ITK v4 Registration Refactoring

Feb 2, 2011

Brian Avants, Nick Tustison, Gang Song, Michael Stauffer, Shreyas Seshamani, James C. Gee

Penn Image Computing and Sciences Laboratory
University of Pennsylvania, Philadelphia, PA.

What moves ITK registration forward

What moves ITK registration forward

- Hans Johnson: why does Mattes MI fail in ITK v4 when the same code worked in ITK320? BSplines?

What moves ITK registration forward

- Hans Johnson: why does Mattes MI fail in ITK v4 when the same code worked in ITK320? BSplines?
- Evaluation studies: these motivate students and benchmark progress. Data with quantitation attached.

What moves ITK registration forward

- Hans Johnson: why does Mattes MI fail in ITK v4 when the same code worked in ITK320? BSplines?
- Evaluation studies: these motivate students and benchmark progress. Data with quantitation attached.
- New problems: interesting data that existing approaches do not easily handle (i.e. use cases).

What moves ITK registration forward

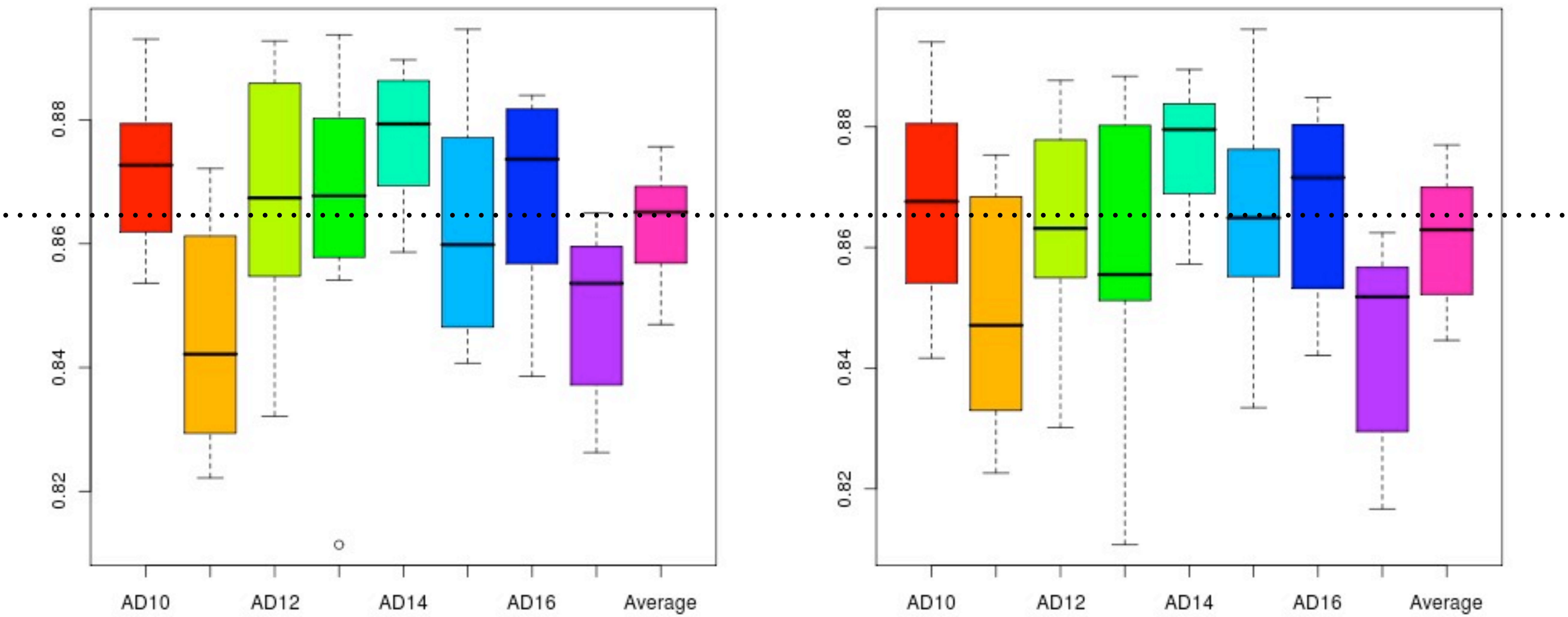
- Hans Johnson: why does Mattes MI fail in ITK v4 when the same code worked in ITK320? BSplines?
- Evaluation studies: these motivate students and benchmark progress. Data with quantitation attached.
- New problems: interesting data that existing approaches do not easily handle (i.e. use cases).
- Slicer: big projects that we want to support. Kilian asks "why is resampling taking longer than registration?"

What moves ITK registration forward

- Hans Johnson: why does Mattes MI fail in ITK v4 when the same code worked in ITK320? BSplines?
- Evaluation studies: these motivate students and benchmark progress. Data with quantitation attached.
- New problems: interesting data that existing approaches do not easily handle (i.e. use cases).
- Slicer: big projects that we want to support. Kilian asks "why is resampling taking longer than registration?"
- New theory: interesting ideas we should be able to easily implement. E.g. LDDMM, longitudinal mapping, multivariate problems ...

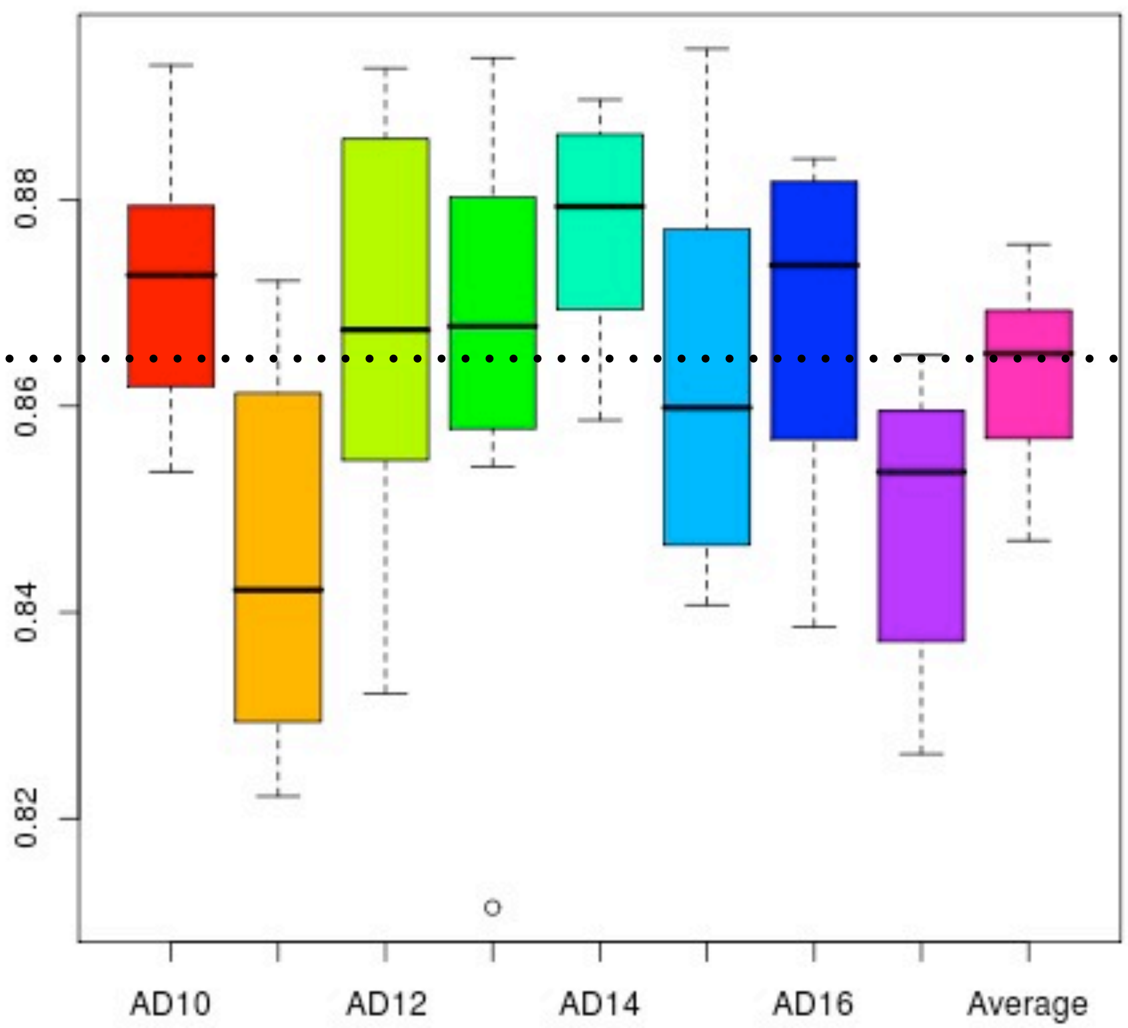
ITK Mattes MI vs FSL correlation ratio

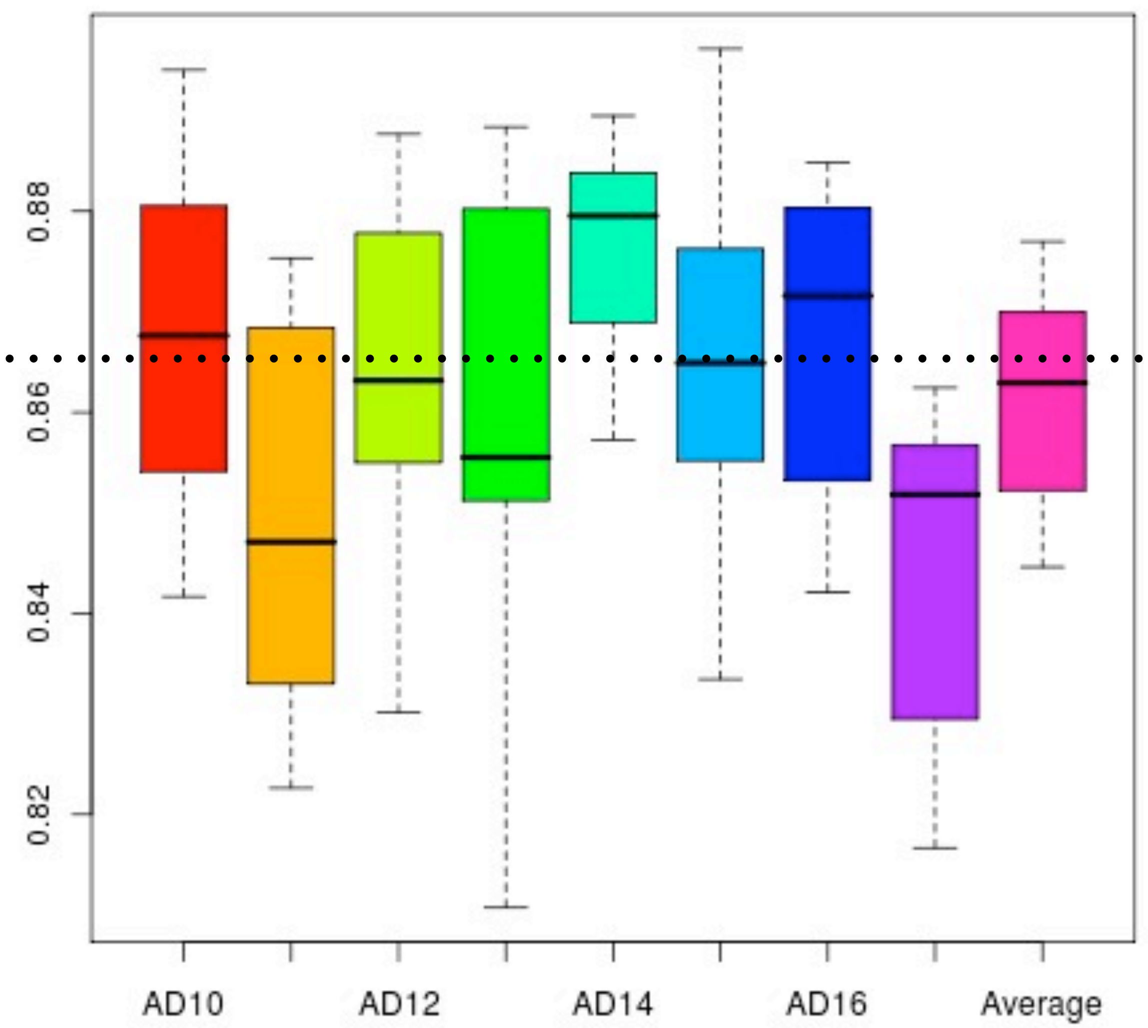
Evaluate brain overlap in neuroimages



ITK Mattes MI vs FSL correlation ratio

Evaluate brain overlap in neuroimages

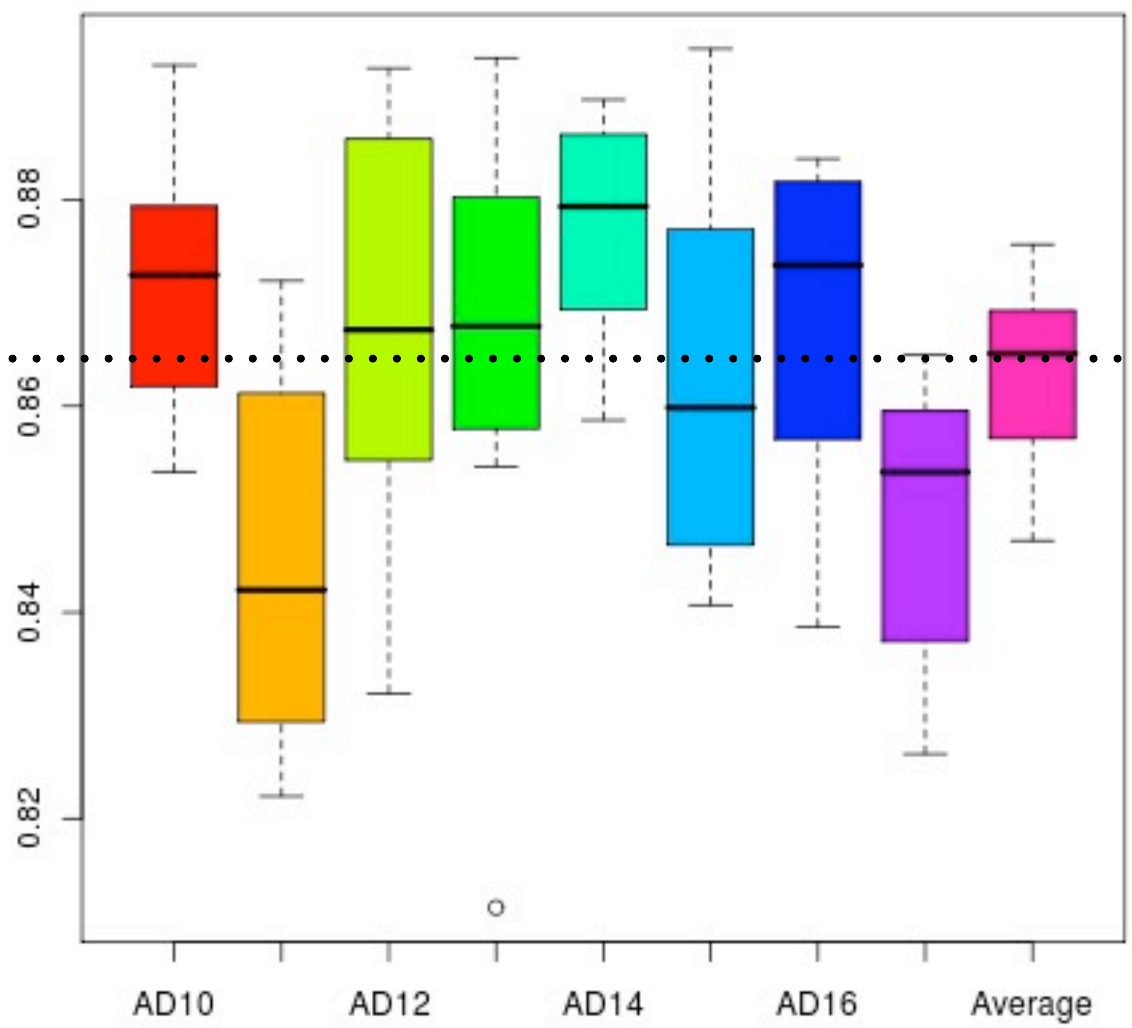




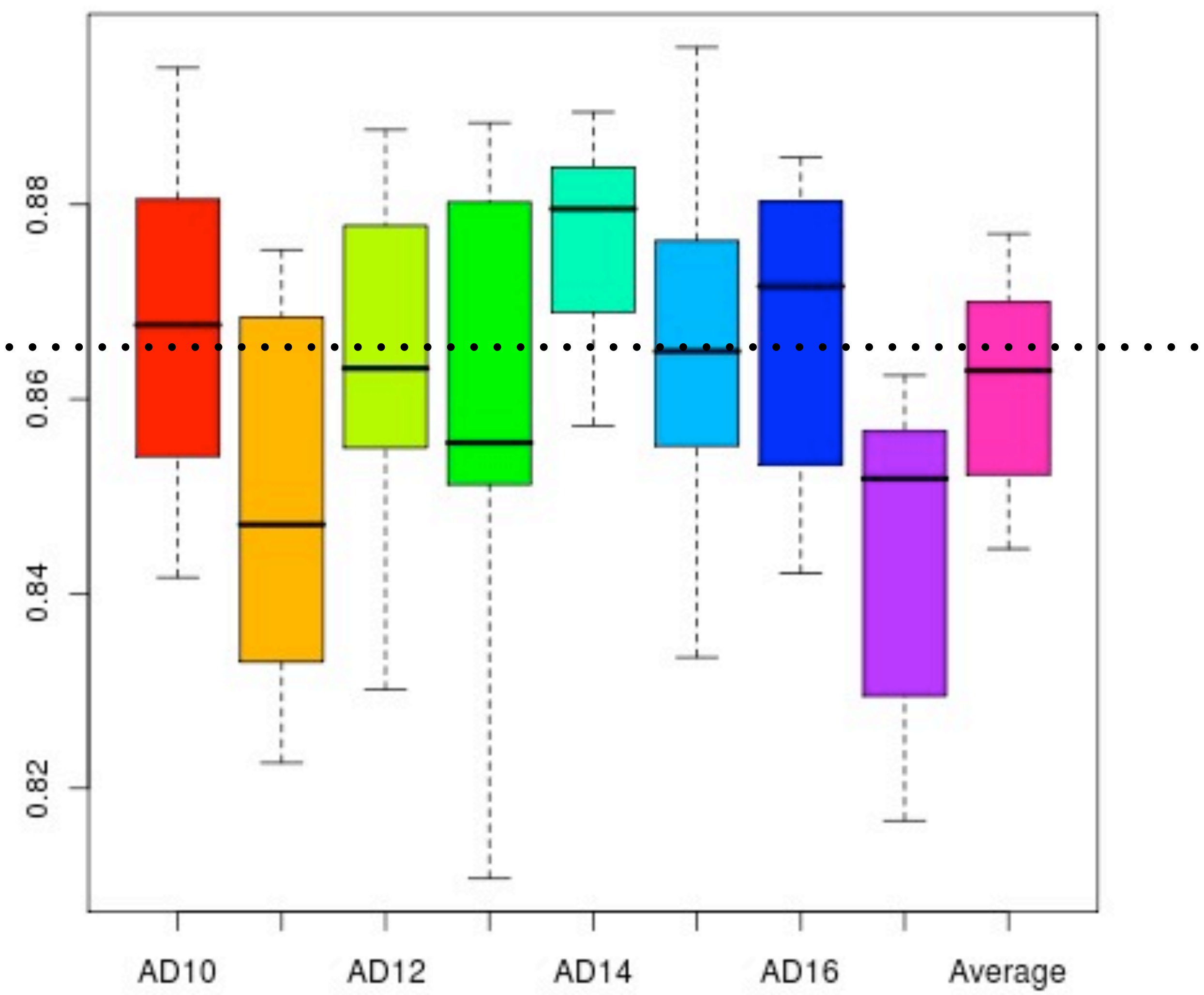
Default ANTs parameters
and optimization.

