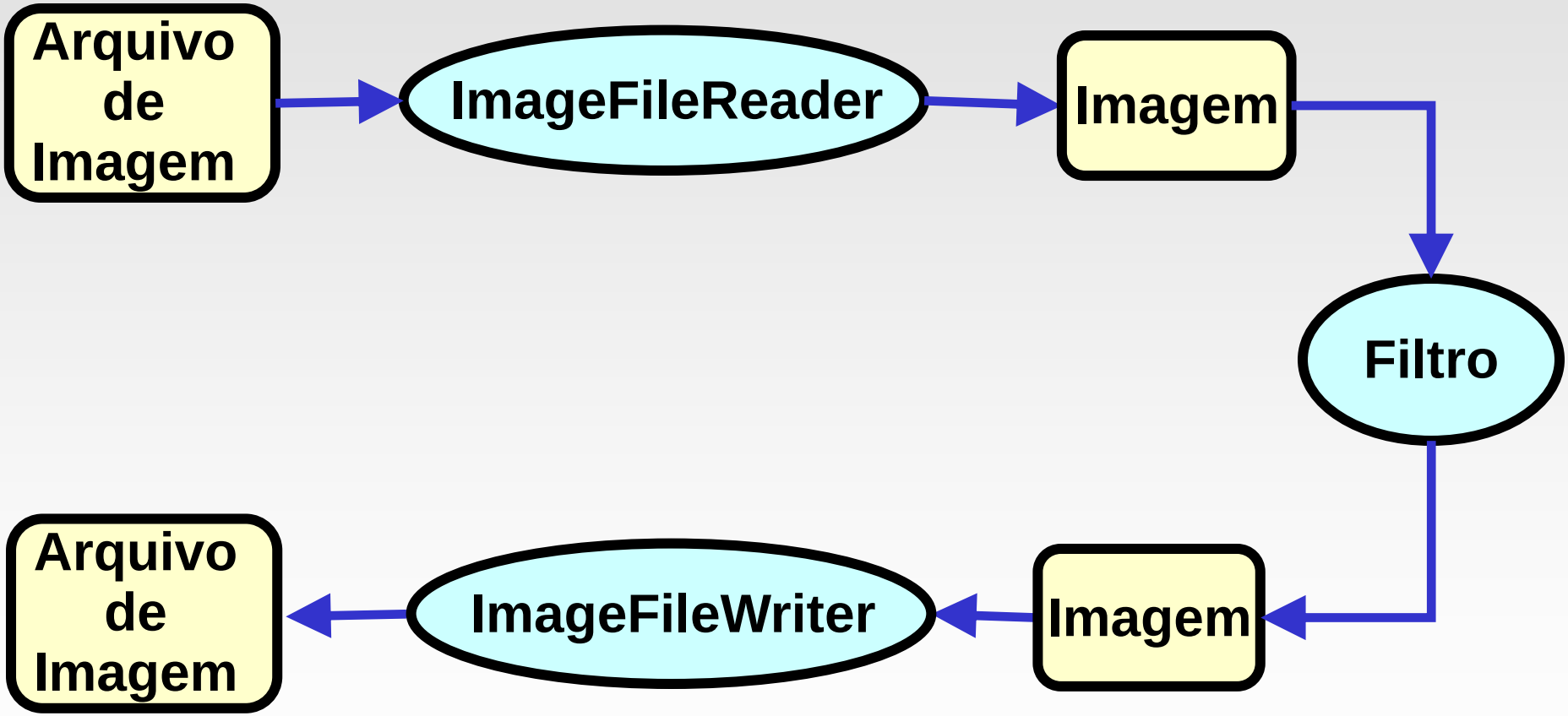
Regiões em Imagens



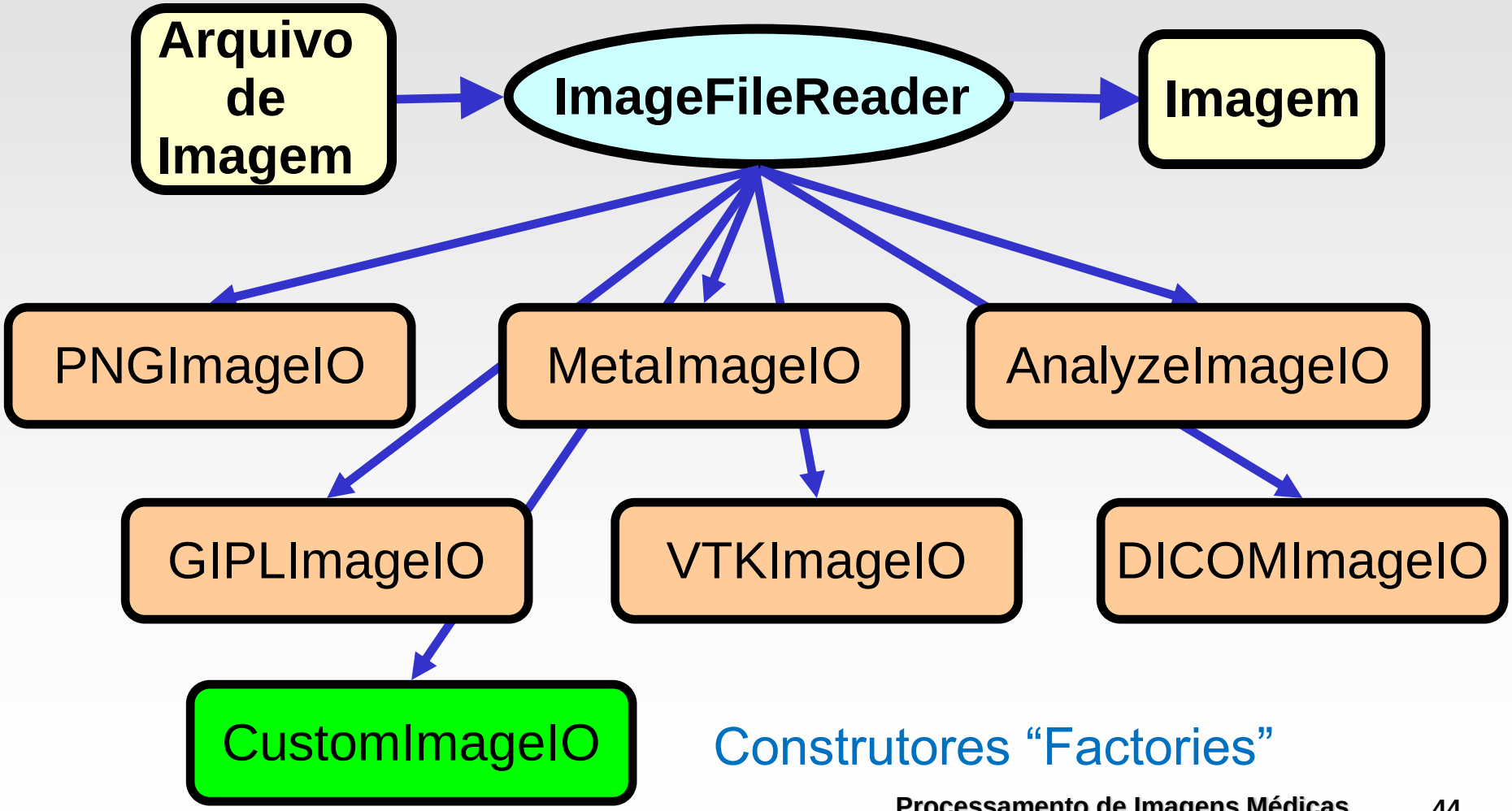
Fluxo de dados



I/O de Imagens Simples



I/O de Imagens Simples



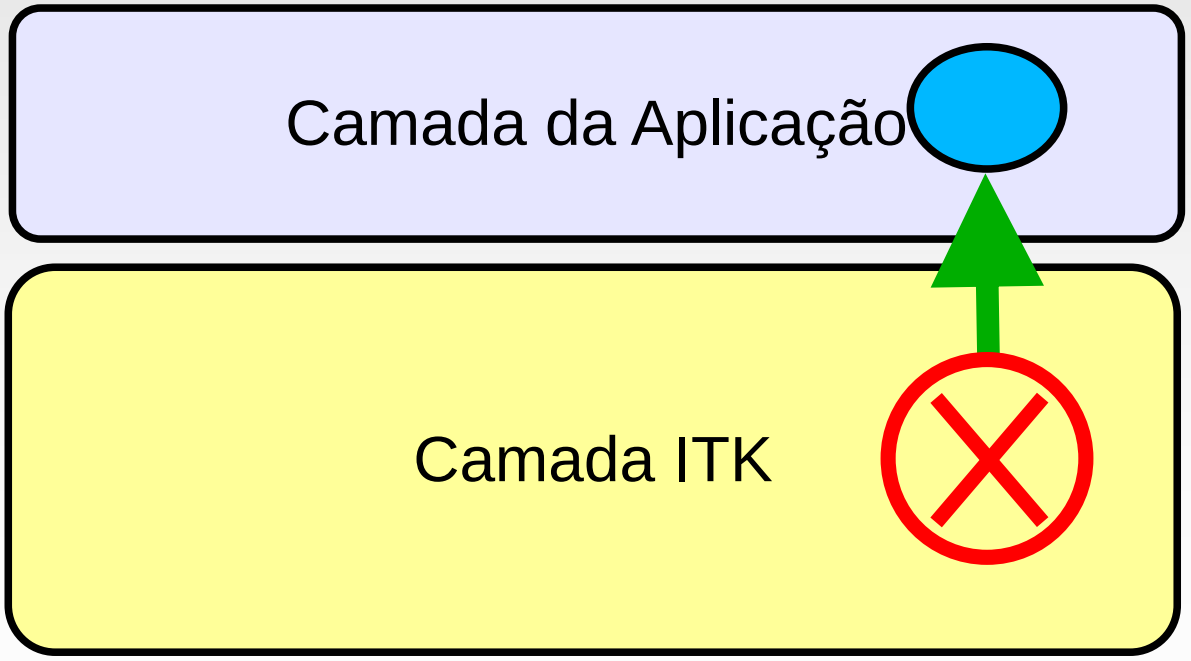
Construtores “Factories”

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

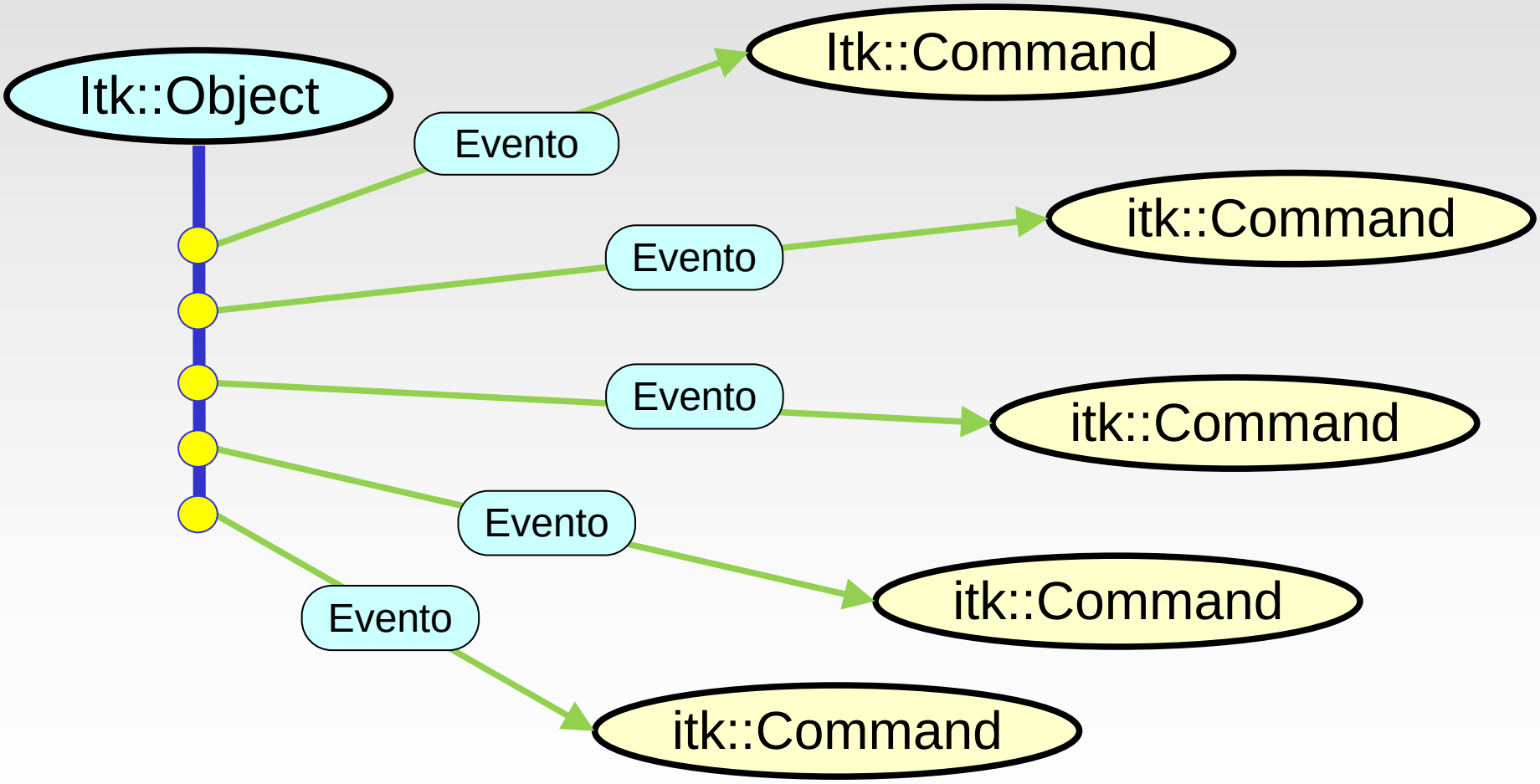
Gerenciamento de Erros



Exceções

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

Eventos e Observadores



Eventos e Observadores

Eventos Comuns

AnyEvent()

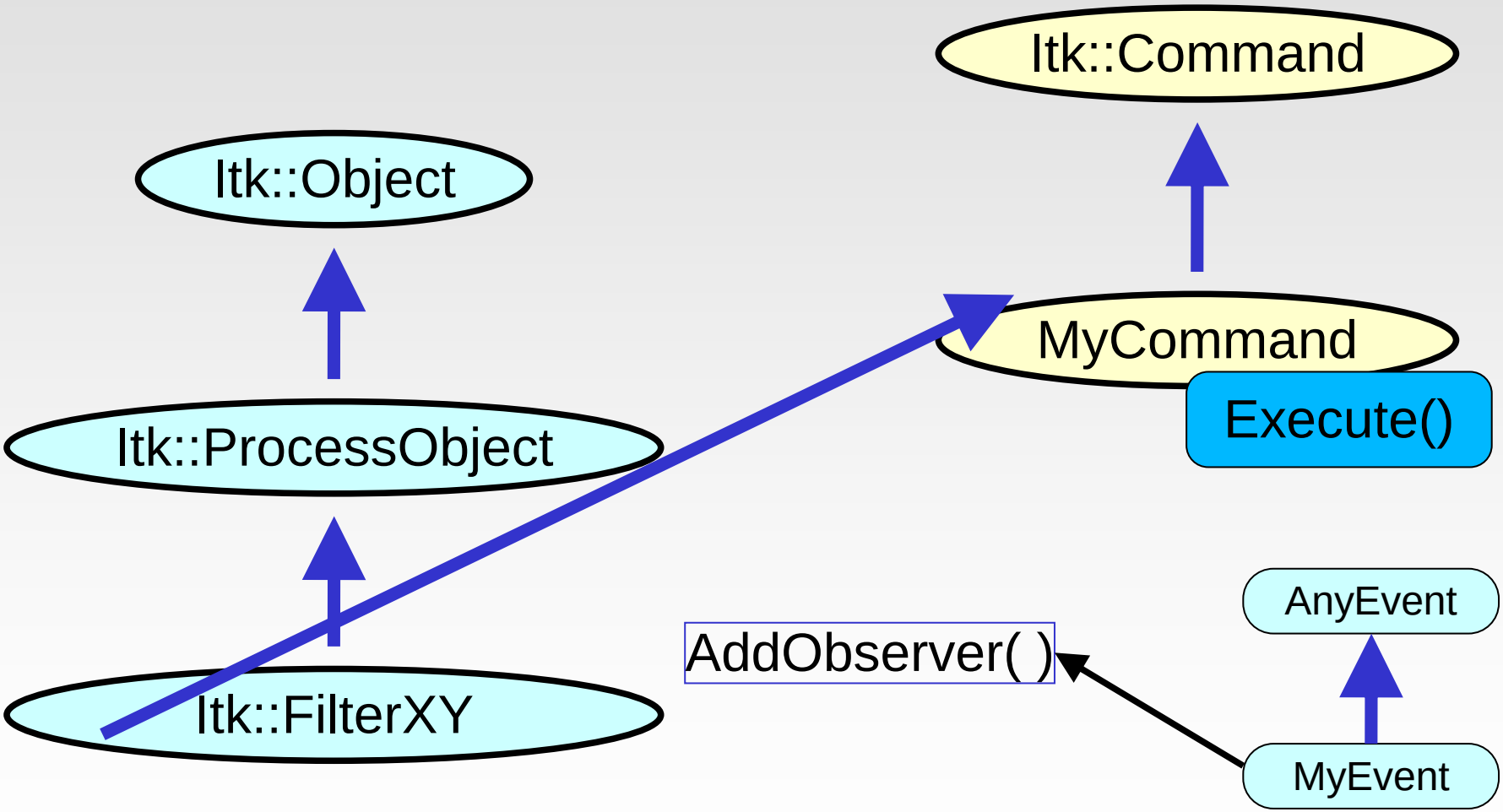
StartEvent()

EndEvent()

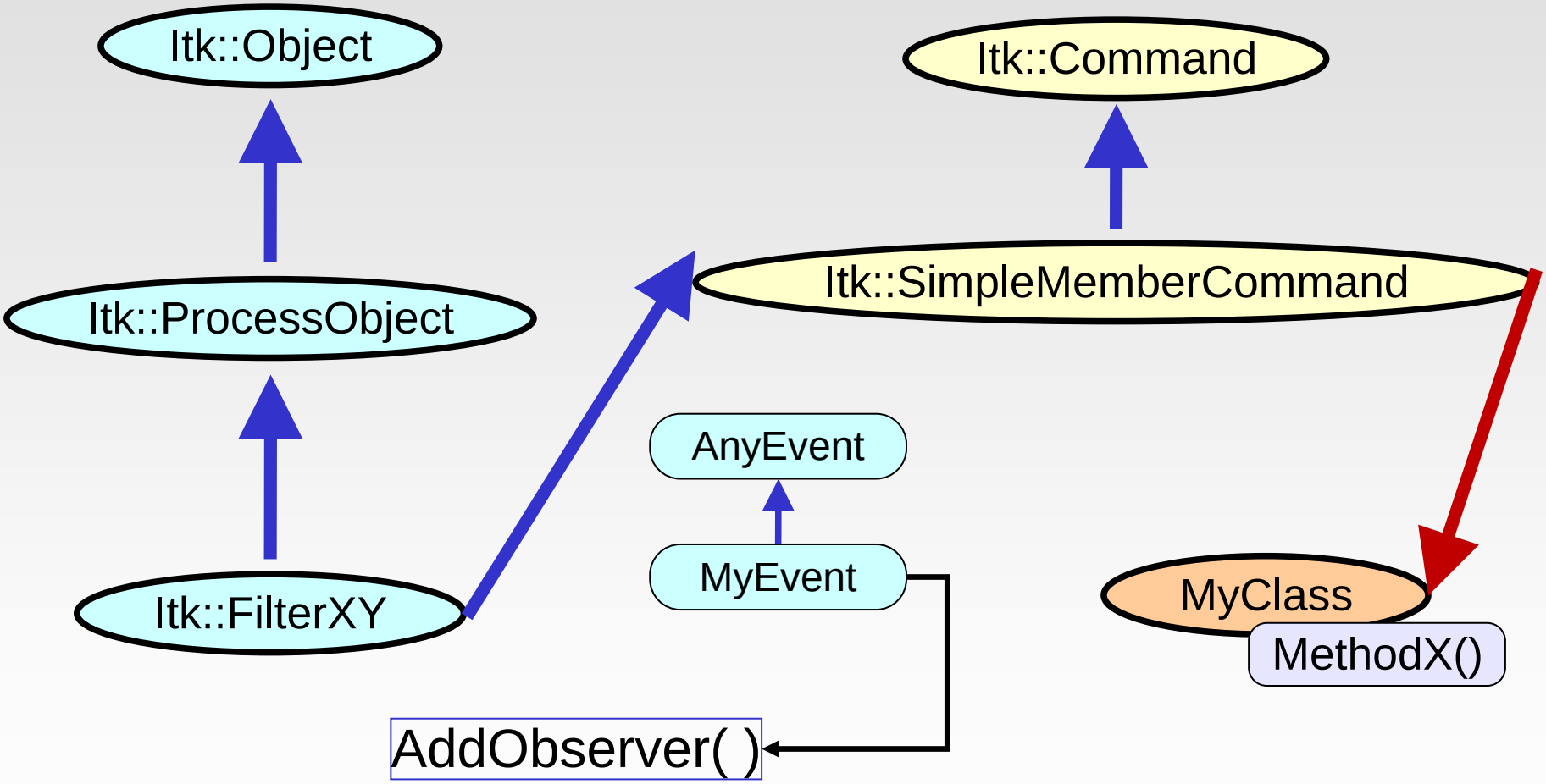
ProgressEvent()

IterationEvent()