ITK Mattes MI vs FSL correlation ratio

Evaluate brain overlap in neuroimages



Default ANTs parameters and optimization.



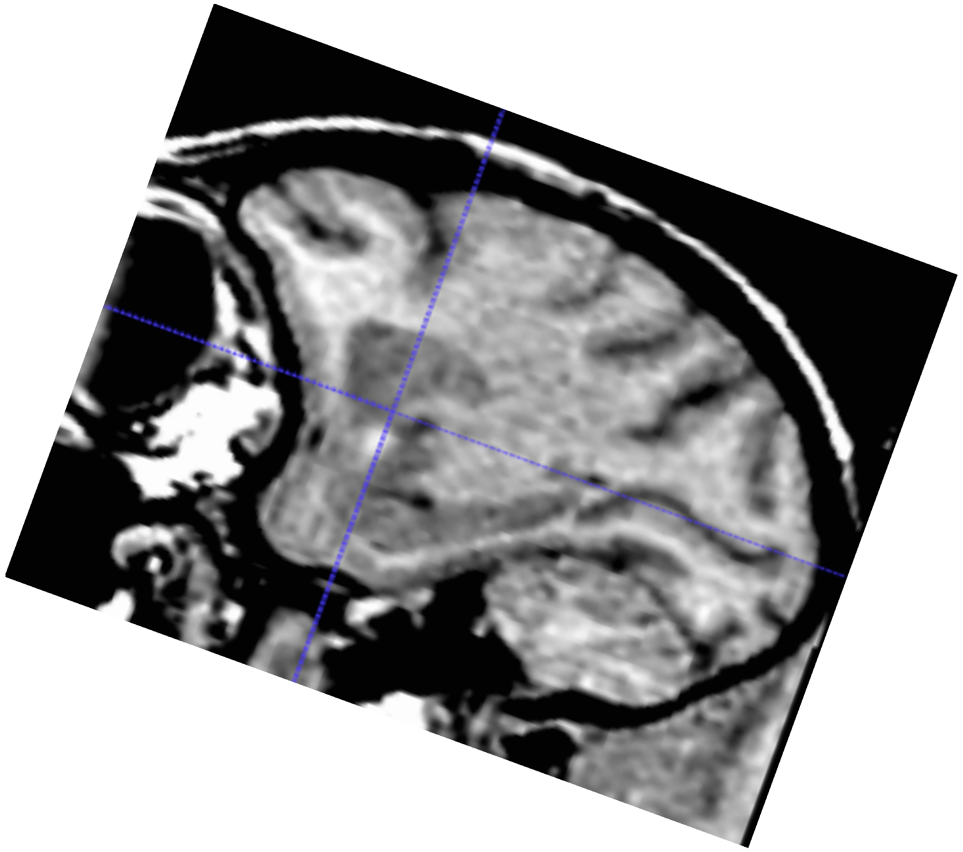
Default FSL parameters and optimization.

Biased registration

A B

Biased registration

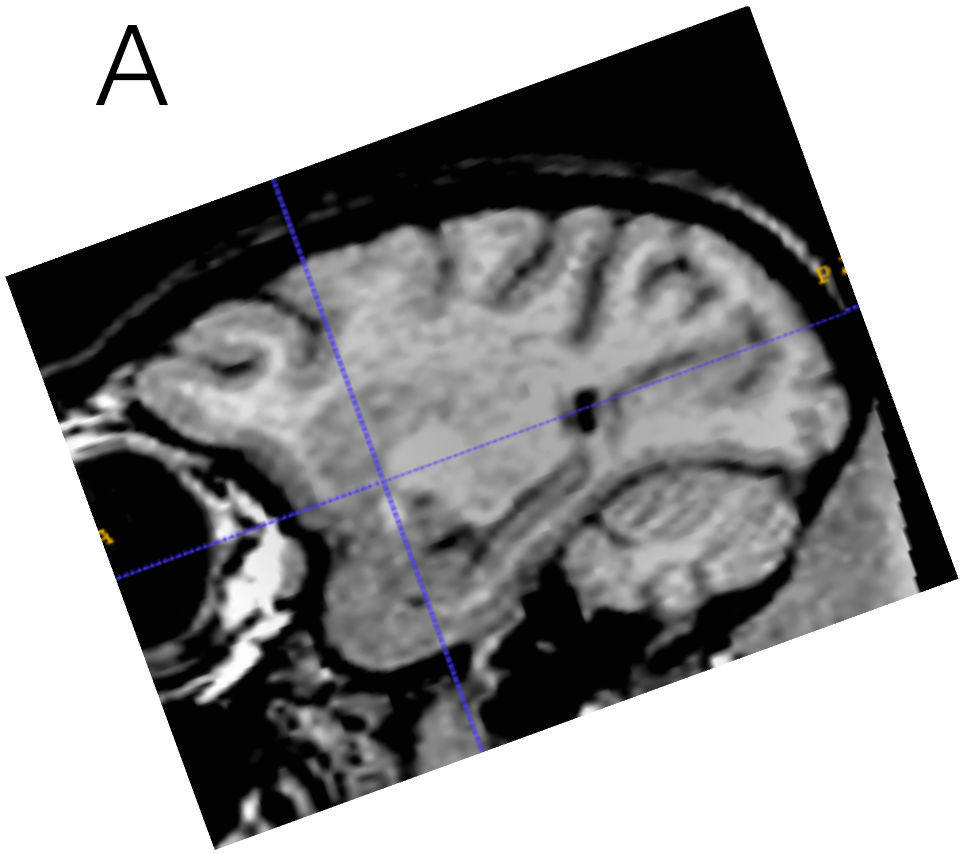
A



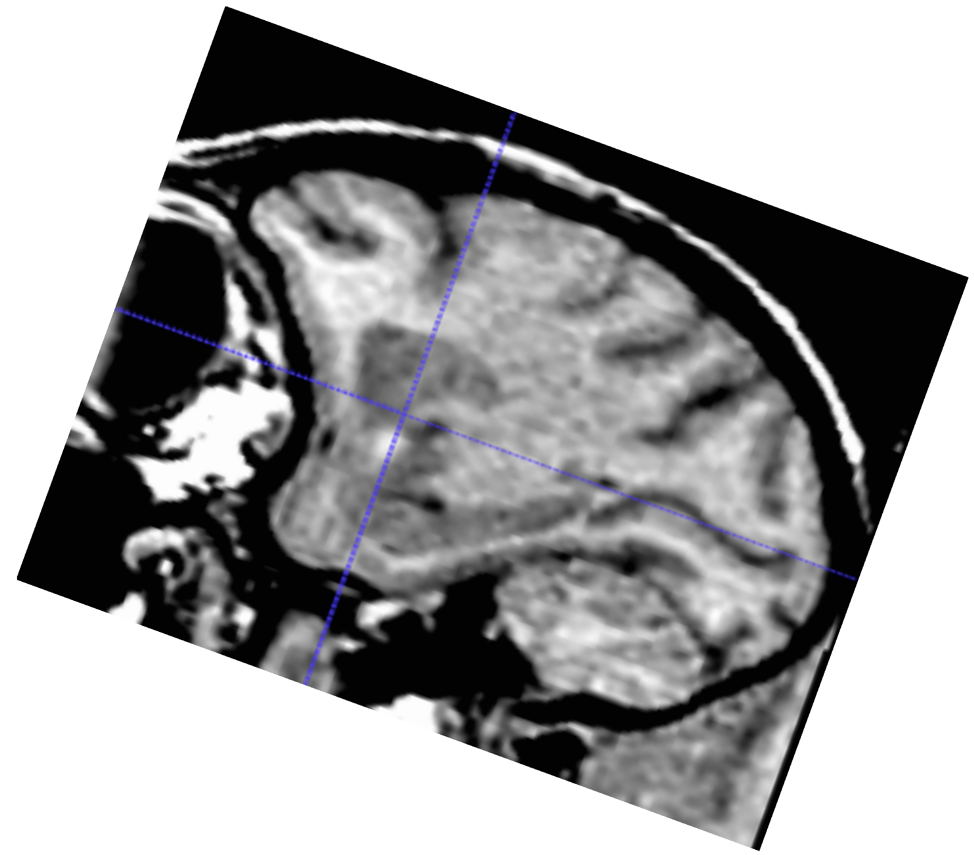
B

Biased registration

A

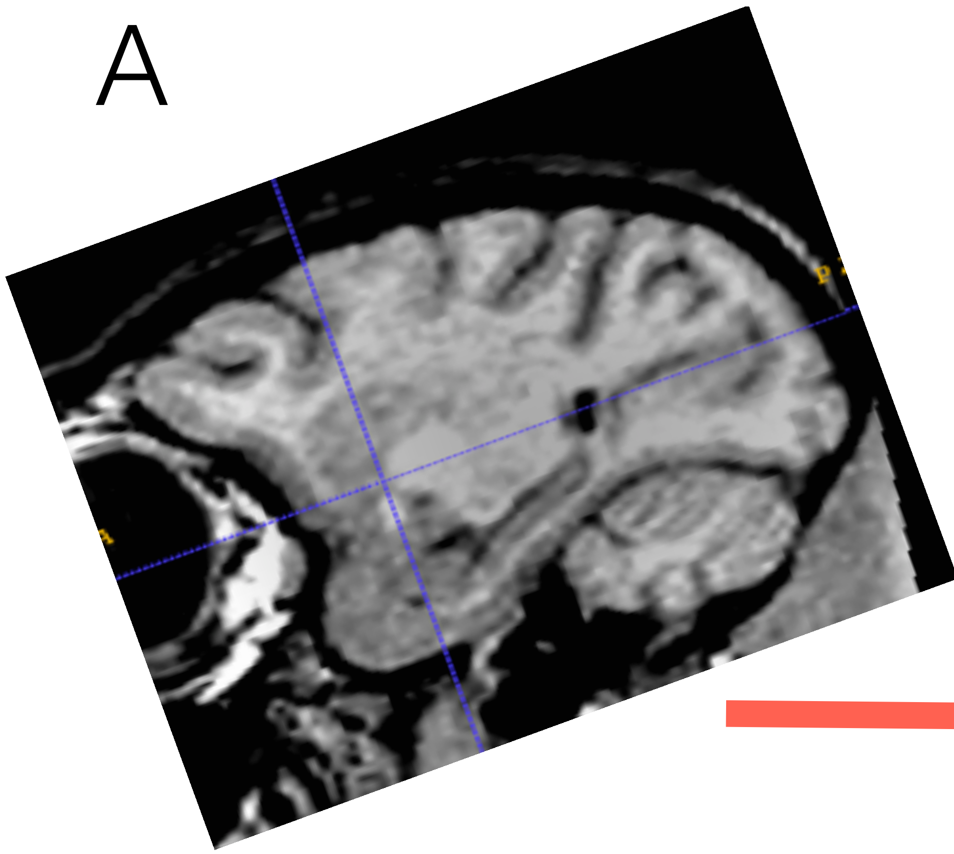


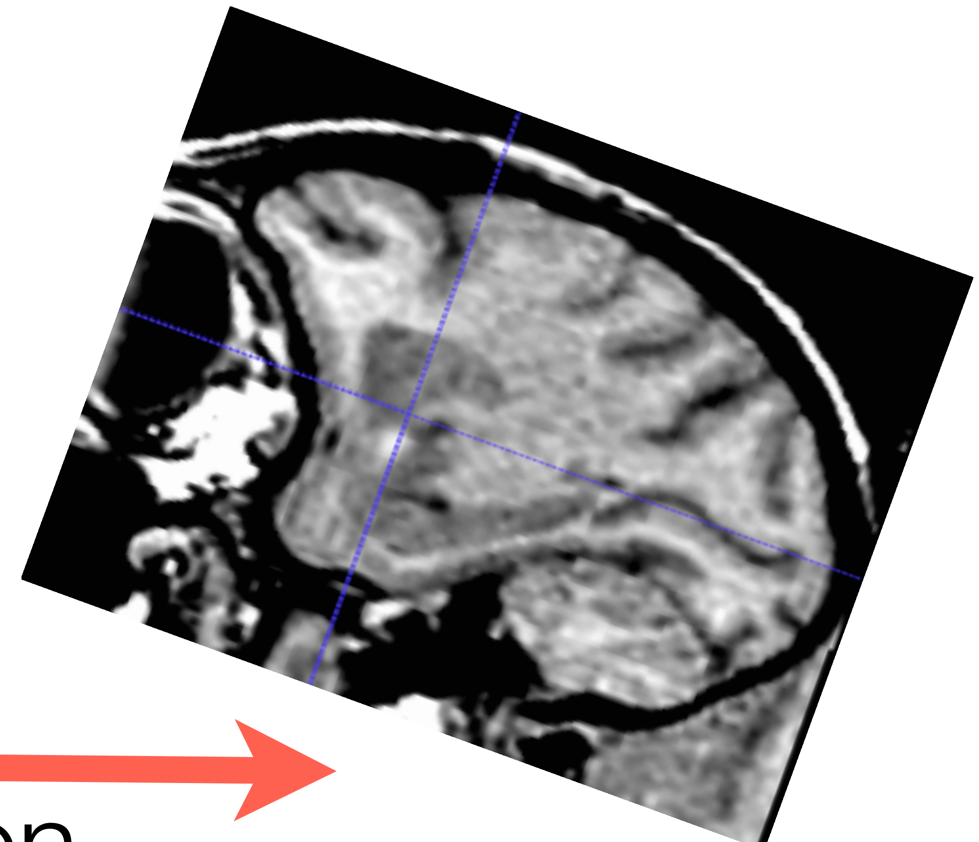
B



Biased registration

A



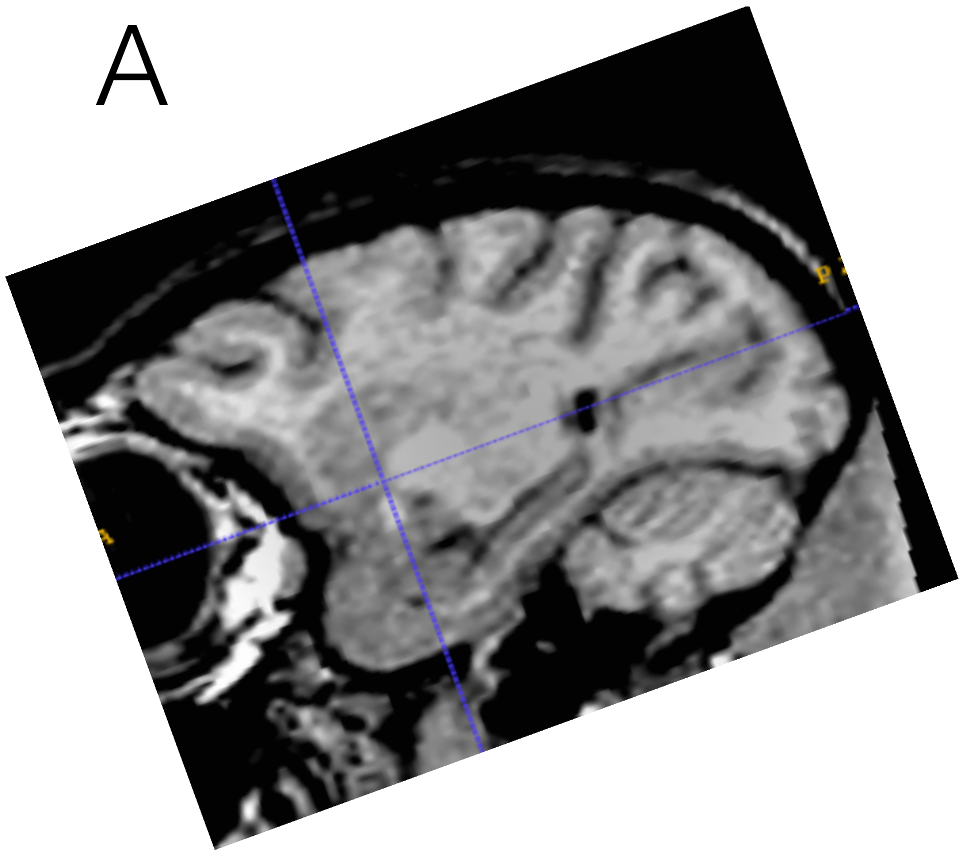


B

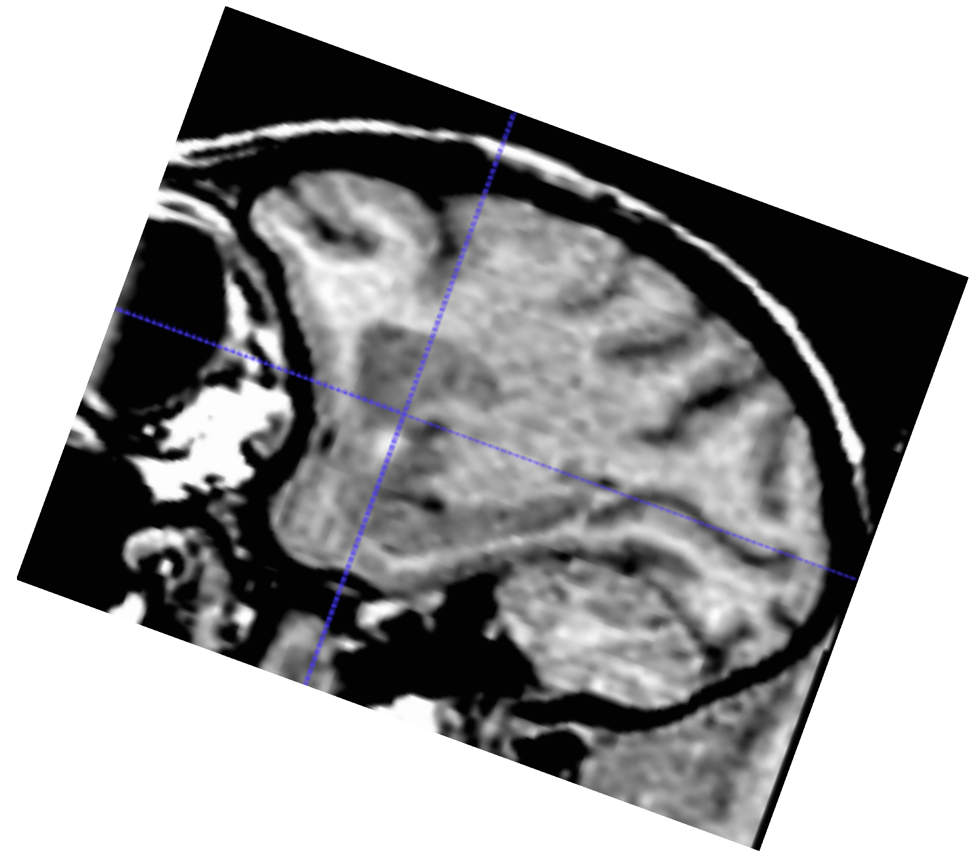
Registration

Unbiased registration

A

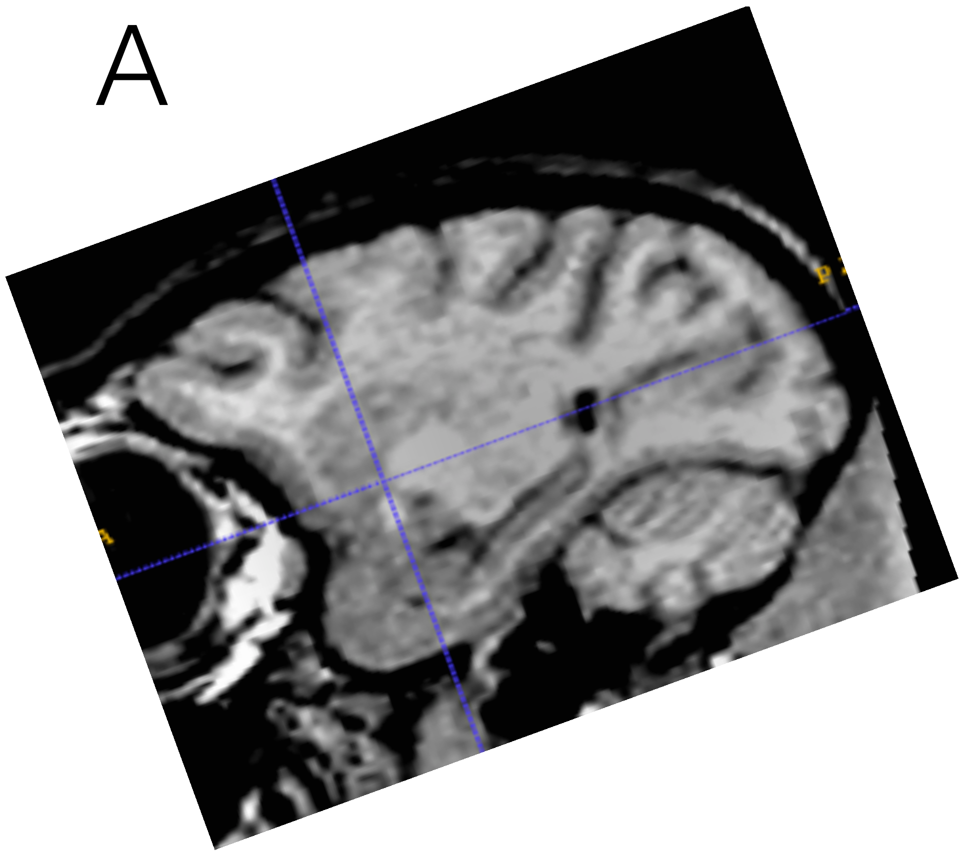


B

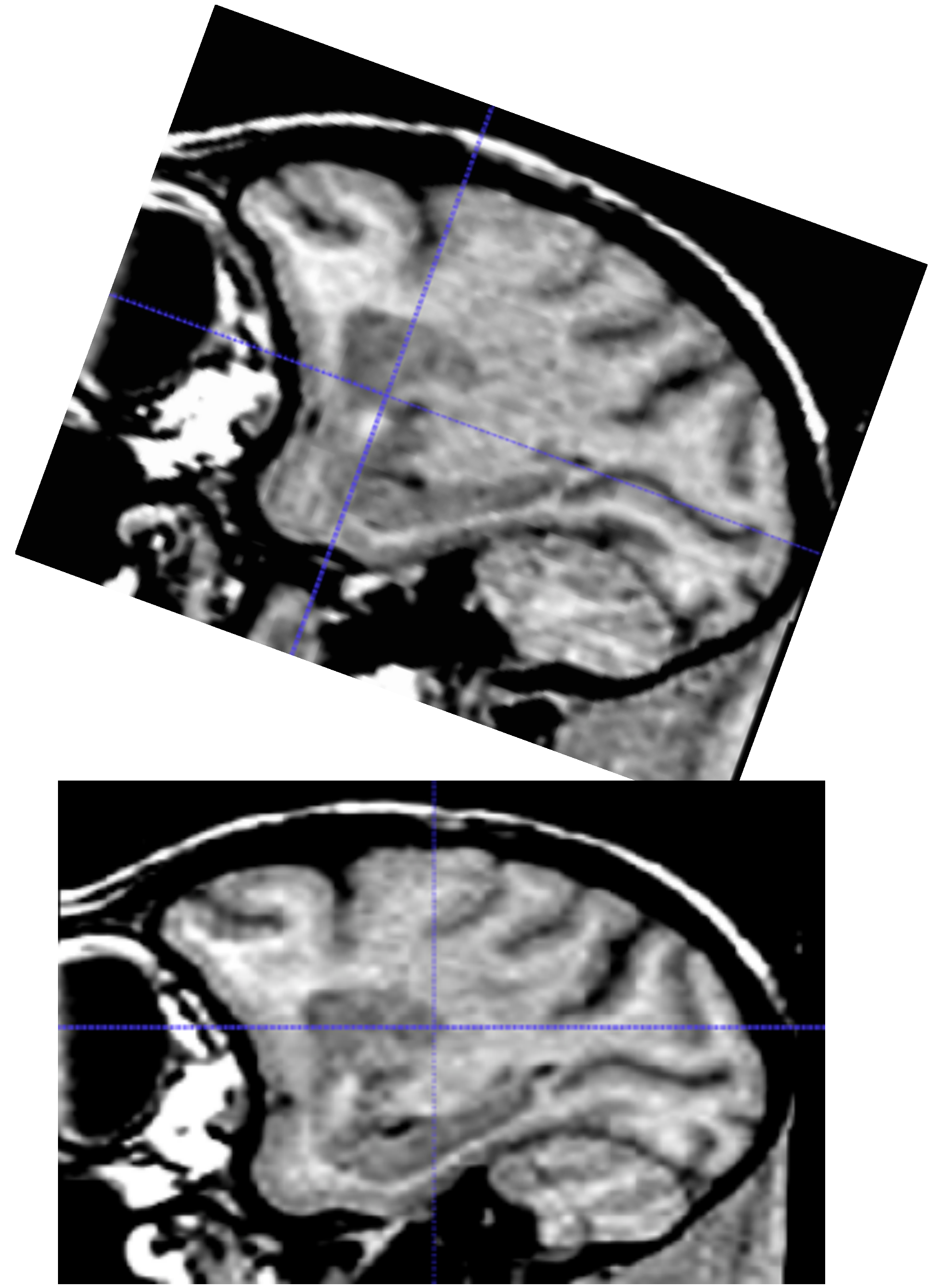


Unbiased registration

A

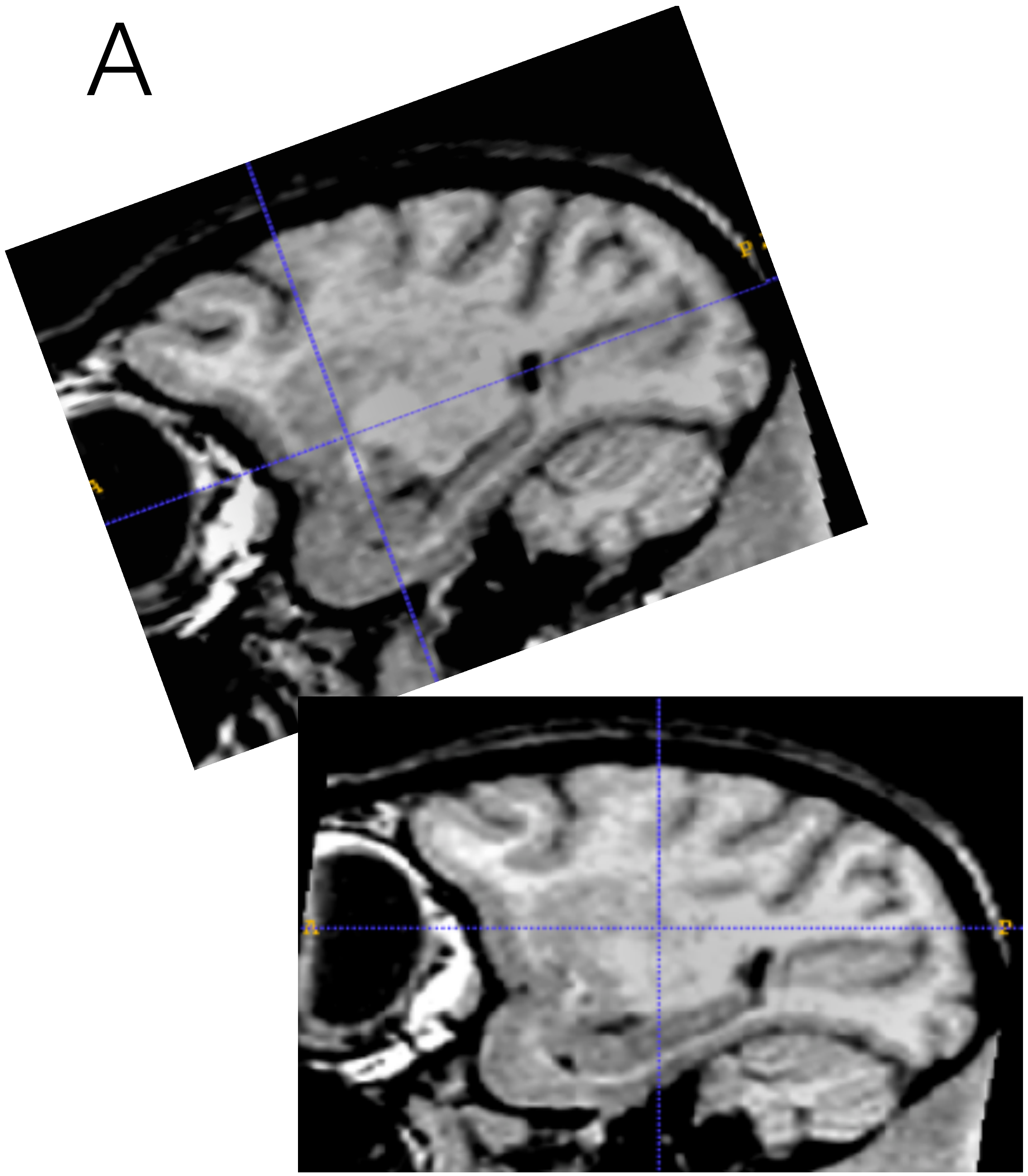


B