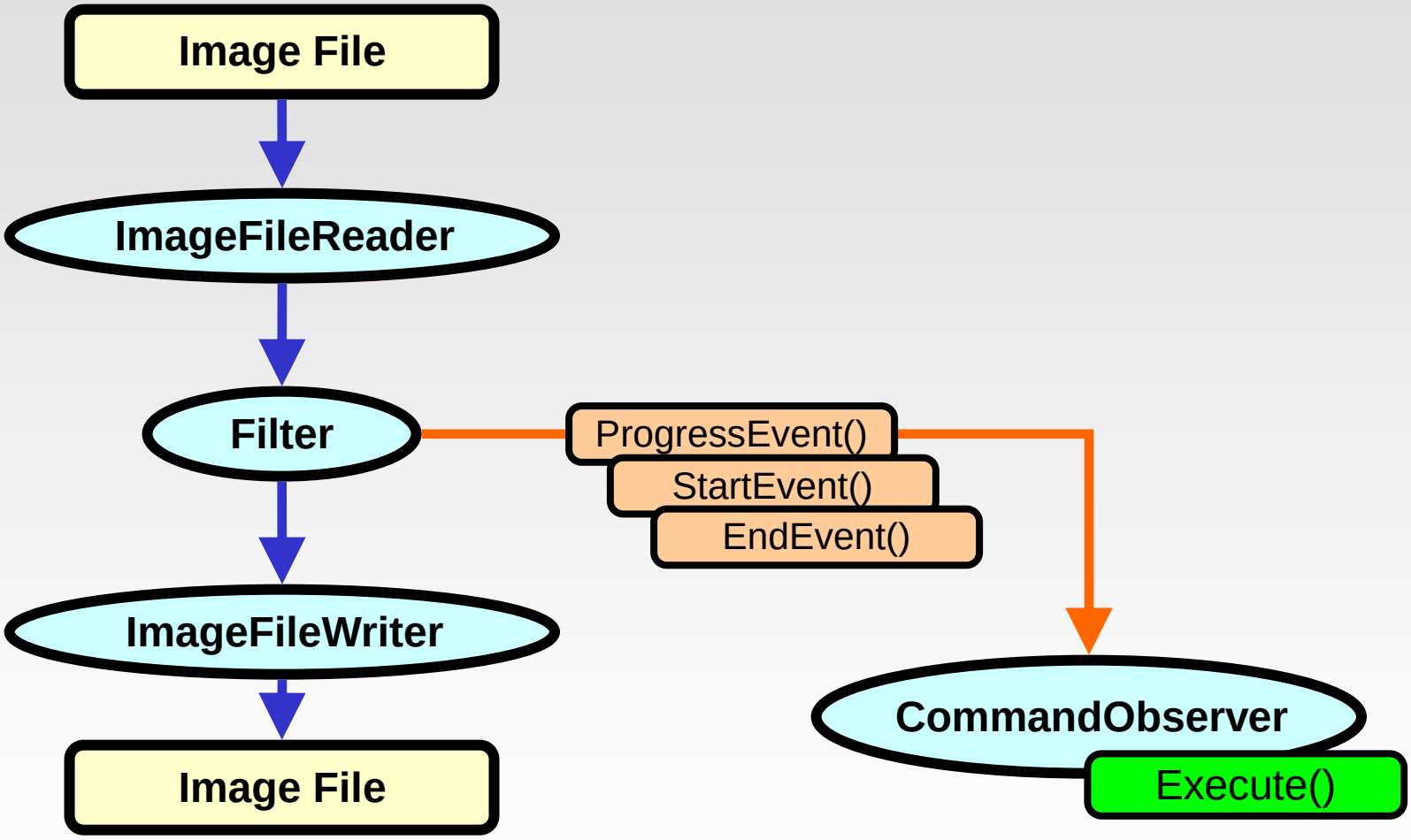
Eventos e Observadores



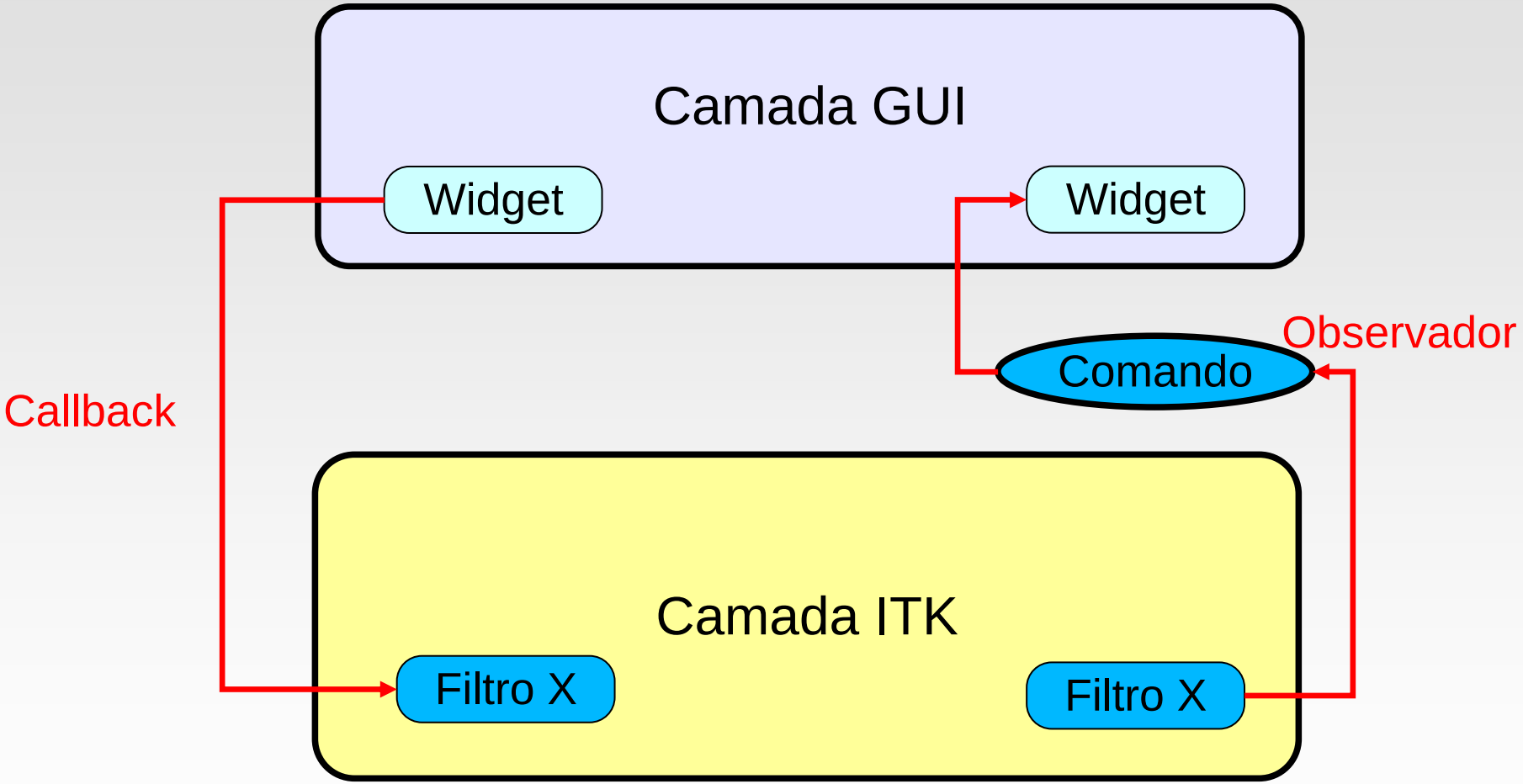
Eventos e Observadores



Eventos e Observadores



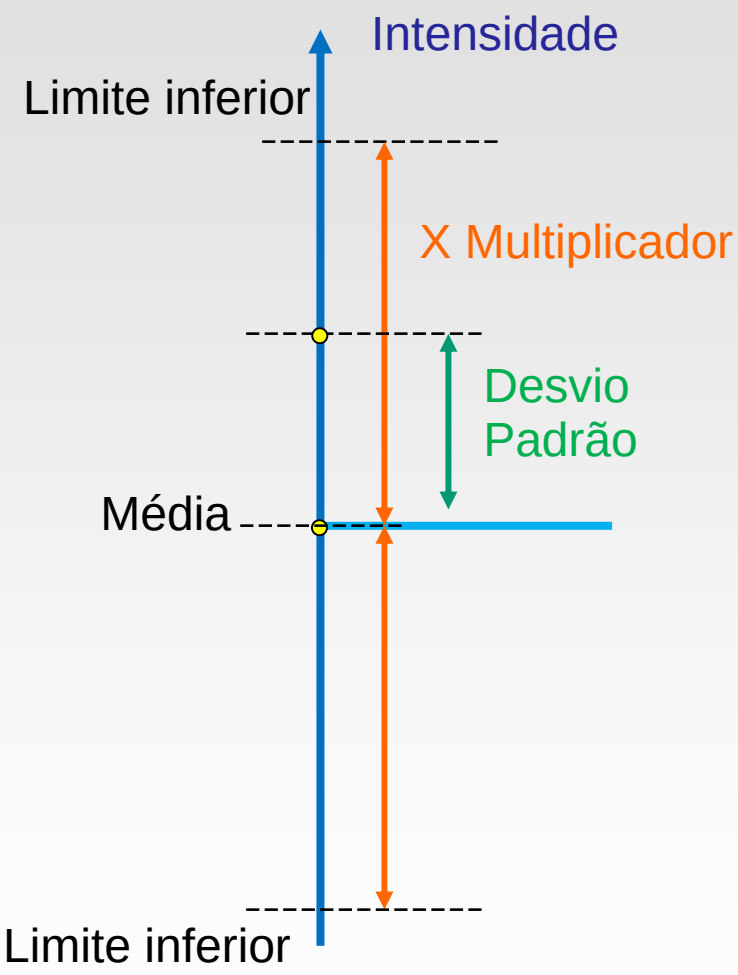
Communicação com GUI

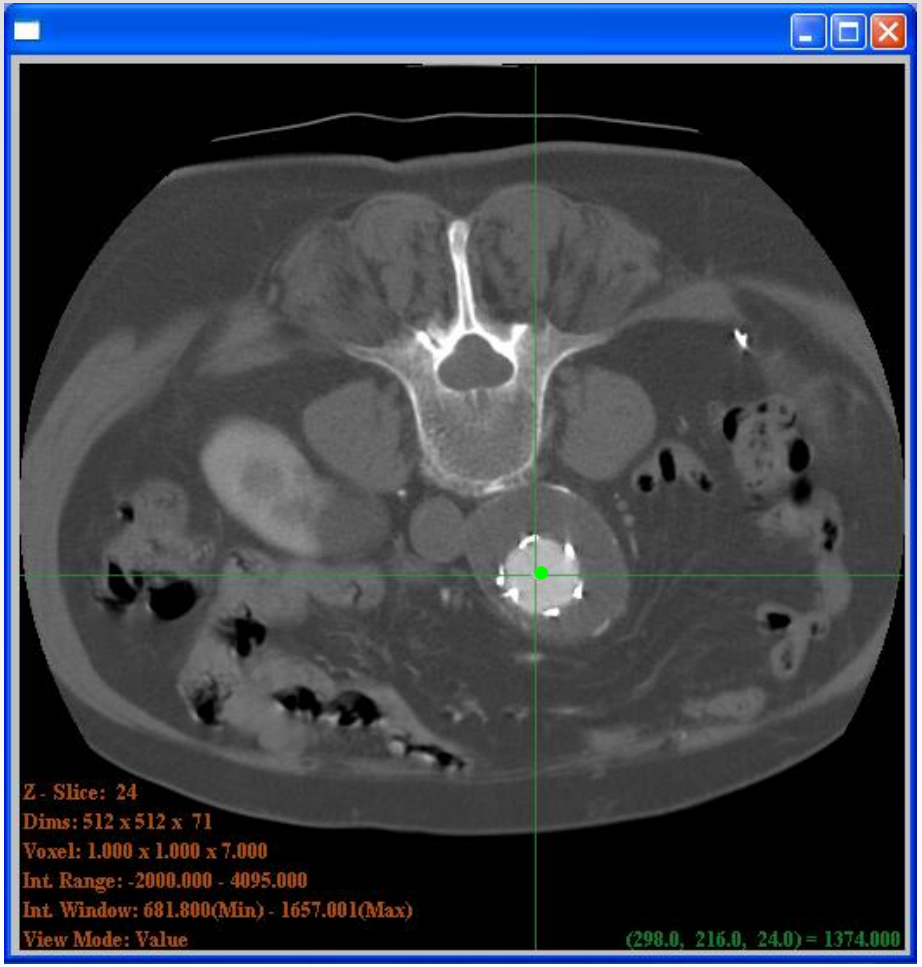


Segmentação

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

Conexão de Confiança



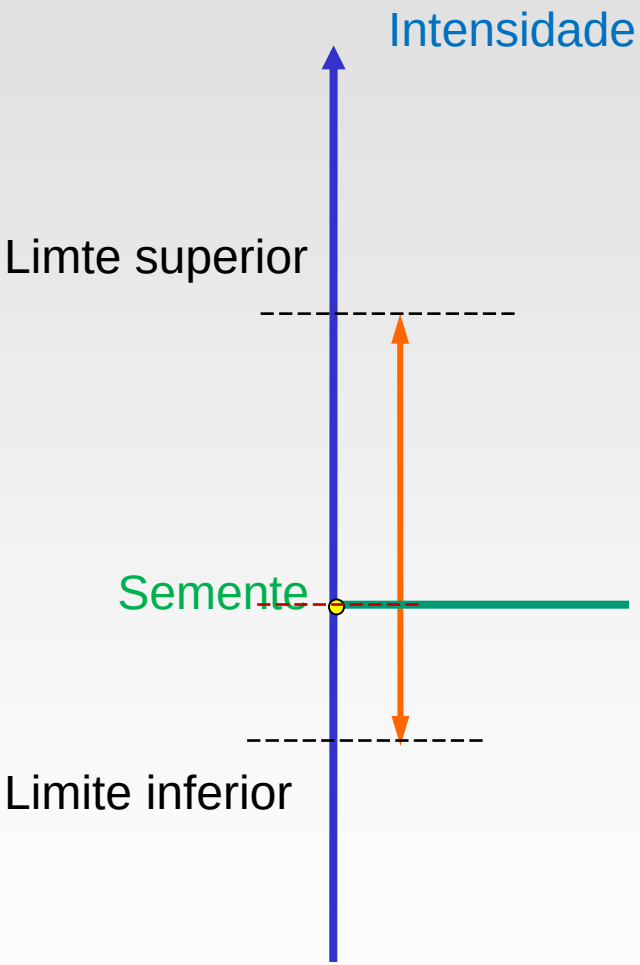


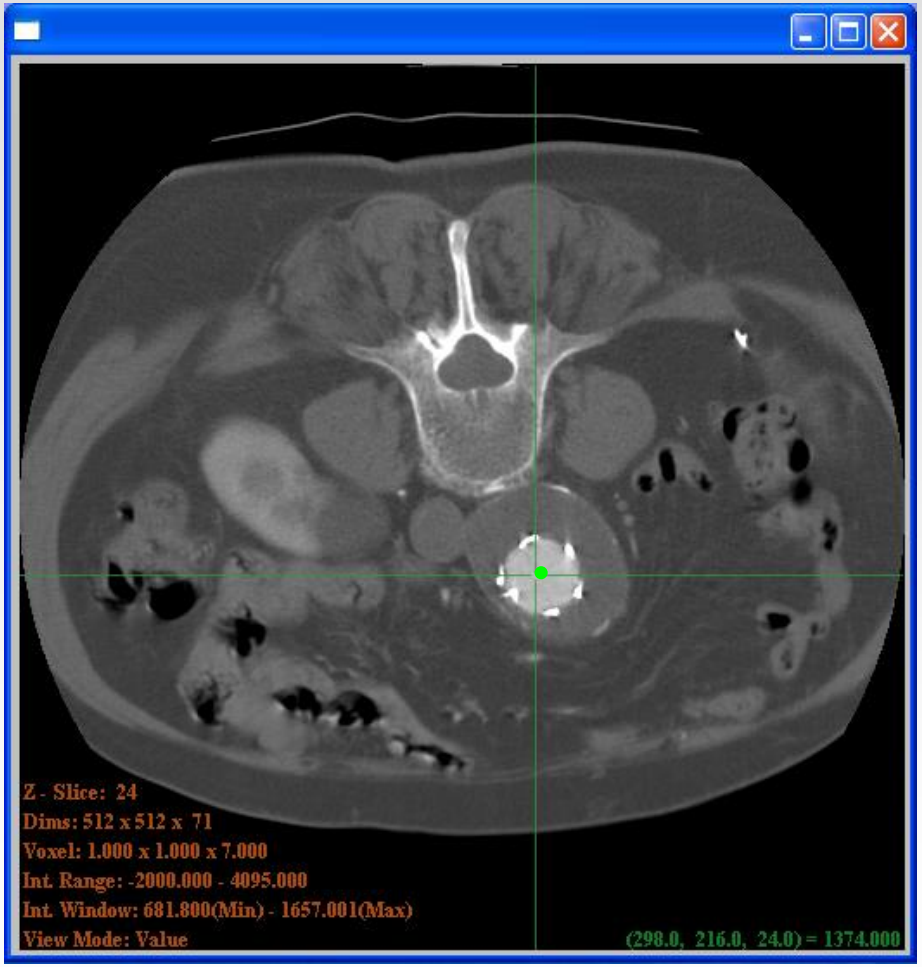
Ponto semente

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



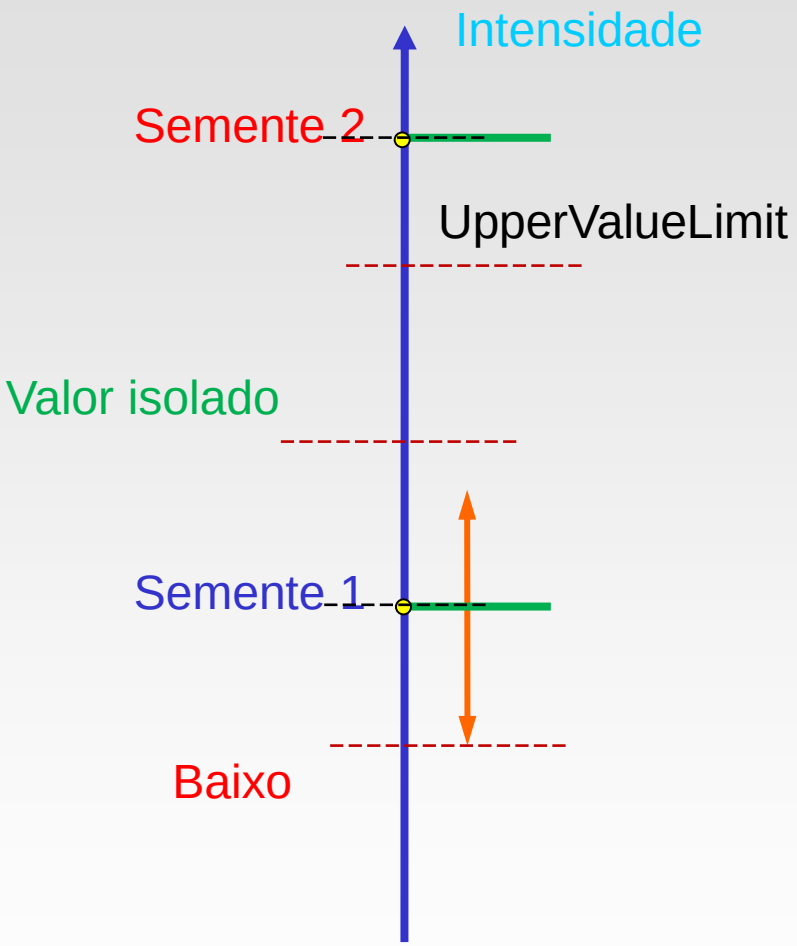


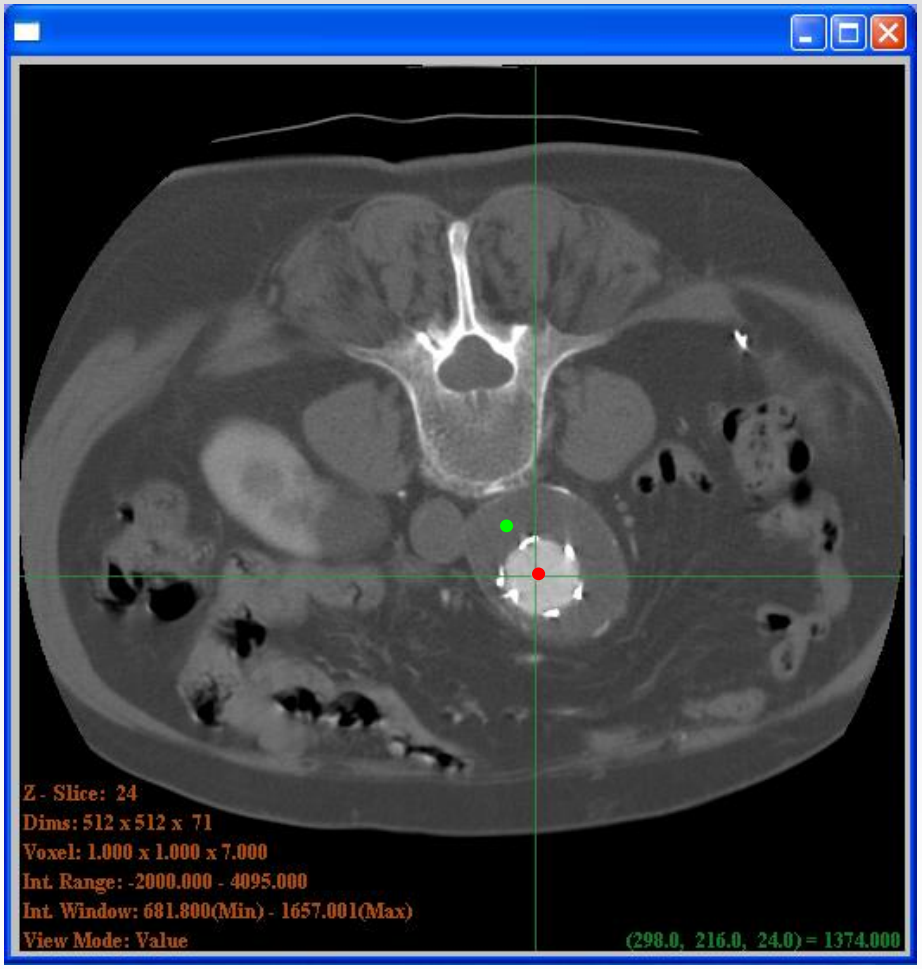
Ponto semente

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





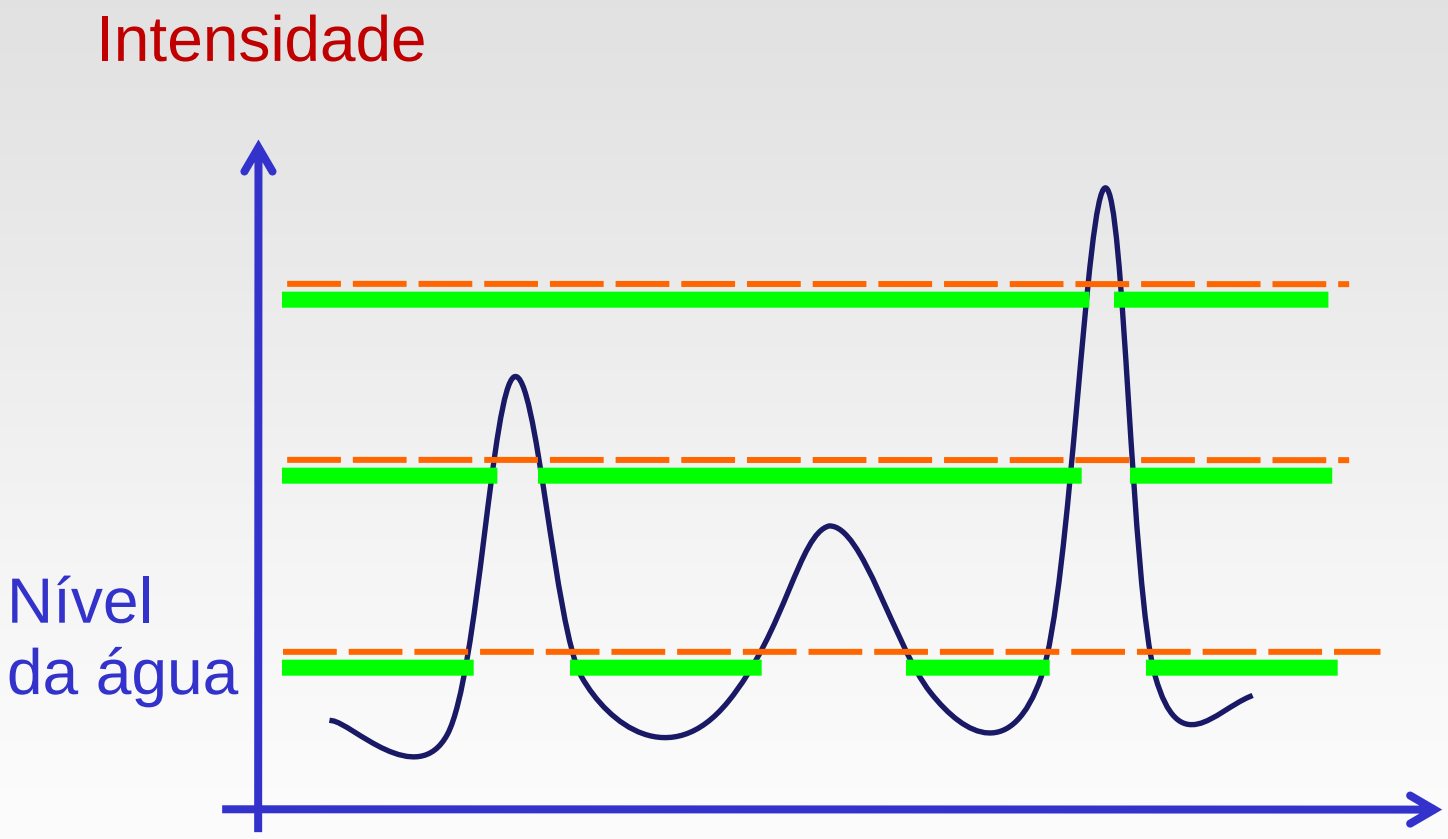
2 pontos semente

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

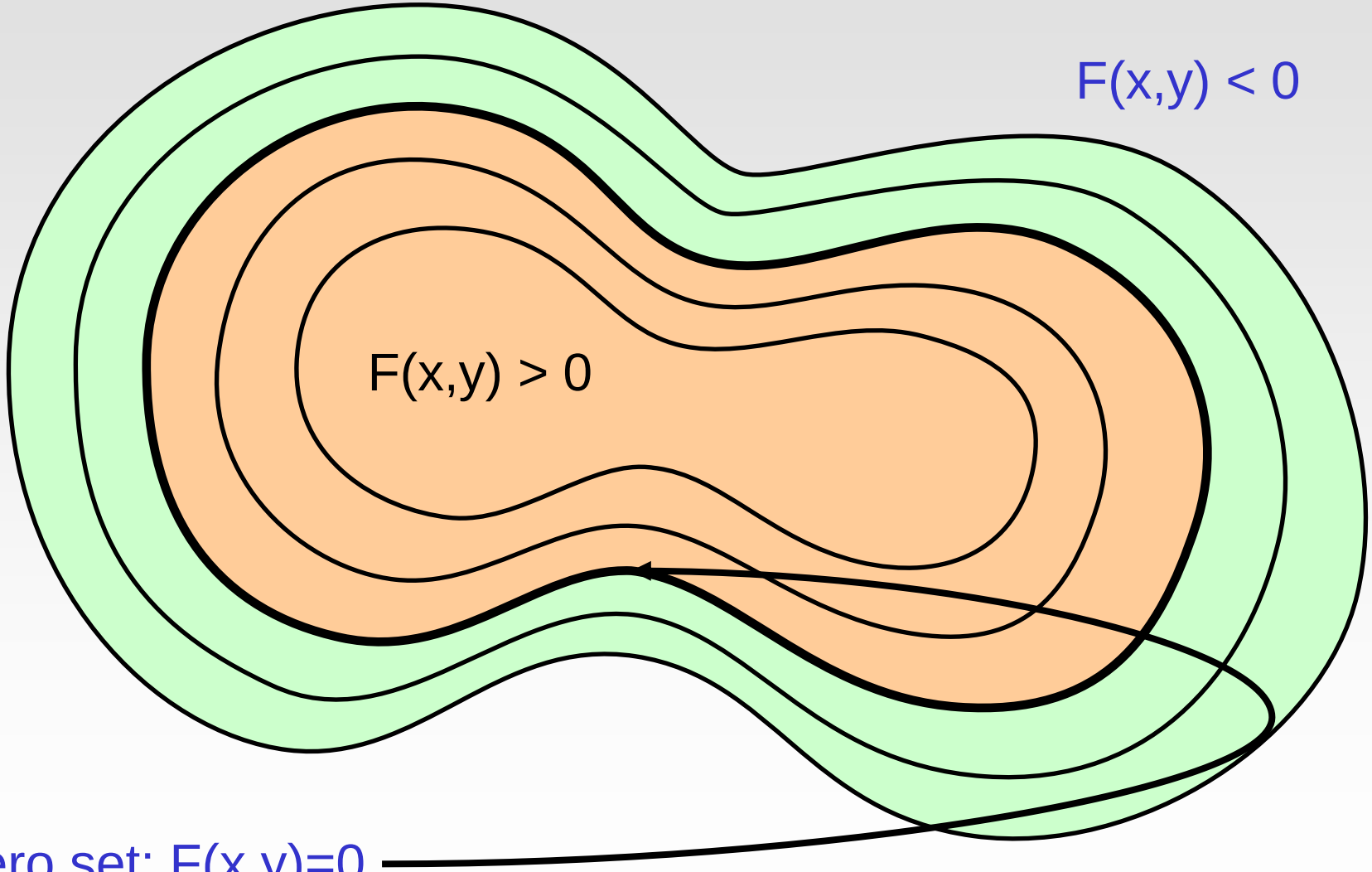
```
typedef   itk::ScalarToRGBPixelFunctor< unsigned long >   FunctorType;
typedef   WatershedFilterType::OutputImageType    LabeledImageType;
typedef   itk::UnaryFunctorImageFilter<         ImageType,
                                        LabeledImageType,
                                        FunctorType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

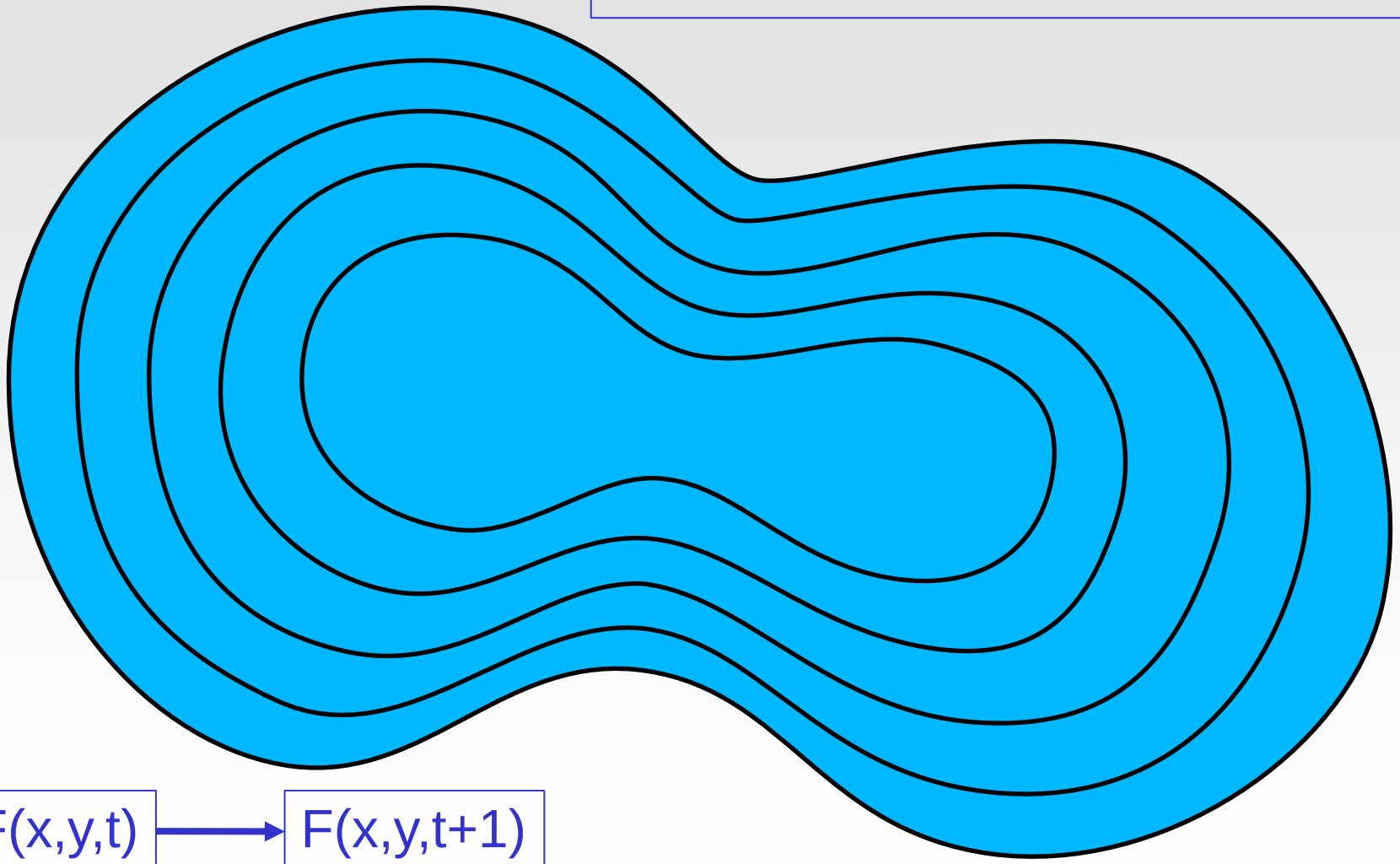
Métodos de Segmentação "Level Set"

Conceito Level set



Evolução Level Set

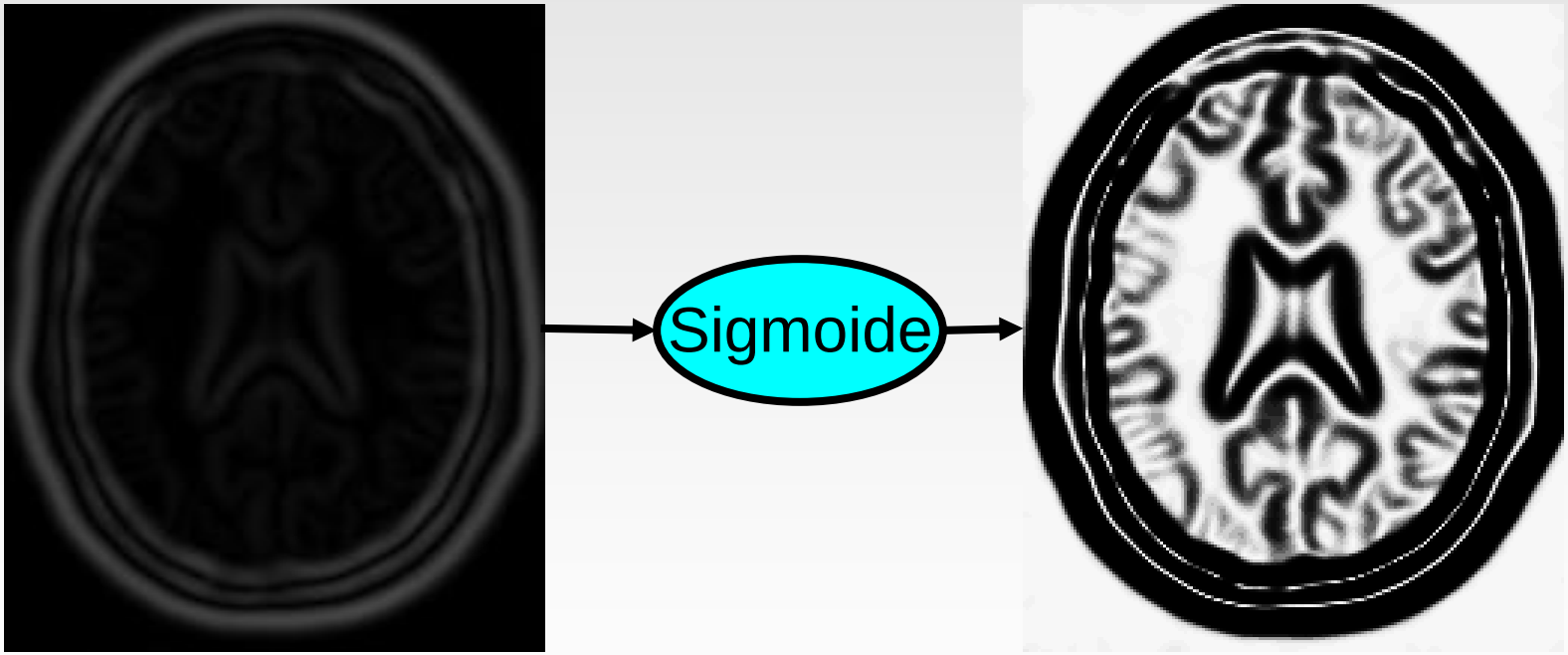
PDE = Automato Celular Restrito



F(x,y,t) → F(x,y,t+1)

Marcha rápida

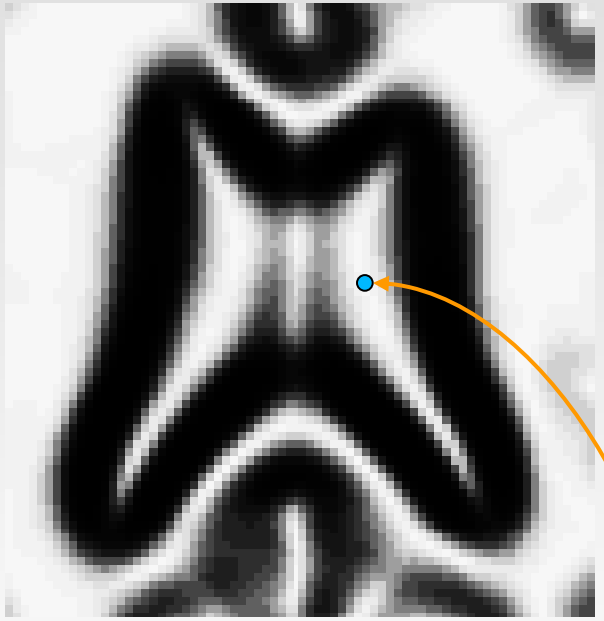
Propagação de Fronte $\Delta x = V \cdot \Delta t$



Magnitude de Gradiente

Imagem de velocidade

Marcha rápida



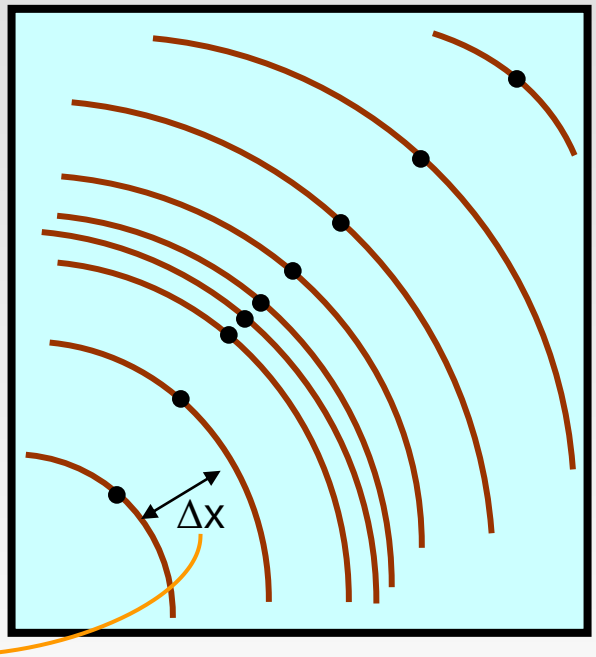


Imagem Velocidade

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 );
```

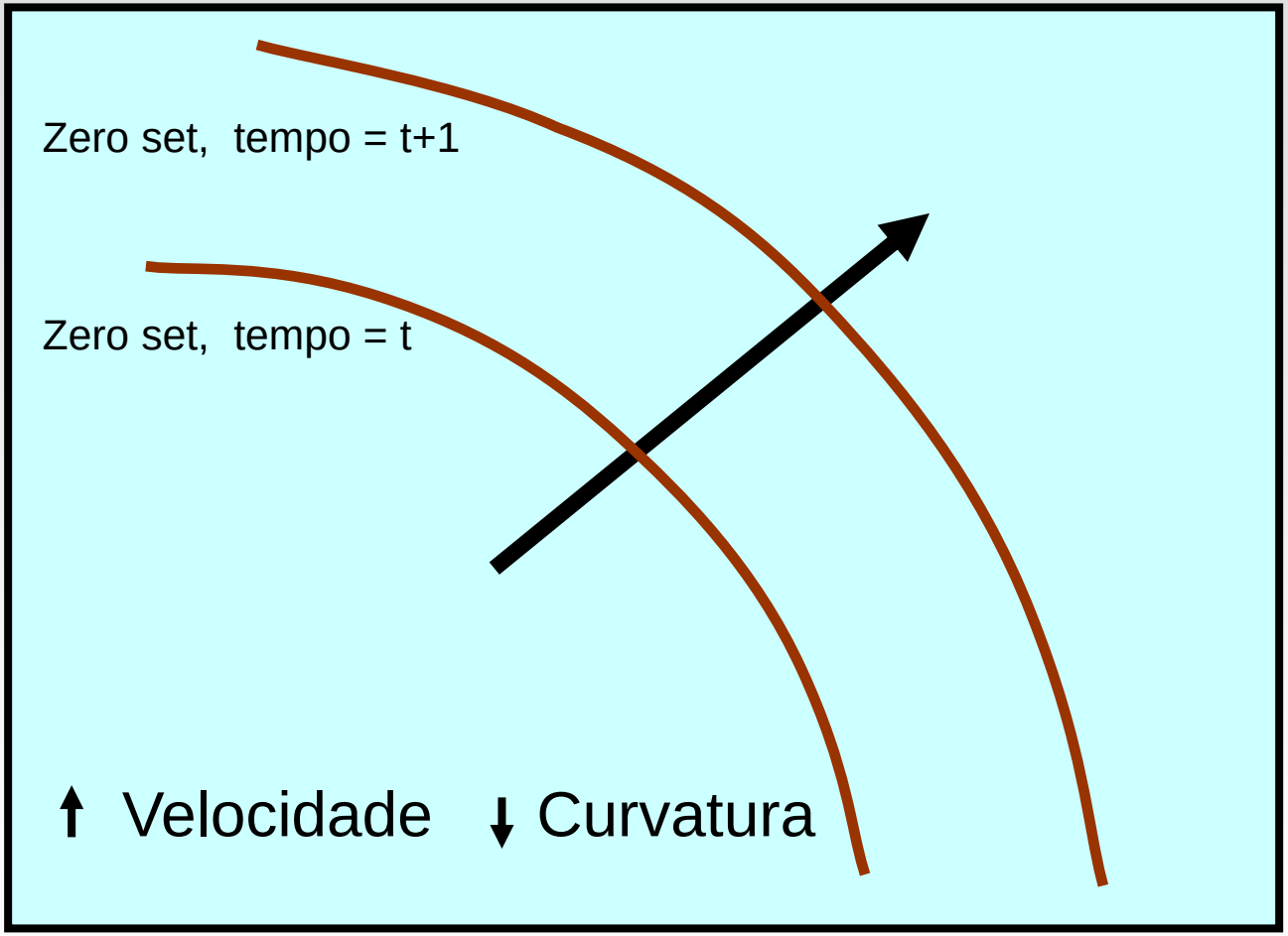
Marcha rápida

```
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();
```

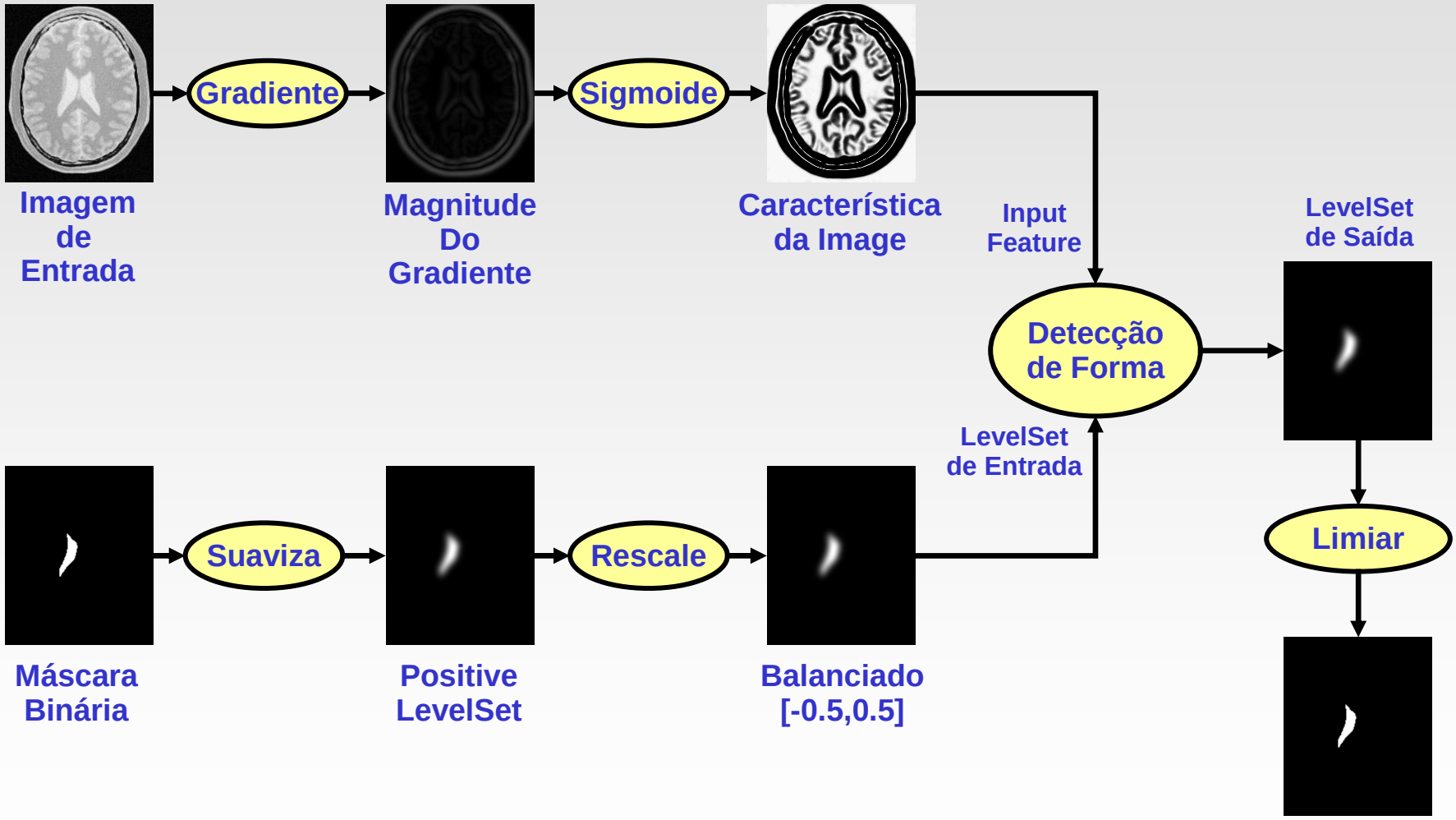
Detecção de Forma



PDE Inclui um termo curvatura

Previne vazamento

Detecção de Forma



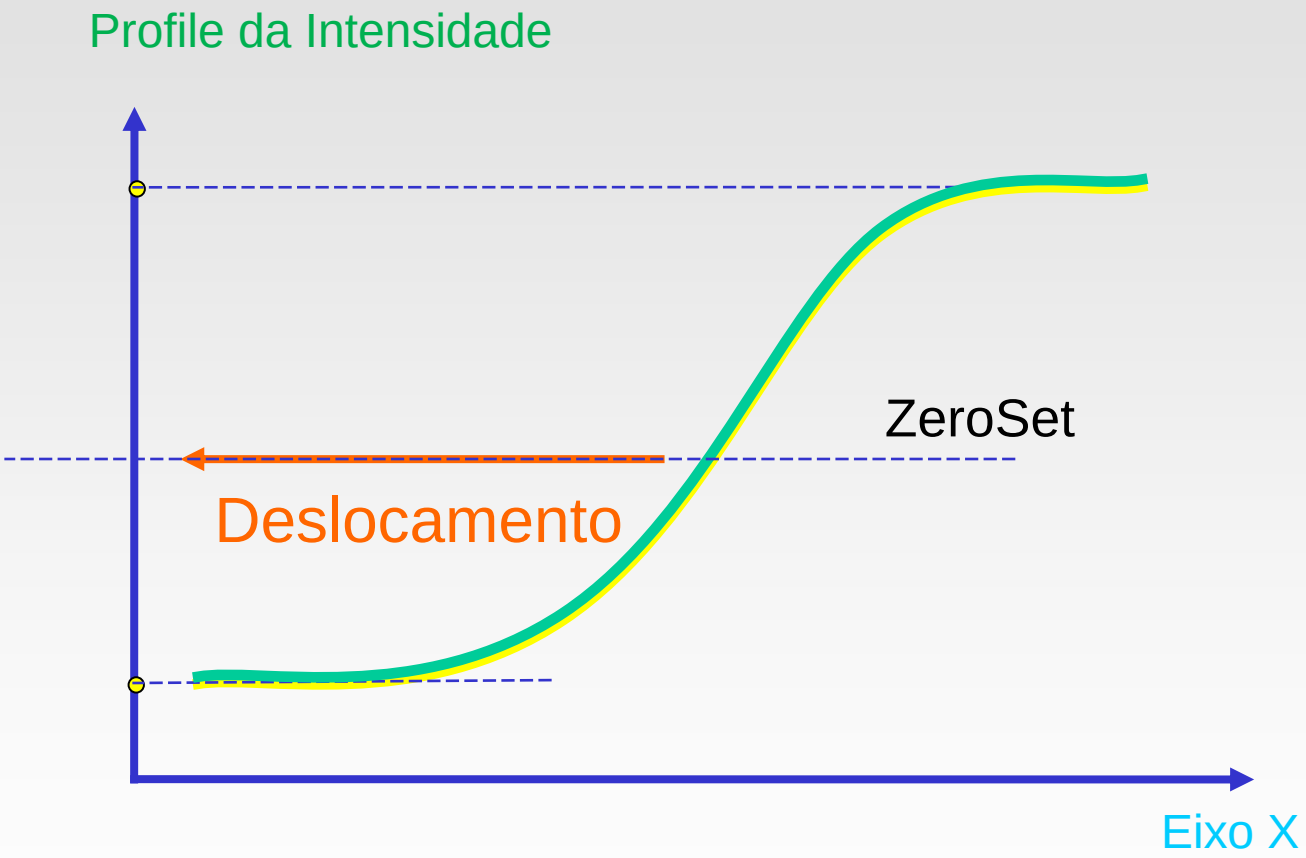
Detecção de Forma

```
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 );
```