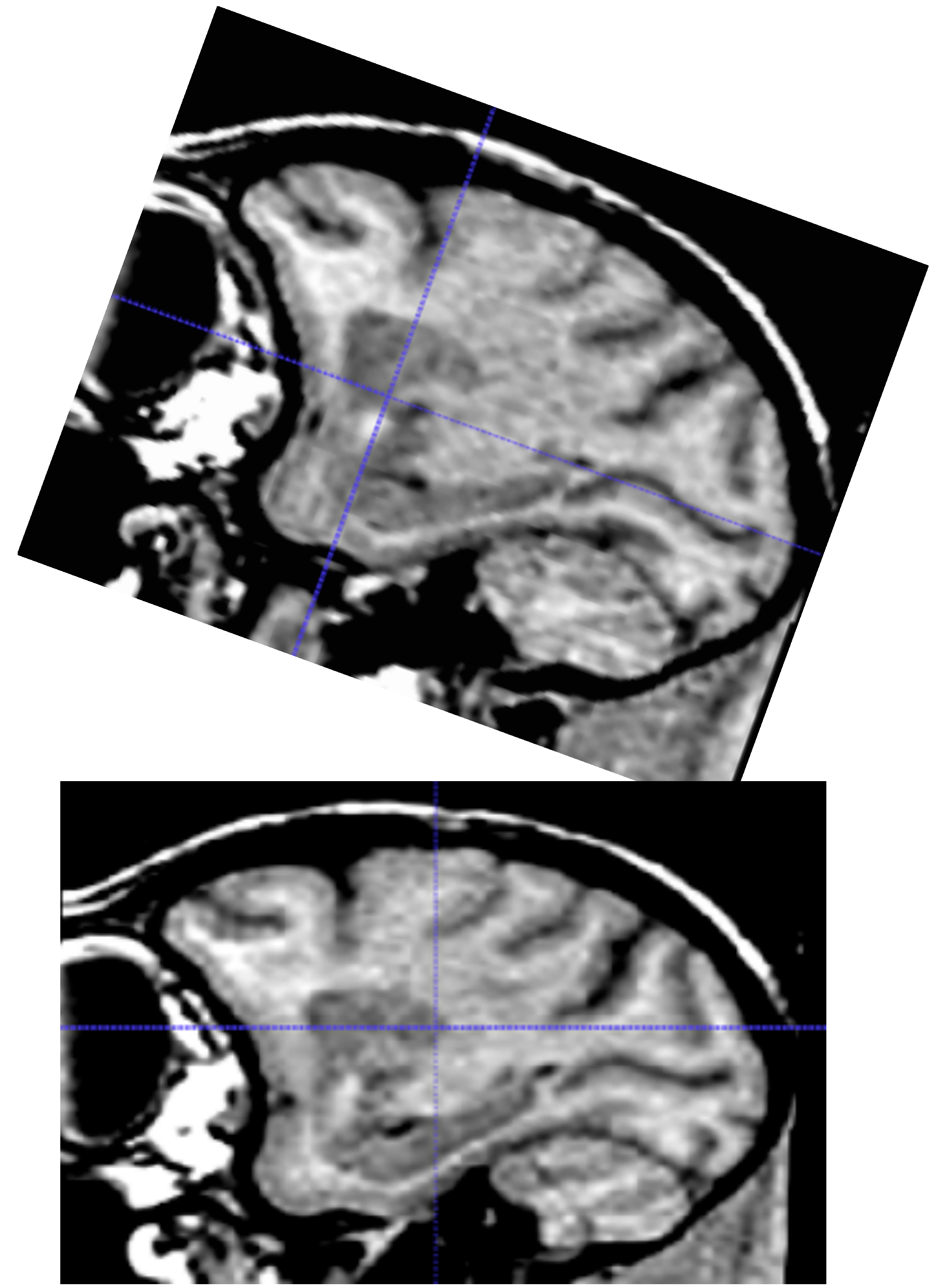


Unbiased registration

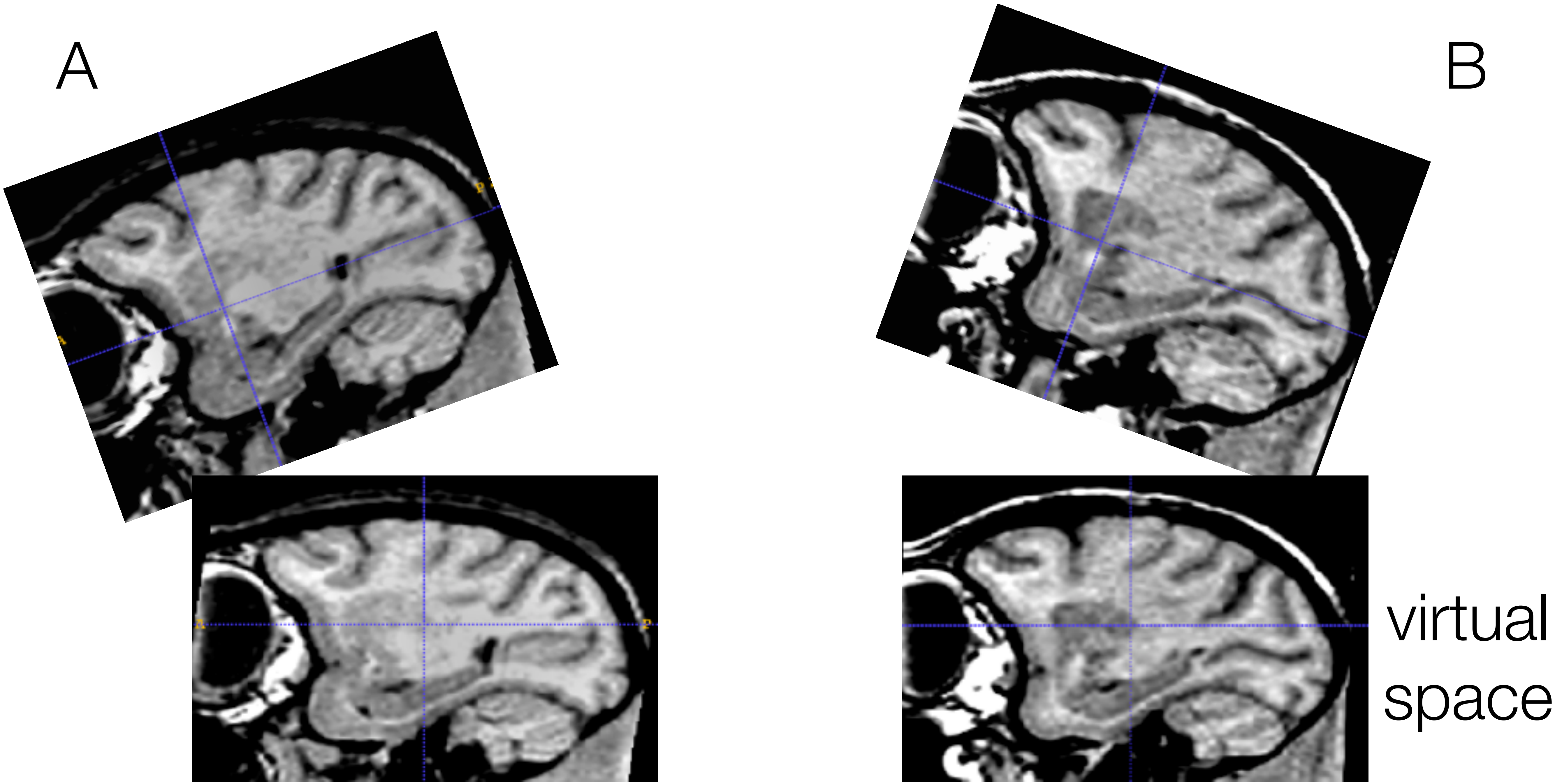
A



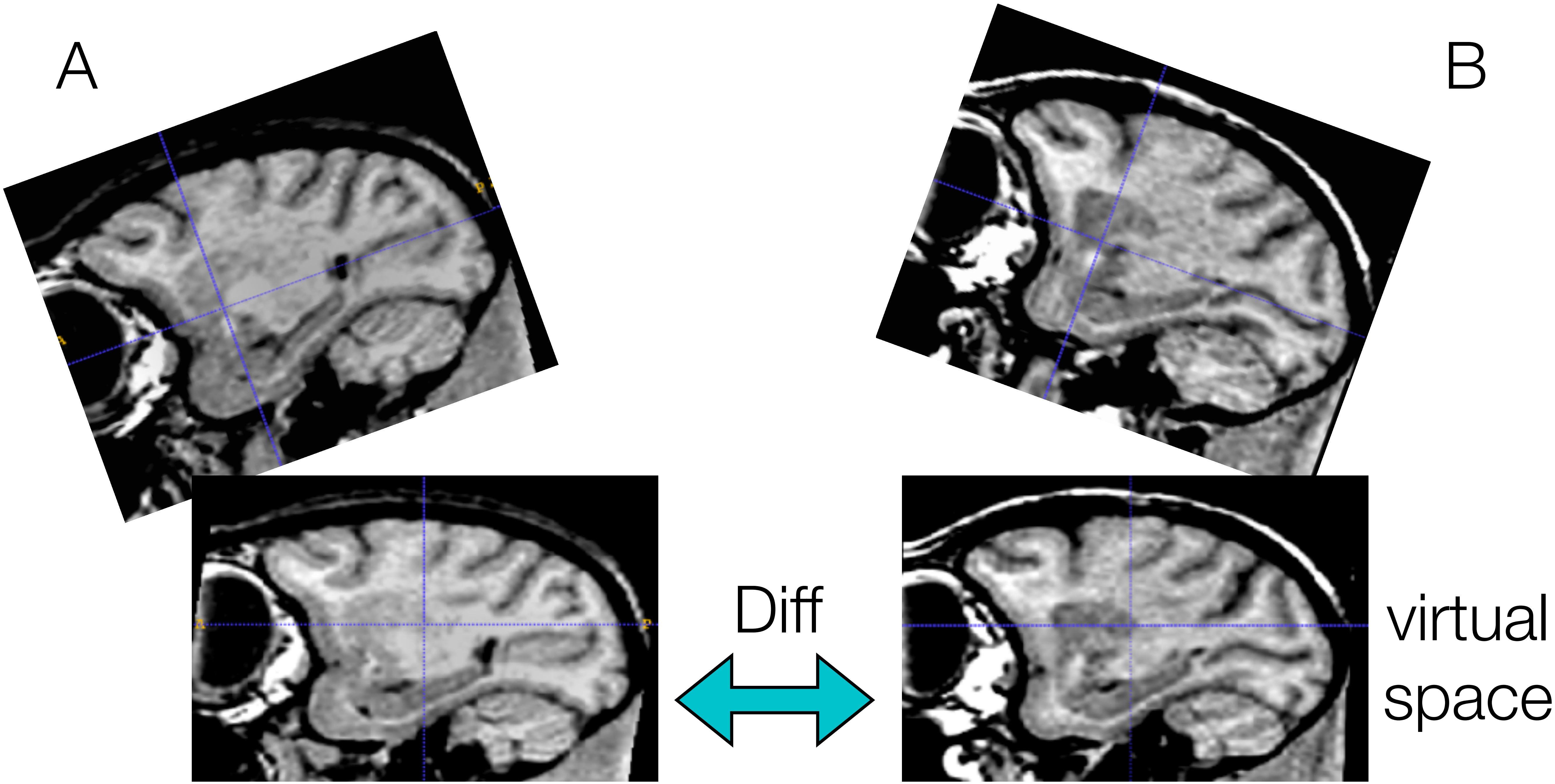
B



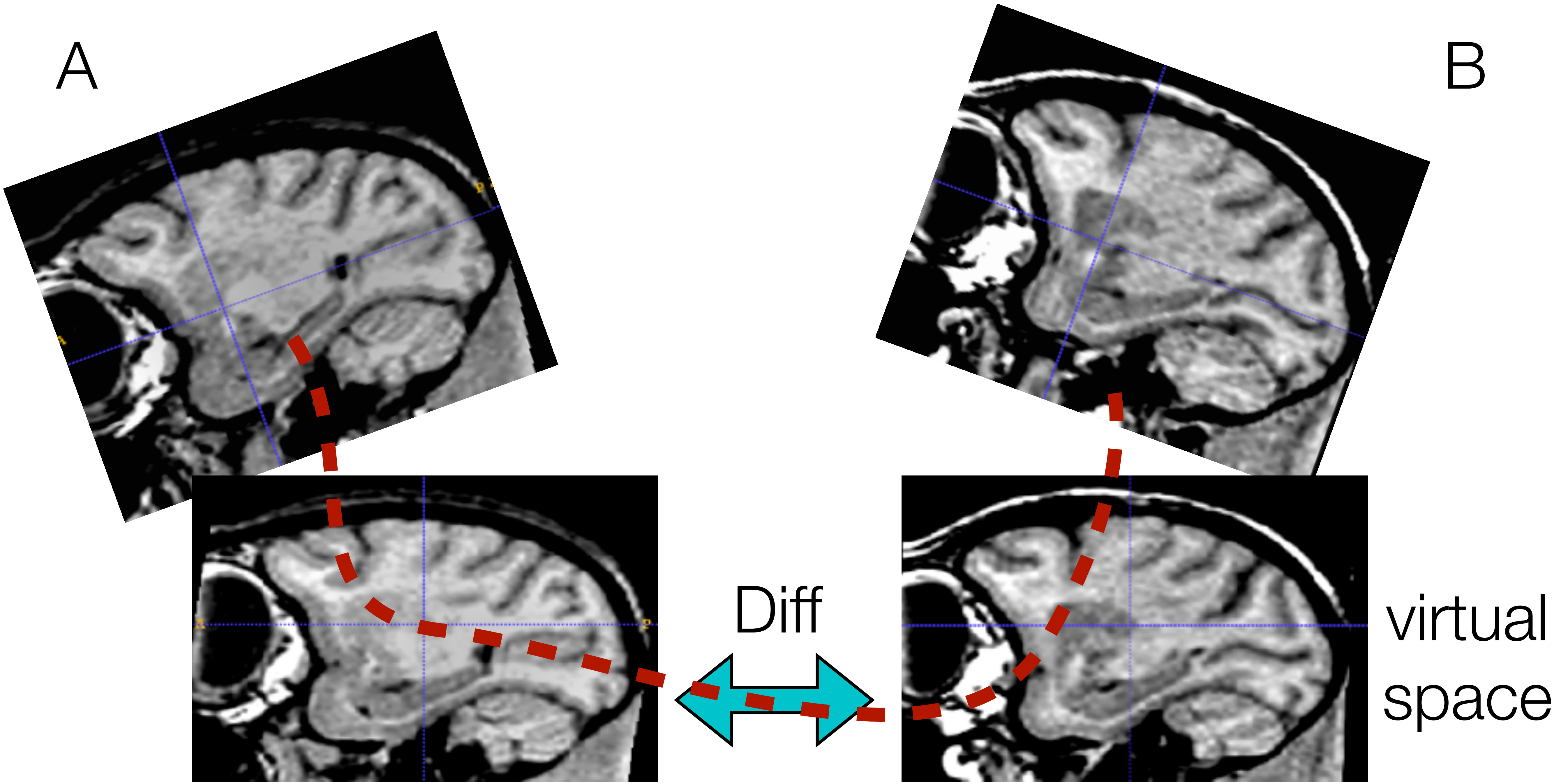
Unbiased registration



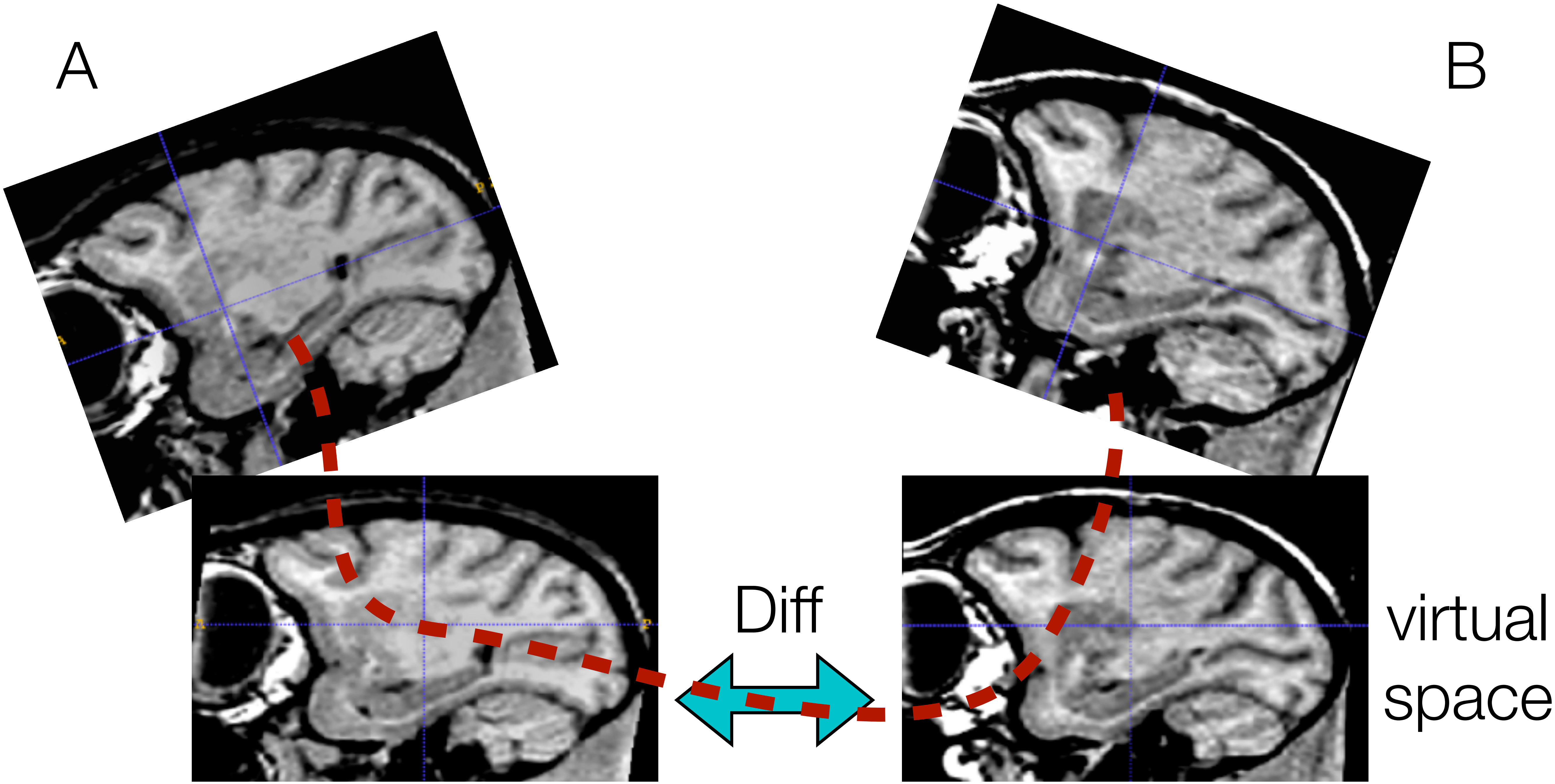
Unbiased registration



Unbiased registration

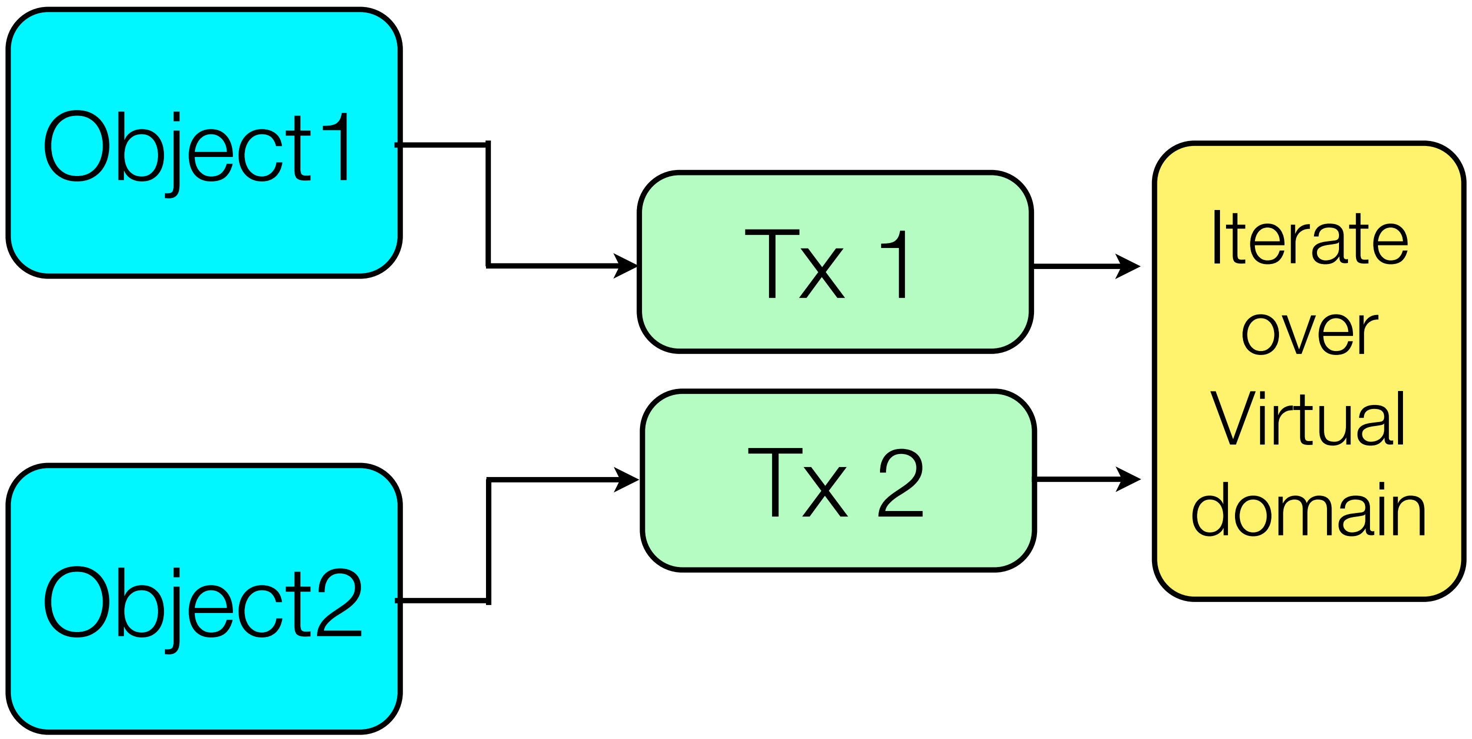


Unbiased registration



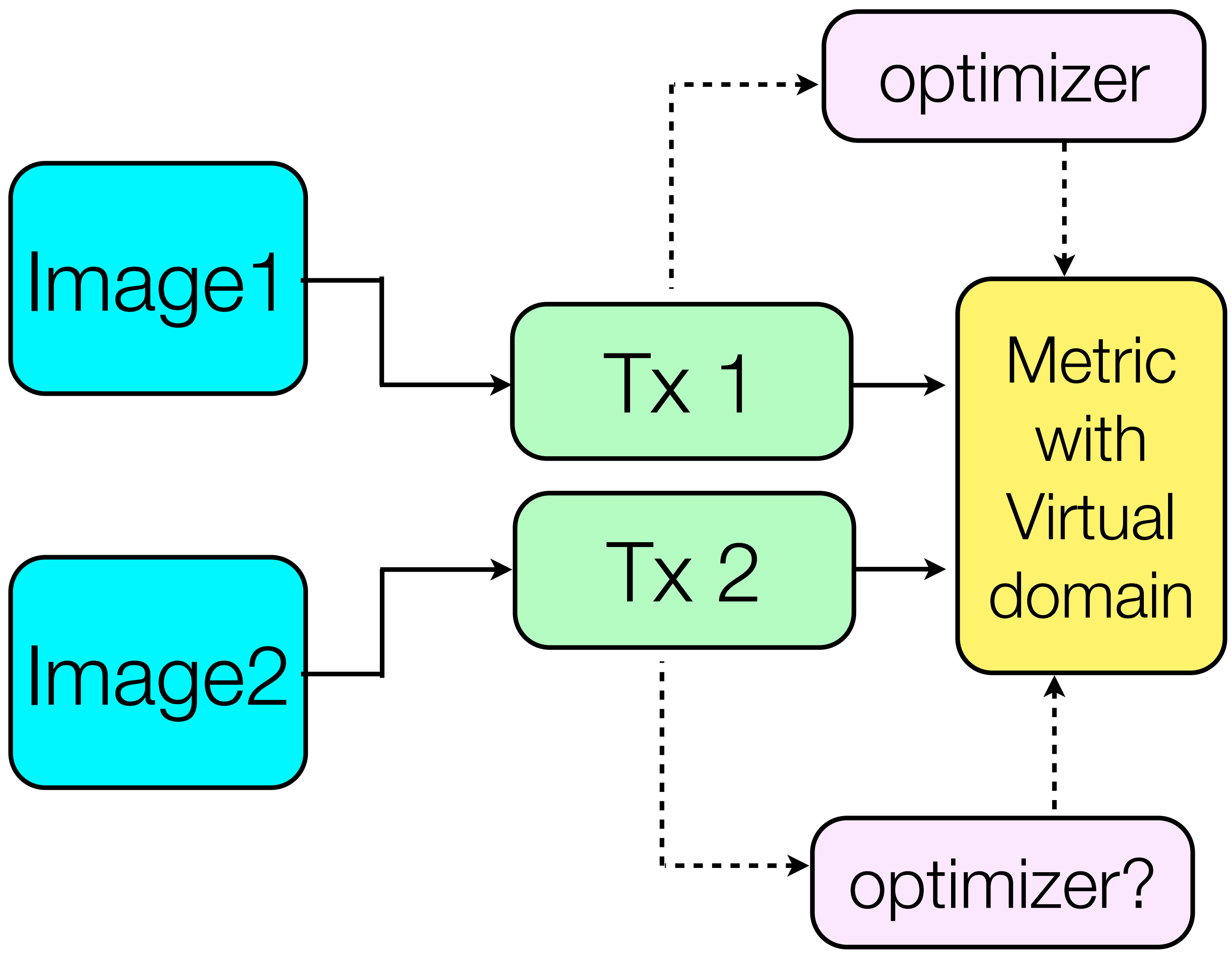
Important for longitudinal studies (Yushkevich, et al, 2010)

Object to object metric

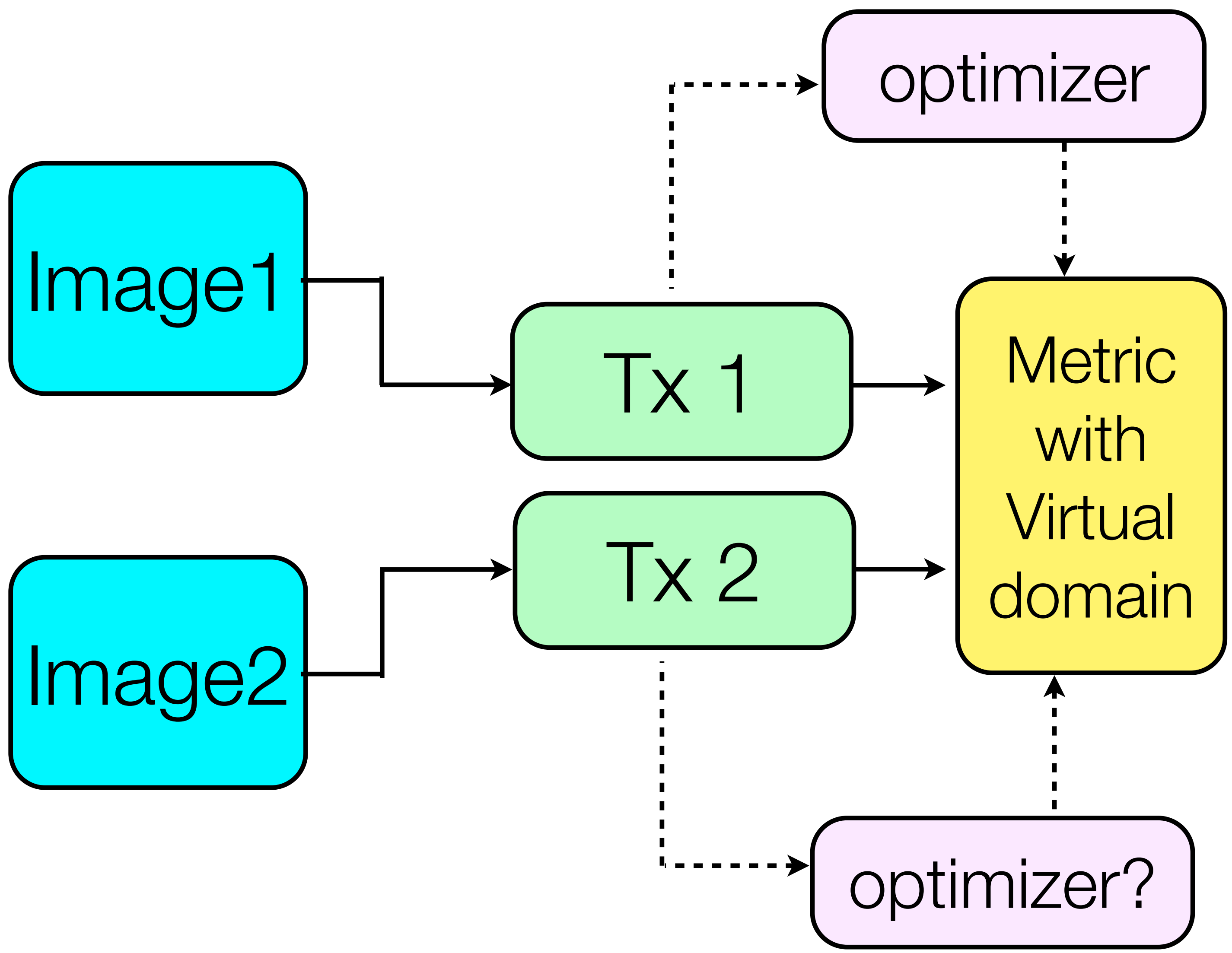


Requires a generic iterator type.