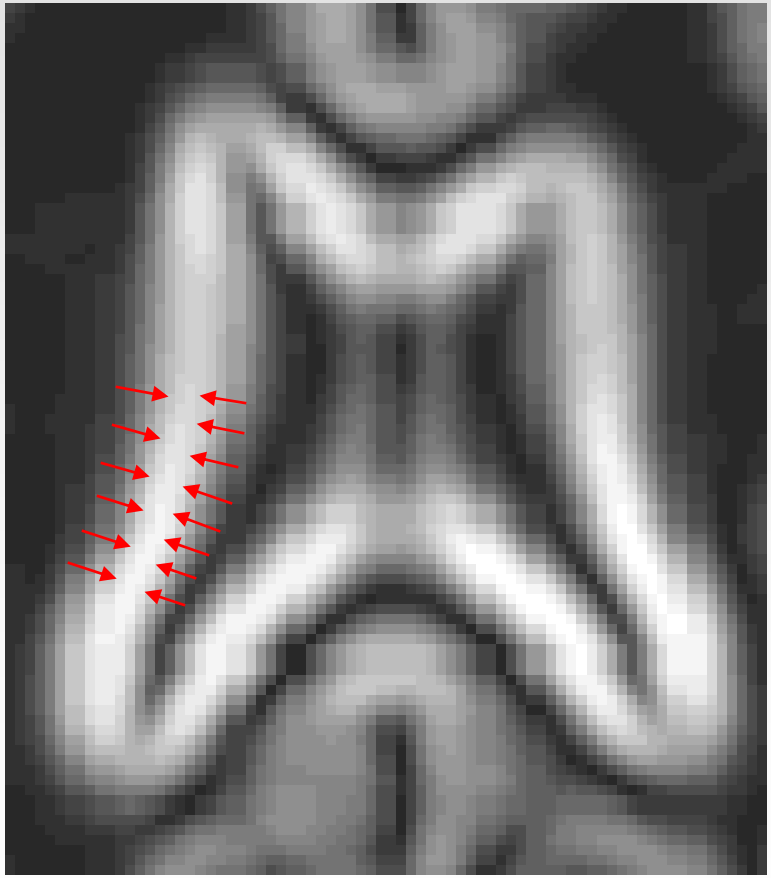
Contorno Ativo Geodésico



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 );
```

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 );
```

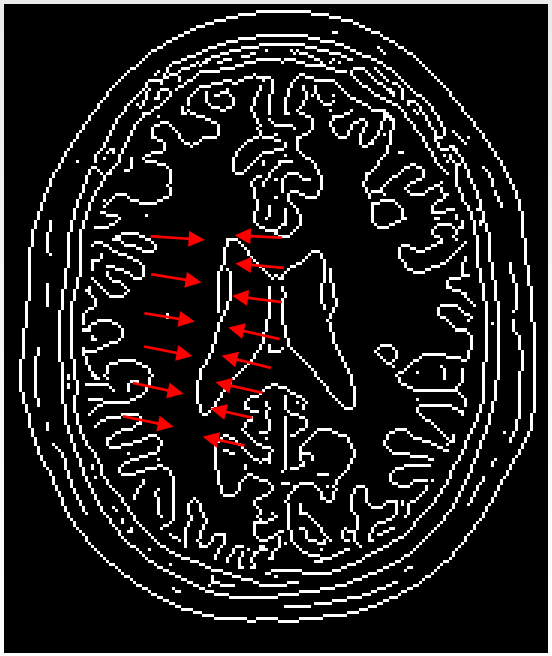
Segmentação por limiar

```
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

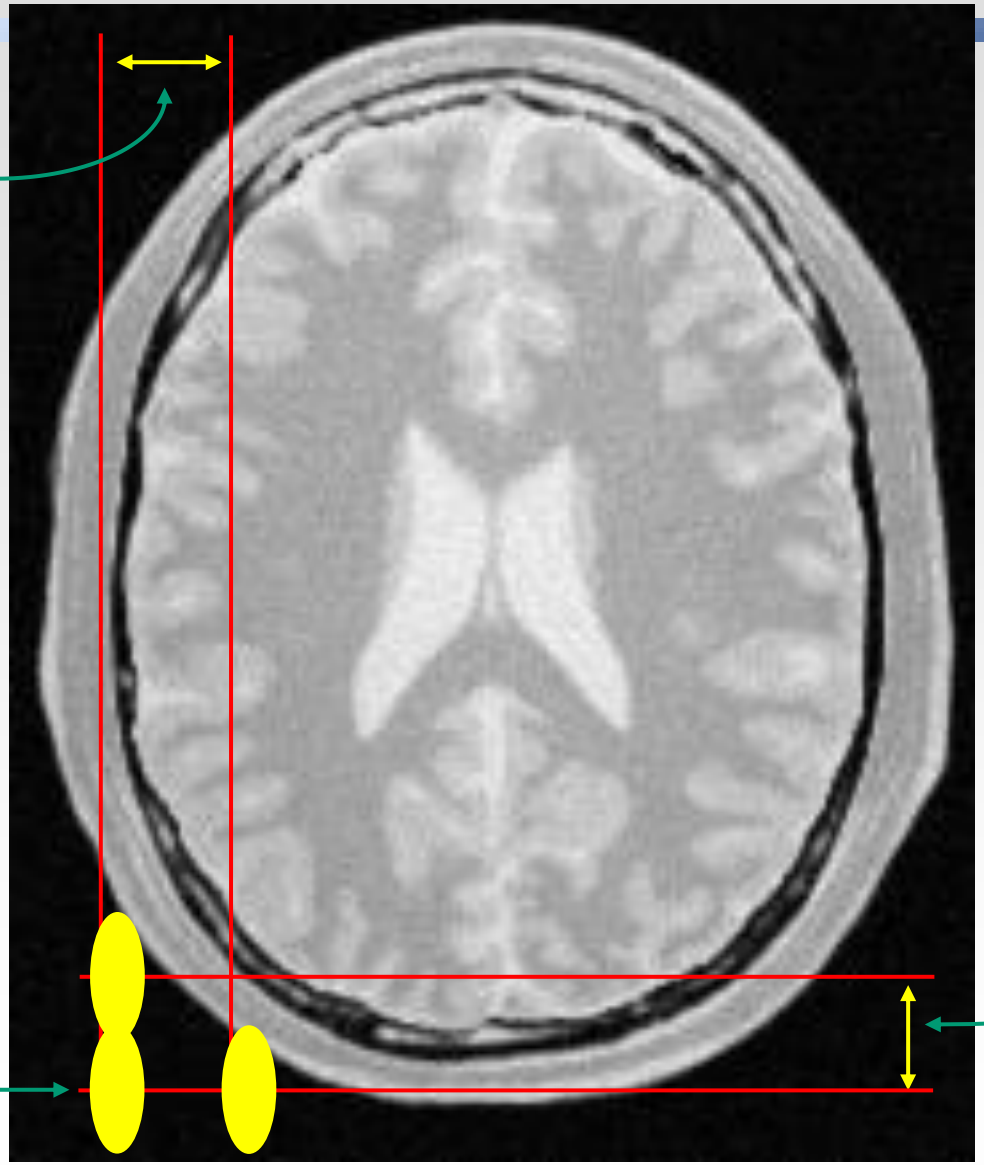
Porquê Reamostragem?

Reamostagem é a Essência
do Corregistro de Imagens
Baseado em Intensidades

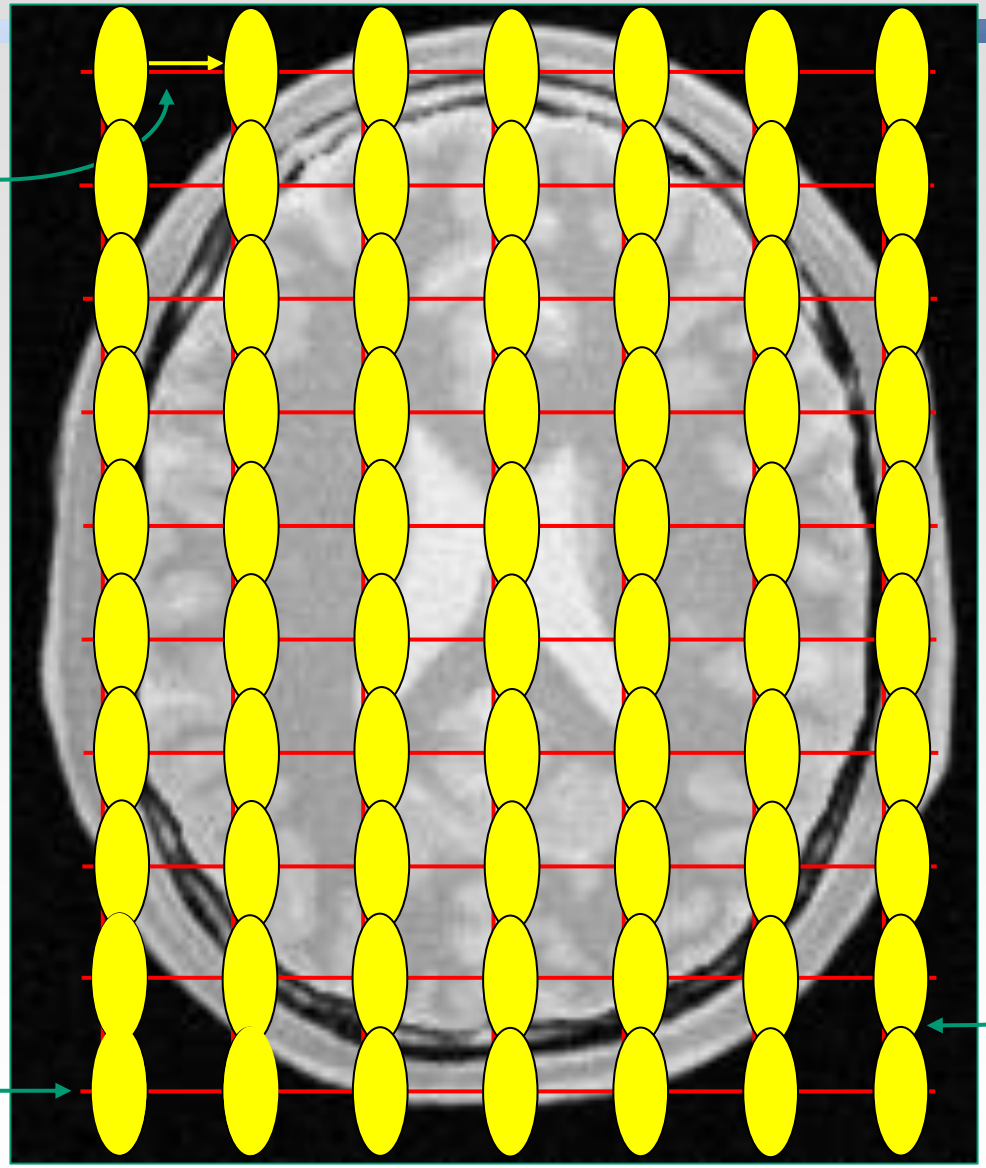
O quê é uma Imagem ?

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

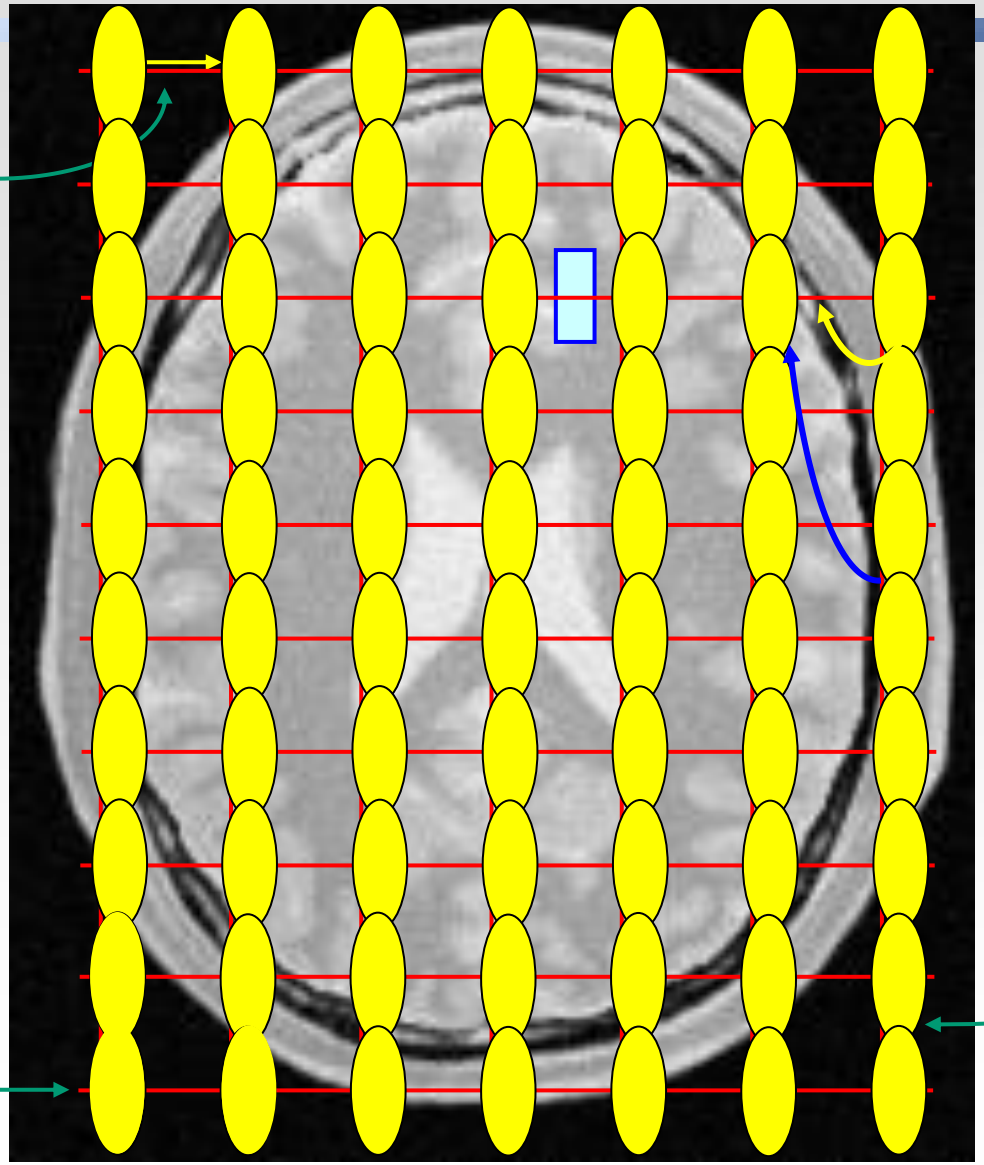
Origem na Imagem & Espaçamento



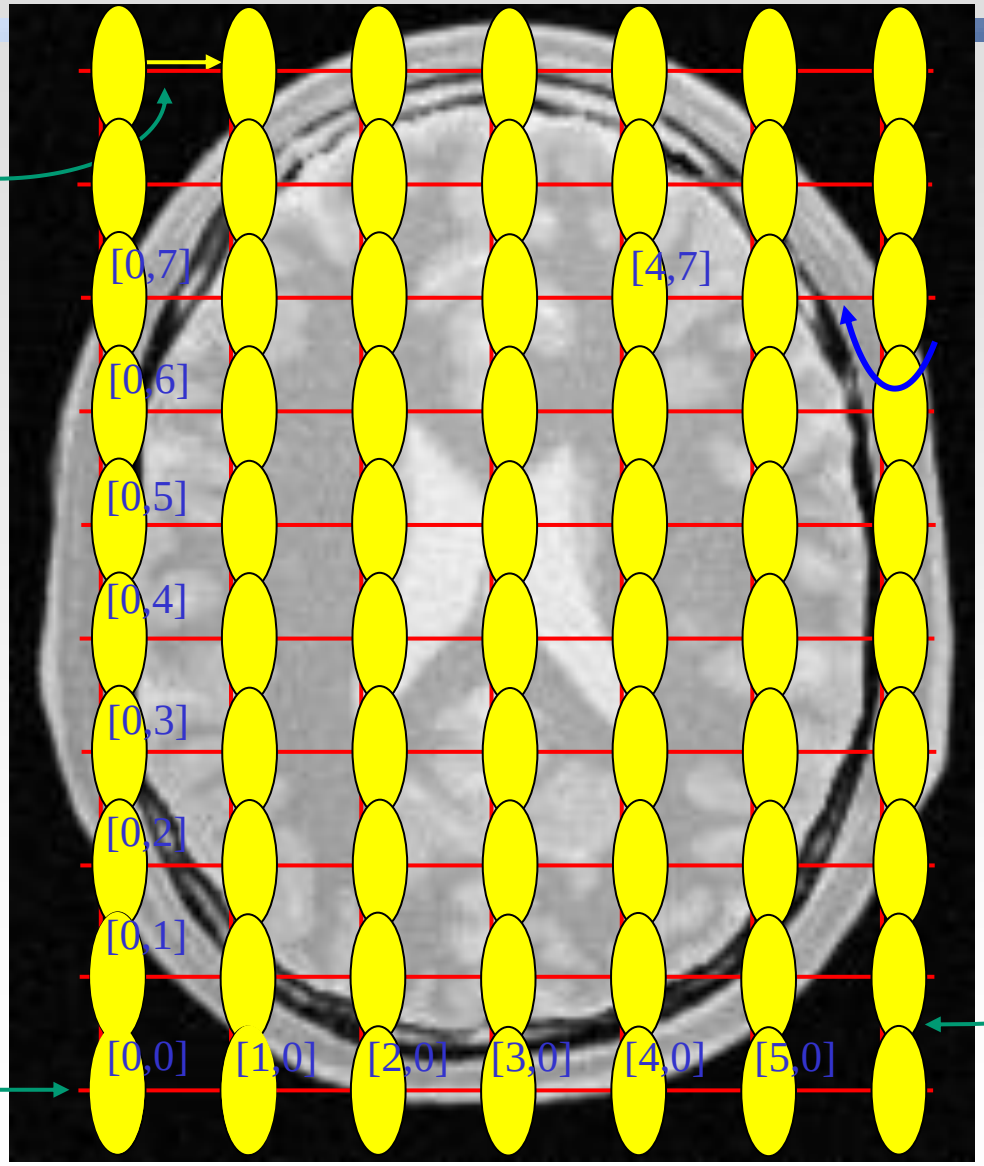
Grade de Amostragem na Imagem



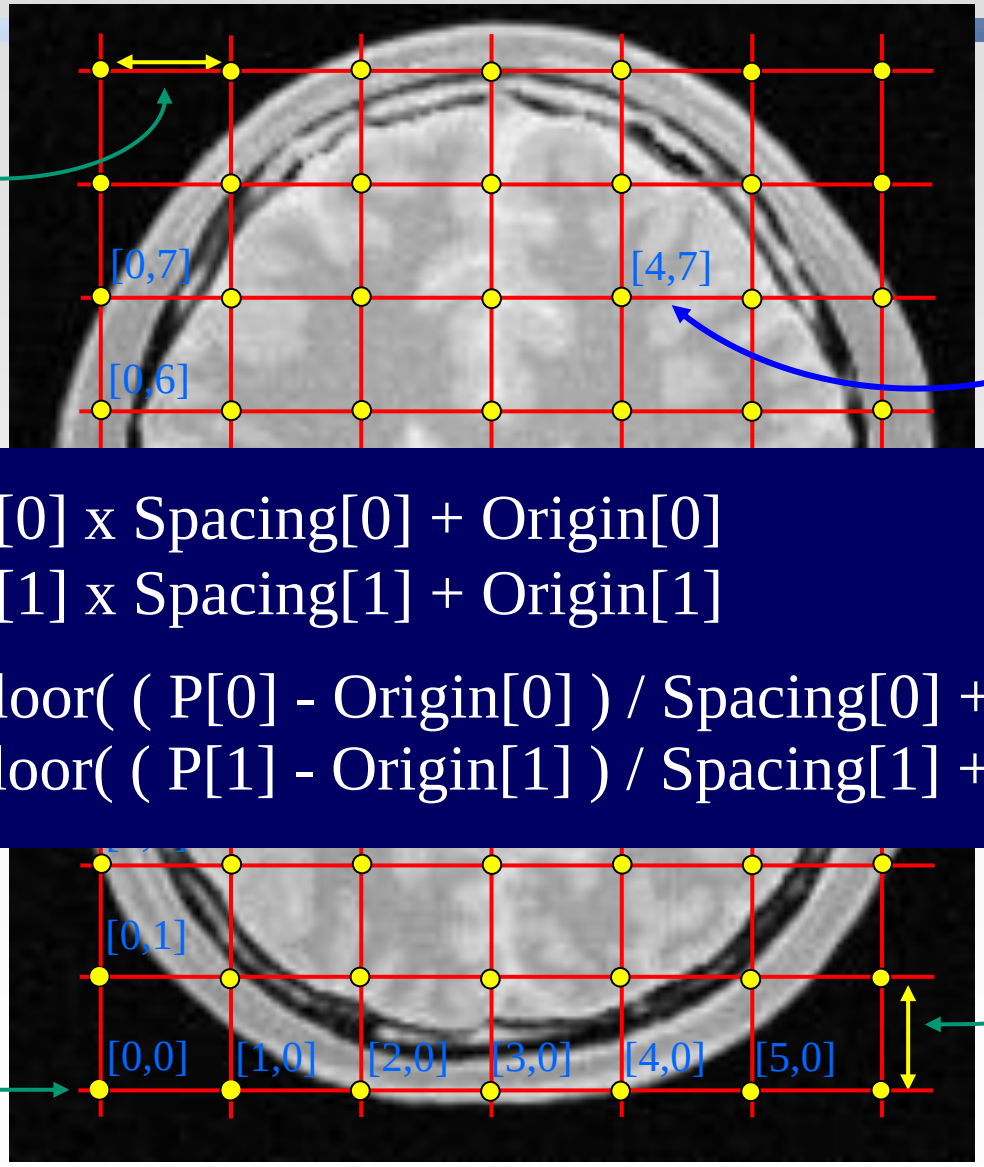
Pixel de Imagem



Índices de Imagens



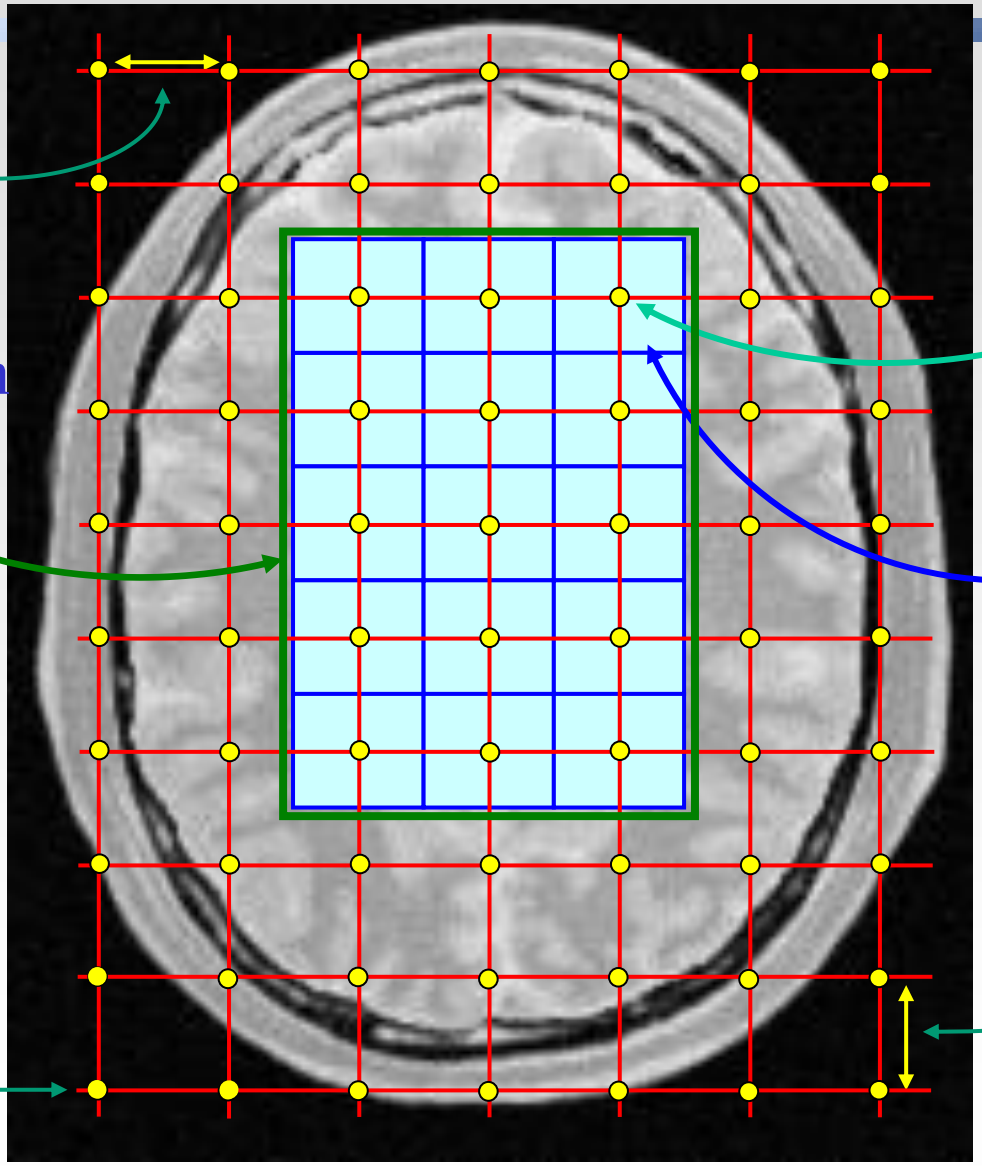
Índice para Coordenadas Físicas



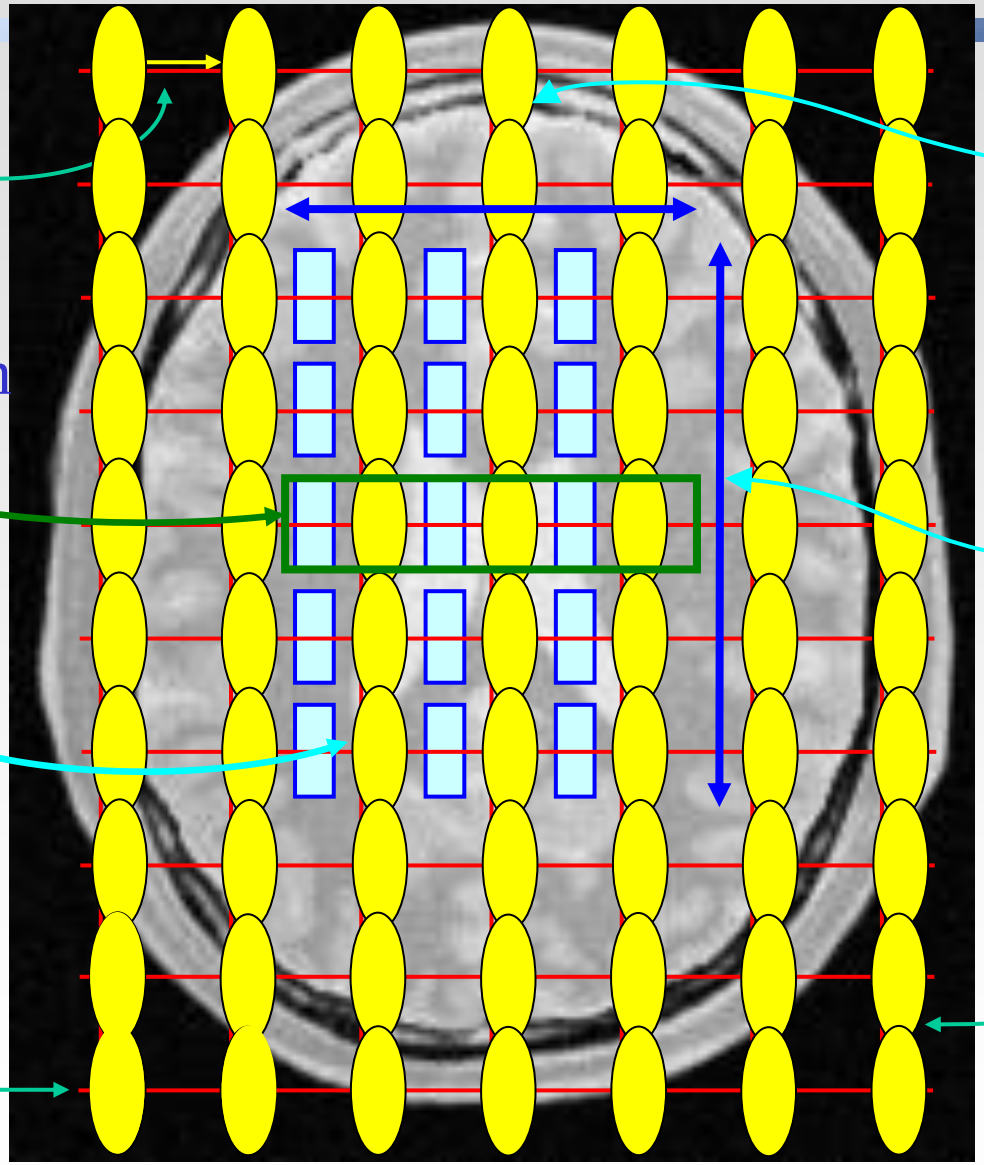
P[0] = Index[0] x Spacing[0] + Origin[0]
P[1] = Index[1] x Spacing[1] + Origin[1]

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



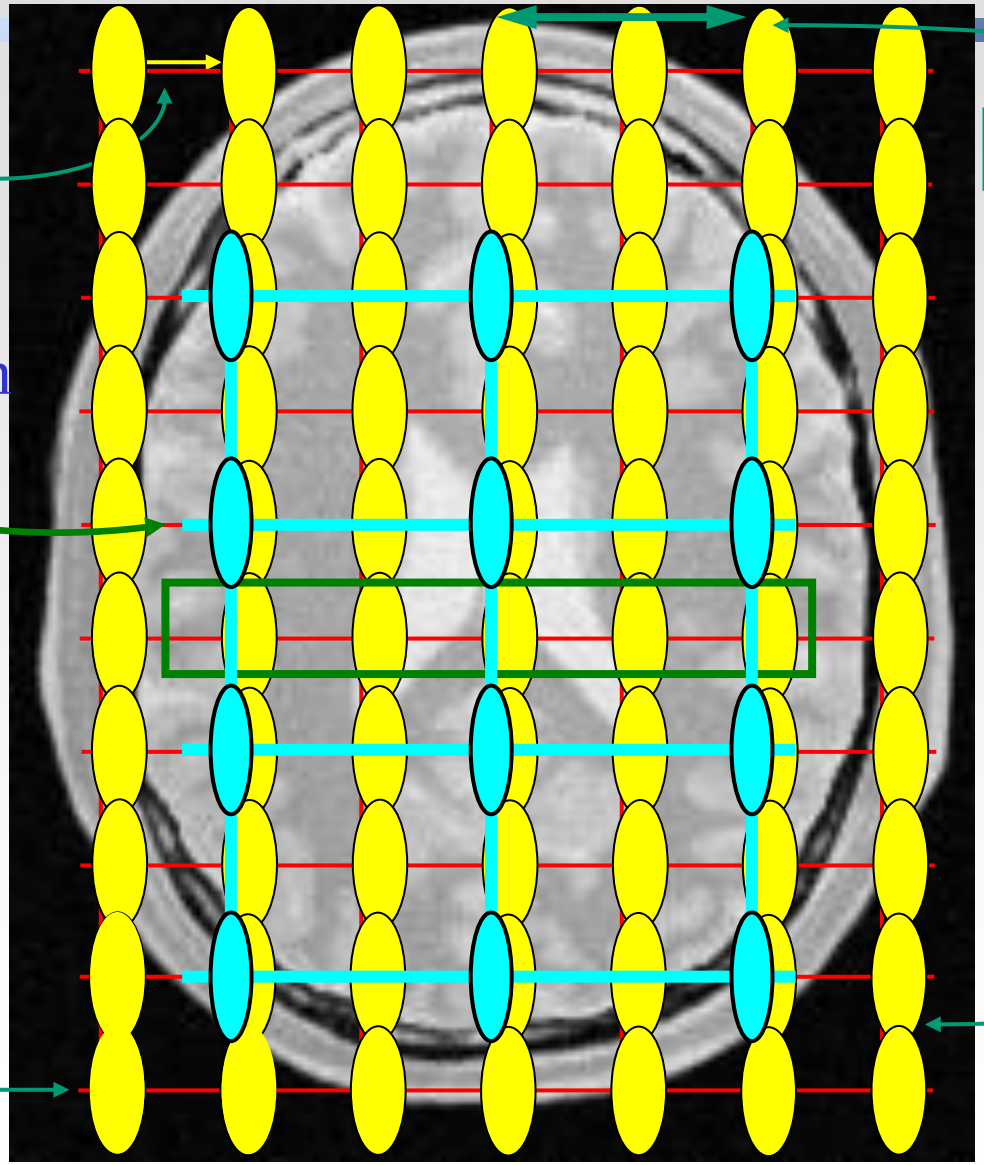
Região da Imagem



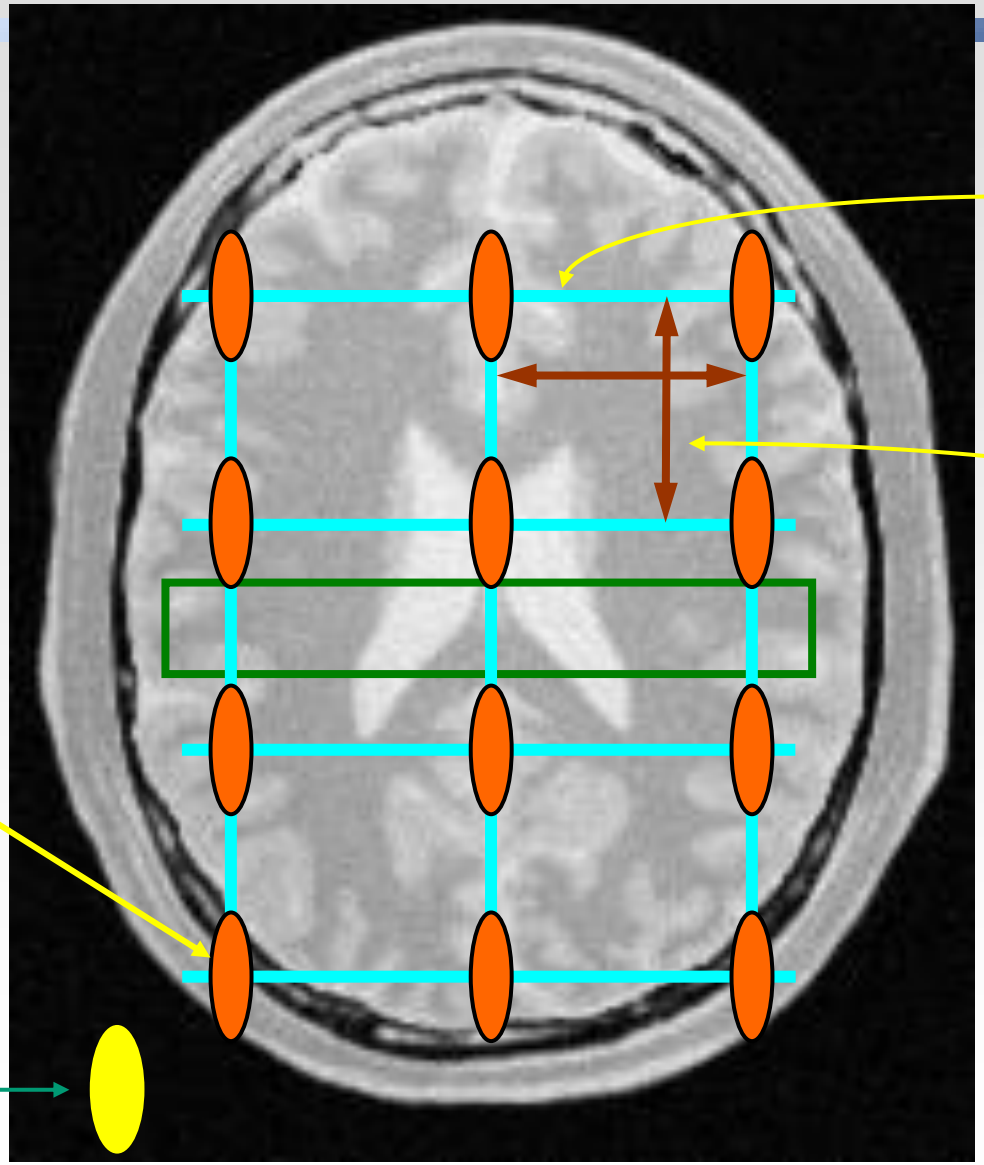
Reamostagem Básica

Reamostragem nos
Casos Triviais

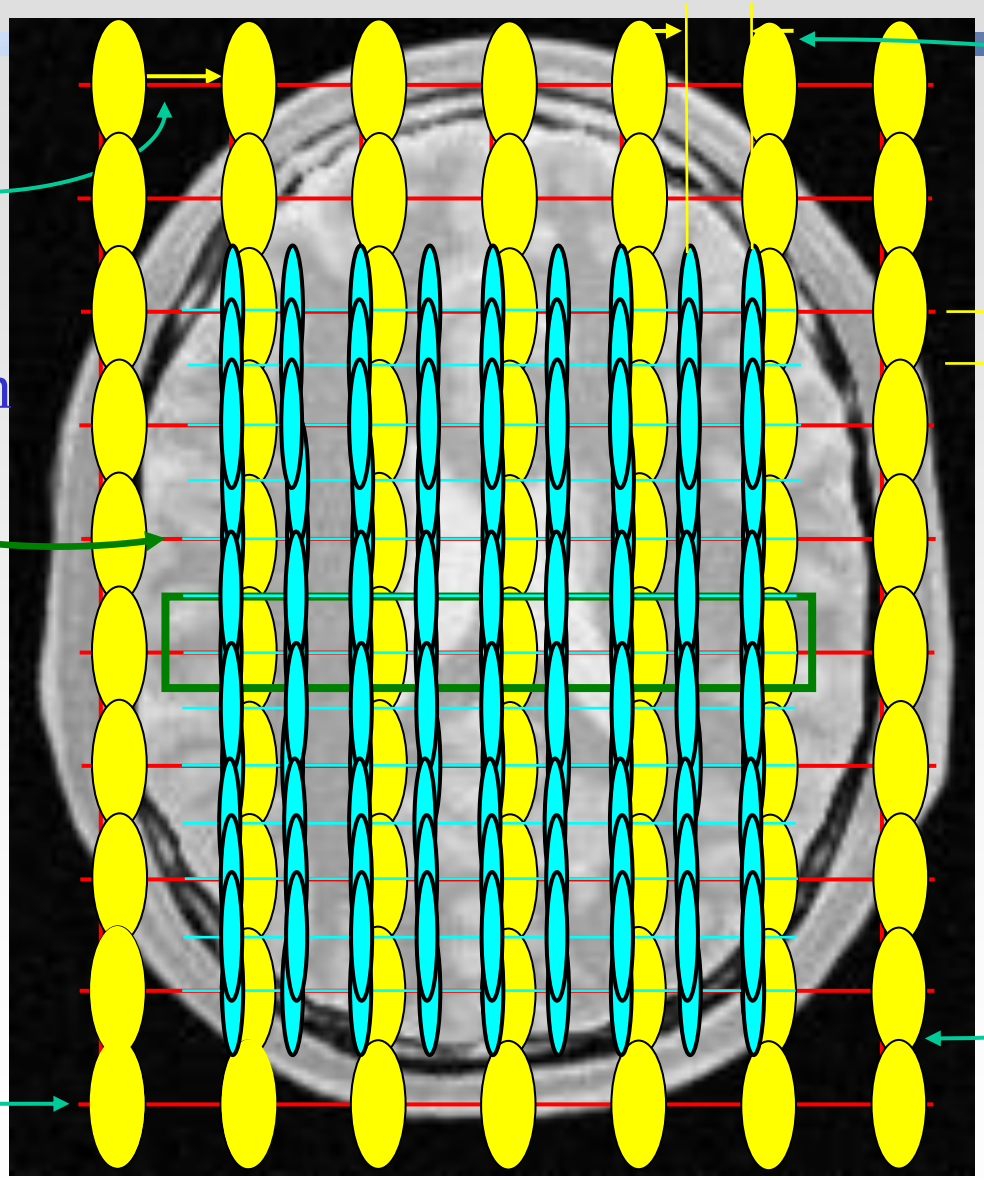
Subamostragem pela metade



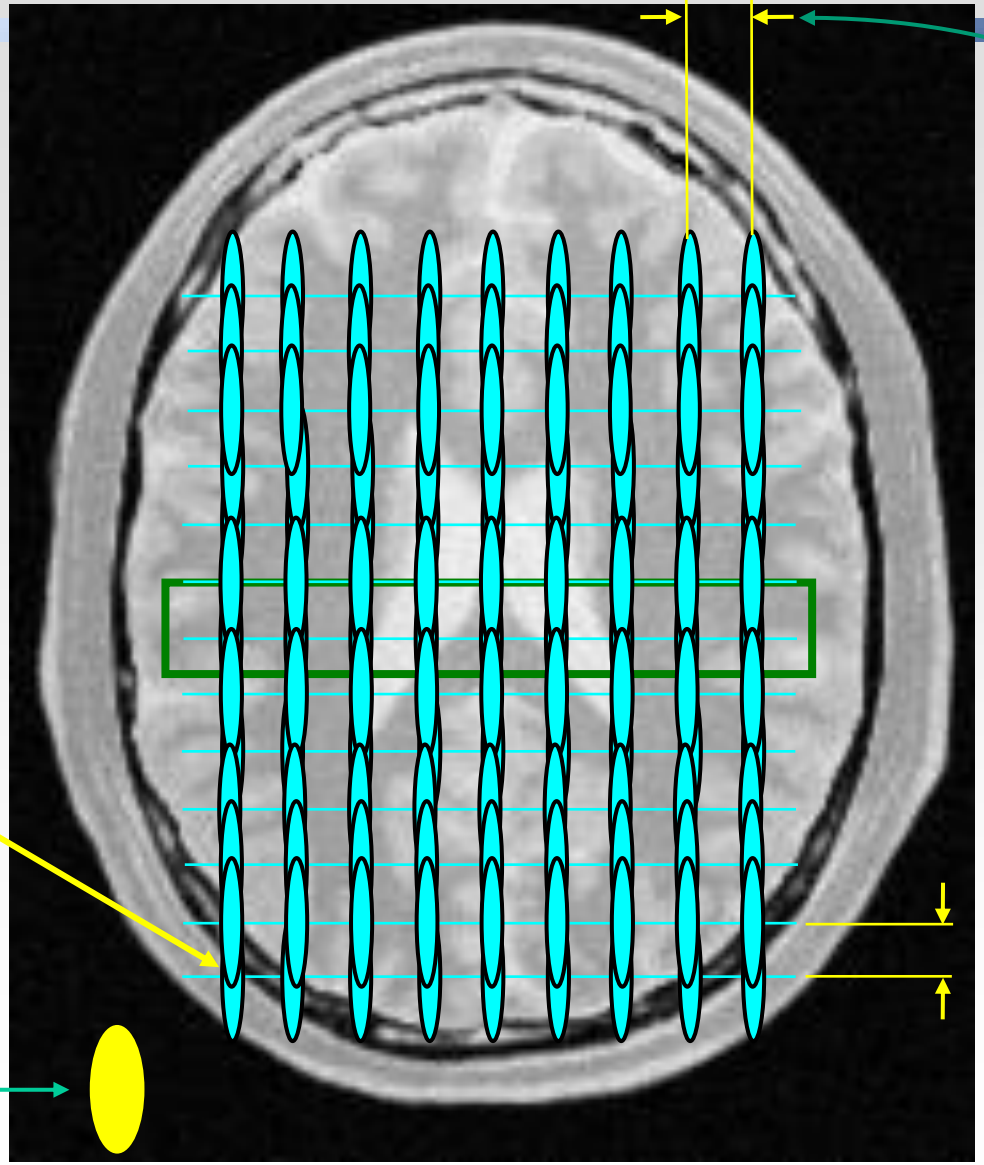
Subamostragem pela metade



Superamostragem pelo dobro



Superamostragem pelo dobro



novo espaçam.
S'x

nova origem
(O'x,O'y)

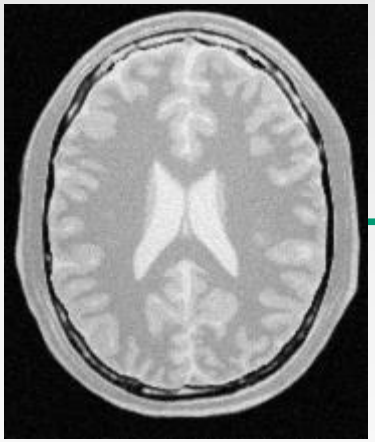
novo espaçam.
S'y

origem (Ox,Oy)

Reamostragem no ITK

itk::ResampleImageFilter

Reamostragem no ITK



Input Image

Origem

Espaçamento

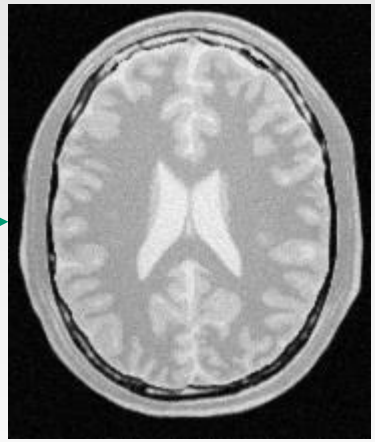
Tamanho da Região

Filtro de reamostragem

Transformada

Início da Região

Interpolador



Output Image

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

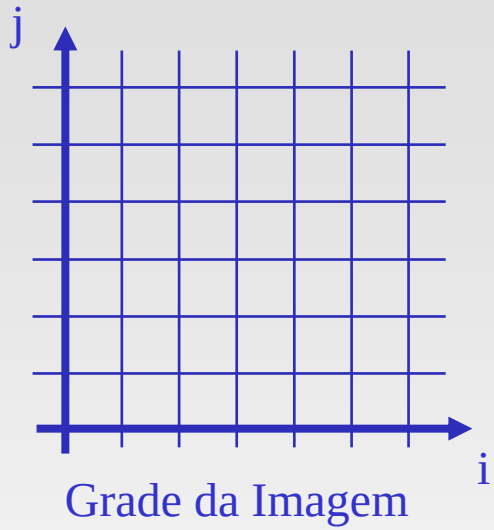
```
ImageType::PointType origin;
origin[0] = 10.0;  // millimeters
origin[1] = 25.5;  // millimeters

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 );
```

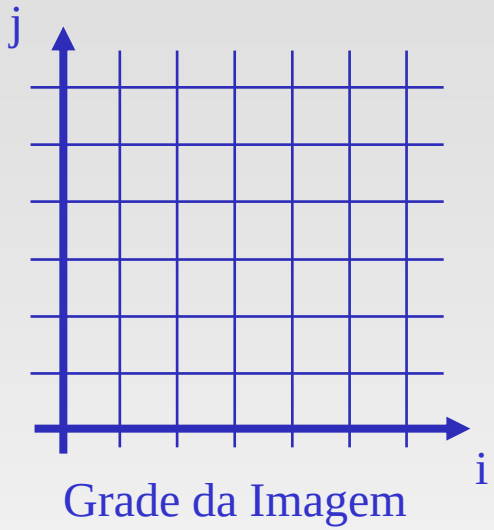
Filtro de Reamostragem

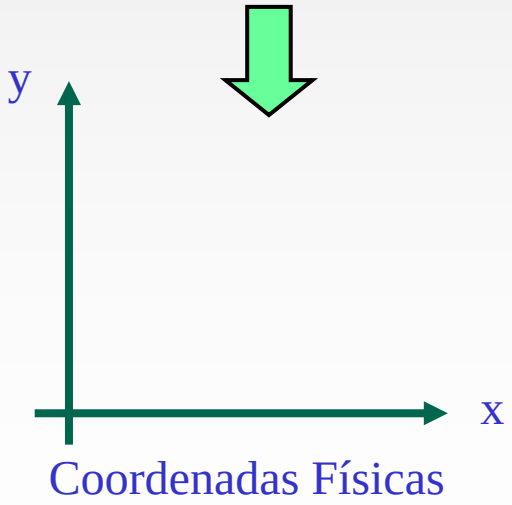
```
typedef itk::LinearInterpolateImageFunction<
                                ImageType,
                                double >     InterpolatorType;

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

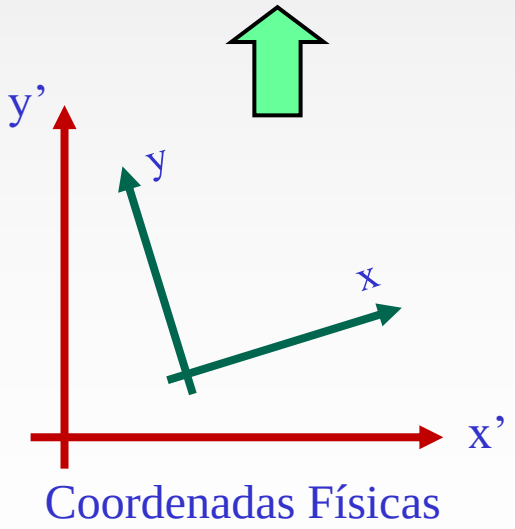
Conversão de Sistemas de Coordenadas







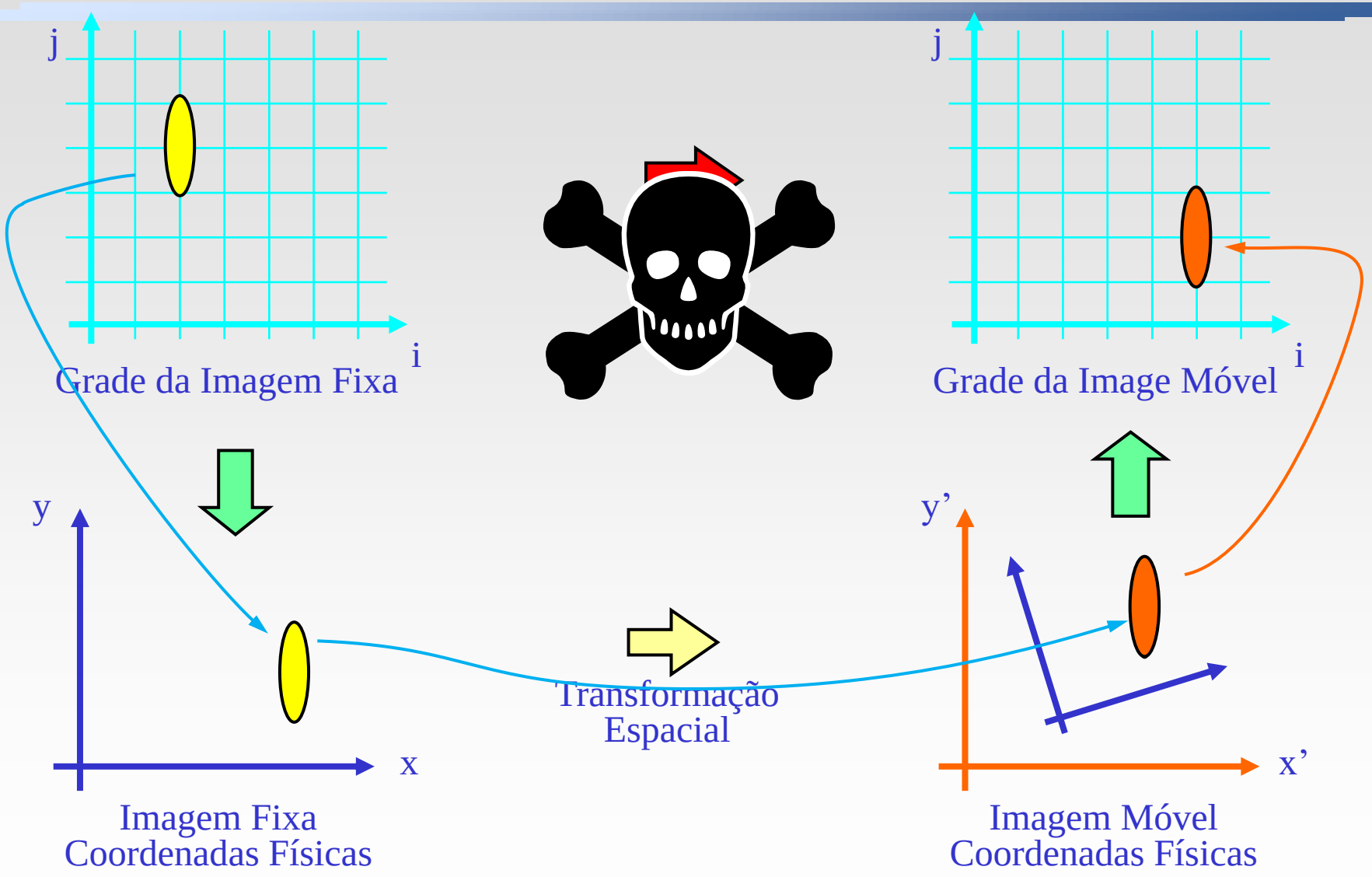
Transformação Espacial



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

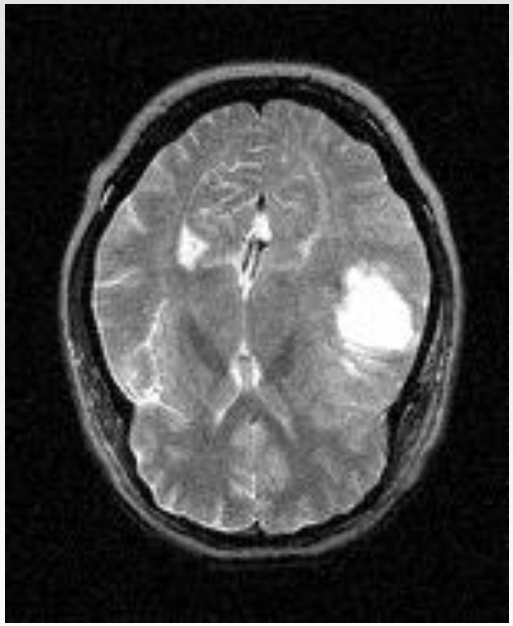
Em princípio, a denominação de
Imagem **Fixa** e Imagem **Móvel**
é arbtrária

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 ?

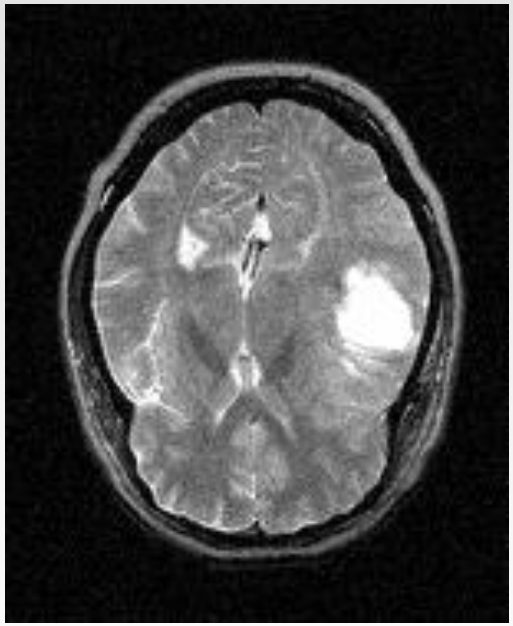
PET

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

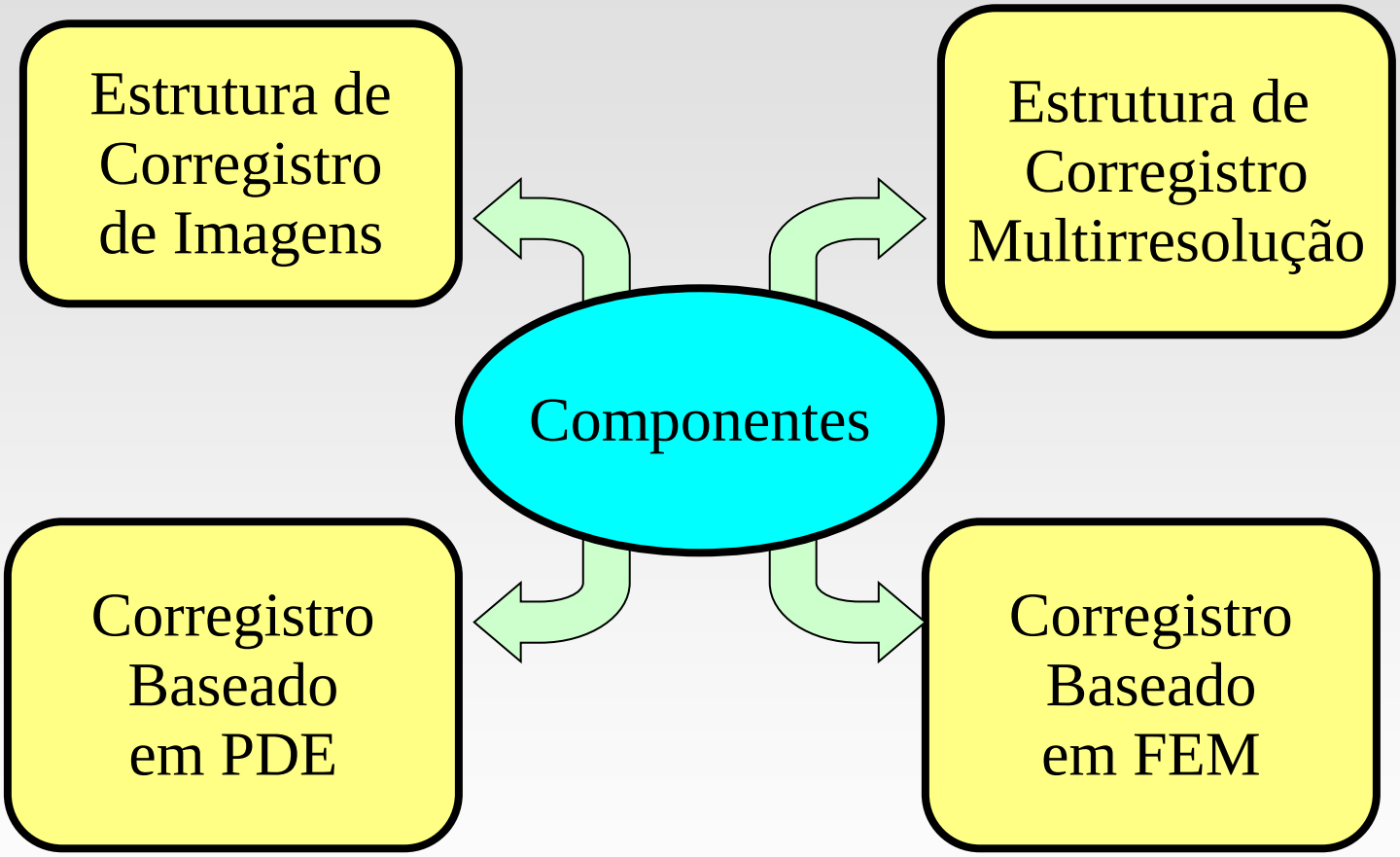
PET

128 x 128 pixels

Coisas que não farei...

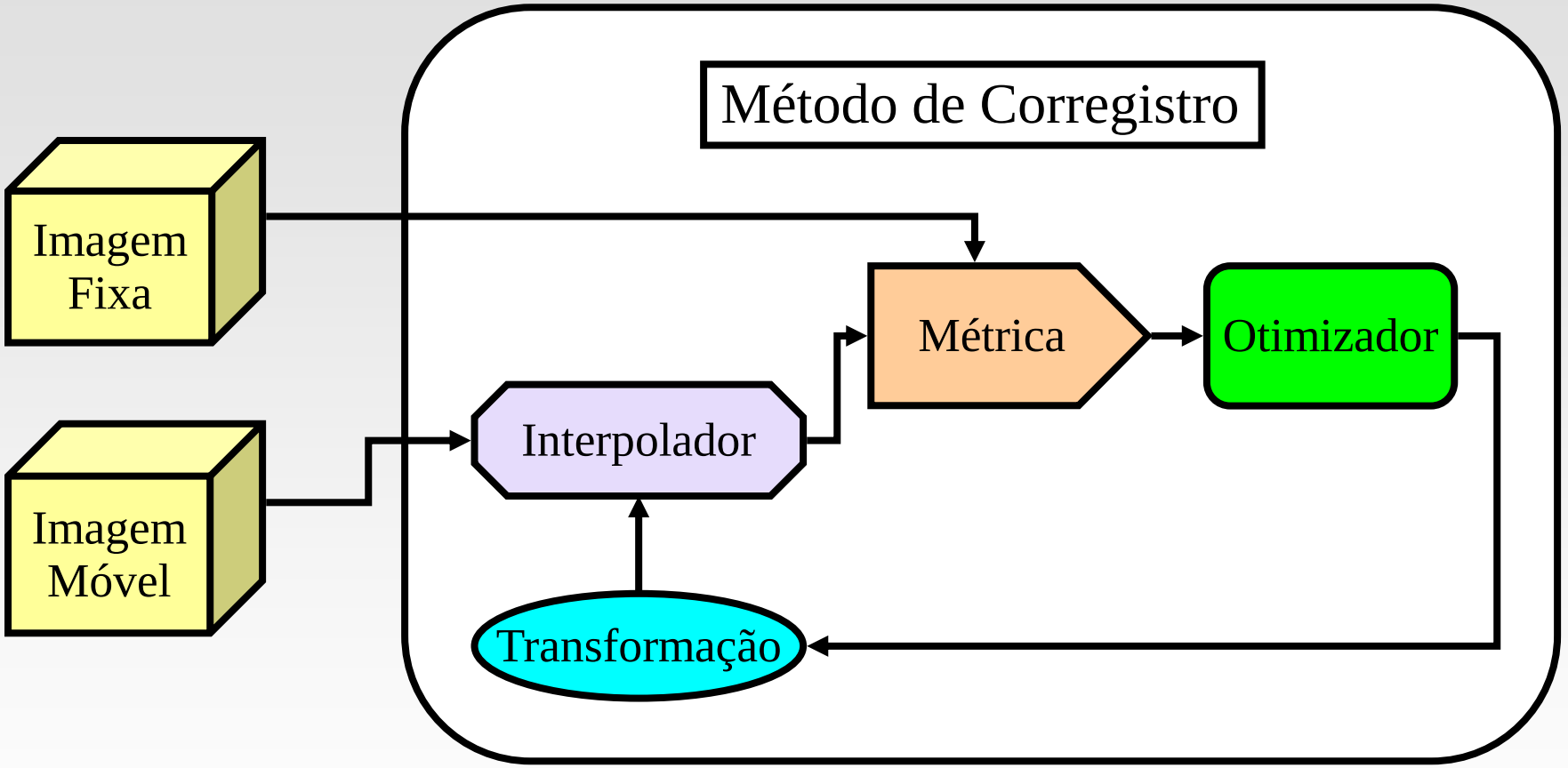
Não corregistrar imagens no espaço de pixels
Não corregistrar imagens no espaço de pixels
Não corregistrar imagens no espaço de pixels
Não corregistrar imagens no espaço de pixels
Não corregistrar imagens no espaço de pixels
Não corregistrar imagens no esp

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

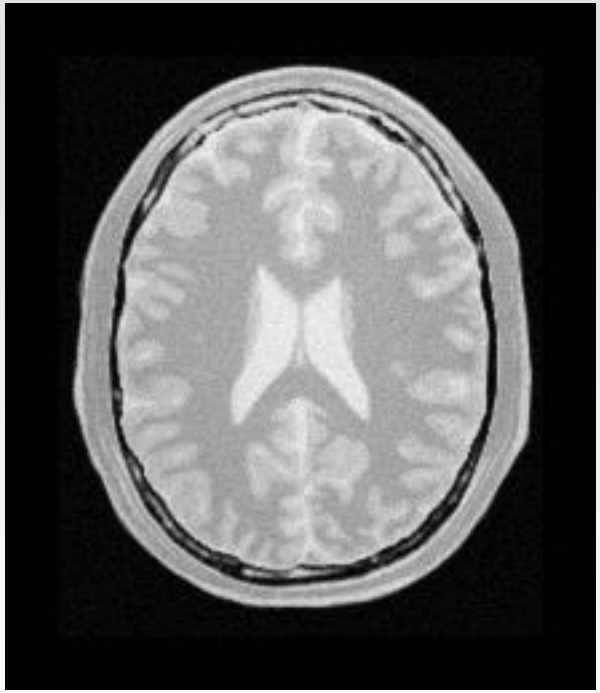


Imagem Fixa

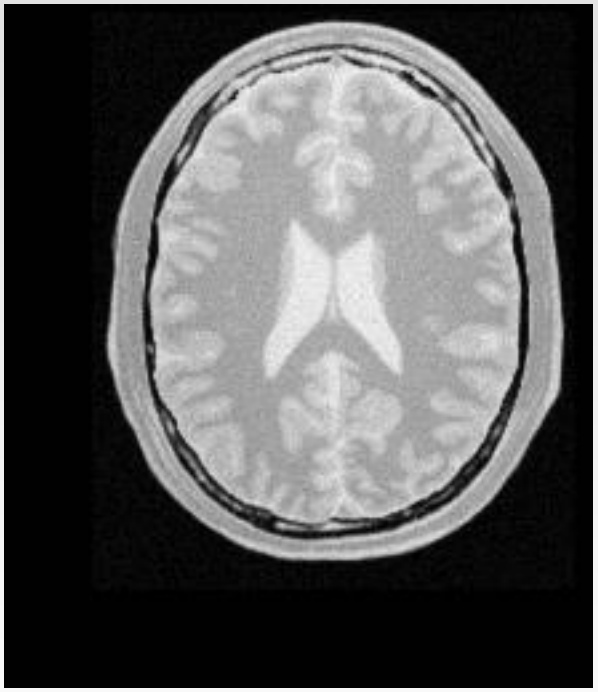


Imagem Móvel

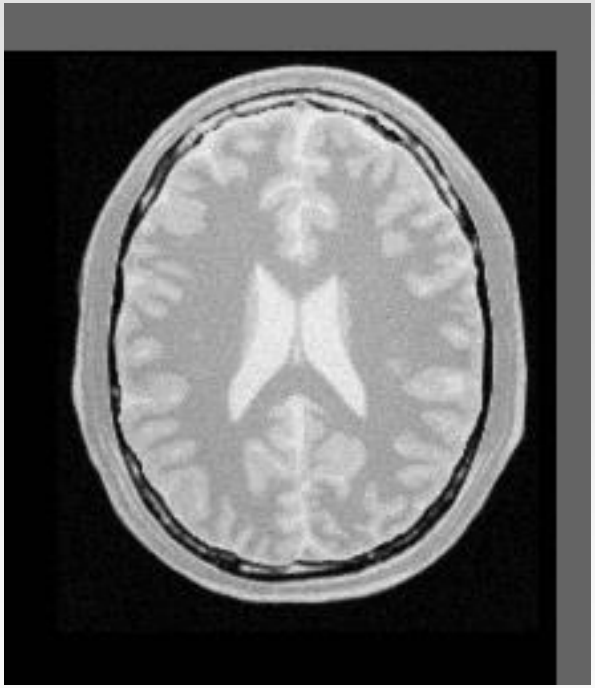


Imagem Móvel Corregistrada

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)