Registration machinery object

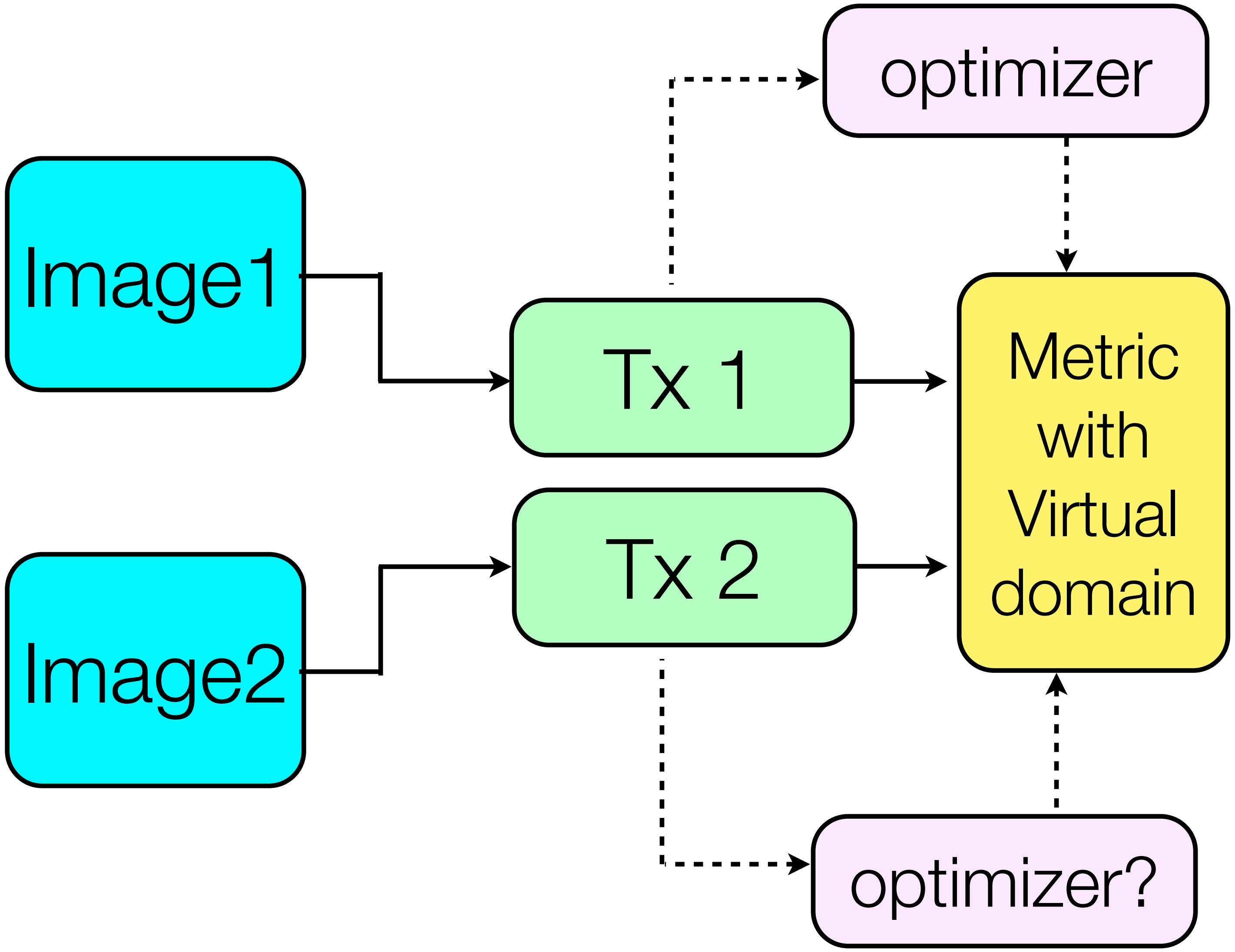


Registration machinery object



Virtual domain a critical component. Downsampling, more.

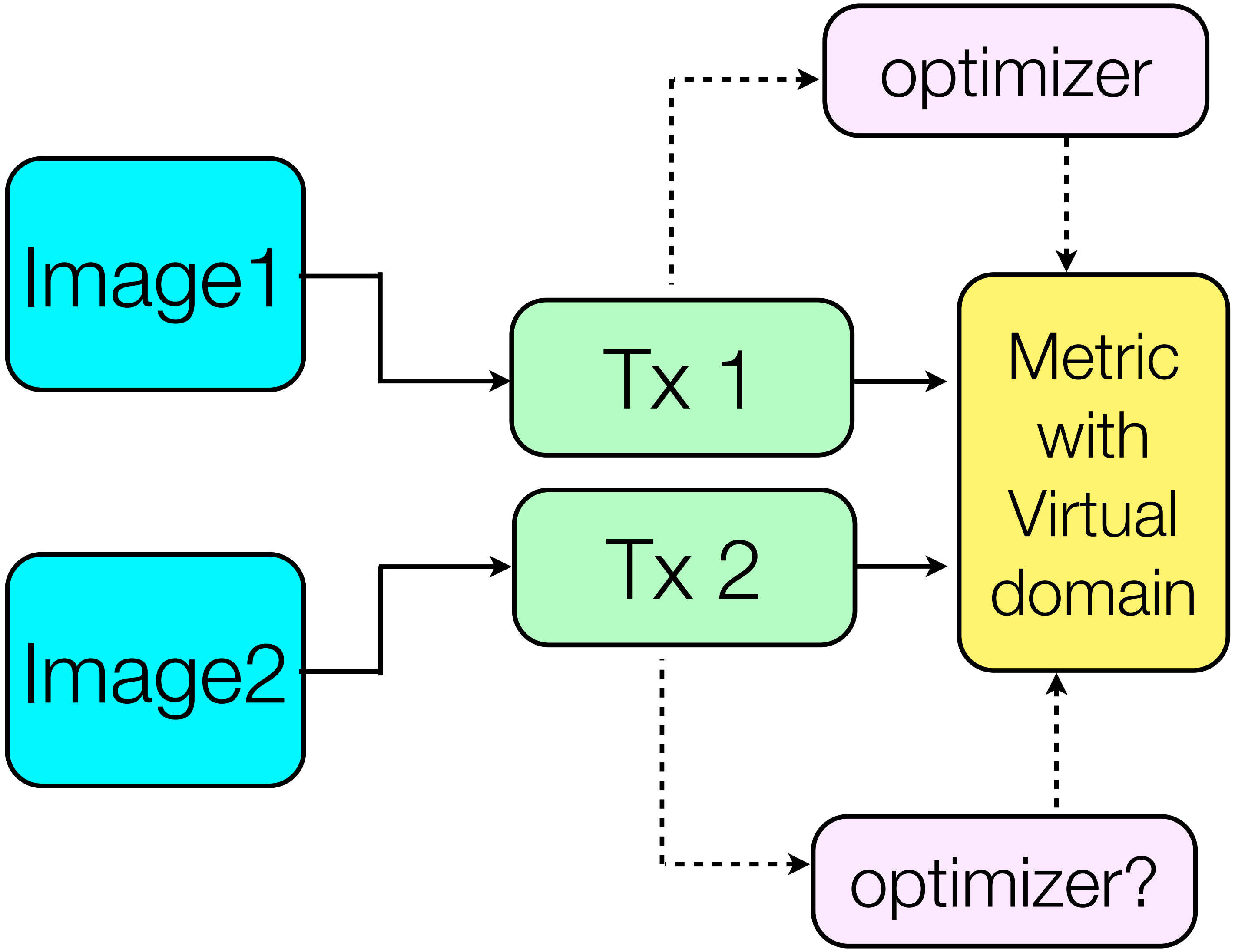
Registration machinery object



Virtual domain a critical component. Downsampling, more.

Tx1 and Tx2 are composite transforms.

Registration machinery object

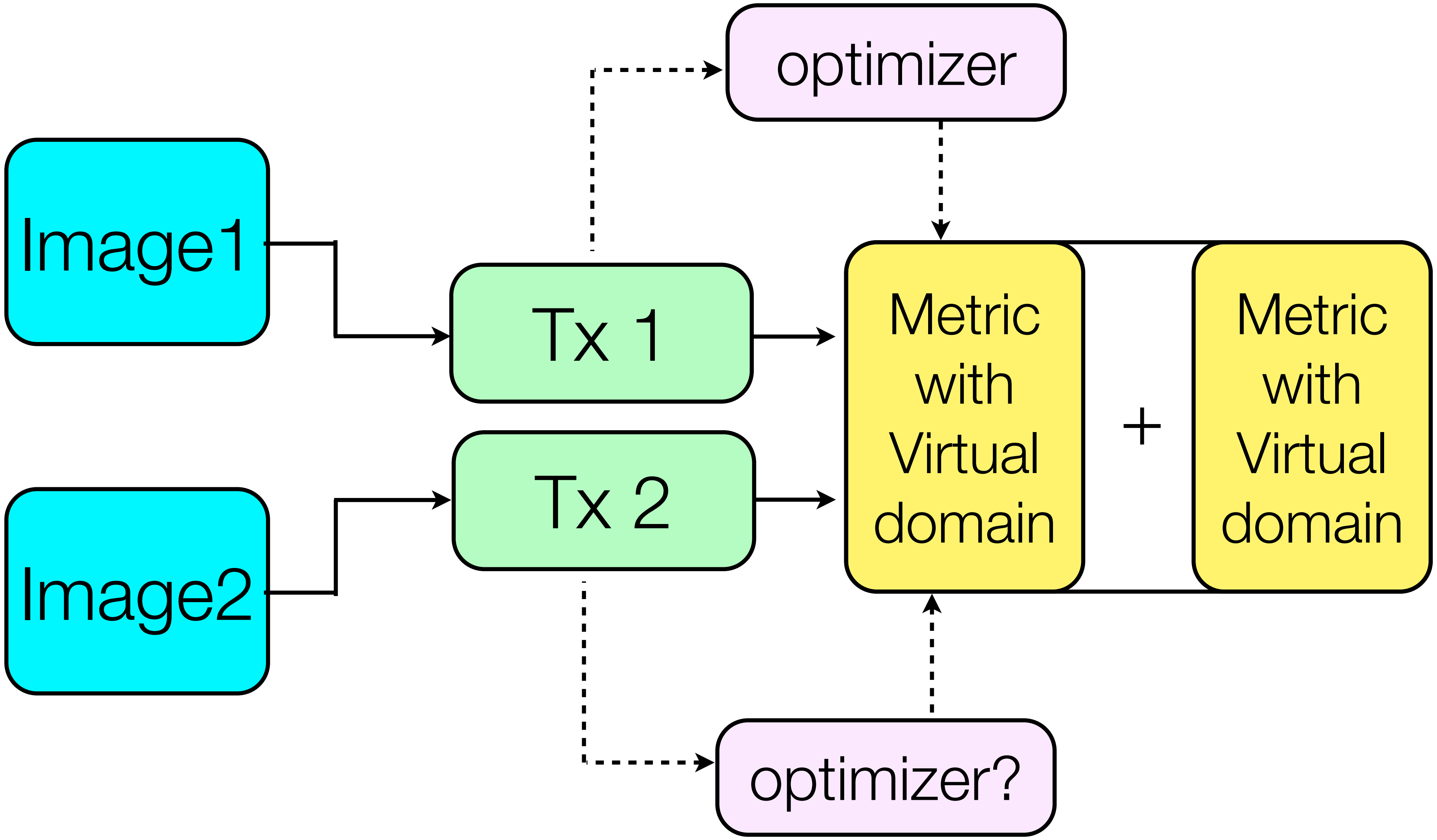


Virtual domain a critical component. Downsampling, more.

Tx1 and Tx2 are composite transforms.

Tx3 (final output) may be a composite of Tx1 and Tx2.

Registration machinery object

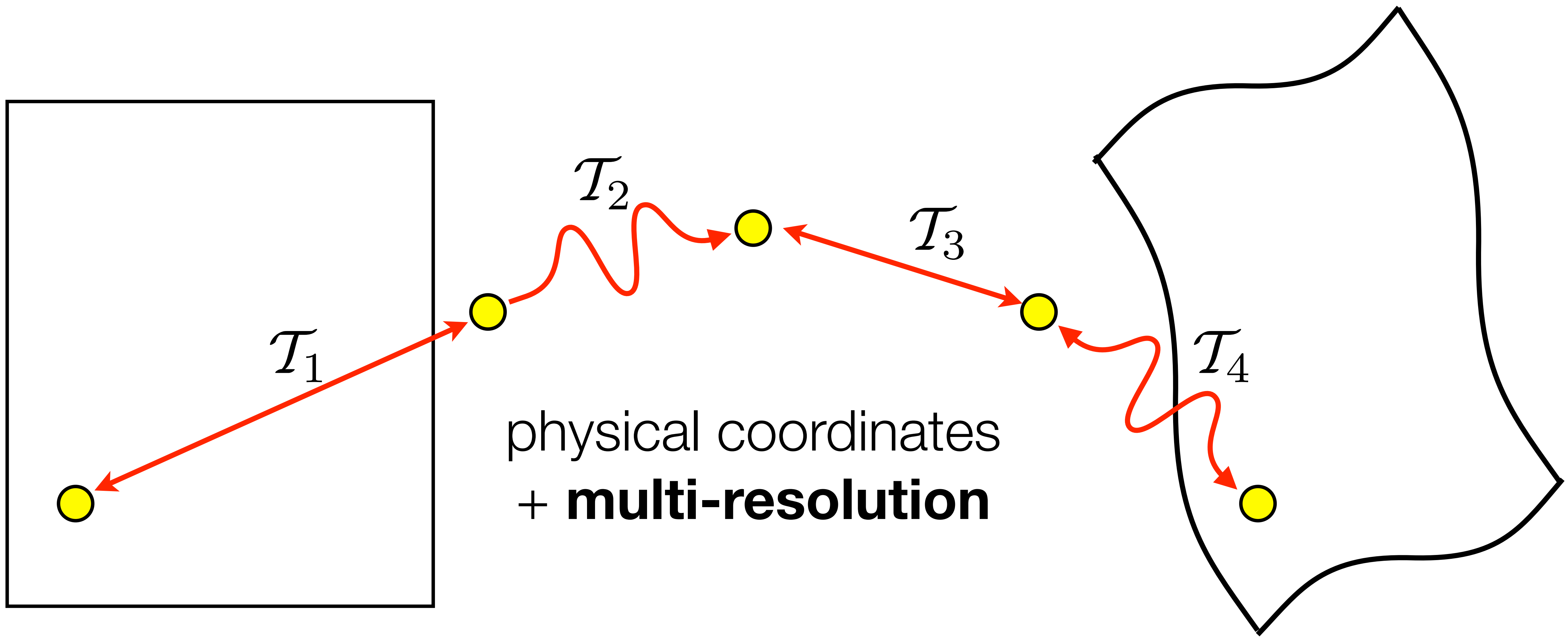


Virtual domain a critical component. Downsampling, more.

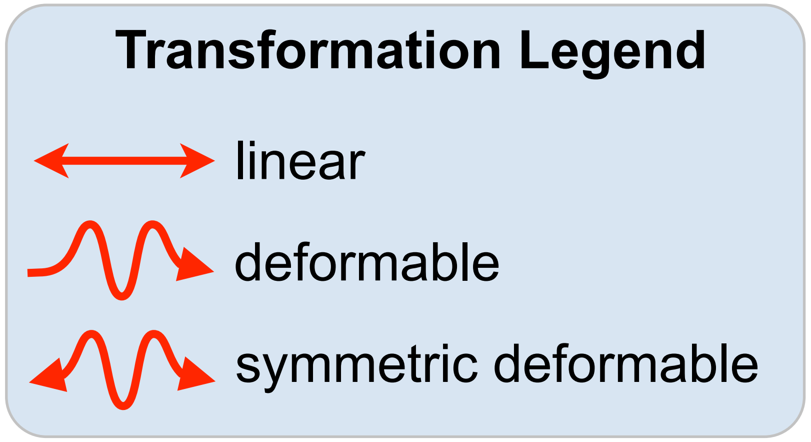
Tx1 and Tx2 are composite transforms.

Tx3 (final output) may be a composite of Tx1 and Tx2.

Composite transform



$$\mathcal{T} = \mathcal{T}_1 \circ \mathcal{T}_2 \circ \mathcal{T}_3 \circ \mathcal{T}_4$$



Accompanied by composite I/O.
Details need to be worked out.

Deformable transformation objects

Deformable transformation objects

./ITK/Code/Review/itkDeformationFieldTransform.h

Deformable transformation objects

./ITK/Code/Review/itkDeformationFieldTransform.h

./ITK/Code/Review/
itkBSplineDeformationFieldTransform.h

Deformable transformation objects

./ITK/Code/Review/itkDeformationFieldTransform.h

./ITK/Code/Review/
itkBSplineDeformationFieldTransform.h

Both can be applied through ResampleImageFilter.

Deformable transformation objects

./ITK/Code/Review/itkDeformationFieldTransform.h

./ITK/Code/Review/
itkBSplineDeformationFieldTransform.h

Both can be applied through ResampleImageFilter.
Should therefore remove WarpImageFilter

Deformable transformation objects

./ITK/Code/Review/itkDeformationFieldTransform.h

./ITK/Code/Review/
itkBSplineDeformationFieldTransform.h

Both can be applied through ResampleImageFilter.
Should therefore remove WarpImageFilter

This helps us unify the PDE and linear registration frameworks.