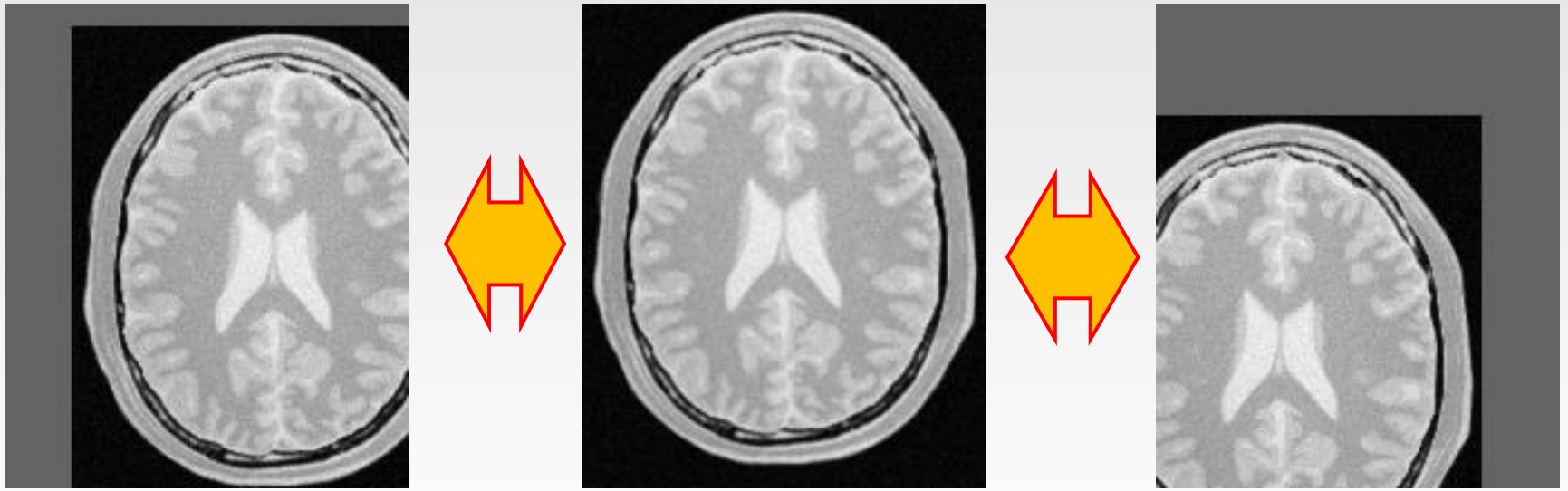


Imagem B Imagem A Imagem C

Métricas de Similaridade

Similaridade(A , B)

Métrica mais Simples

Diferença Quadrática Média

Diferenças Quadráticas Média

Para cada pixel em A

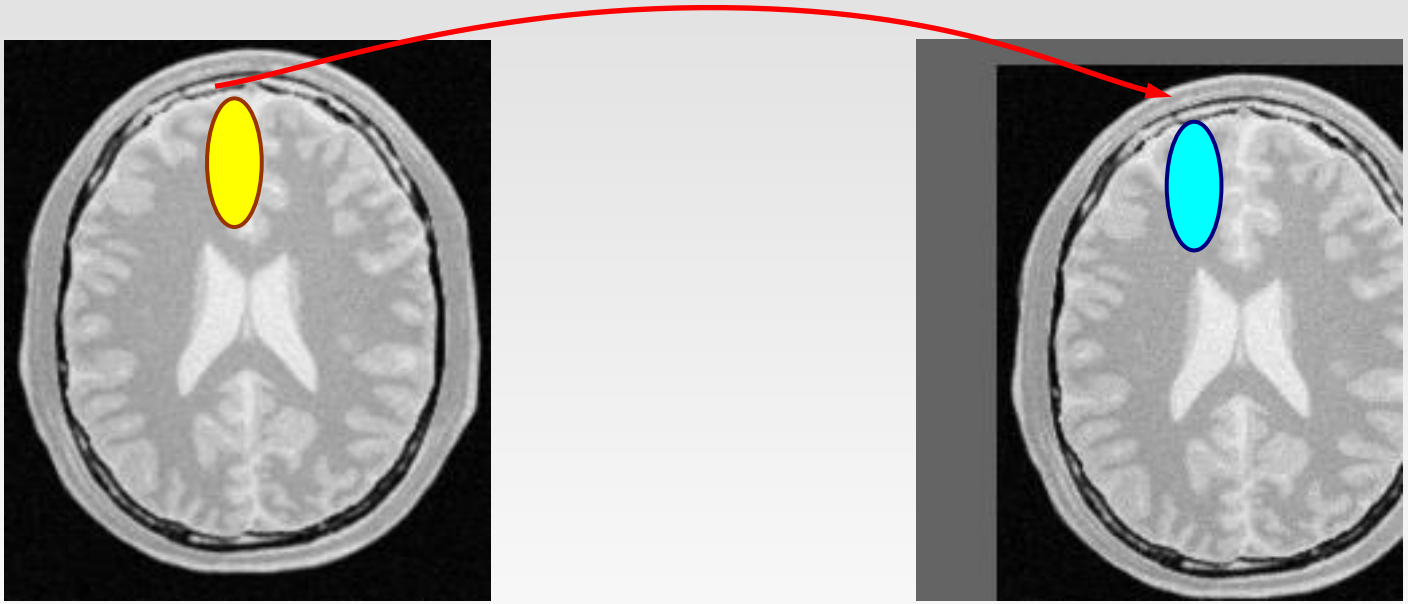


Imagem A

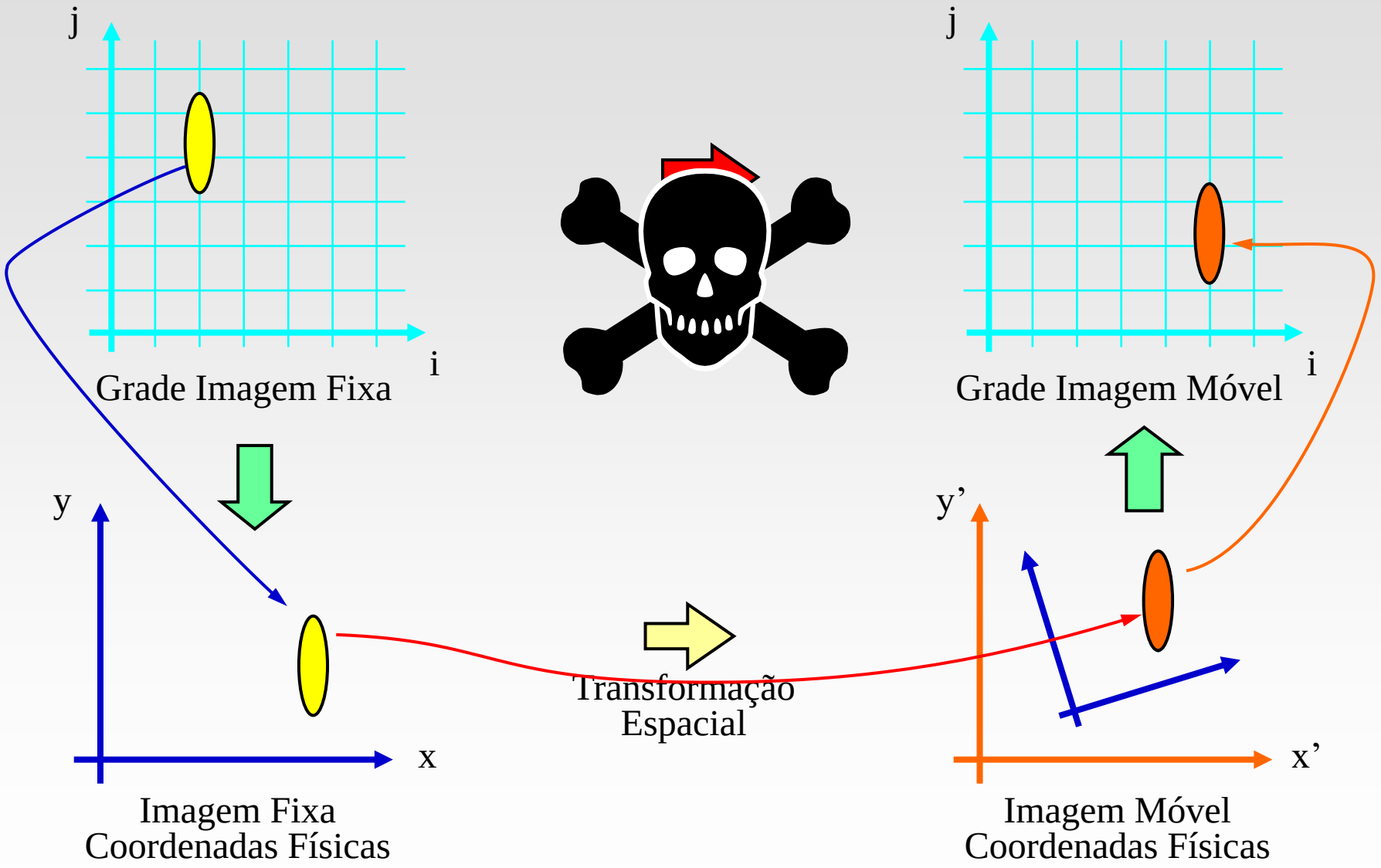
Imagem B

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

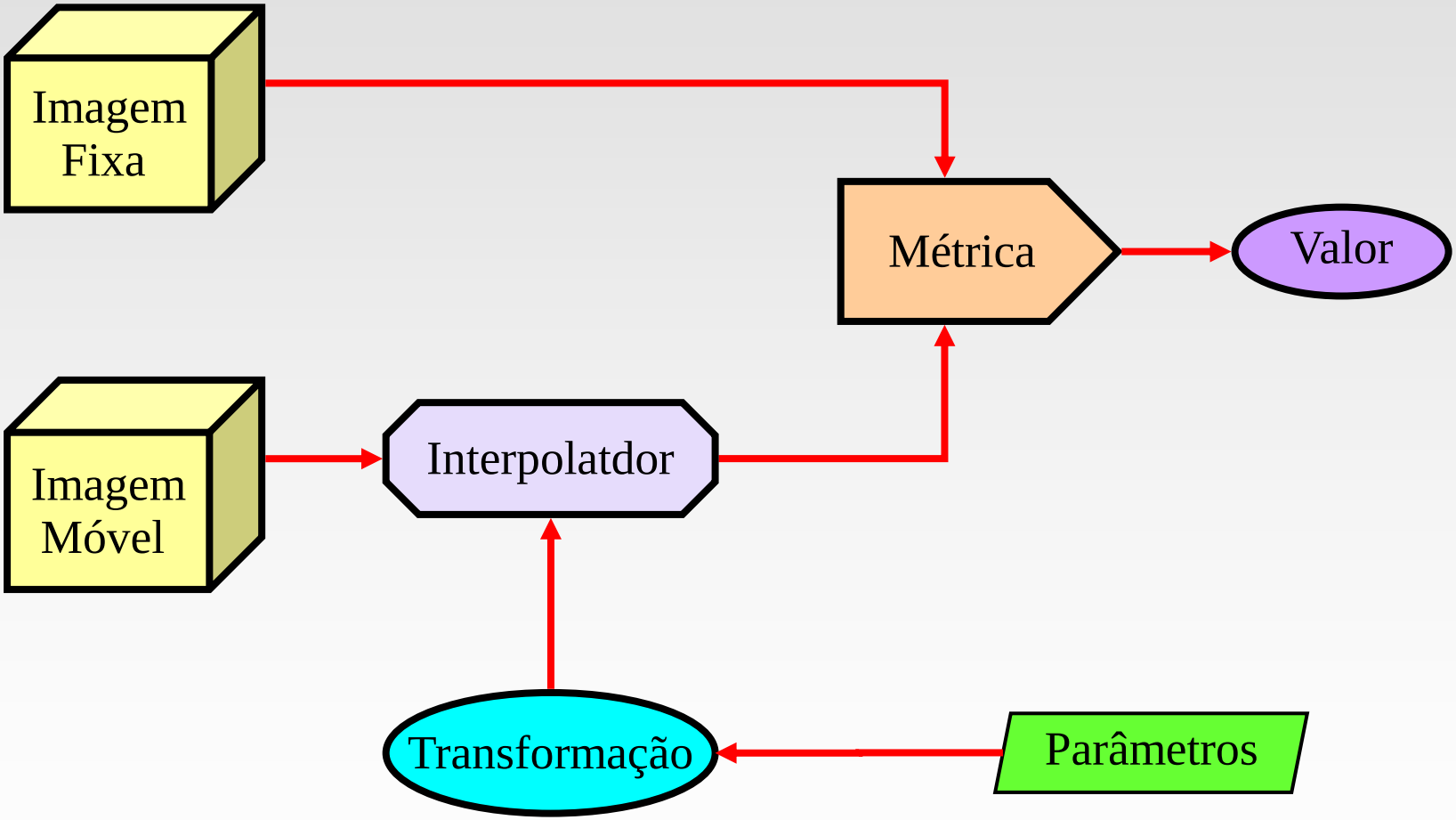
Soma += Diferença(índice) ^{2}

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();
```

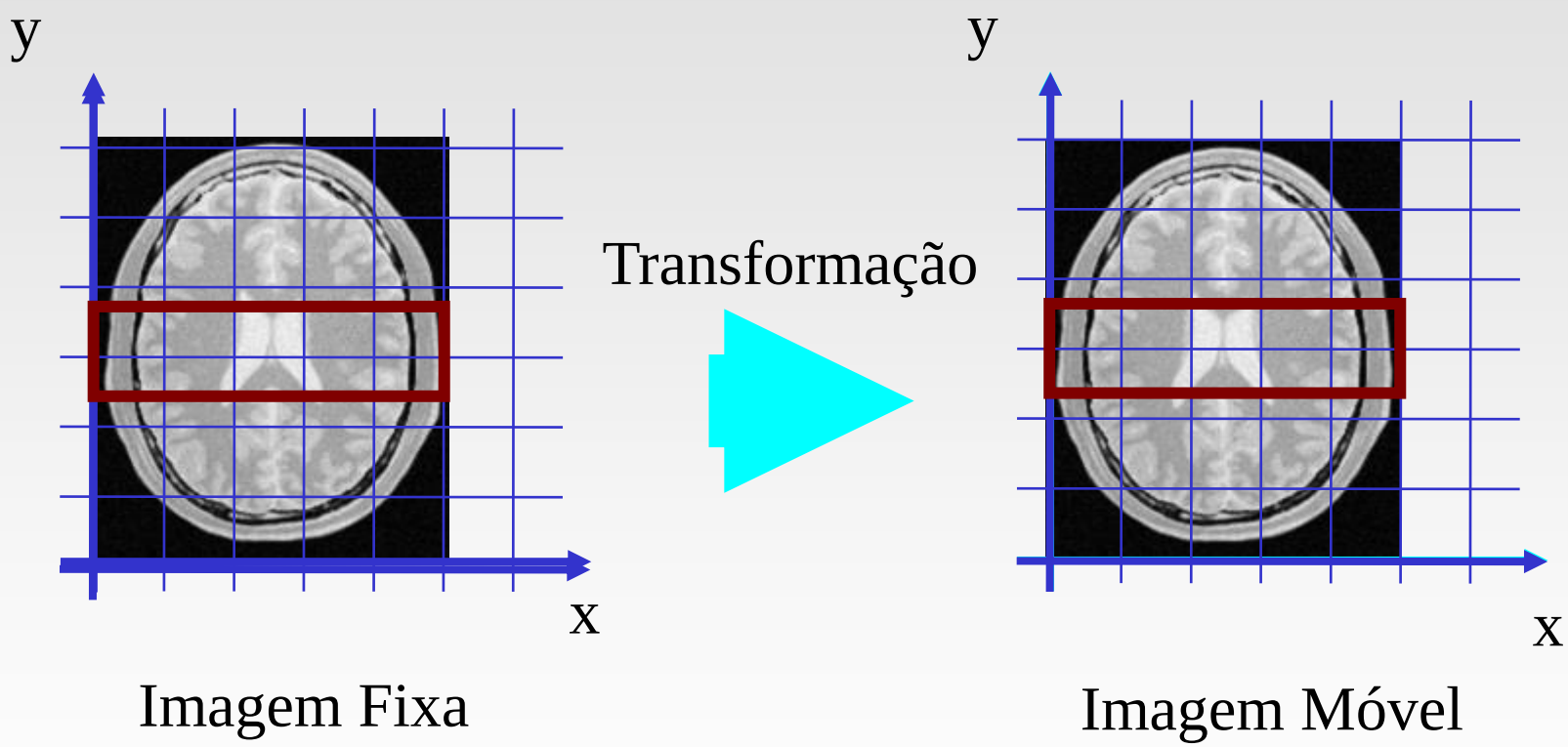
Diferenças Quadráticas Média