Registration machinery code 1

```
typedef itk::Transform<double,ImageDimension>  TransformType;
typedef itk::IdentityTransform<double,ImageDimension>  IdentityTransformType;
typedef itk::CompositeTransform<double,ImageDimension>  CompositeTransformType;
typedef itk::TranslationTransform<double,ImageDimension>  TranslationTransformType;
typedef itk::DeformationFieldTransform<double,ImageDimension>  DeformationTransformTy
typedef DeformationTransformType::DeformationFieldType FieldType;
IdentityTransformType::Pointer transformF = IdentityTransformType::New();
DeformationTransformType::Pointer transformM1 = DeformationTransformType::New();
TranslationTransformType::Pointer transformM2 = TranslationTransformType::New();
TranslationTransformType::Pointer transformM3 = TranslationTransformType::New();
CompositeTransformType::Pointer transformM = CompositeTransformType::New();
```

Registration machinery code 2

```
typedef itk::DemonsImageToImageMetric<ImageType, ImageType> ObjectMetric
ObjectMetricType::Pointer objectMetric = ObjectMetricType::New();

/** Set up the virtual reference space */
objectMetric->SetVirtualDomainSpacing(fixed_image->GetSpacing());
objectMetric->SetVirtualDomainSize(fixed_image->GetLargestPossibleRegion
objectMetric->SetVirtualDomainOrigin(fixed_image->GetOrigin());
objectMetric->SetVirtualDomainDirection(fixed_image->GetDirection());
```

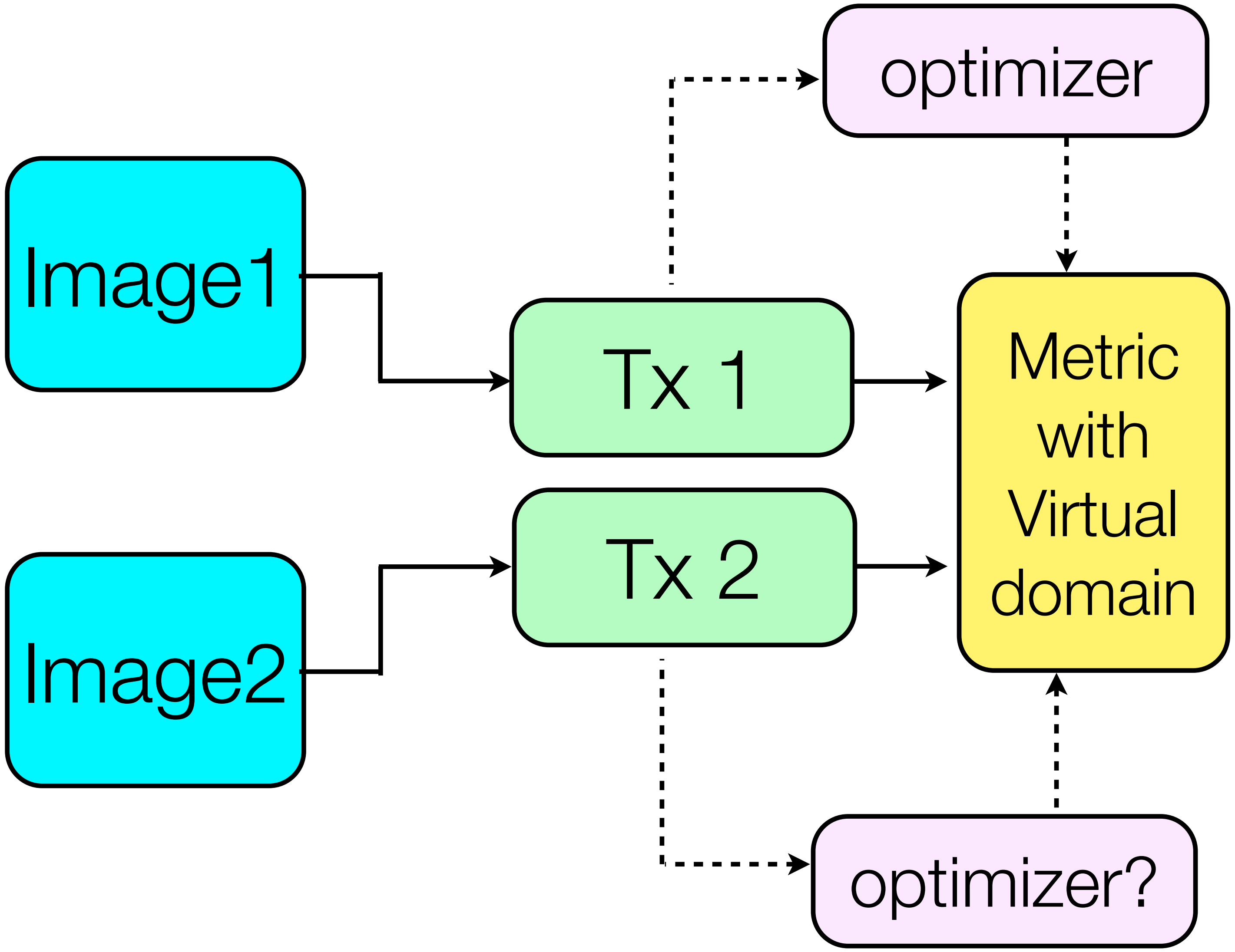
Registration machinery code 3

```
transformM1->SetDeformationField(field);
transformM1->SetInverseDeformationField(field);

transformM->AddTransform(transformM1);
transformM->AddTransform(transformM2);

/** Define the input images and their transforms */
objectMetric->SetFixedImage(fixed_image);
objectMetric->SetMovingImage(moving_image);
objectMetric->SetFixedImageTransform(transformF);
objectMetric->SetMovingImageTransform(transformM);

/** Compute one iteration of the metric */
objectMetric->ComputeMetricAndDerivative();
```

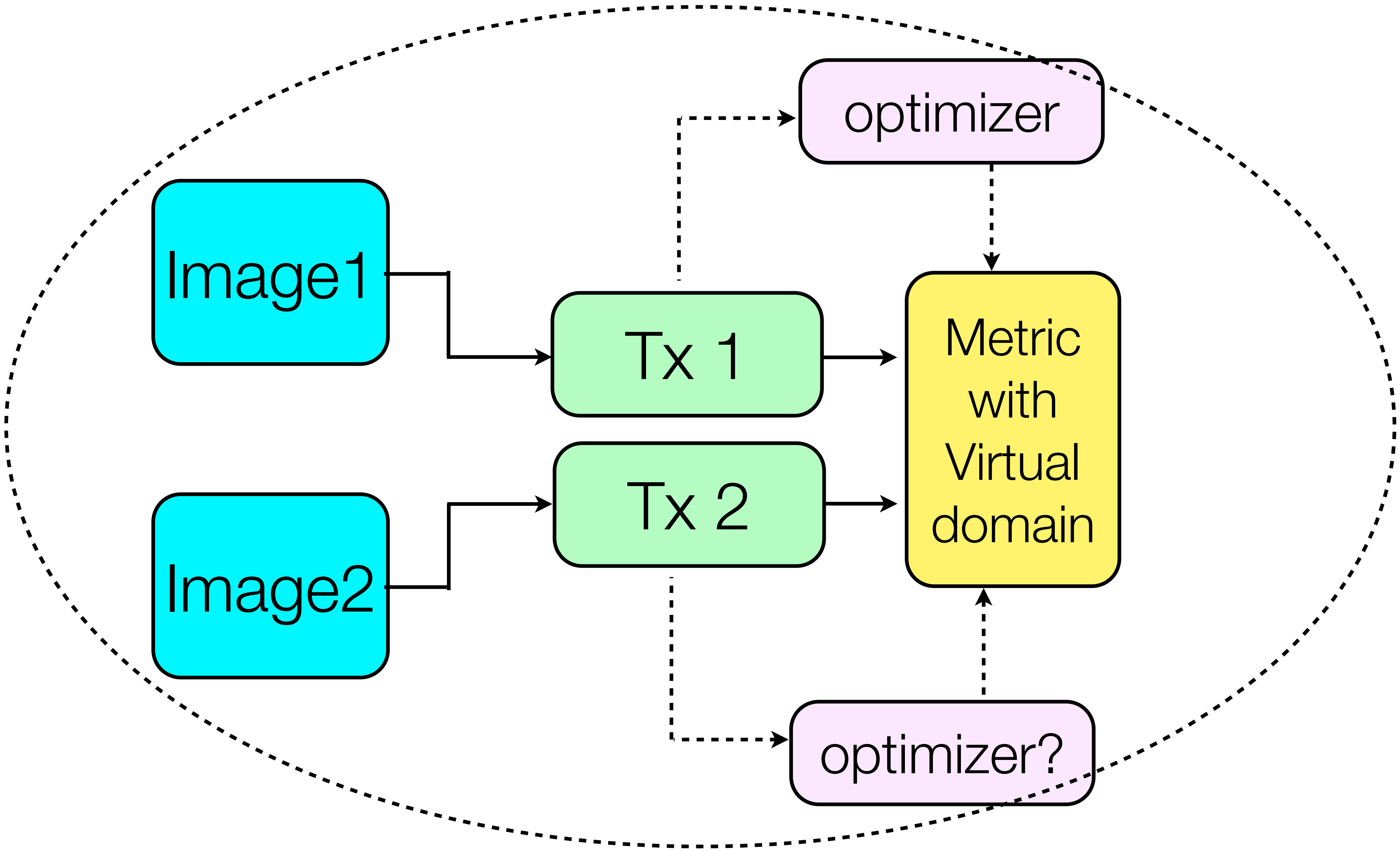
Object to object metric code

```
this->m_VirtualImage->TransformIndexToPhysicalPoint(ItV.GetIndex(),mappedPoint);

// Use generic transform to compute mapped position
mappedFixedPoint = this->m_FixedImageTransform->TransformPoint(mappedPoint);
mappedMovingPoint = this->m_MovingImageTransform->TransformPoint(mappedPoint);
if ( !this->m_FixedInterpolator->IsInsideBuffer(mappedFixedPoint) ||
     !this->m_MovingInterpolator->IsInsideBuffer(mappedMovingPoint) )
  sampleOk=false;
if ( sampleOk )
  {
    double metricval=this->ComputeLocalContributionToMetricAndDerivative(mappedFixedPoint,ma
```

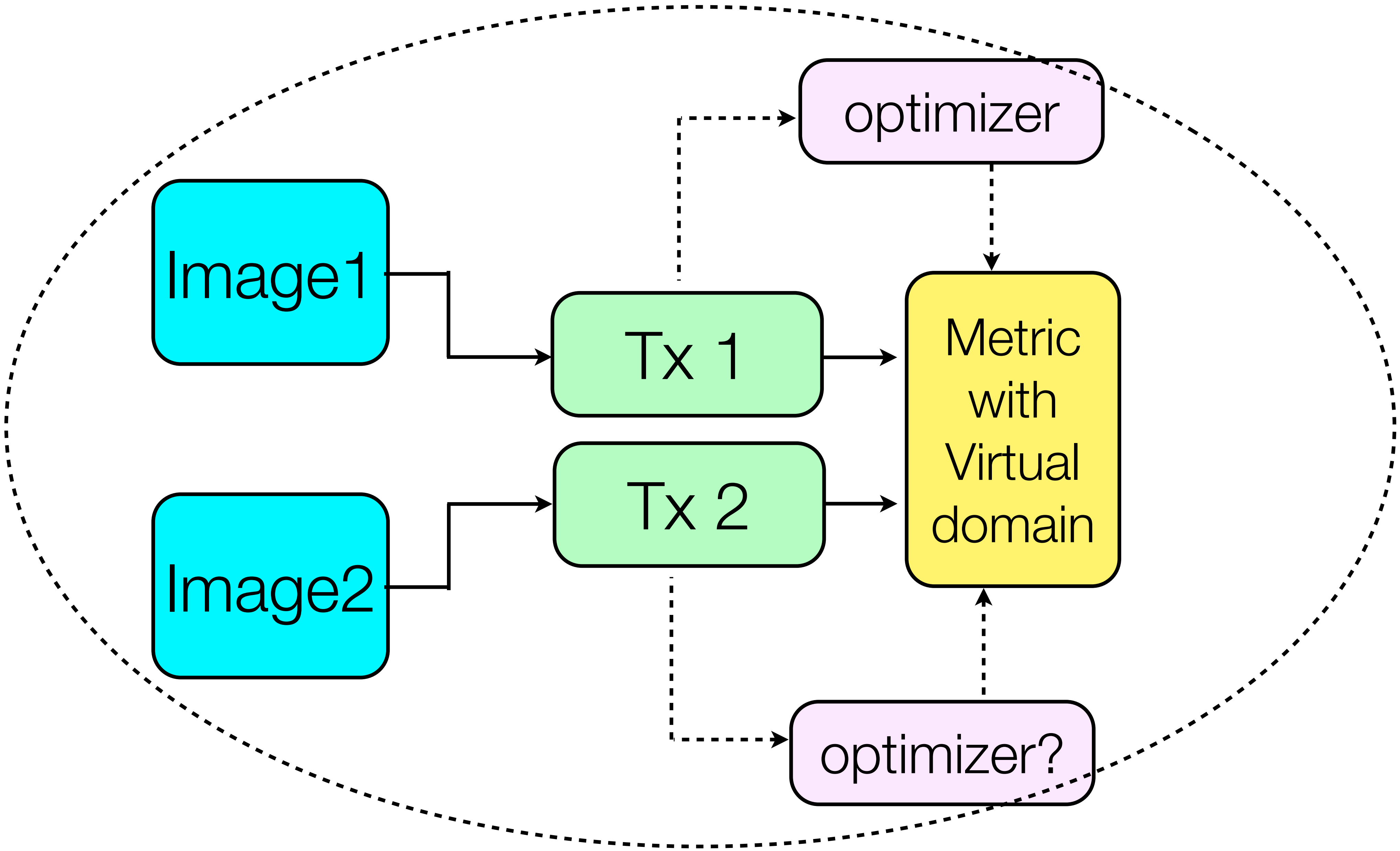
Automated parameter setting



Automated parameter setting

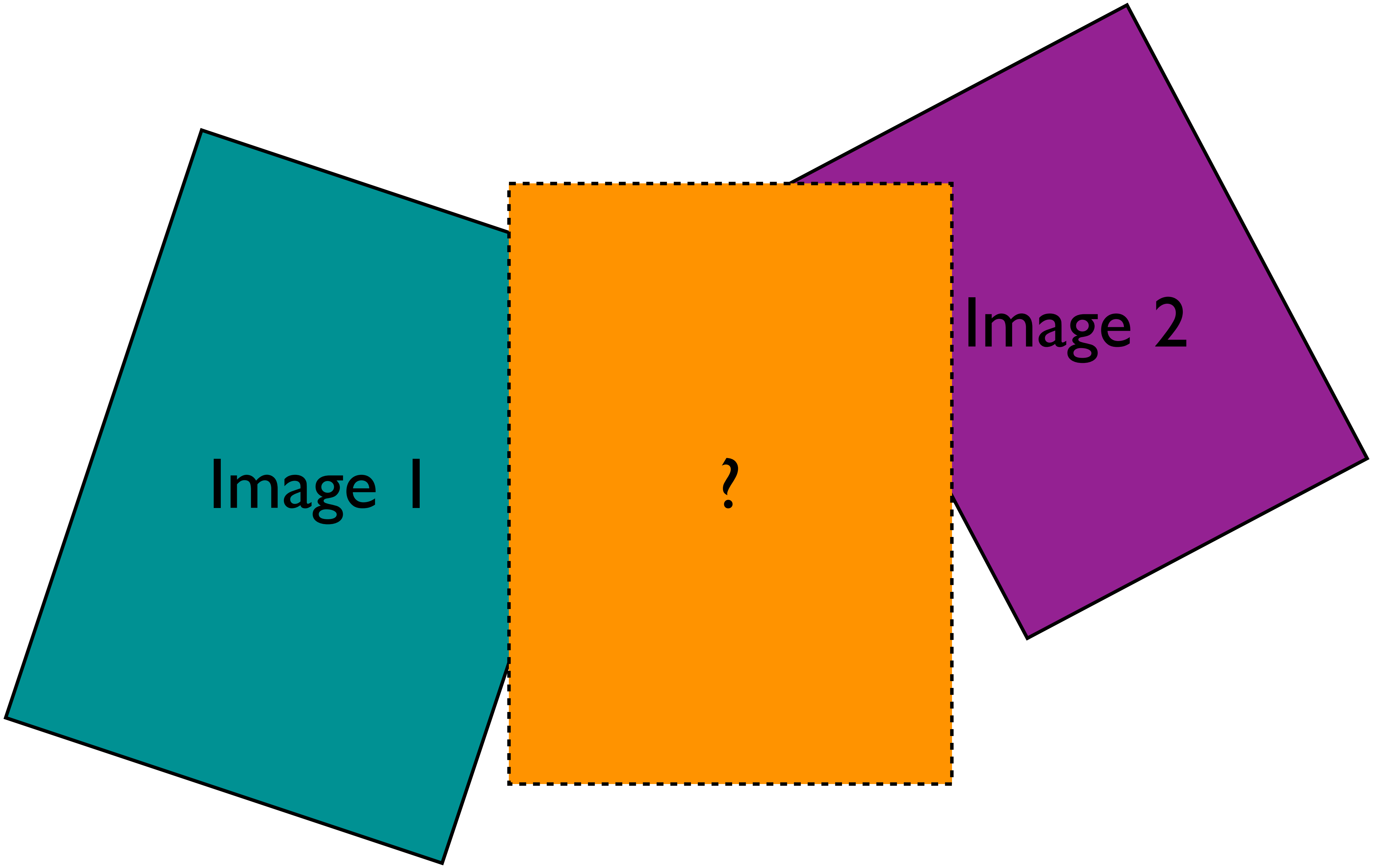


Automated parameter setting

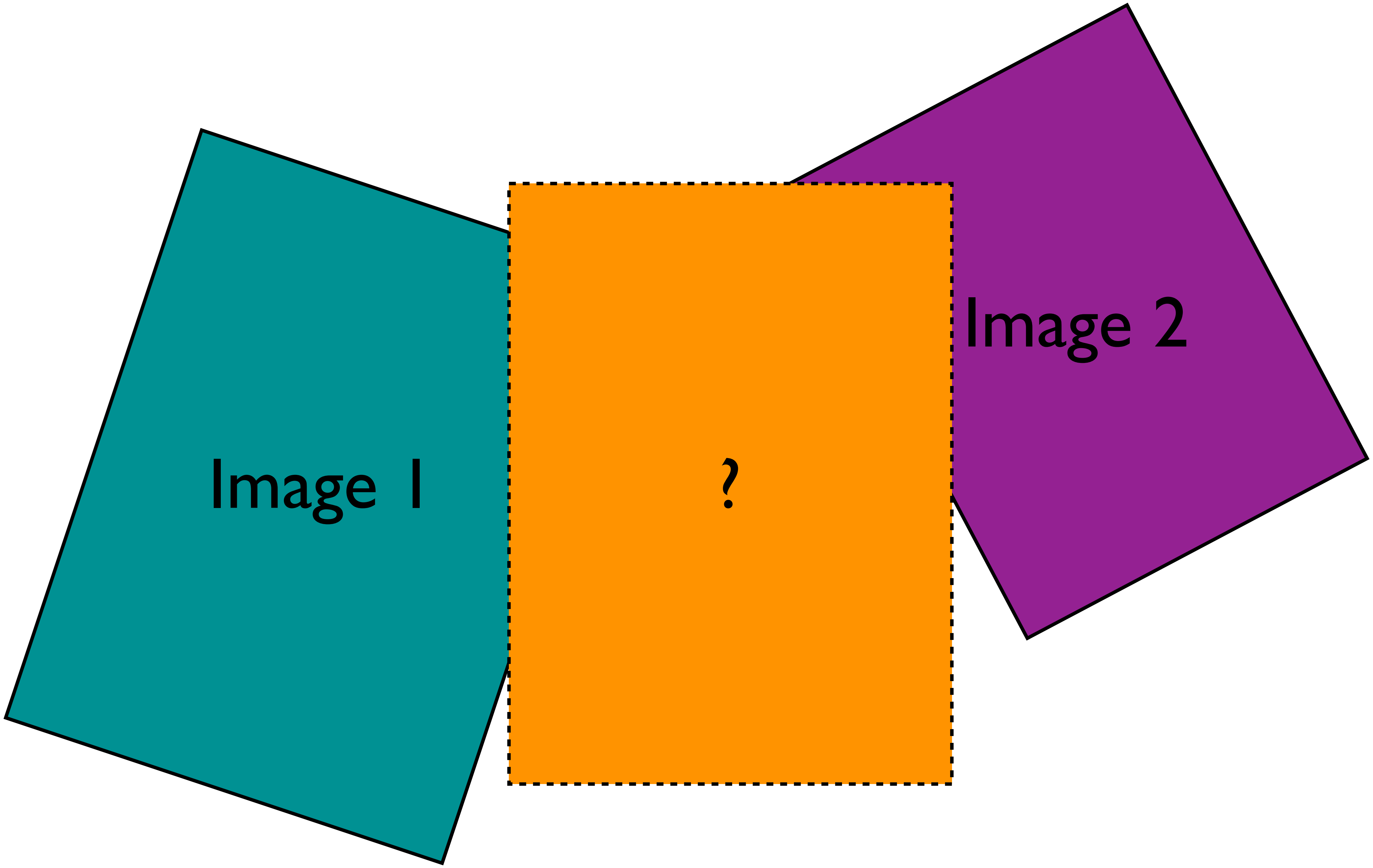


Takes the registration machinery object as input and sets the optimizer parameters according to a TBD algorithm.

Virtual space initializer

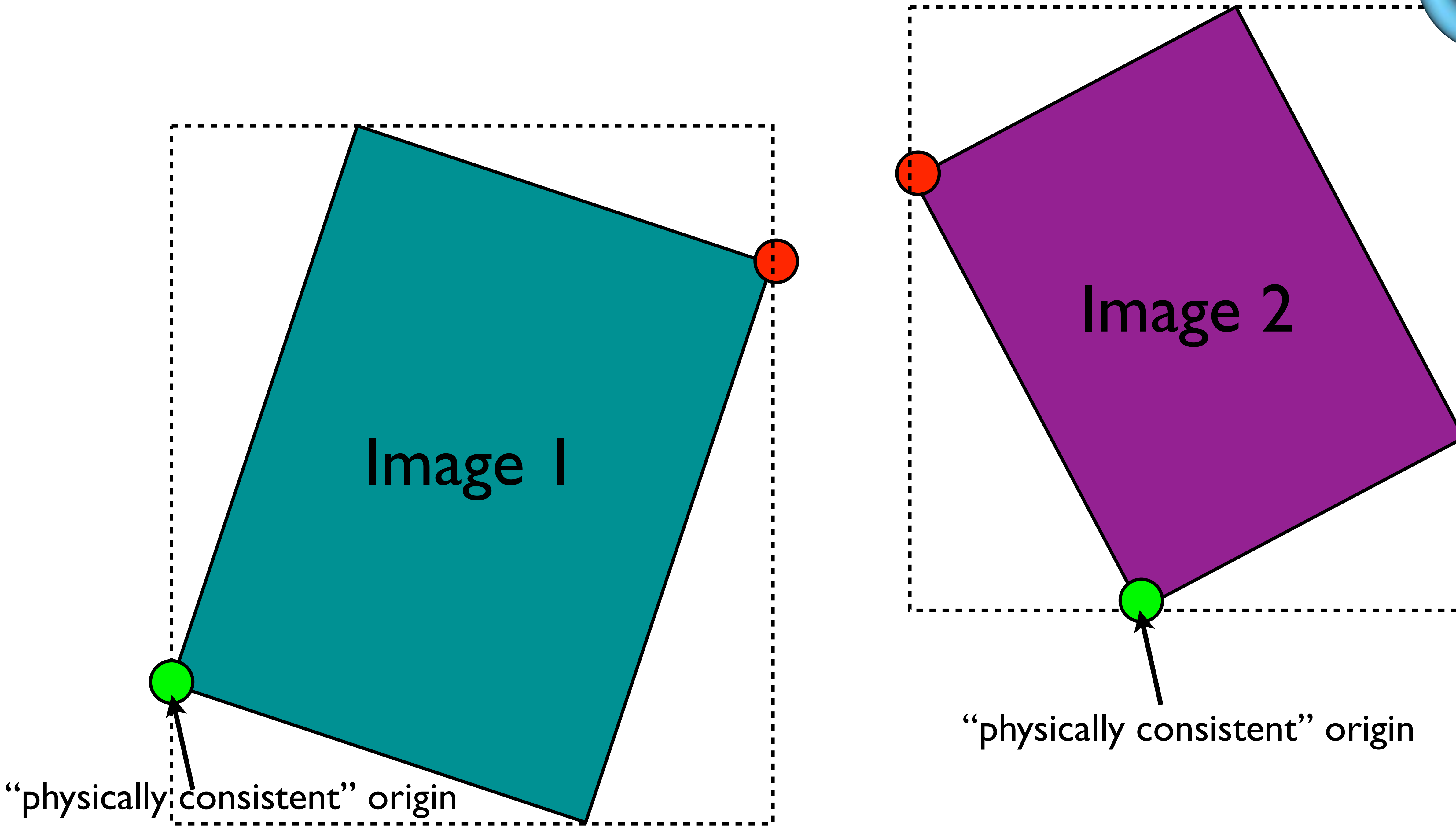


Virtual space initializer

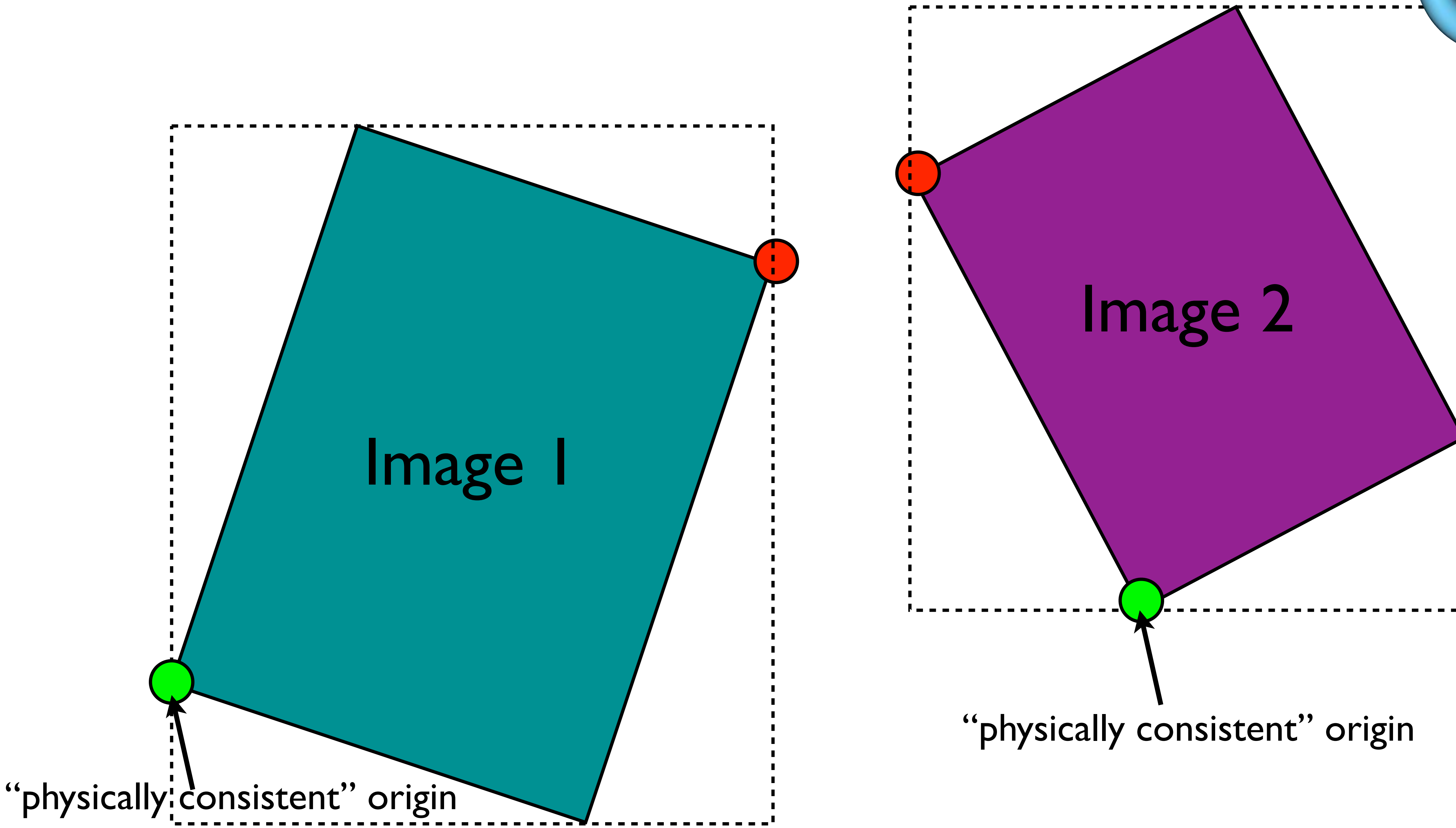


what is the best “average domain”?

Virtual space initializer

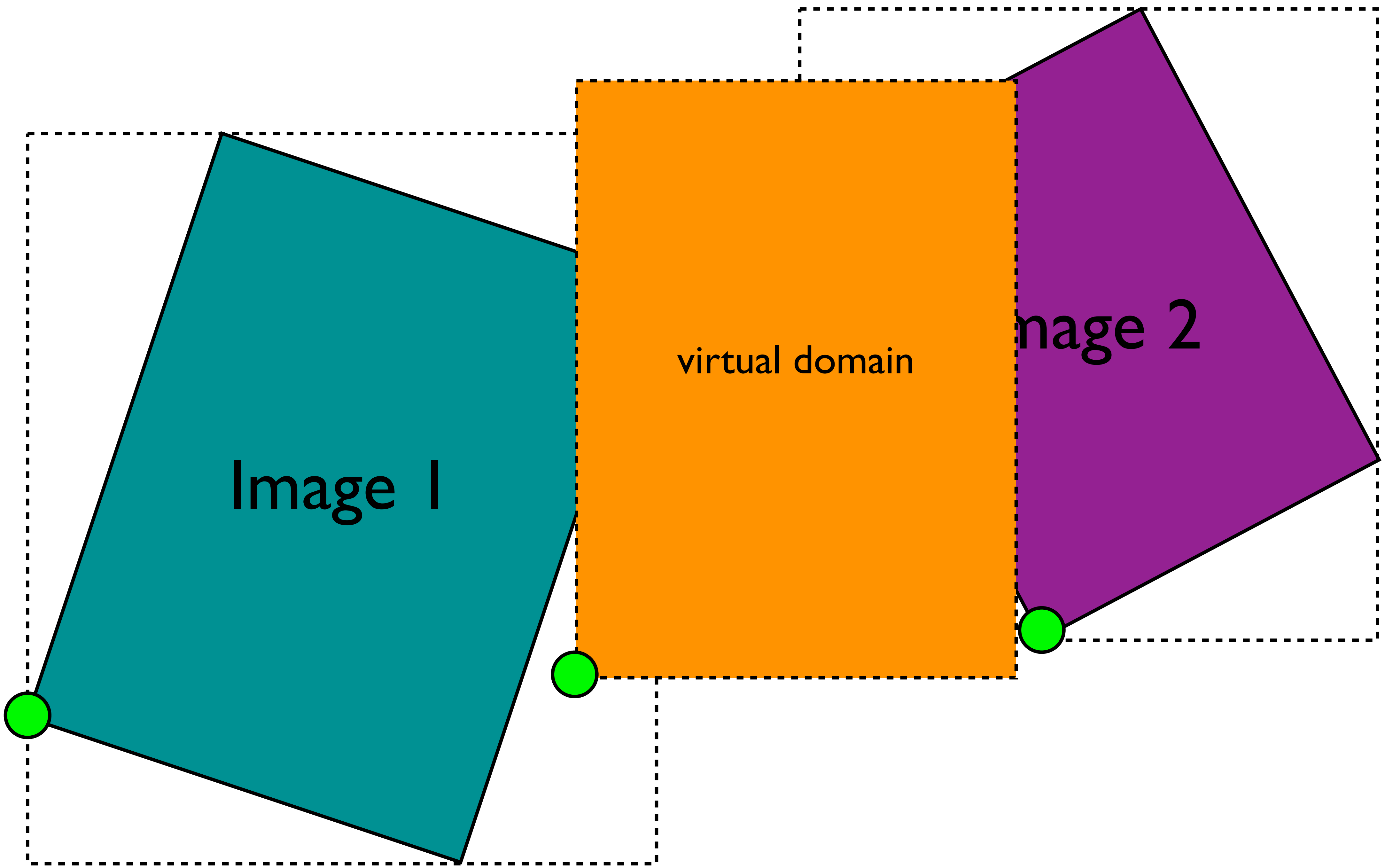


Virtual space initializer

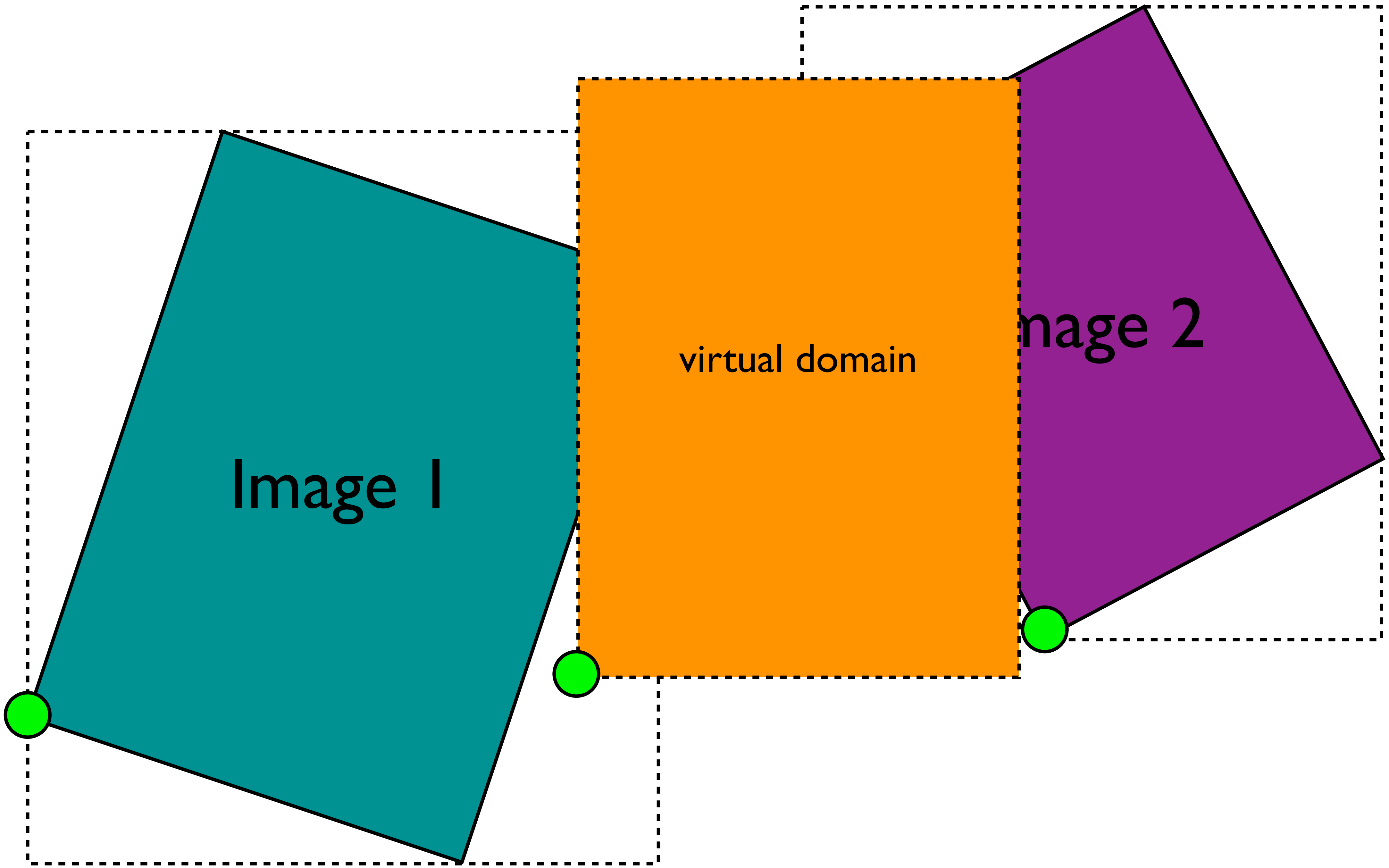


what is the best “average domain”?

Virtual space initializer



Virtual space initializer



what is the best “average domain”?

Changes to transform + metric

Changes to transform + metric

GetLocalJacobian

Changes to transform + metric

GetLocalJacobian

GetLocalContributionToMetricAndDerivative

Changes to transform + metric

GetLocalJacobian

GetLocalContributionToMetricAndDerivative

Revised framework supports dense metric computation with or without use of Jacobian

Changes to transform + metric

GetLocalJacobian

GetLocalContributionToMetricAndDerivative

Revised framework supports dense metric computation with or without use of Jacobian

Critical for efficient deformable registration

Neighborhood correlation metric

Neighborhood correlation metric

Sliding window implementation of both metric and derivative.

Neighborhood correlation metric

Sliding window implementation of both metric and derivative.

Users sets the window radius.

Neighborhood correlation metric

Sliding window implementation of both metric and derivative.

Users sets the window radius.

Multi-threaded. MI, MSQ, Global CC goals. NMI?

Neighborhood correlation metric

Sliding window implementation of both metric and derivative.

Users sets the window radius.

Multi-threaded. MI, MSQ, Global CC goals. NMI?

Is currently on Github.

Neighborhood correlation metric

Sliding window implementation of both metric and derivative.

Users sets the window radius.

Multi-threaded. MI, MSQ, Global CC goals. NMI?

Is currently on Github.

Independent metric development needs to be unified.