```
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

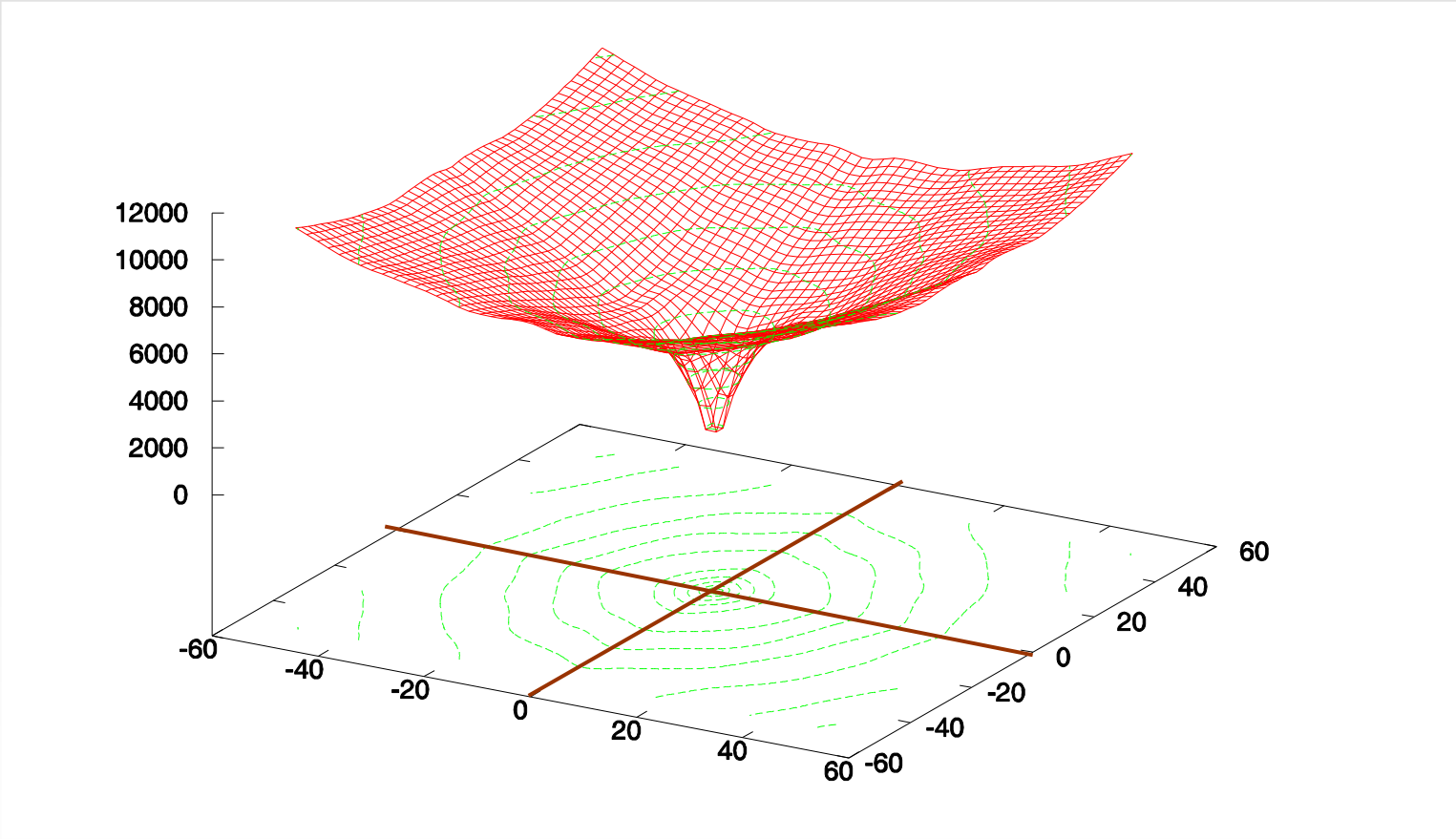
```
MetricType::TransformParametersType translation( Dimension );

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

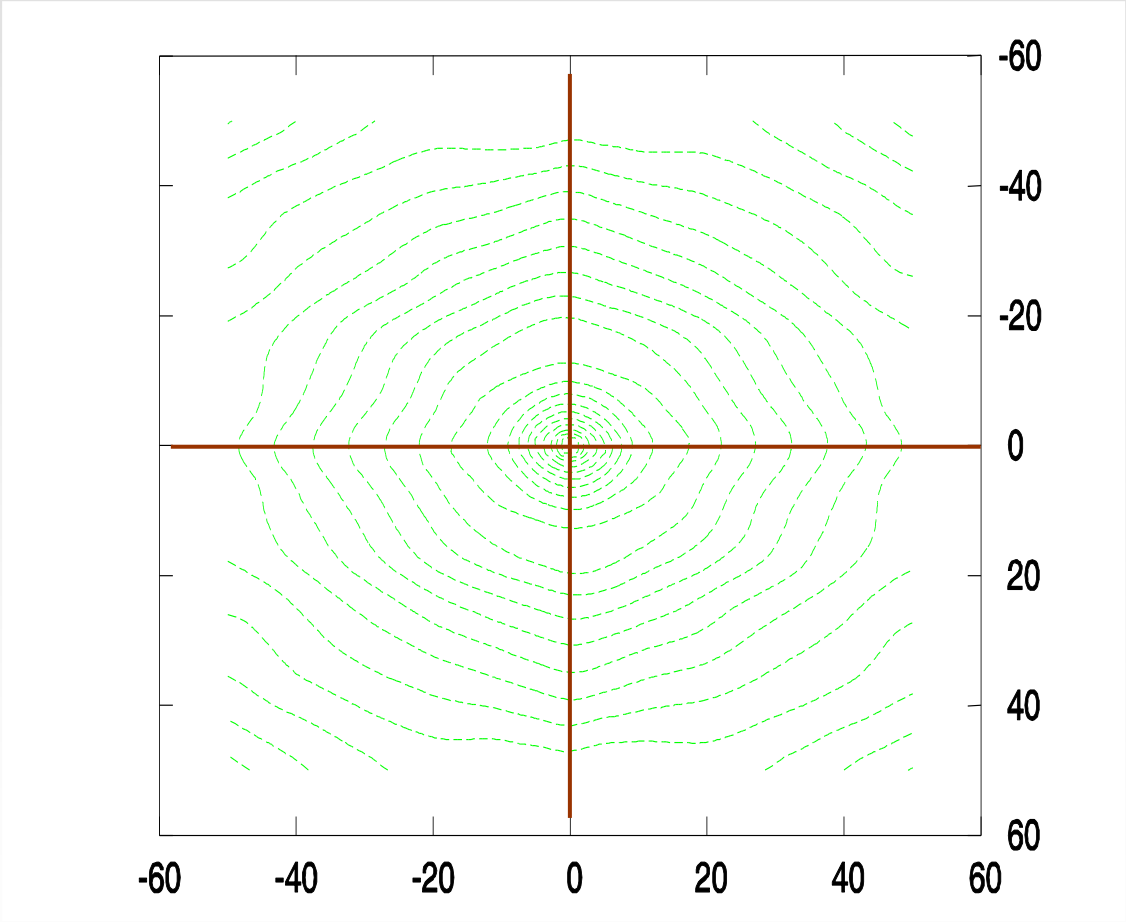
Diferenças Quadráticas Média



Espaço de parâmetros da transformação

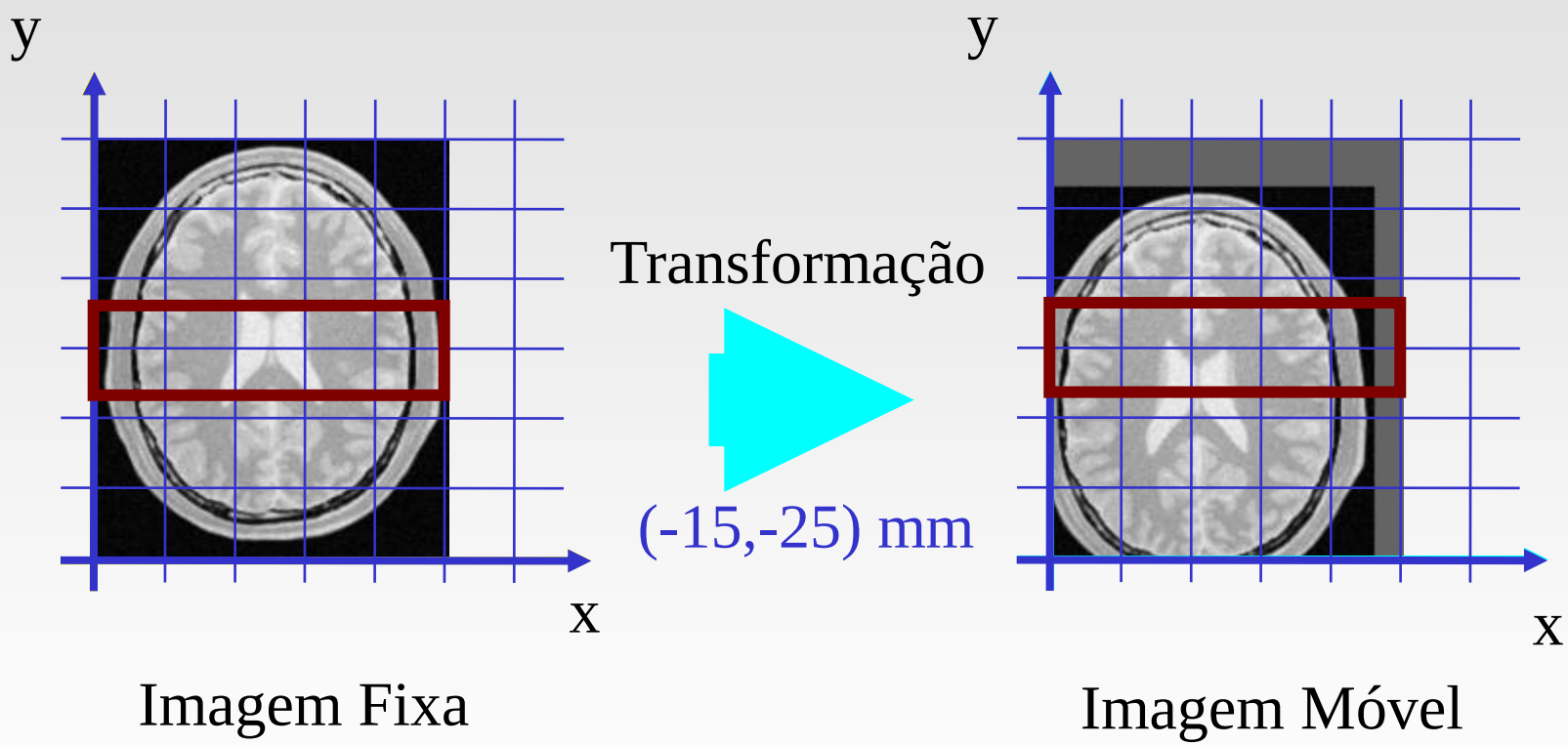
Plotando a Métrica

Diferenças Quadráticas Média



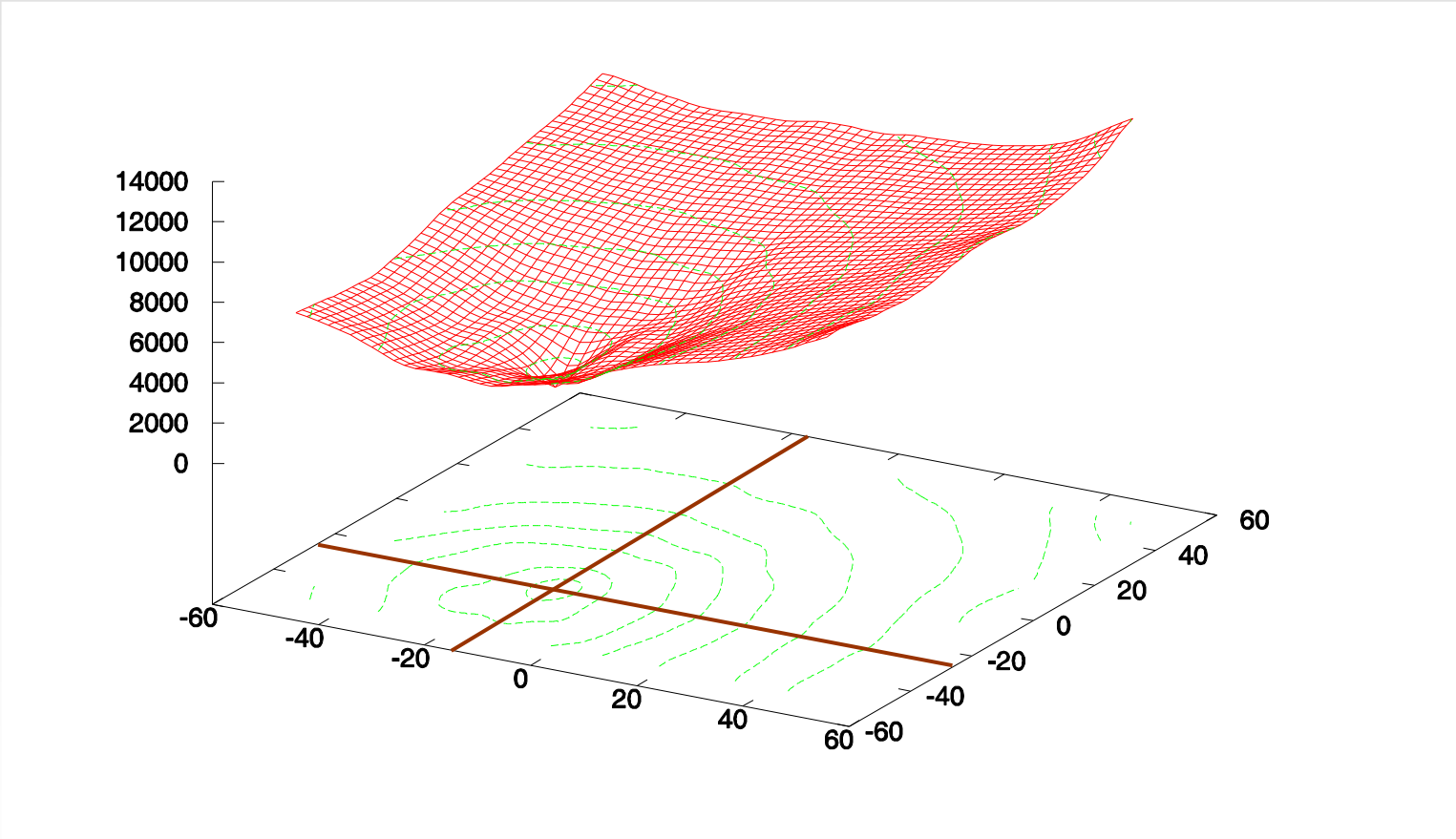
Espaço de parâmetros da transformação

Avaliando muitas comparações



Plotando a Métrica

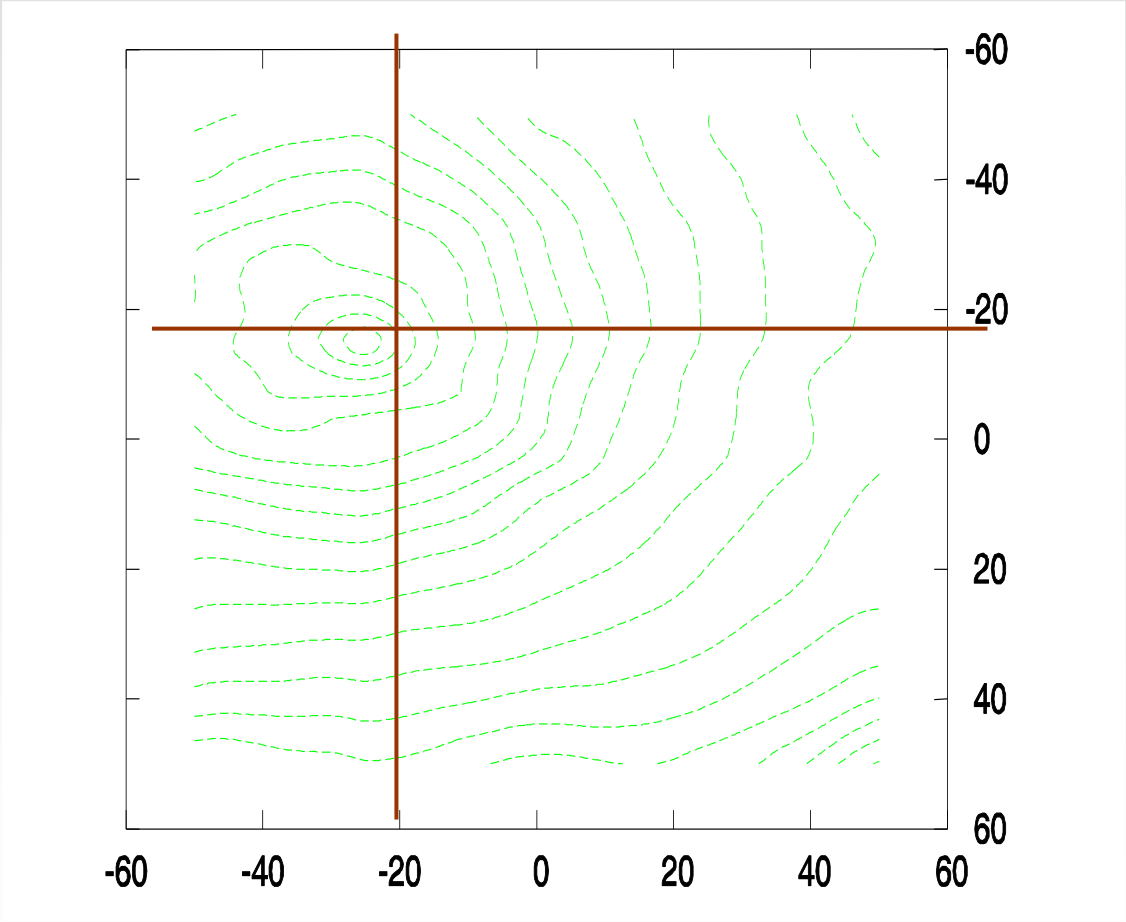
Diferenças Quadráticas Média



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

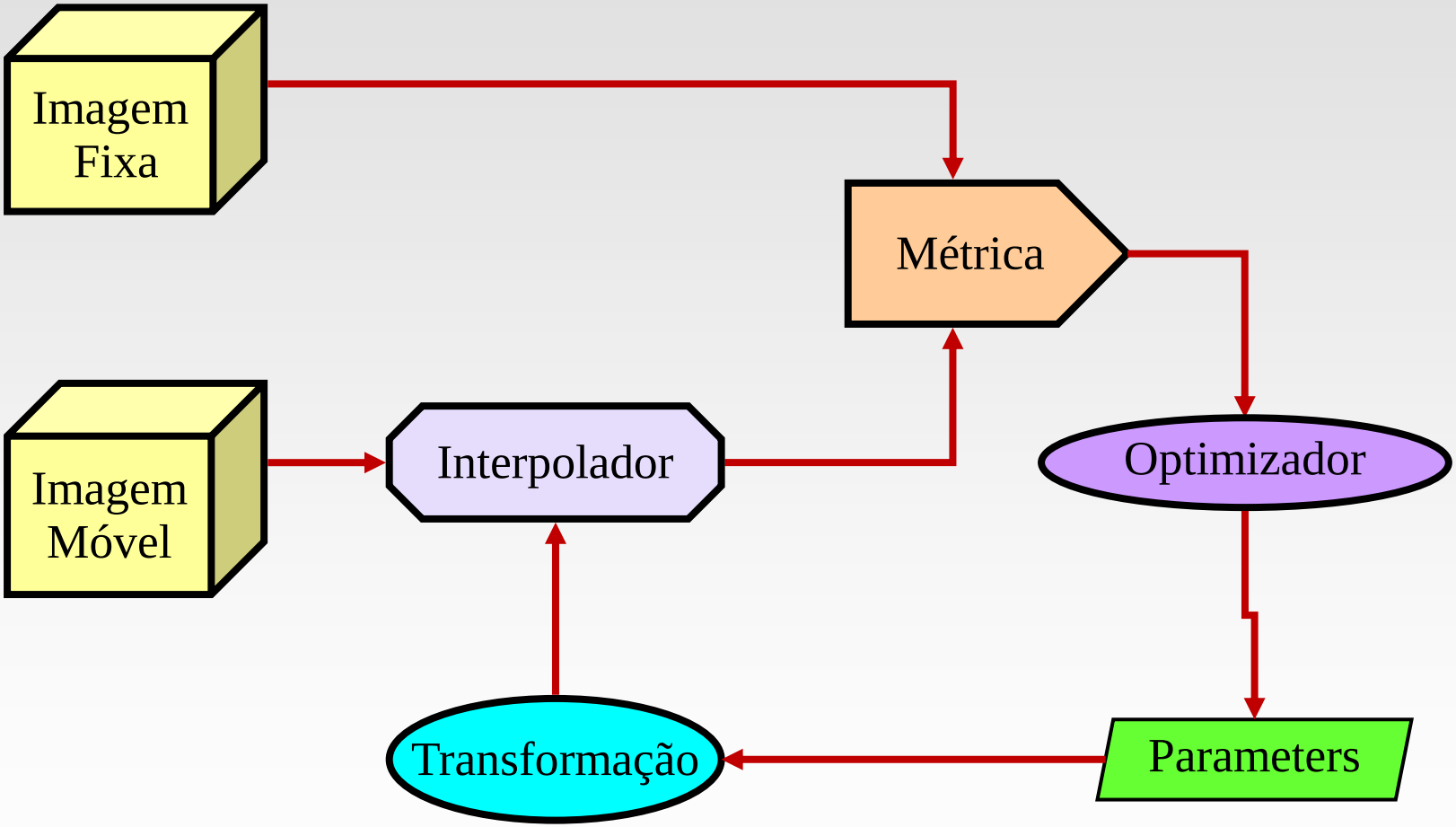
O melhor conjunto de parâmetros de transformação

Melhores Métodos de Optimização

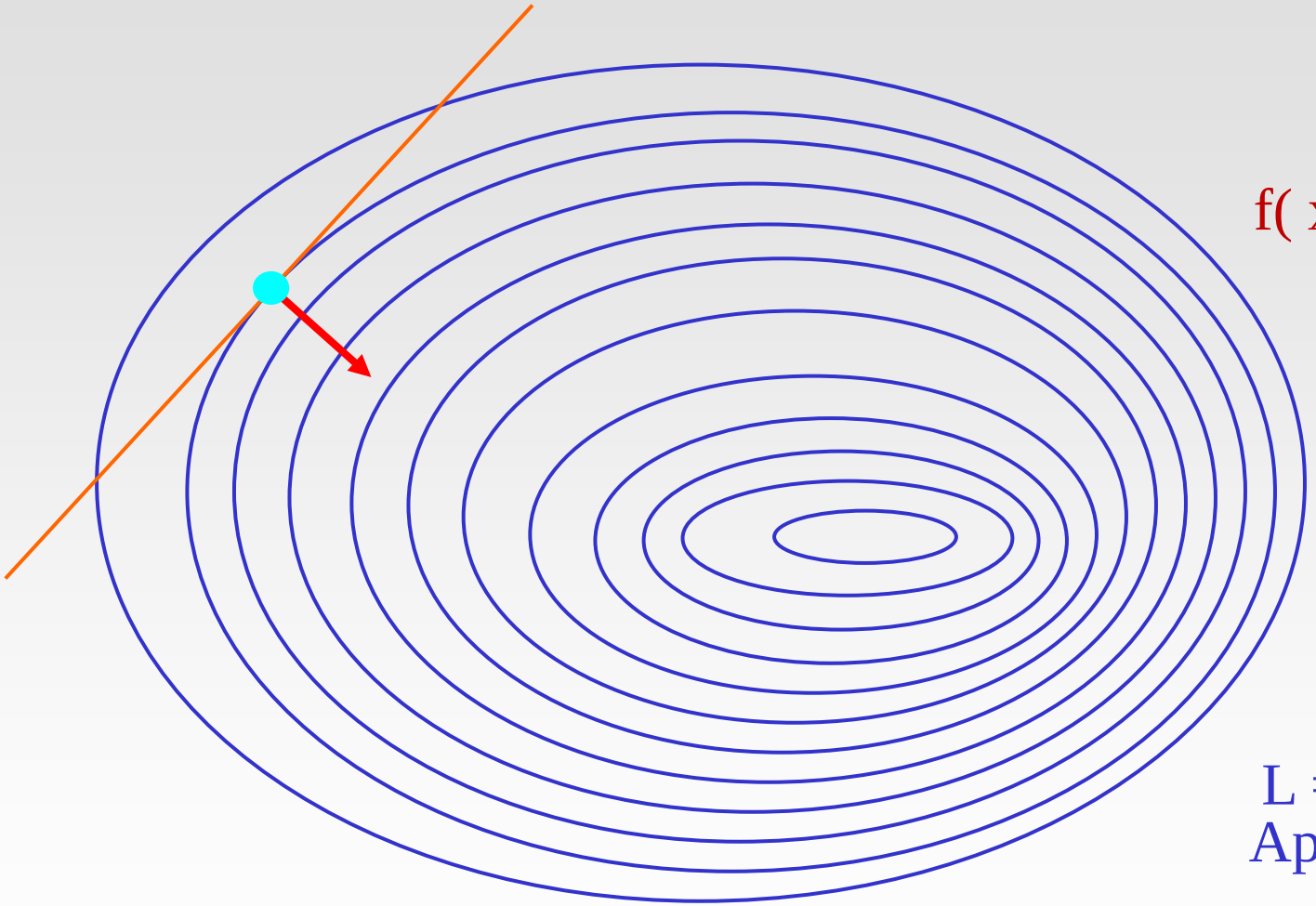
por exemplo

Gradiente Descente

Estrutura de Corregistro de imagens



Otimizador Gradiente Descente



f(x , y)

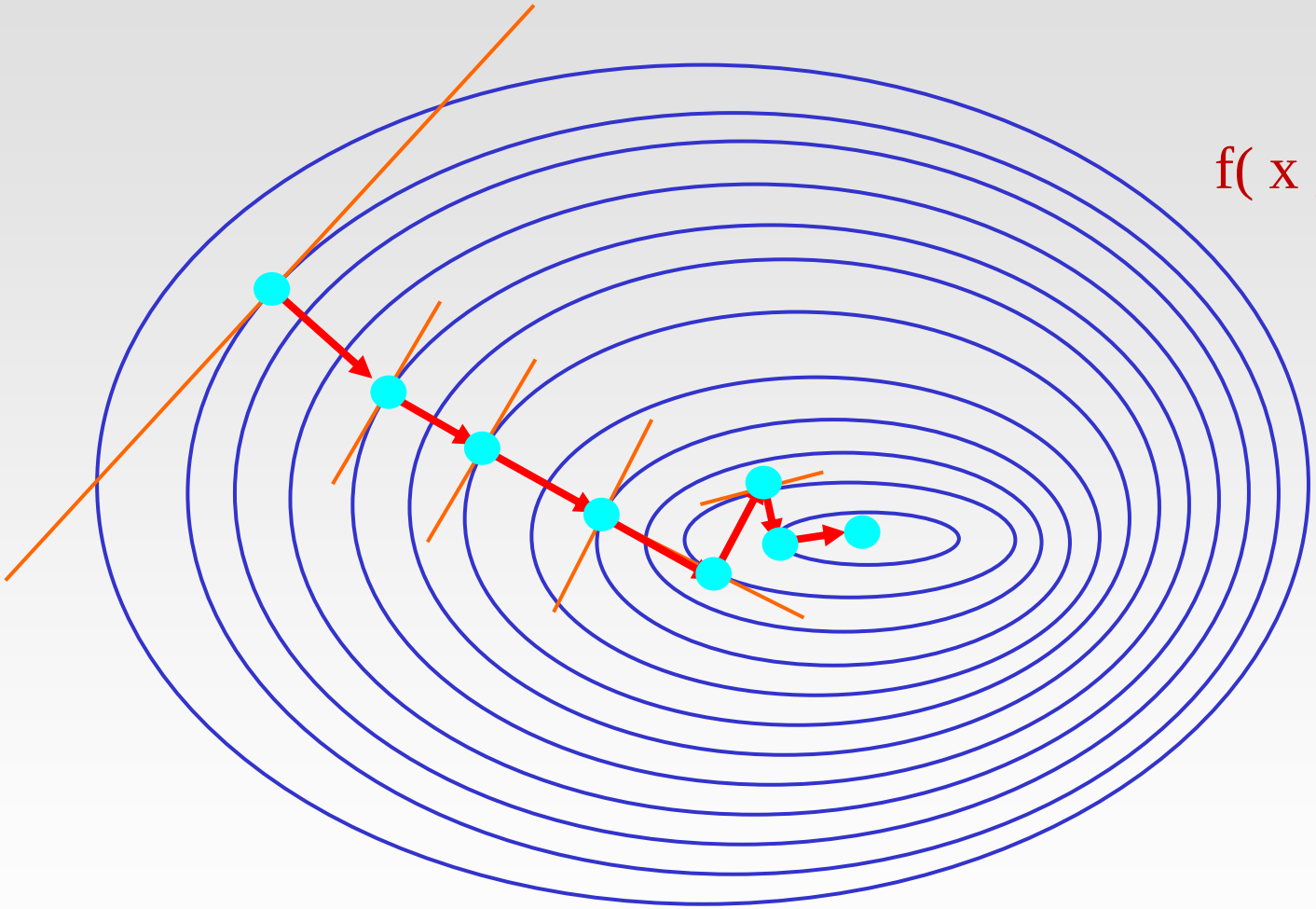
S = Passo

L = Taxa de Aprendizado

G(x , y) ∇f(x , y)

S = L · G(x , y)

Otimizador Gradiente Descente



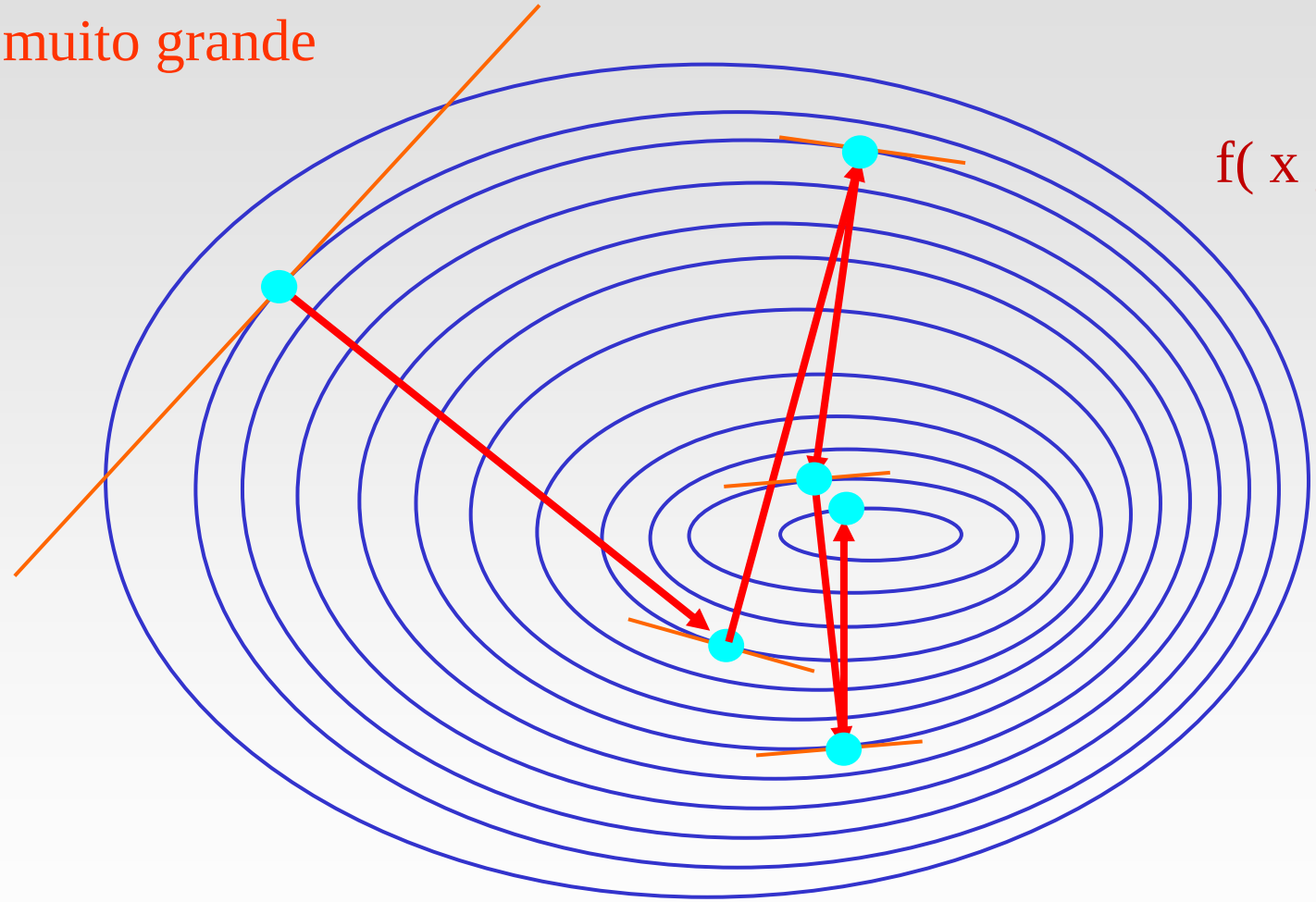
f(x , y)

G(x , y) ∇f(x , y)

S = L · G(x , y)

Otimizador Gradiente Descente

L muito grande



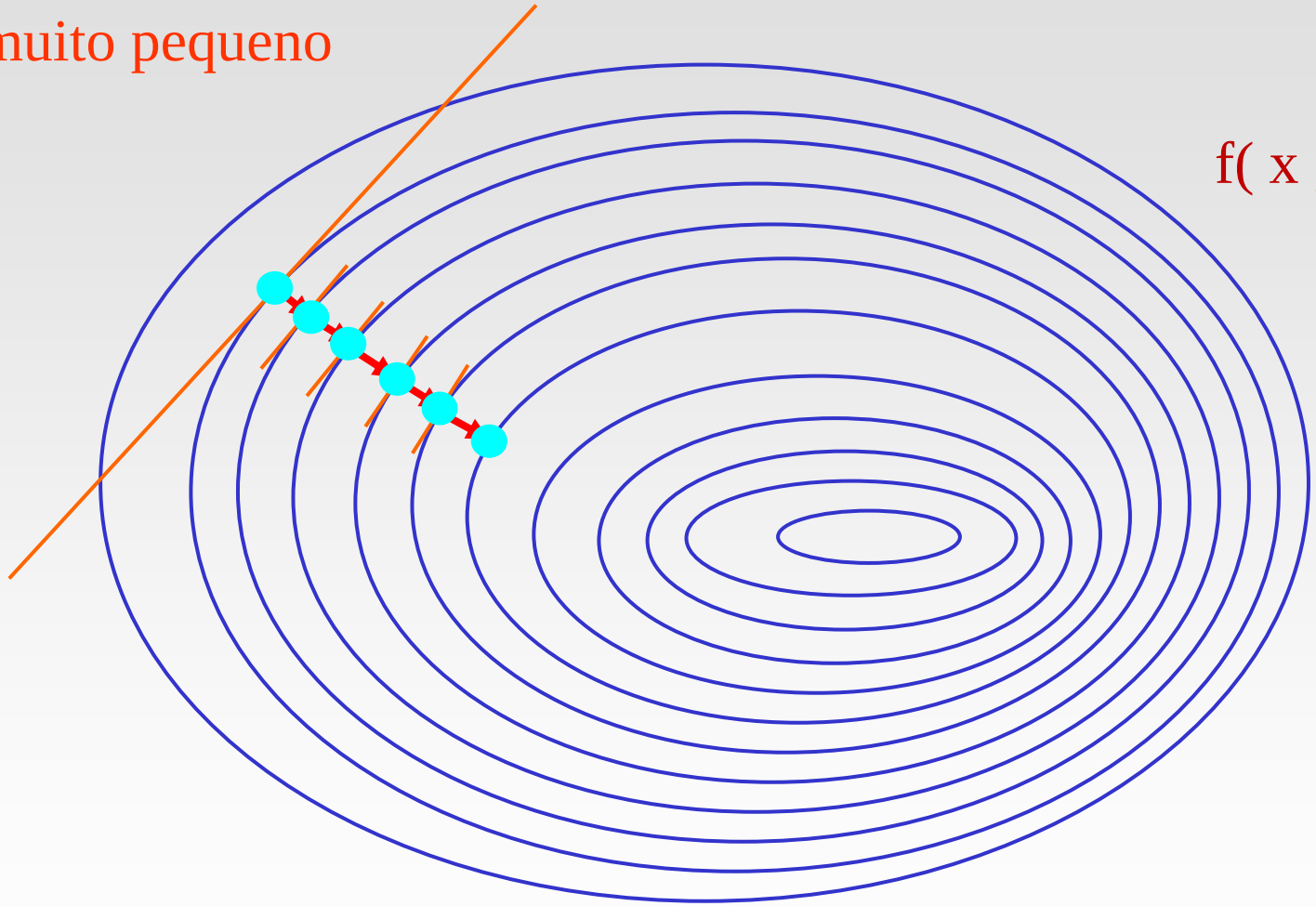
f(x , y)

G(x , y) ∇f(x , y)

S = L · G(x , y)

Otimizador Gradiente Descente

L muito pequeno



f(x , y)

G(x , y) ∇f(x , y)

S = L · G(x , y)

Otimizador Gradiente Descendente

O quê há de errado com este algoritmo ?

Otimizador Gradiente Descendente

S Unidade ? = milimetros

f(x,y) Unidade ? = intensidade

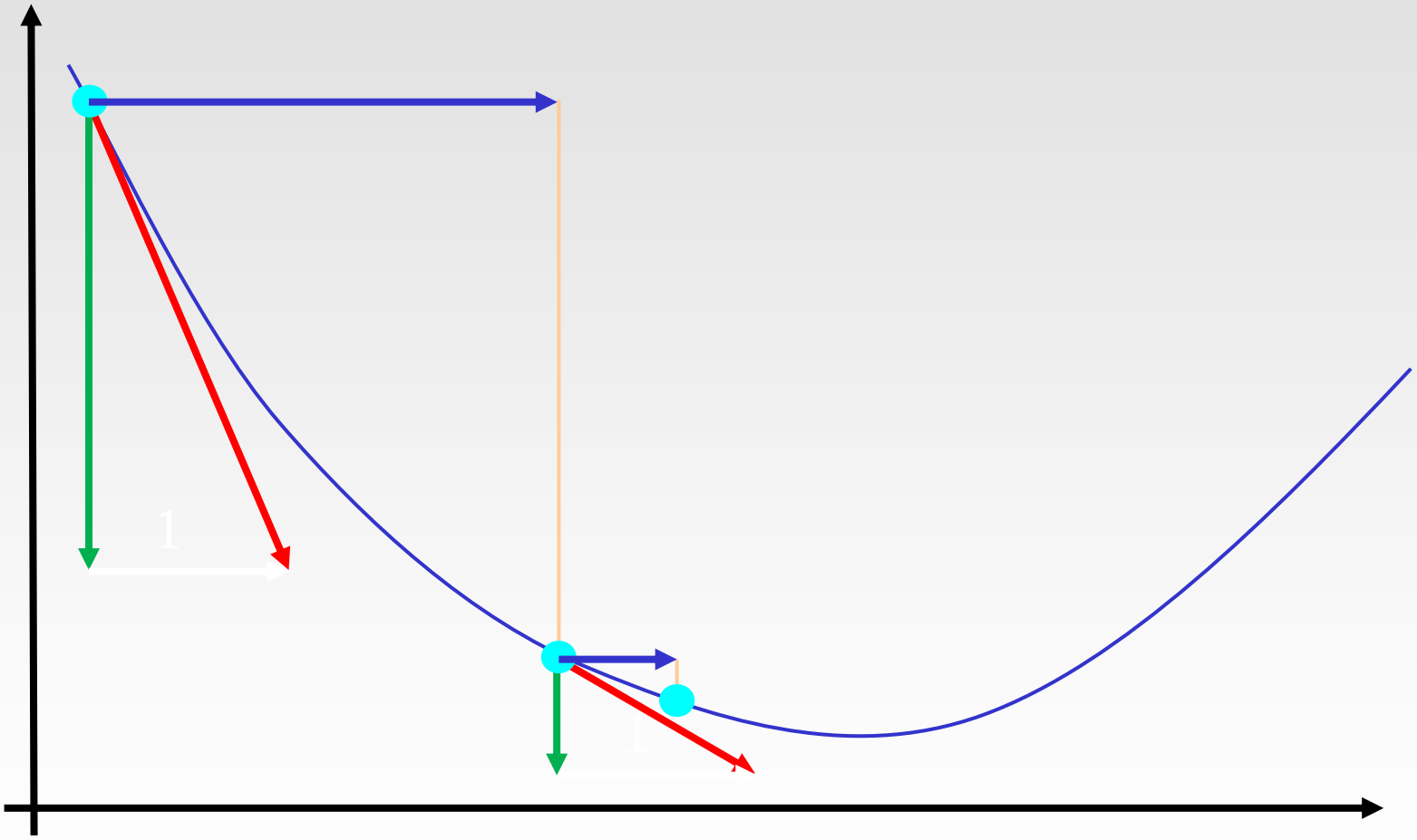
G(x,y) Unidade ? = intensidade / millimetros

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

L Unidade ? = millimetros^2 / intensidade

Otimizador Gradiente Descendente

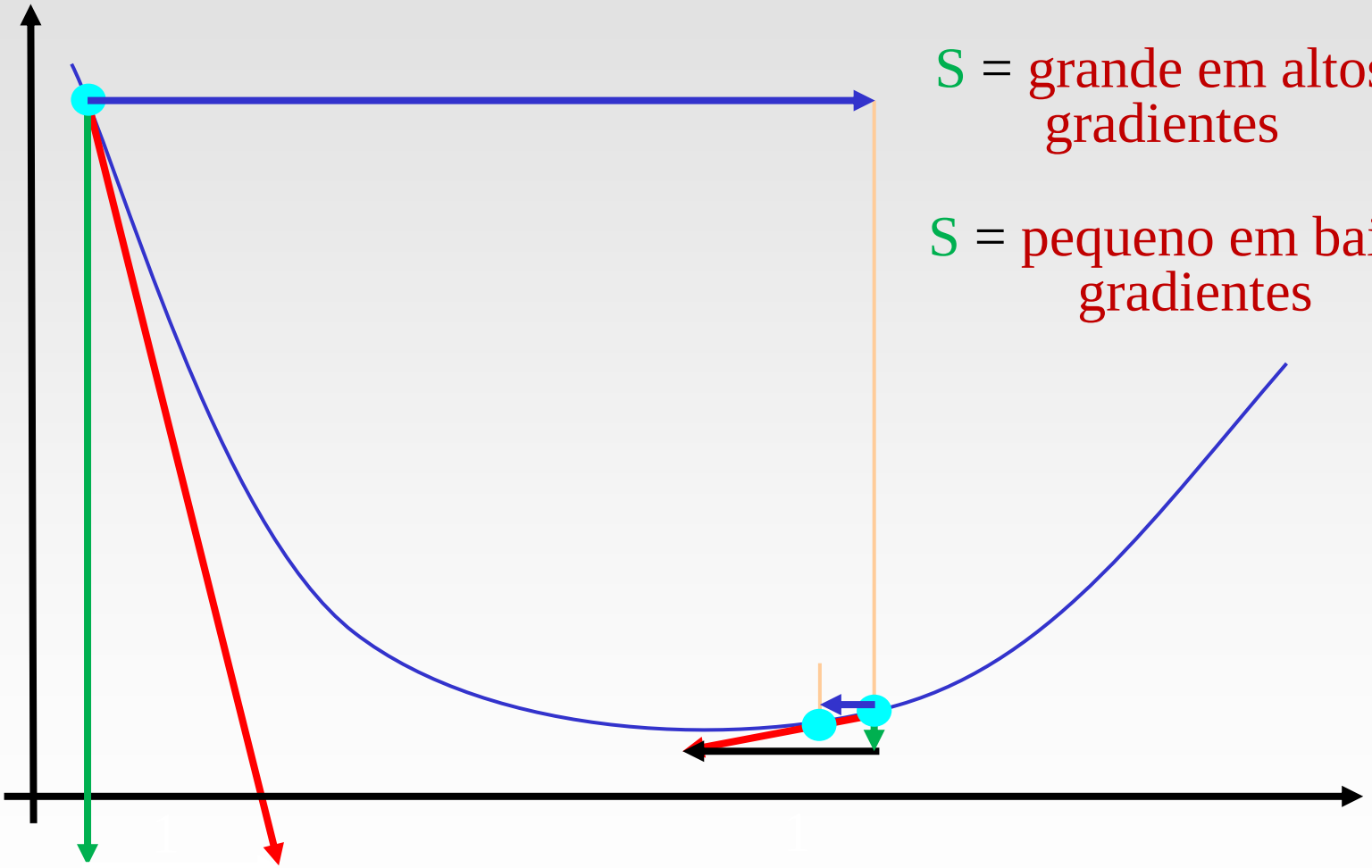
$S = L \cdot G(x)$ $f(x)$



Otimizador Gradiente Descendente

$S = L \cdot G(x)$

$f(x)$



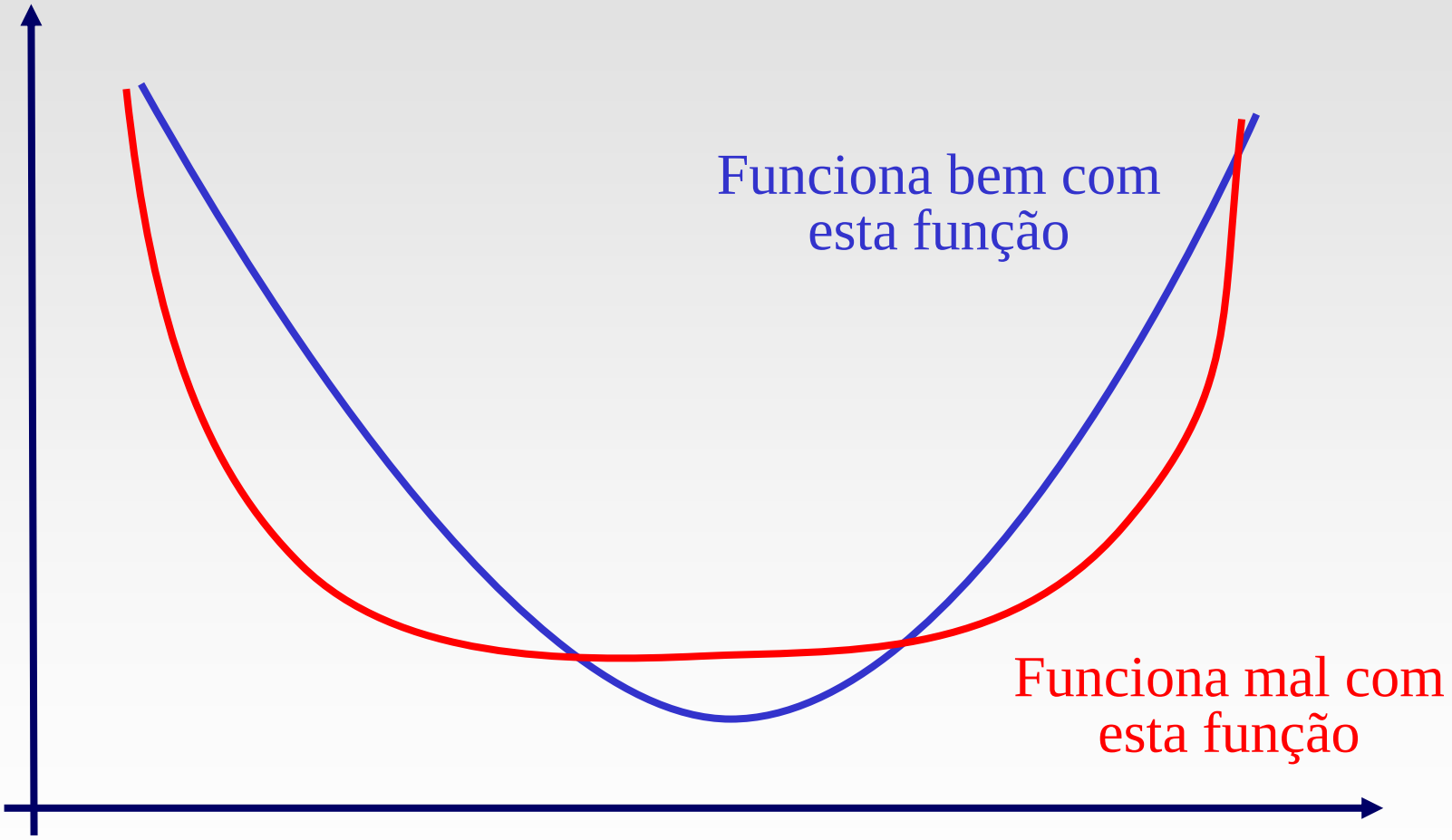
S = grande em altos gradientes

S = pequeno em baixos gradientes

Otimizador Gradiente Descendente

$S = L \cdot G(x)$

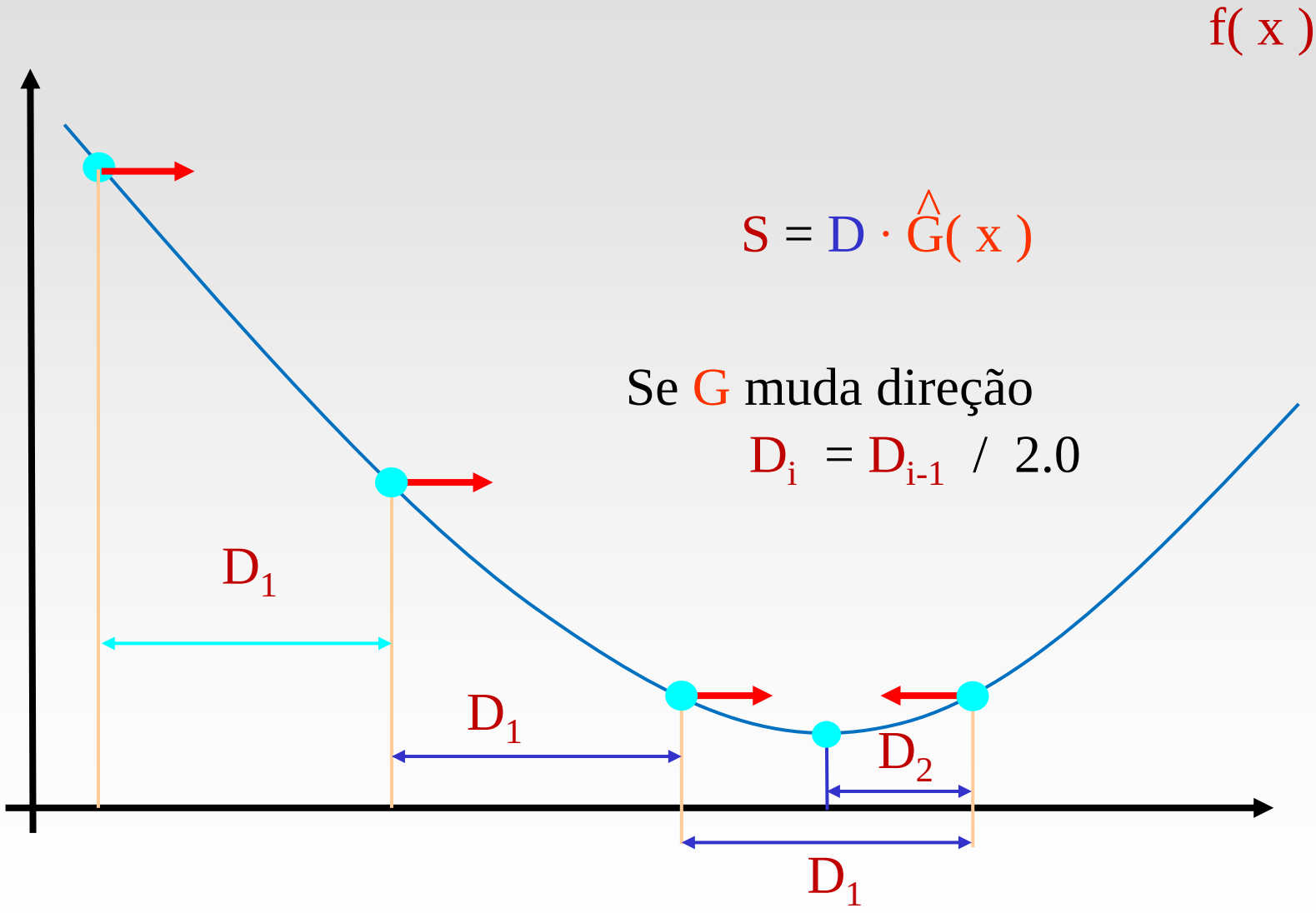
$f(x)$



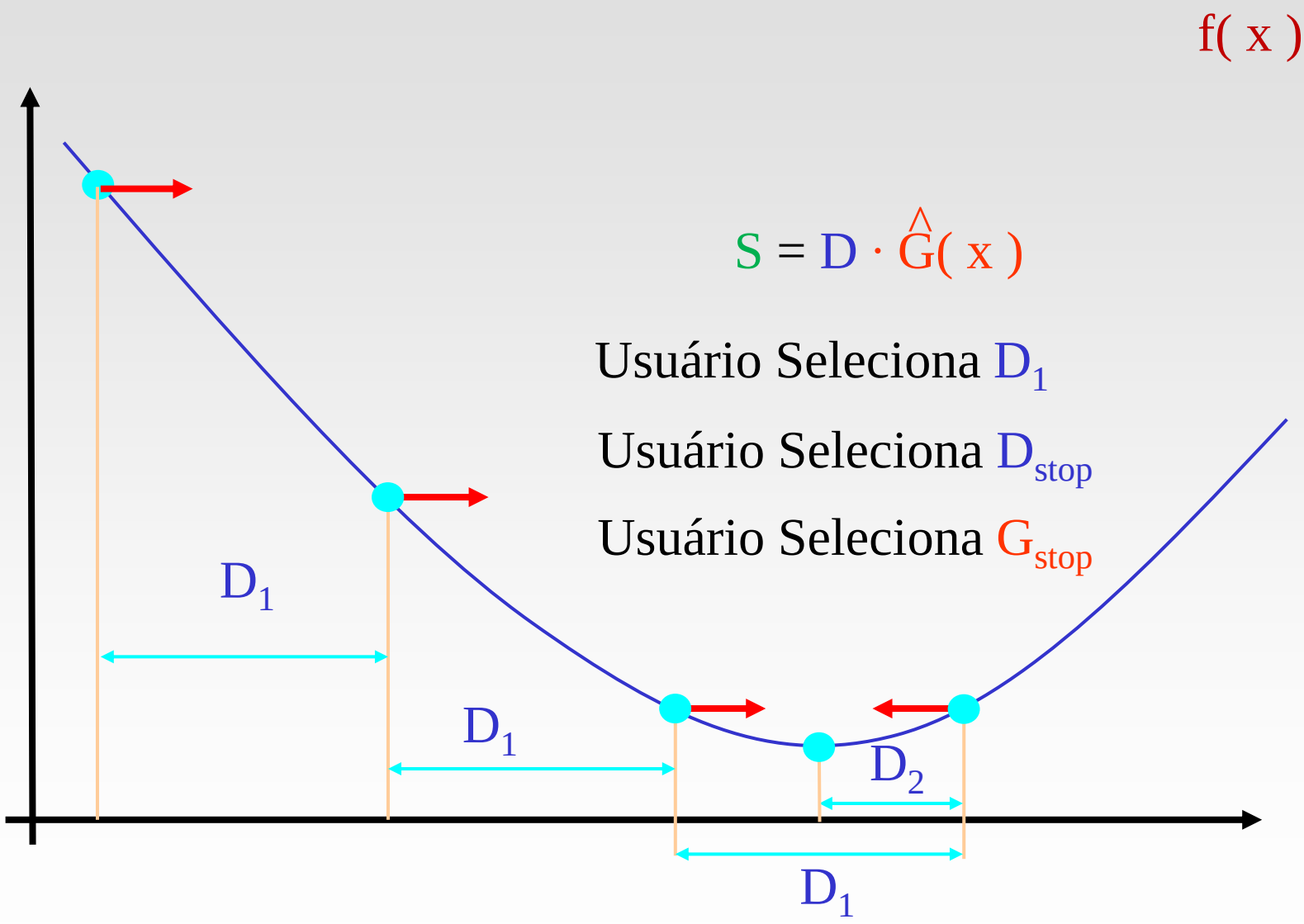
Variante do Gradiente Descendente

Direção Segura !

Gradiente Descendente de Passo Regular



Gradiente Descendente de Passo Regular



Optimizadores são como carro

Cuidado quando estiver dirigindo !

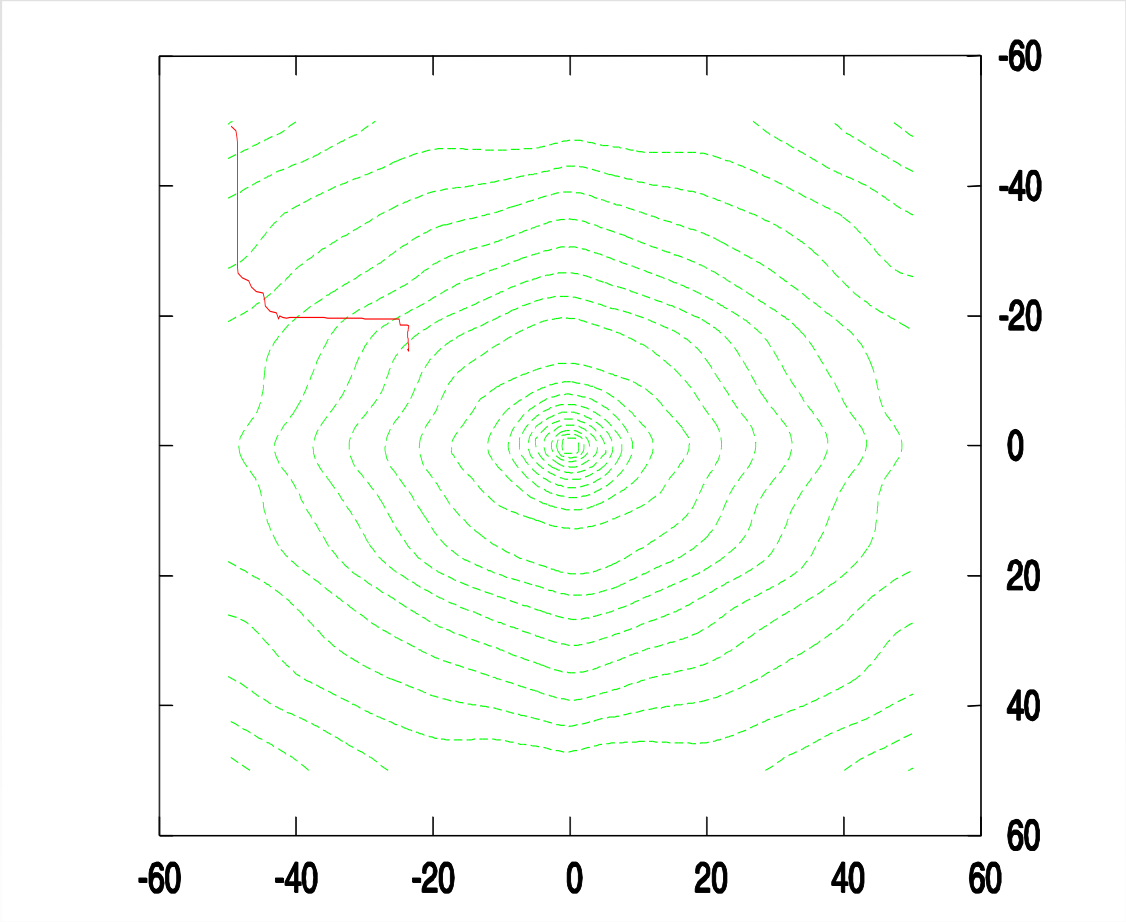
Cuidado com o seu otimizador

Exemplo:

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

Plotando o trajeto do Otimizador

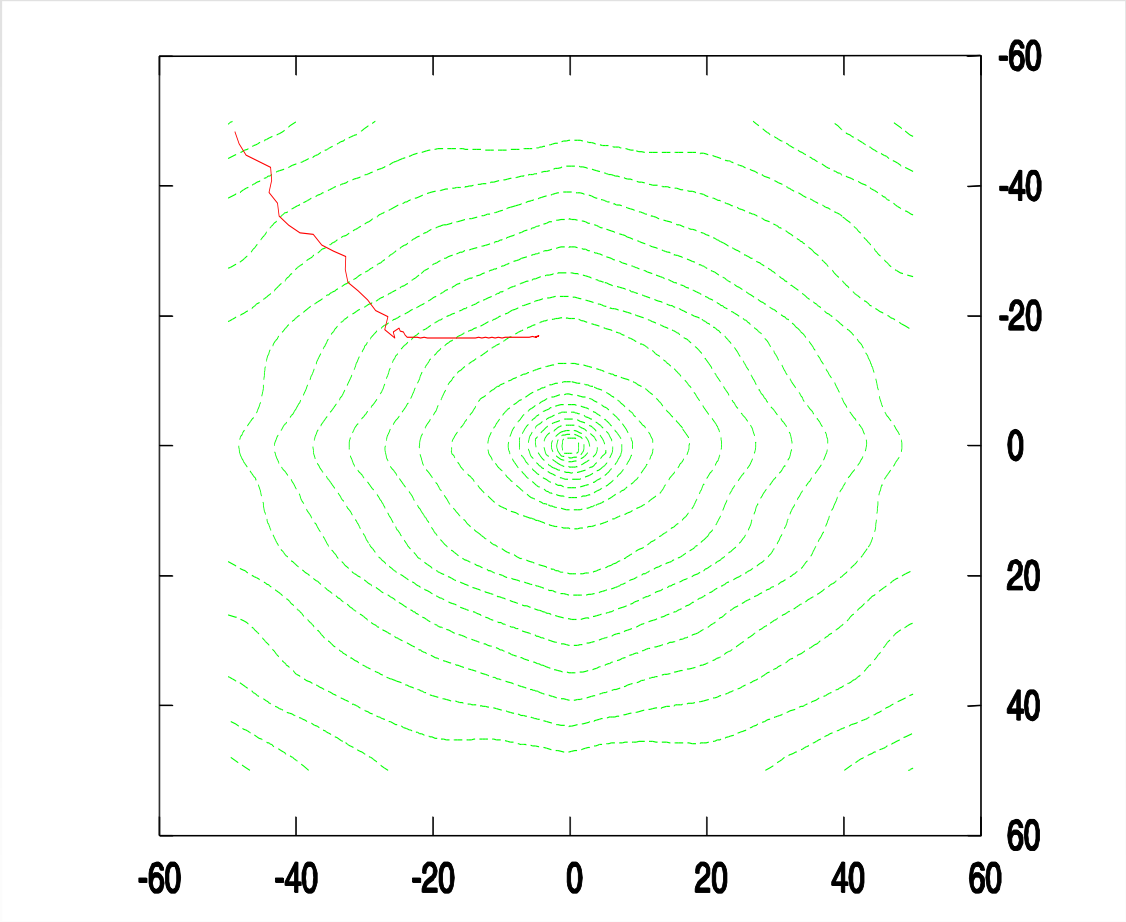
Diferenças Quadráticas Média



Tamanho do passo = 1.0 mm

Plotando o trajeto do Otimizador

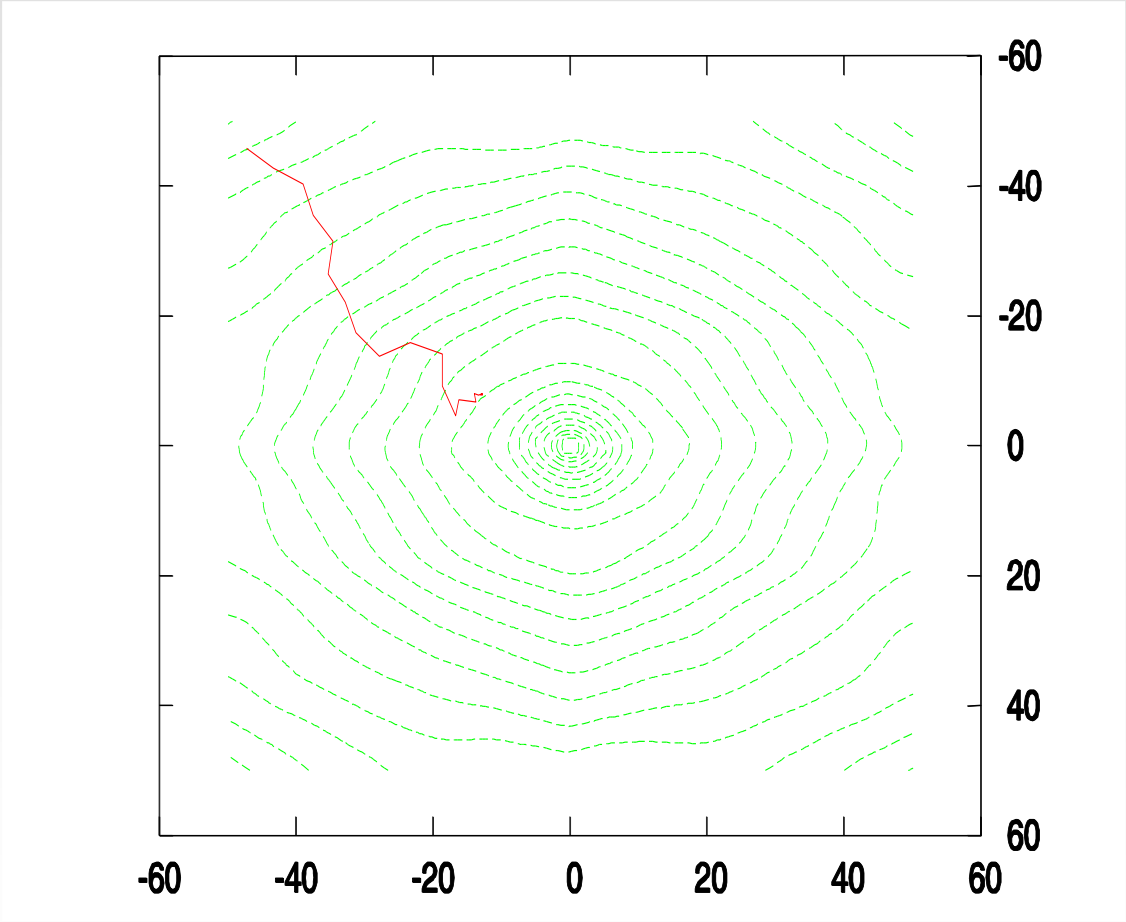
Diferenças Quadráticas Média



Tamanho do passo = 2.0 mm

Plotando o trajeto do Otimizador

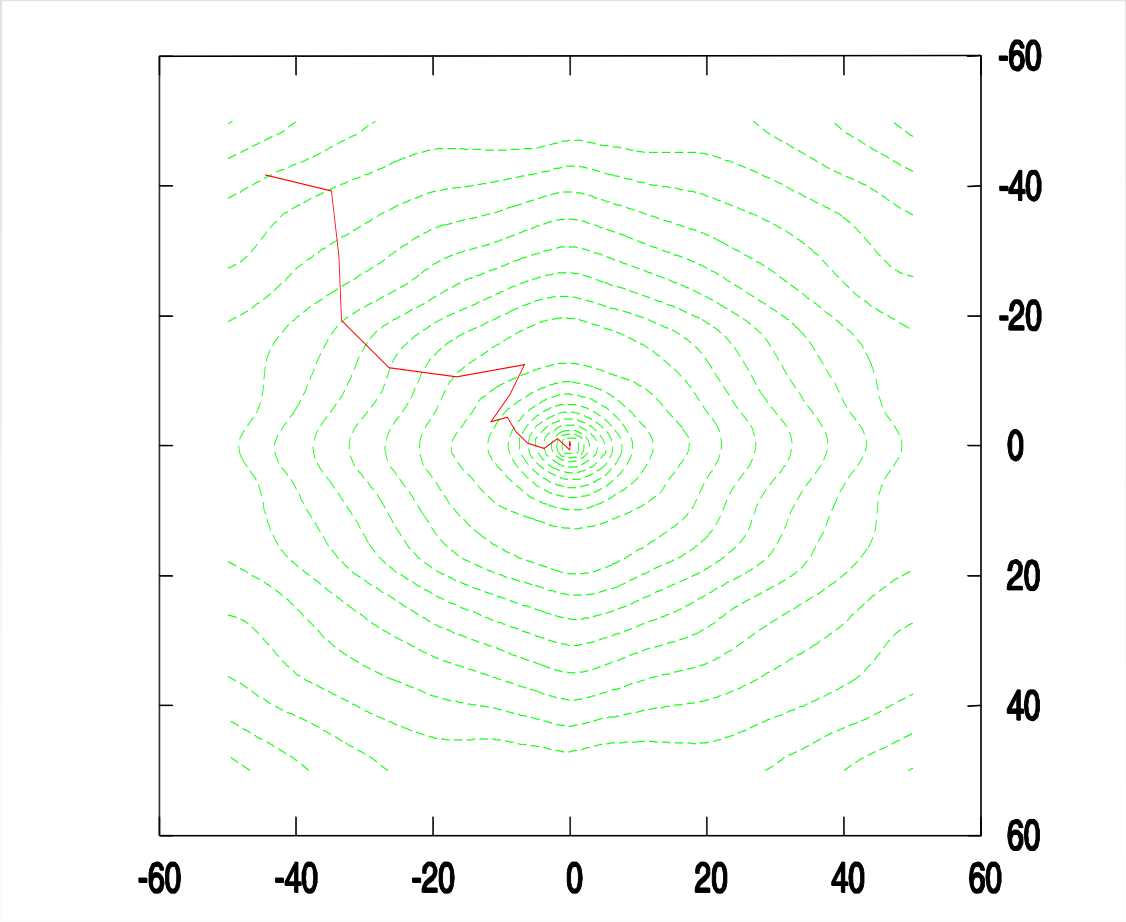
Diferenças Quadráticas Média



Tamanho do passo = 5.0 mm

Plotando o trajeto do Otimizador

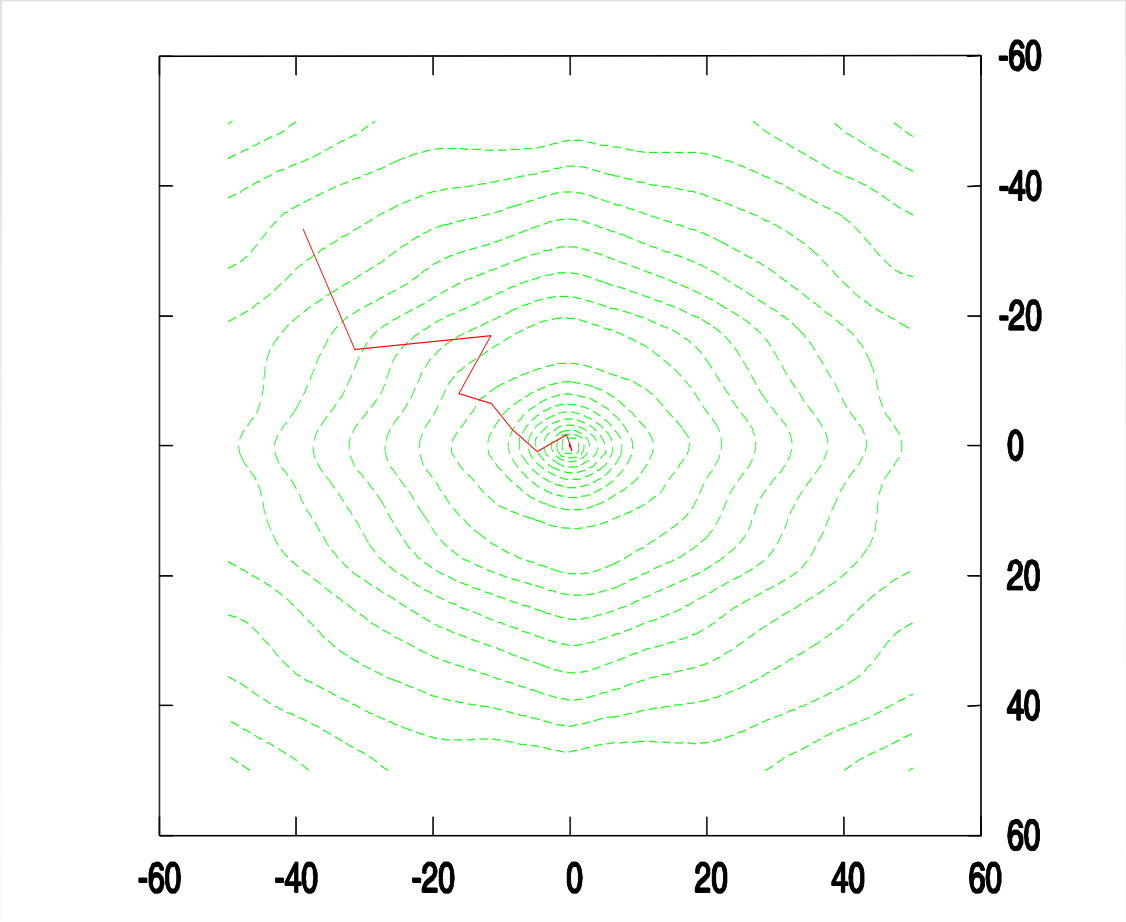
Diferenças Quadráticas Média



Tamanho do passo = 10.0 mm

Plotando o trajeto do Otimizador

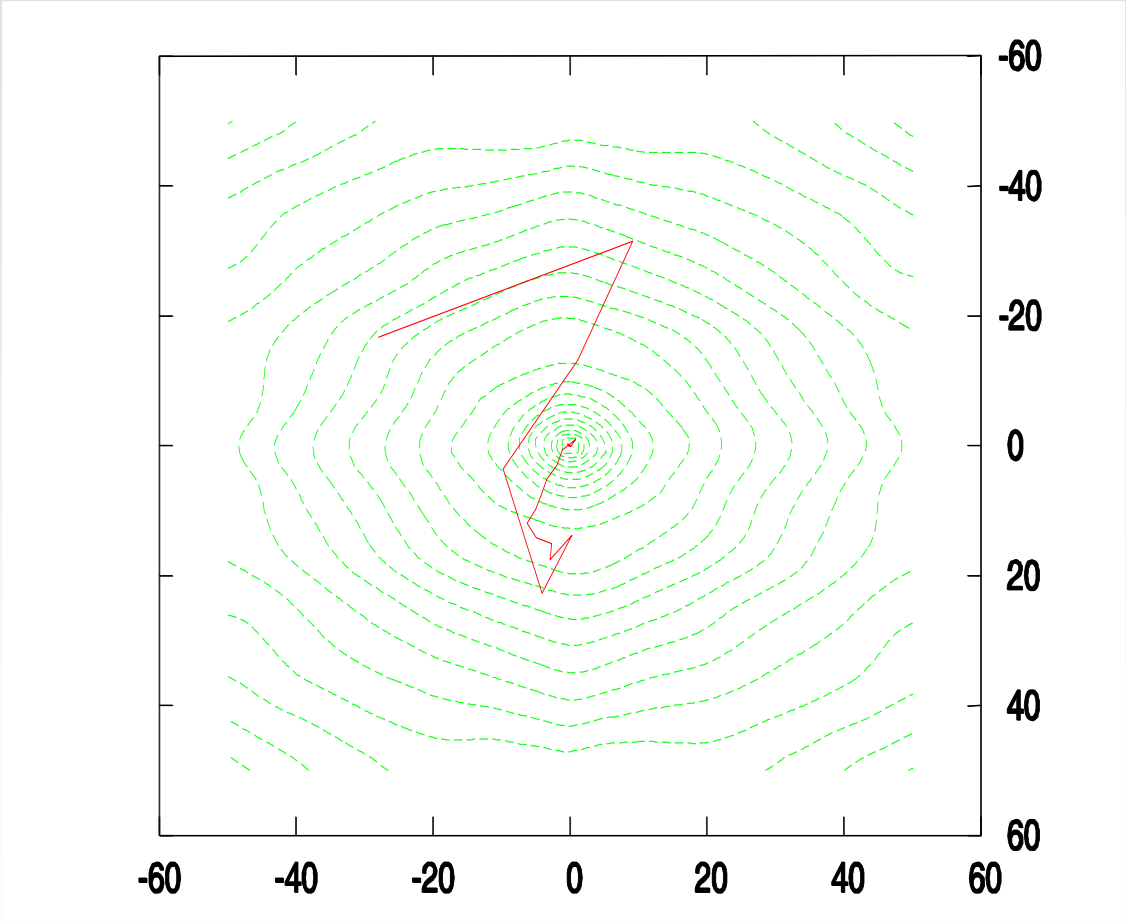
Diferenças Quadráticas Média



Tamanho do passo = 20.0 mm

Plotando o trajeto do Otimizador

Diferenças Quadráticas Média



Tamanho do passo = 40.0 mm

Observe o seu Otimizador

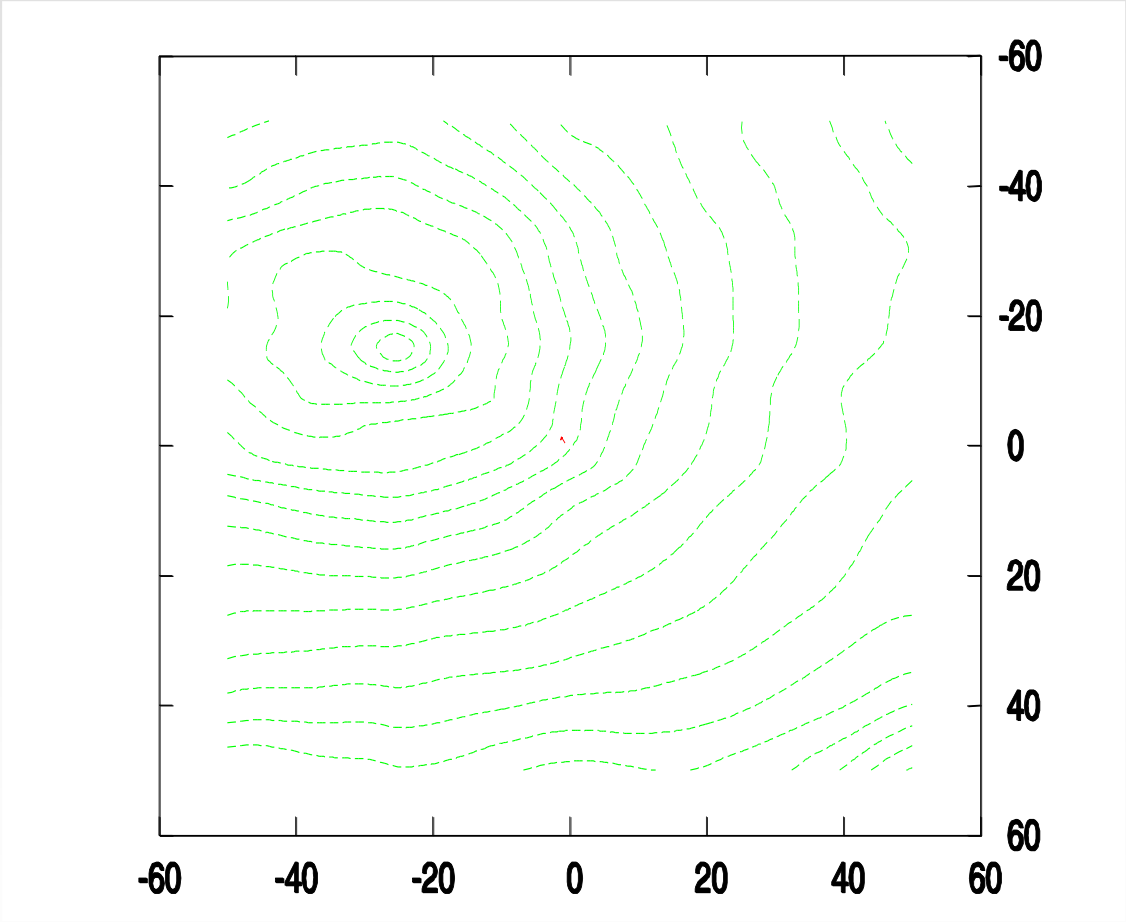
Exemplo:

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

O otimizador começa com (0mm,0mm)

Plotando o trajeto do Otimizador

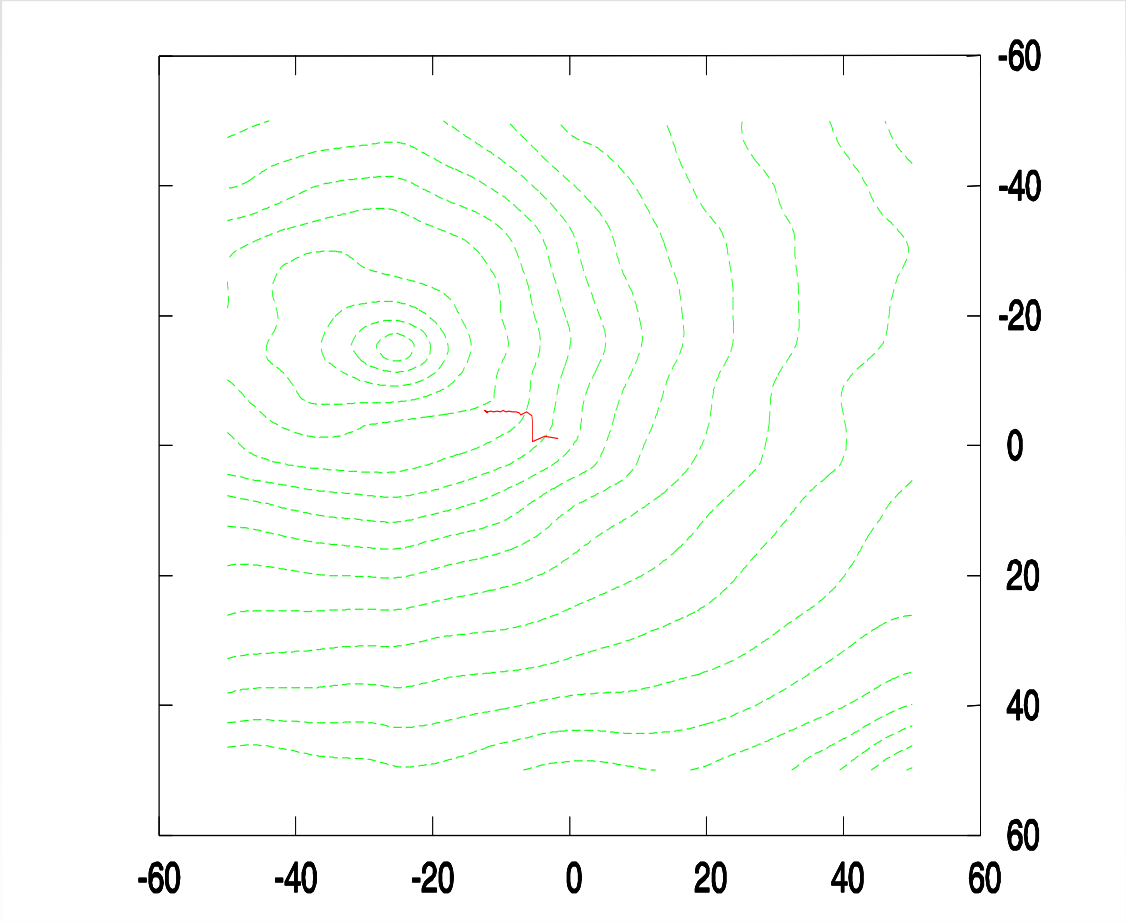
Diferenças Quadráticas Média



Tamanho do passo = 1.0 mm

Plotando o trajeto do Otimizador

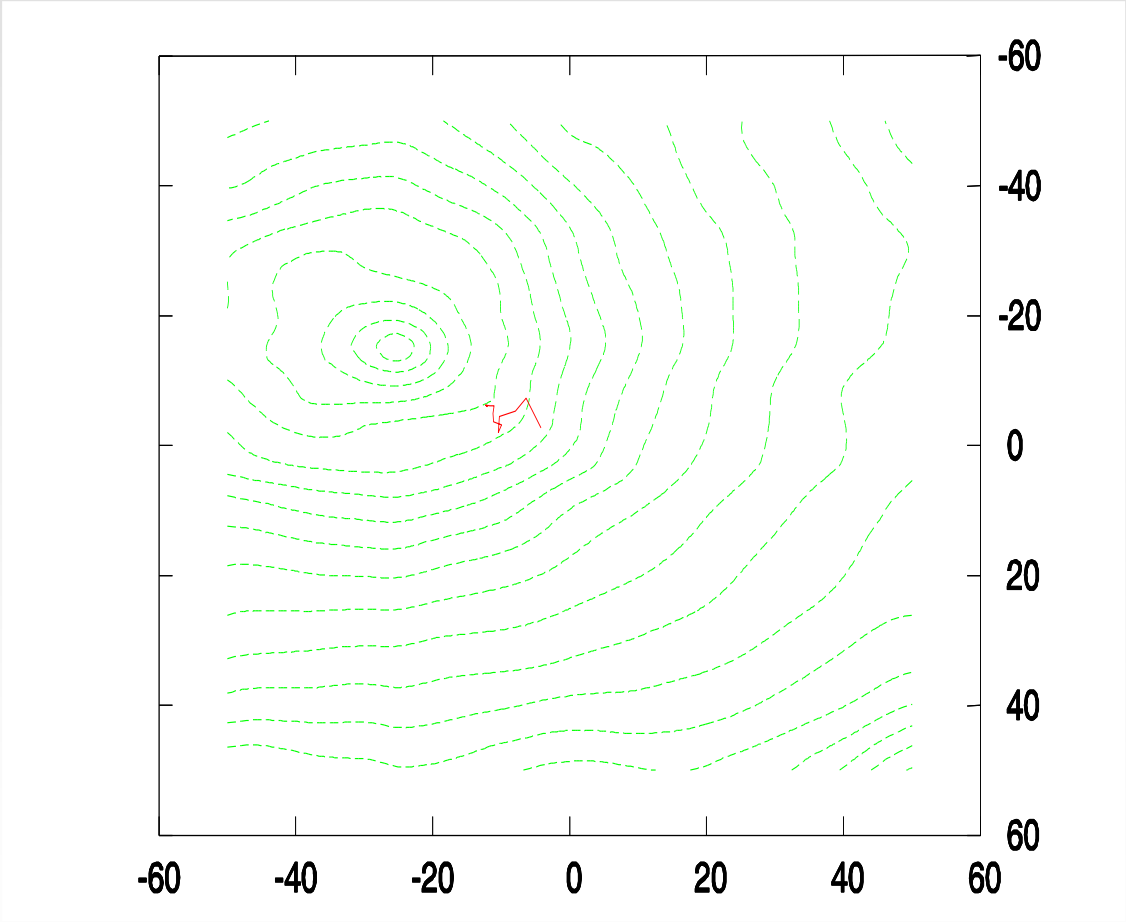
Diferenças Quadráticas Média



Tamanho do passo = 2.0 mm

Plotando o trajeto do Otimizador

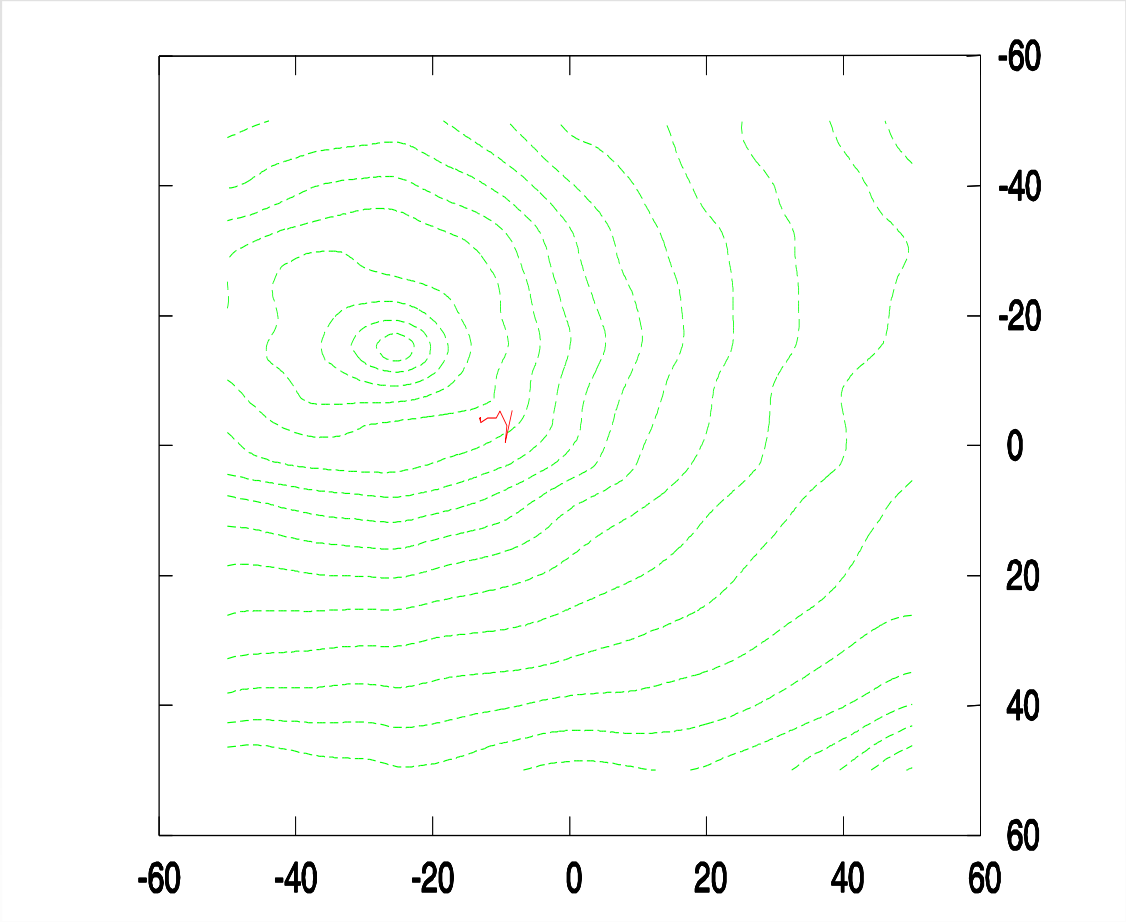
Diferenças Quadráticas Média



Tamanho do passo = 5.0 mm

Plotando o trajeto do Otimizador

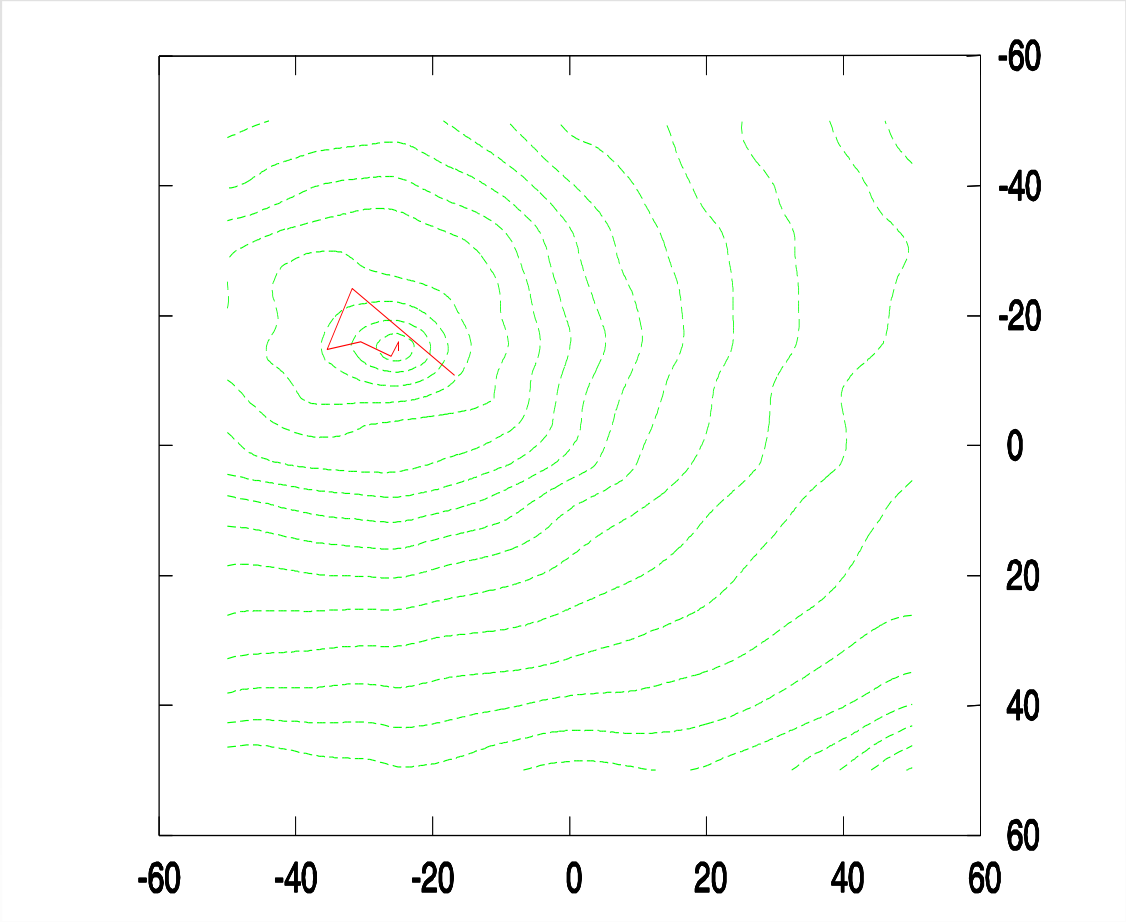
Diferenças Quadráticas Média



Tamanho do passo = 10.0 mm

Plotando o trajeto do Otimizador

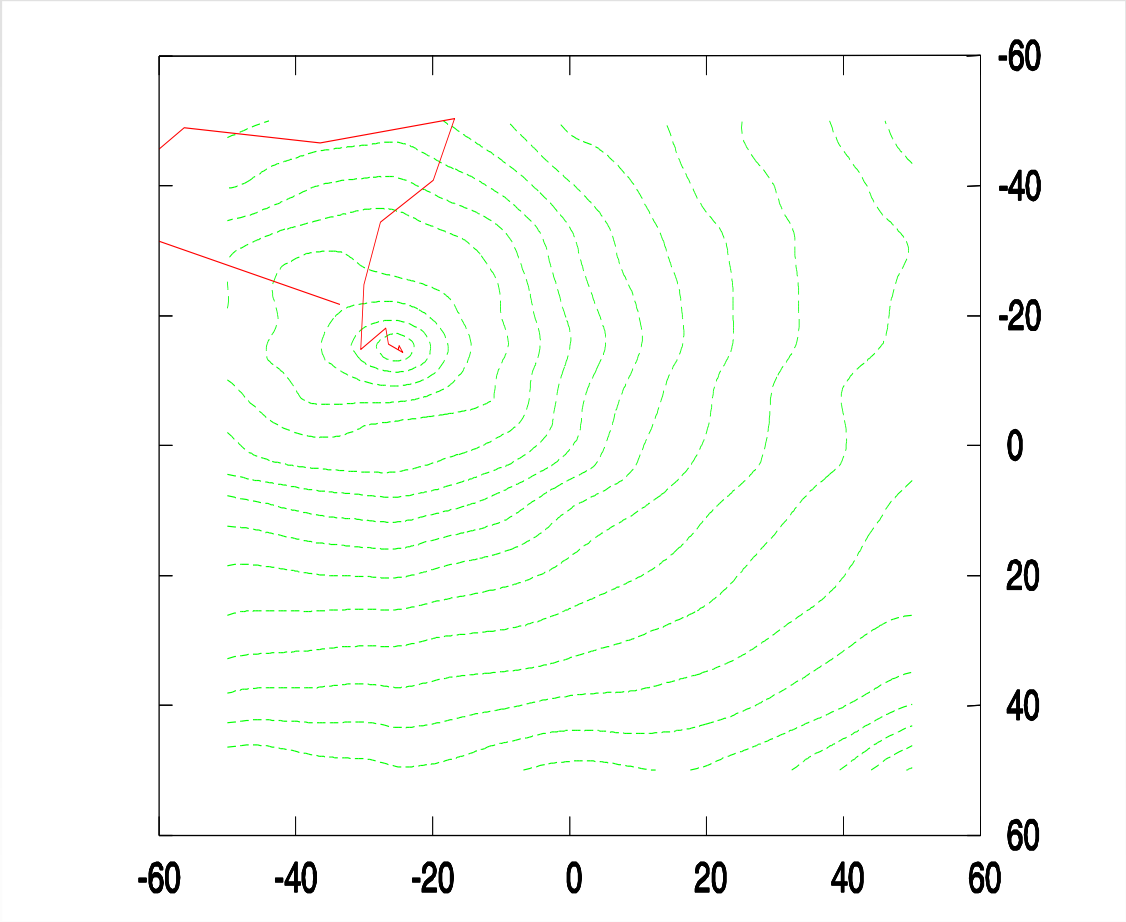
Diferenças Quadráticas Média



Tamanho do passo = 20.0 mm

Plotando o trajeto do Otimizador

Diferenças Quadráticas Média



Tamanho do passo = 40.0 mm

Multimodalidade

Corregistro Multimodal

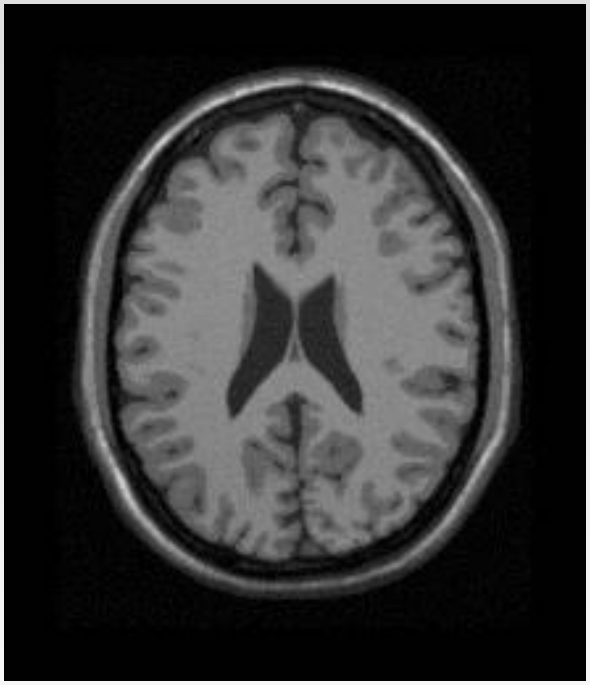


Imagem Fixa

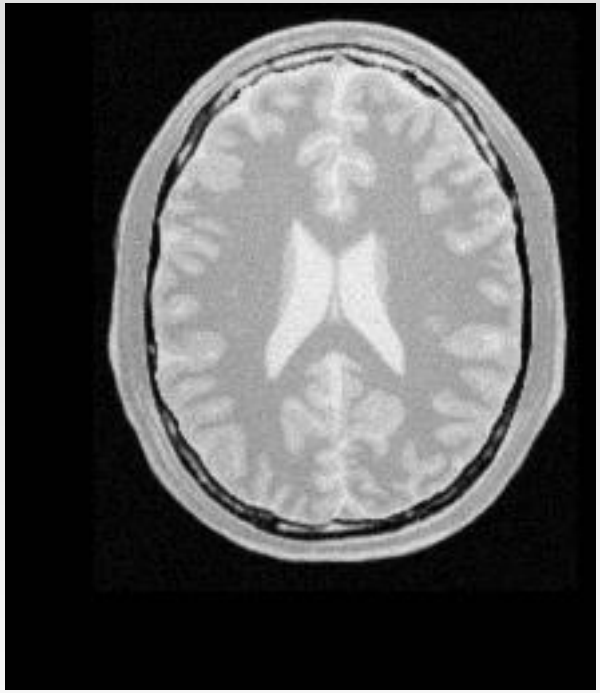


Imagem Móvel

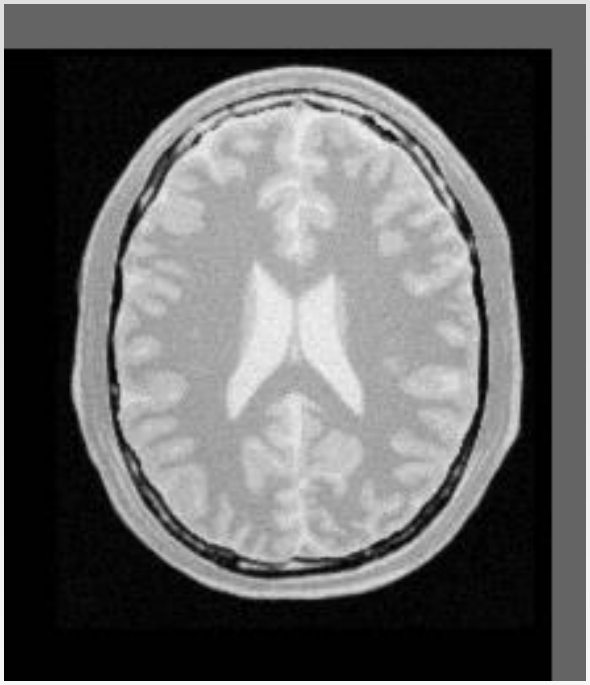
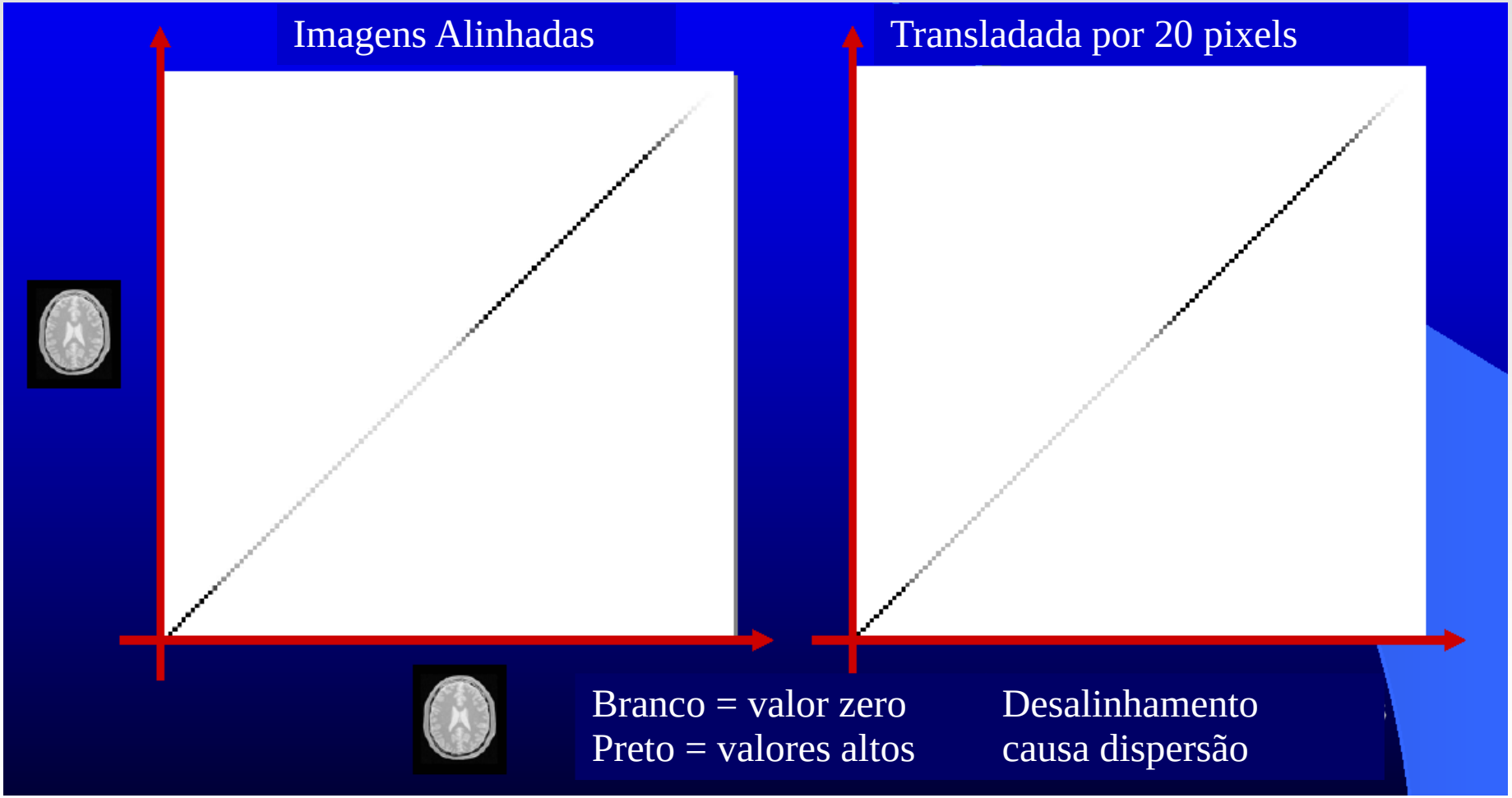
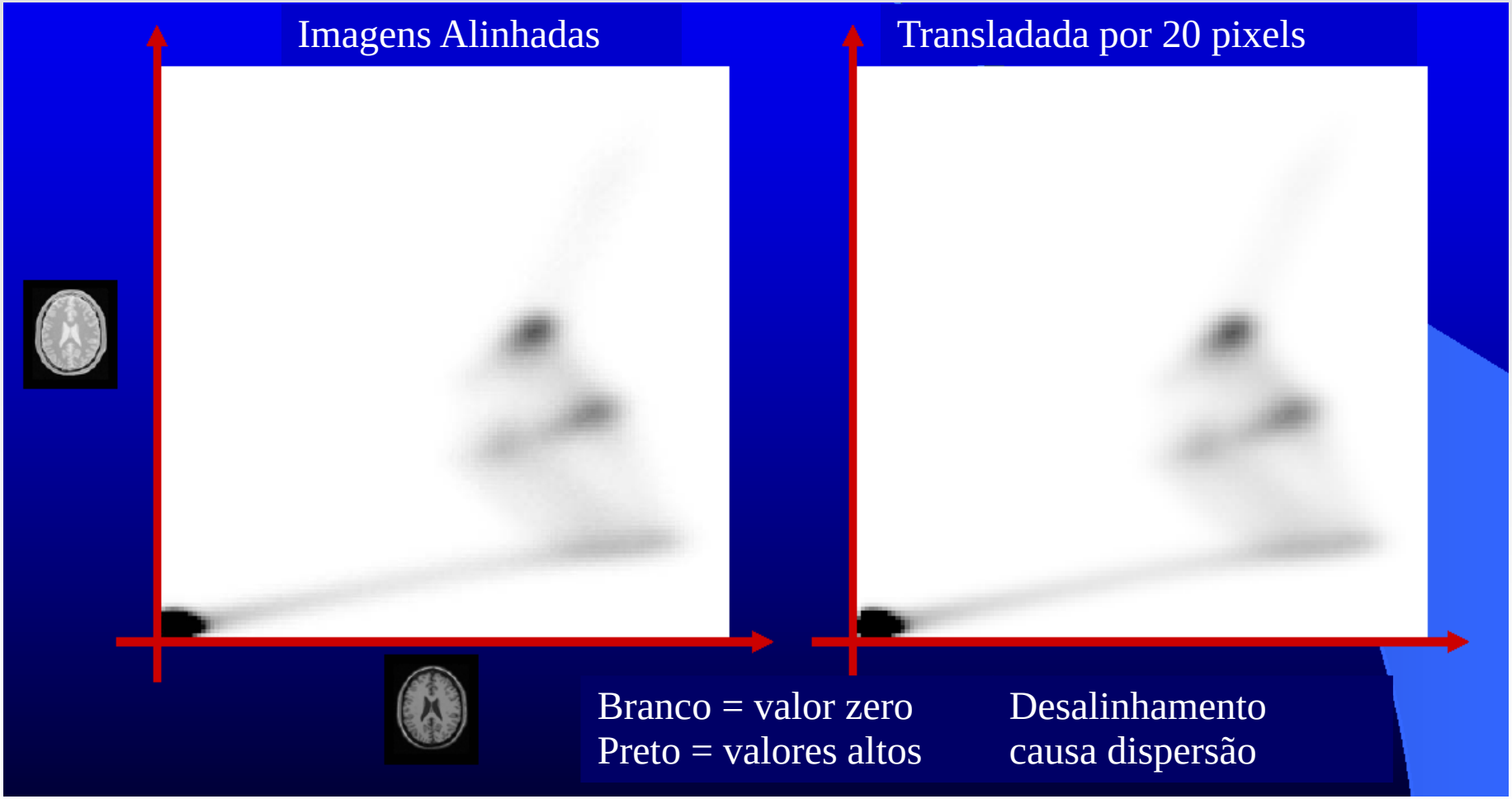


Imagem Móvel Corregistrada

Informação Mutua



Informação Mutua



Corregistro de imagens

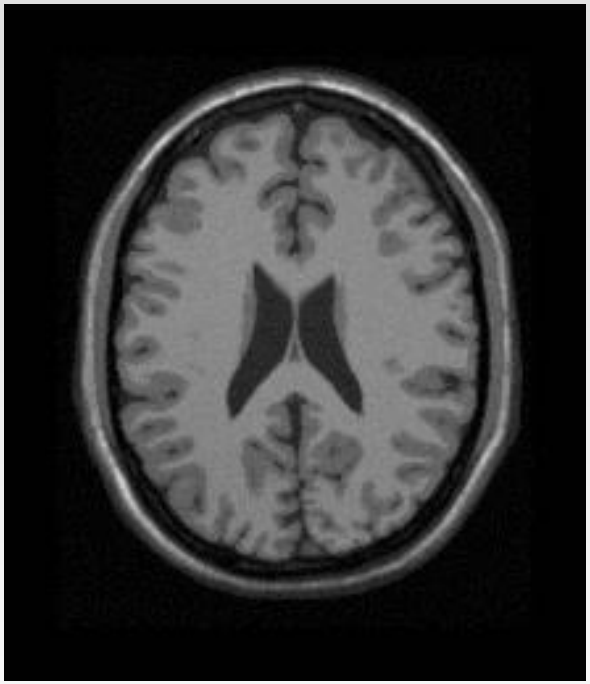


Imagem Fixa

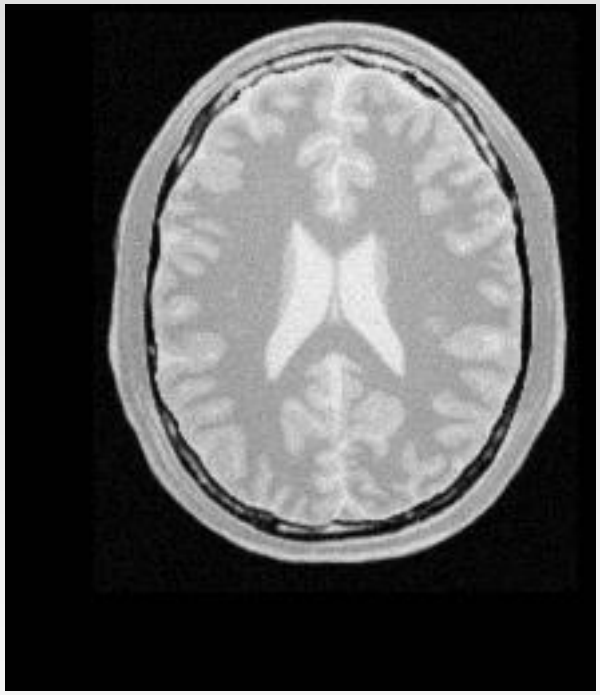


Imagem Móvel

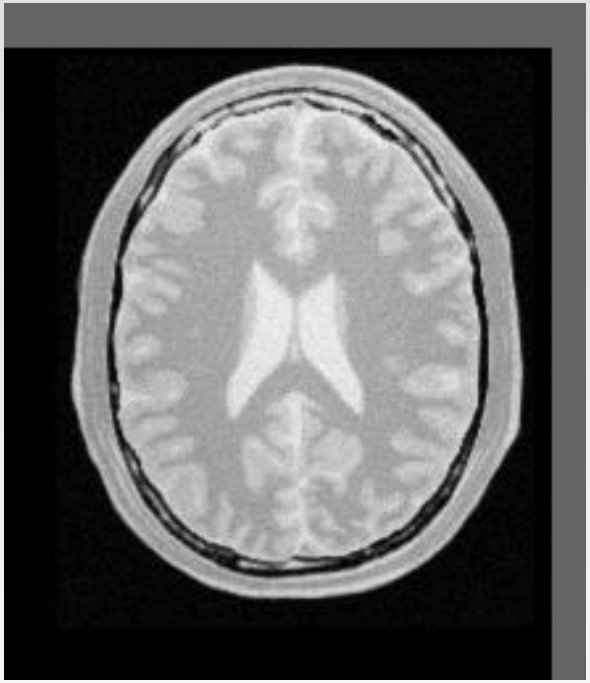


Imagem Móvel Corregistrada

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

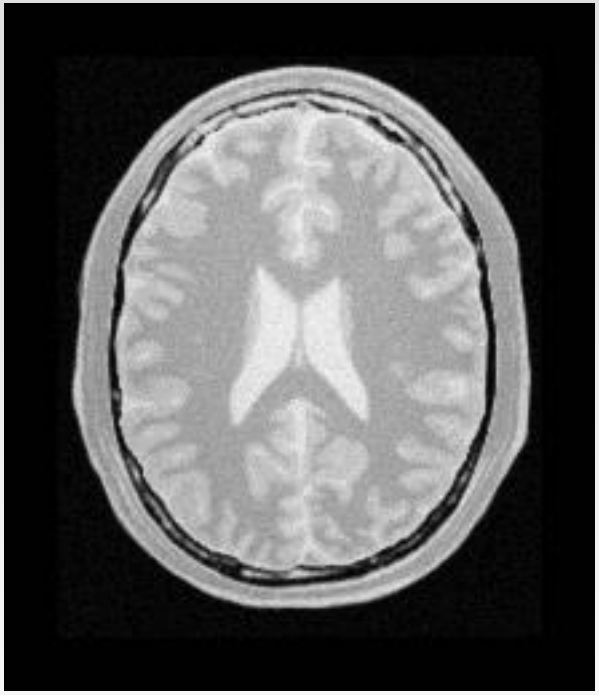


Imagem Fixa

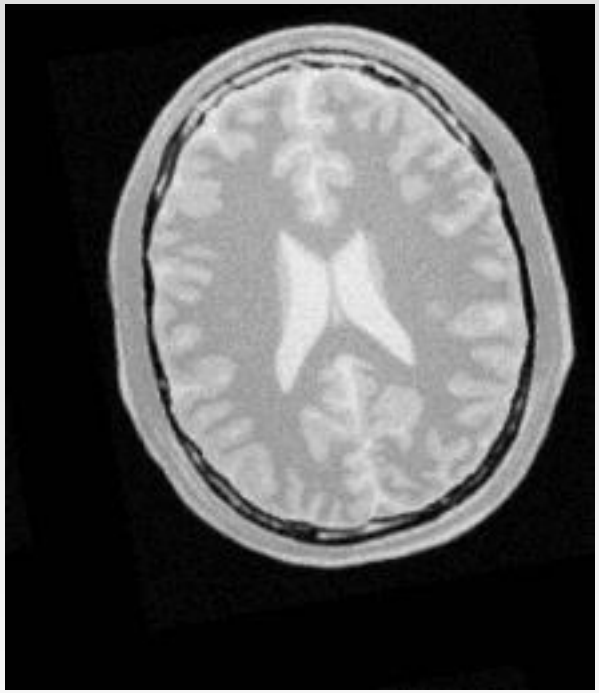


Imagem Móvel

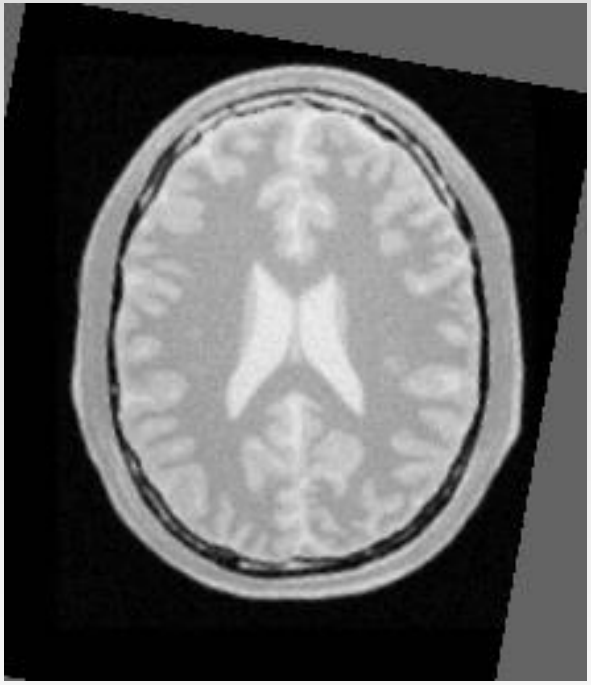


Imagem Móvel Corregistrada

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

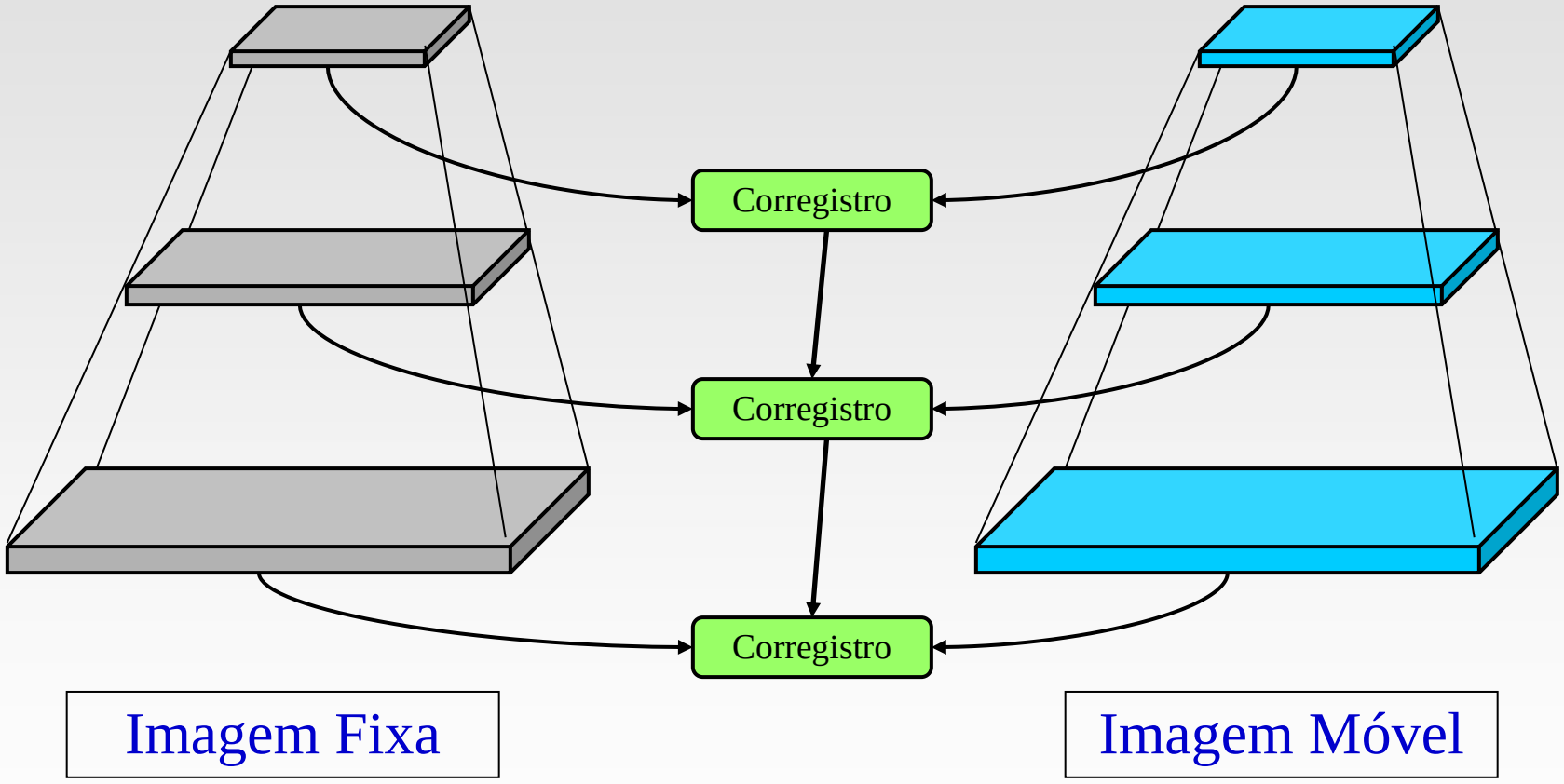
```
typedef   itk::ResampleImageFilter<
                    FixedImageType , MovingImageType >     ResamplerType;

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