Transducer function (@ github)

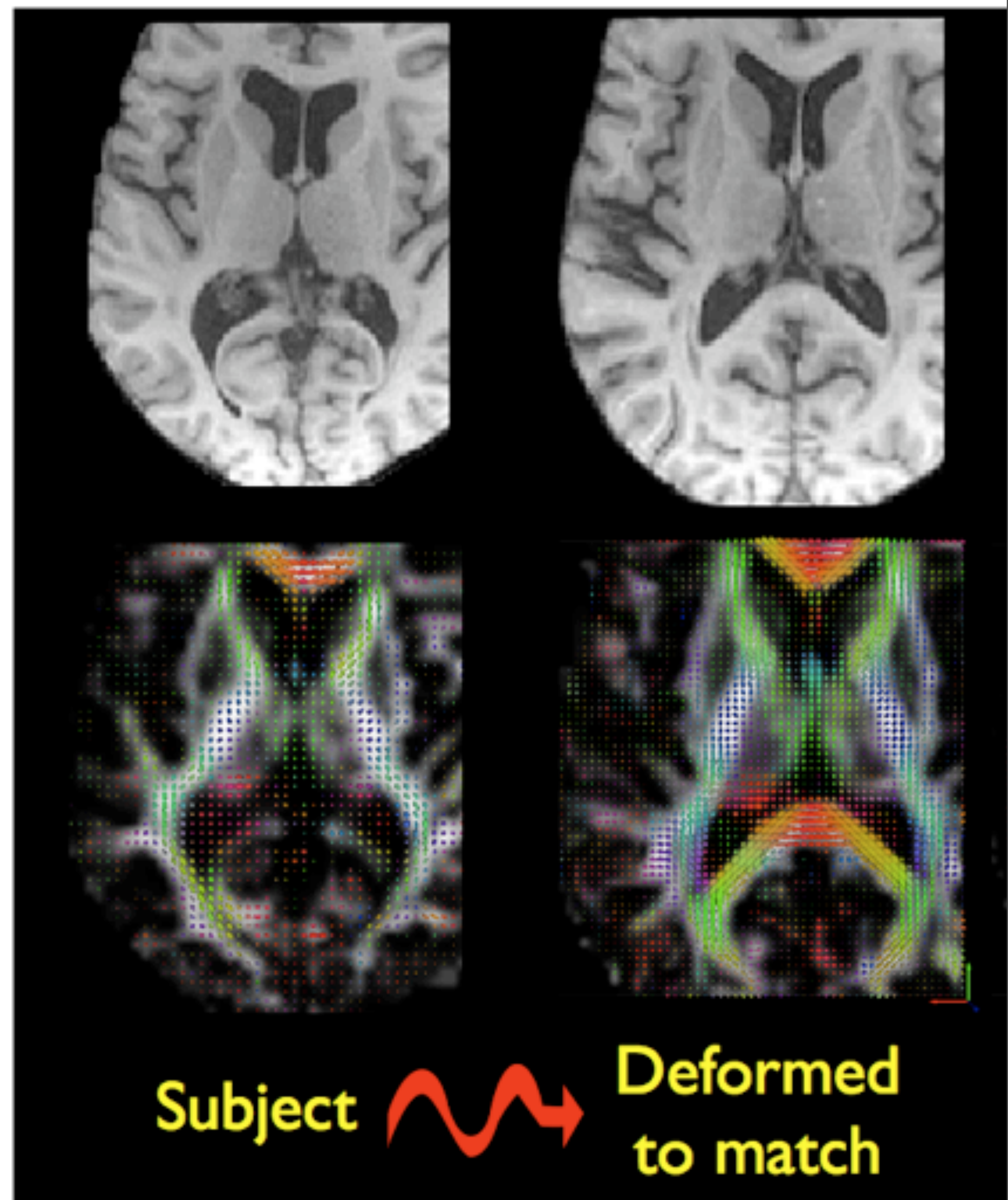
$$E(\phi, \mathbf{I}, \mathbf{J}) = \sum_i \lambda_i S_i(\phi, I_i, J_i)$$

$$\phi \in Diff_0$$

$$S_i \in \{\ MI,\ CC,\ \|\cdot\|^2,$$

$$PSE,\ \|\cdot\|_{Dev},\ JTB\ \}$$

$$\phi T = (D\phi)T(\phi)$$



Transducer function (@ github)

Add to the resample image filter.

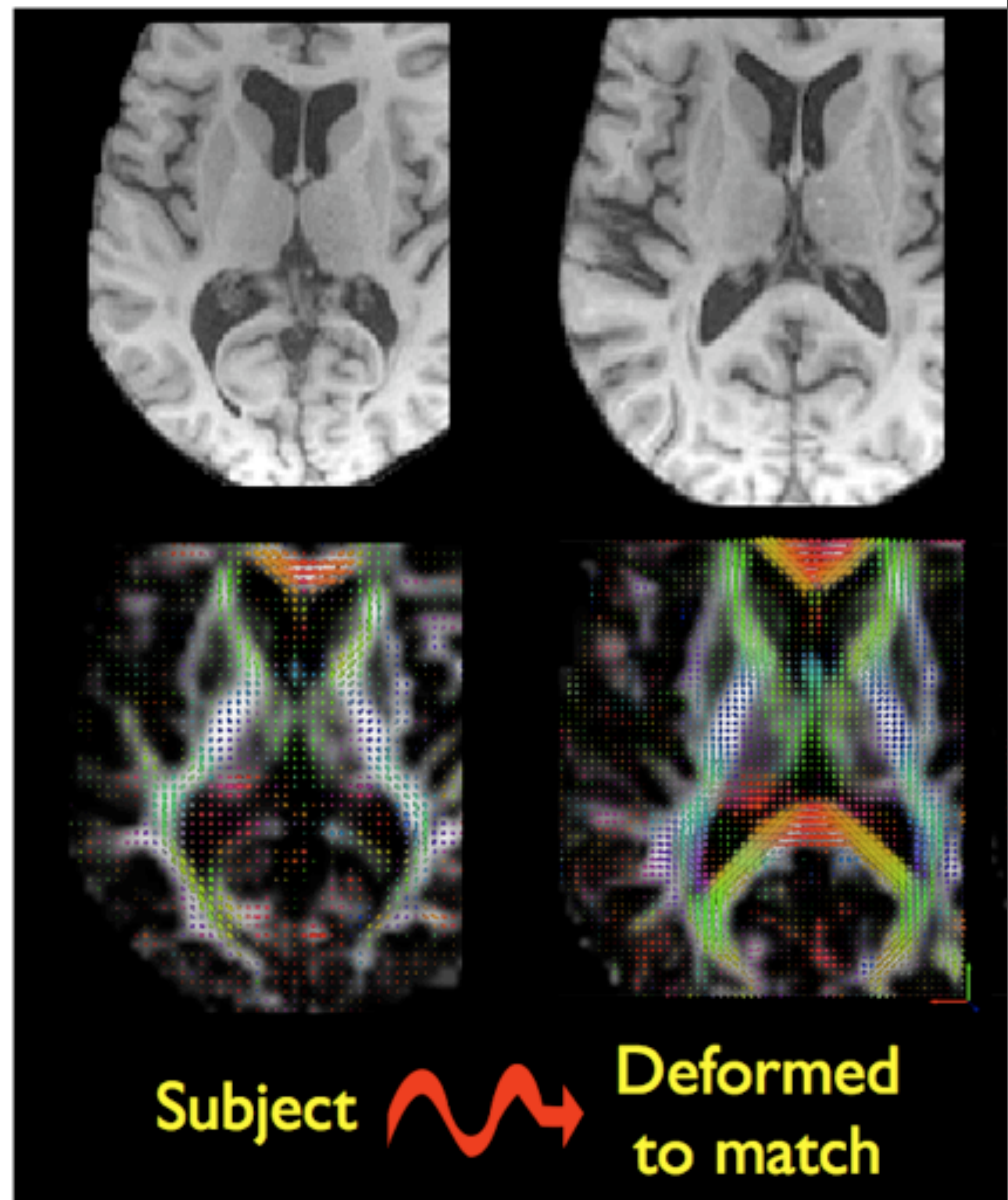
$$E(\phi, \mathbf{I}, \mathbf{J}) = \sum_i \lambda_i S_i(\phi, I_i, J_i)$$

$$\phi \in Diff_0$$

$$S_i \in \{\ MI,\ CC,\ \|\cdot\|^2,$$

$$PSE,\ \|\cdot\|_{Dev},\ JTB\ \}$$

$$\phi T = (D\phi)T(\phi)$$



Transducer function (@ github)

Add to the resample image filter.
Need to make efficient for the case when Transducer=Identity

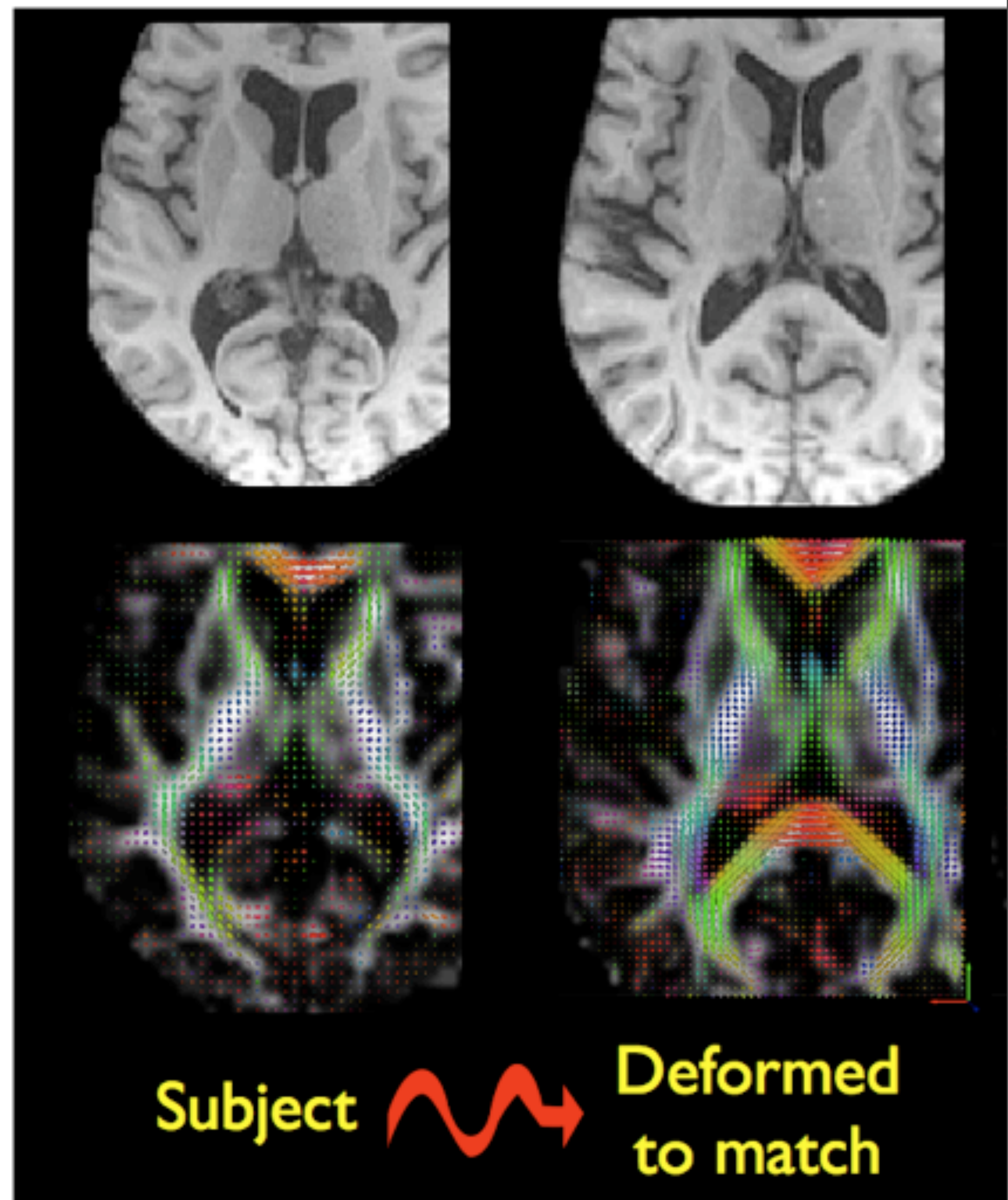
$$E(\phi, \mathbf{I}, \mathbf{J}) = \sum_i \lambda_i S_i(\phi, I_i, J_i)$$

$$\phi \in Diff_0$$

$$S_i \in \{\ MI,\ CC,\ \|\cdot\|^2,$$

$$PSE,\ \|\cdot\|_{Dev},\ JTB\ \}$$

$$\phi T = (D\phi) T(\phi)$$



B-splines in the Insight Toolkit

B-splines in the Insight Toolkit

A couple classes to discuss:

B-splines in the Insight Toolkit

A couple classes to discuss:

- BSplineScatteredData

B-splines in the Insight Toolkit

A couple classes to discuss:

- BSplineScatteredData
  - nick cleverly sped this up

B-splines in the Insight Toolkit

A couple classes to discuss:

- BSplineScatteredData
  - nick cleverly sped this up
  - core machinery for all BSpline derived work

B-splines in the Insight Toolkit

A couple classes to discuss:

- BSplineScatteredData
  - nick cleverly sped this up
  - core machinery for all BSpline derived work
- BSplineDeformableTransform

B-splines in the Insight Toolkit

A couple classes to discuss:

- BSplineScatteredData
  - nick cleverly sped this up
  - core machinery for all BSpline derived work
- BSplineDeformableTransform
  - this is a revision of the old class

B-splines in the Insight Toolkit

A couple classes to discuss:

- BSplineScatteredData
  - nick cleverly sped this up
  - core machinery for all BSpline derived work
- BSplineDeformableTransform
  - this is a revision of the old class
  - took out the bulk transform

B-splines in the Insight Toolkit

A couple classes to discuss:

- BSplineScatteredData
  - nick cleverly sped this up
  - core machinery for all BSpline derived work
- BSplineDeformableTransform
  - this is a revision of the old class
  - took out the bulk transform
  - redefining how users sets b-spline grid in a more intuitive way

B-splines in the Insight Toolkit

A couple classes to discuss:

- BSplineScatteredData
  - nick cleverly sped this up
  - core machinery for all BSpline derived work
- BSplineDeformableTransform
  - this is a revision of the old class
  - took out the bulk transform
  - redefining how users sets b-spline grid in a more intuitive way
- BSplineDeformationFieldTransform or BSplineDenseDeformationFieldTransform

B-splines in the Insight Toolkit

A couple classes to discuss:

- BSplineScatteredData
  - nick cleverly sped this up
  - core machinery for all BSpline derived work
- BSplineDeformableTransform
  - this is a revision of the old class
  - took out the bulk transform
  - redefining how users sets b-spline grid in a more intuitive way
- BSplineDeformationFieldTransform or BSplineDenseDeformationFieldTransform
- BSplineTransformInitializer (rename this)

B-splines in the Insight Toolkit

A couple classes to discuss:

- BSplineScatteredData
  - nick cleverly sped this up
  - core machinery for all BSpline derived work
- BSplineDeformableTransform
  - this is a revision of the old class
  - took out the bulk transform
  - redefining how users sets b-spline grid in a more intuitive way
- BSplineDeformationFieldTransform or BSplineDenseDeformationFieldTransform
- BSplineTransformInitializer (rename this)
  - fixed bug in domain definition

B-splines in the Insight Toolkit

A couple classes to discuss:

- BSplineScatteredData
  - nick cleverly sped this up
  - core machinery for all BSpline derived work
- BSplineDeformableTransform
  - this is a revision of the old class
  - took out the bulk transform
  - redefining how users sets b-spline grid in a more intuitive way
- BSplineDeformationFieldTransform or BSplineDenseDeformationFieldTransform
- BSplineTransformInitializer (rename this)
  - fixed bug in domain definition
  - nick generalized the initialization to N-dimensions

B-splines in the Insight Toolkit

A couple classes to discuss:

- BSplineScatteredData
  - nick cleverly sped this up
  - core machinery for all BSpline derived work
- BSplineDeformableTransform
  - this is a revision of the old class
  - took out the bulk transform
  - redefining how users sets b-spline grid in a more intuitive way
- BSplineDeformationFieldTransform or BSplineDenseDeformationFieldTransform
- BSplineTransformInitializer (rename this)
  - fixed bug in domain definition
  - nick generalized the initialization to N-dimensions
- Rename these transforms?

B-spline transform domain

itkBSplineDeformableTransform (lines 179-187)

```
// Set the valid region
// If the grid spans the interval [start, last].
// The valid interval for evaluation is [start+offset, last-offset]
// when spline order is even.
// The valid interval for evaluation is [start+offset, last-offset)
// when spline order is odd.
// Where offset = floor(spline / 2 ).
// Note that the last pixel is not included in the valid region
// with odd spline orders.
```

B-spline transform domain

itkBSplineDeformableTransform (lines 179-187)

```
// Set the valid region
// If the grid spans the interval [start, last].
// The valid interval for evaluation is [start+offset, last-offset]
// when spline order is even.
// The valid interval for evaluation is [start+offset, last-offset)
// when spline order is odd.
// Where offset = floor(spline / 2 ).
// Note that the last pixel is not included in the valid region
// with odd spline orders.
```

This is slightly different from what I understand about B-spline domains.

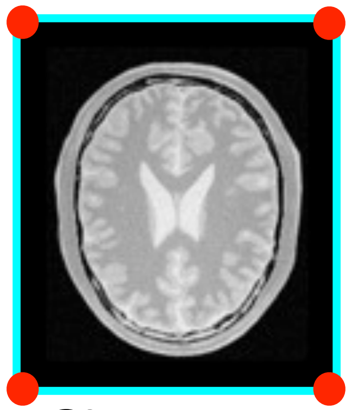
Single element domains with c.p. grid

C^0 splines
SplineOrder = 1

C^2 splines
SplineOrder = 3

C^1 splines
SplineOrder = 2

Single element domains with c.p. grid

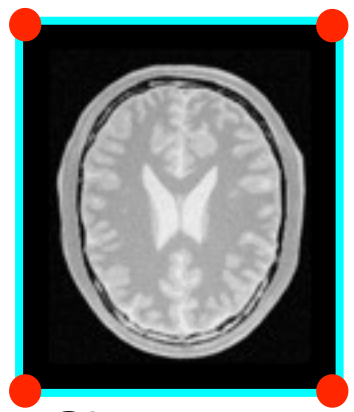


C^0 splines
SplineOrder = 1

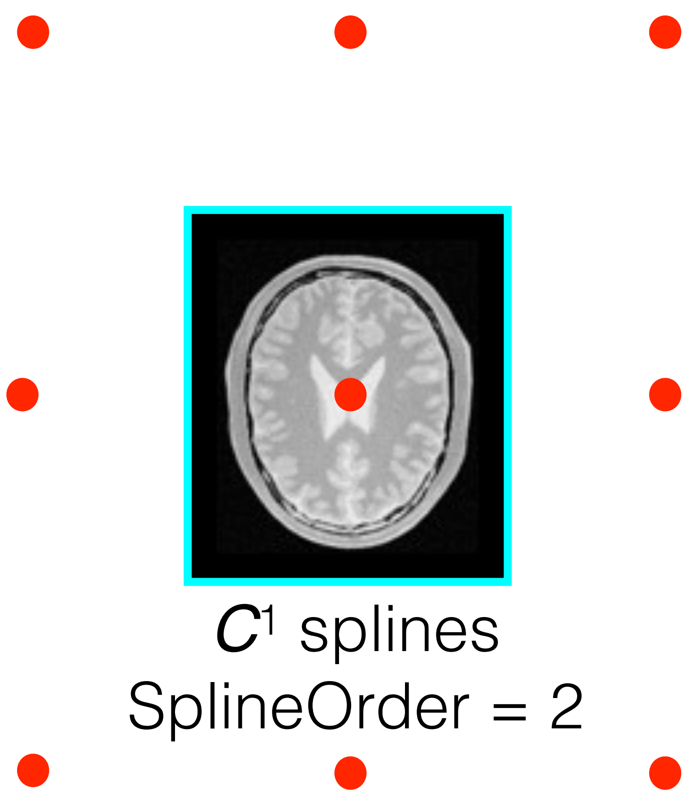
C^2 splines
SplineOrder = 3

C^1 splines
SplineOrder = 2

Single element domains with c.p. grid



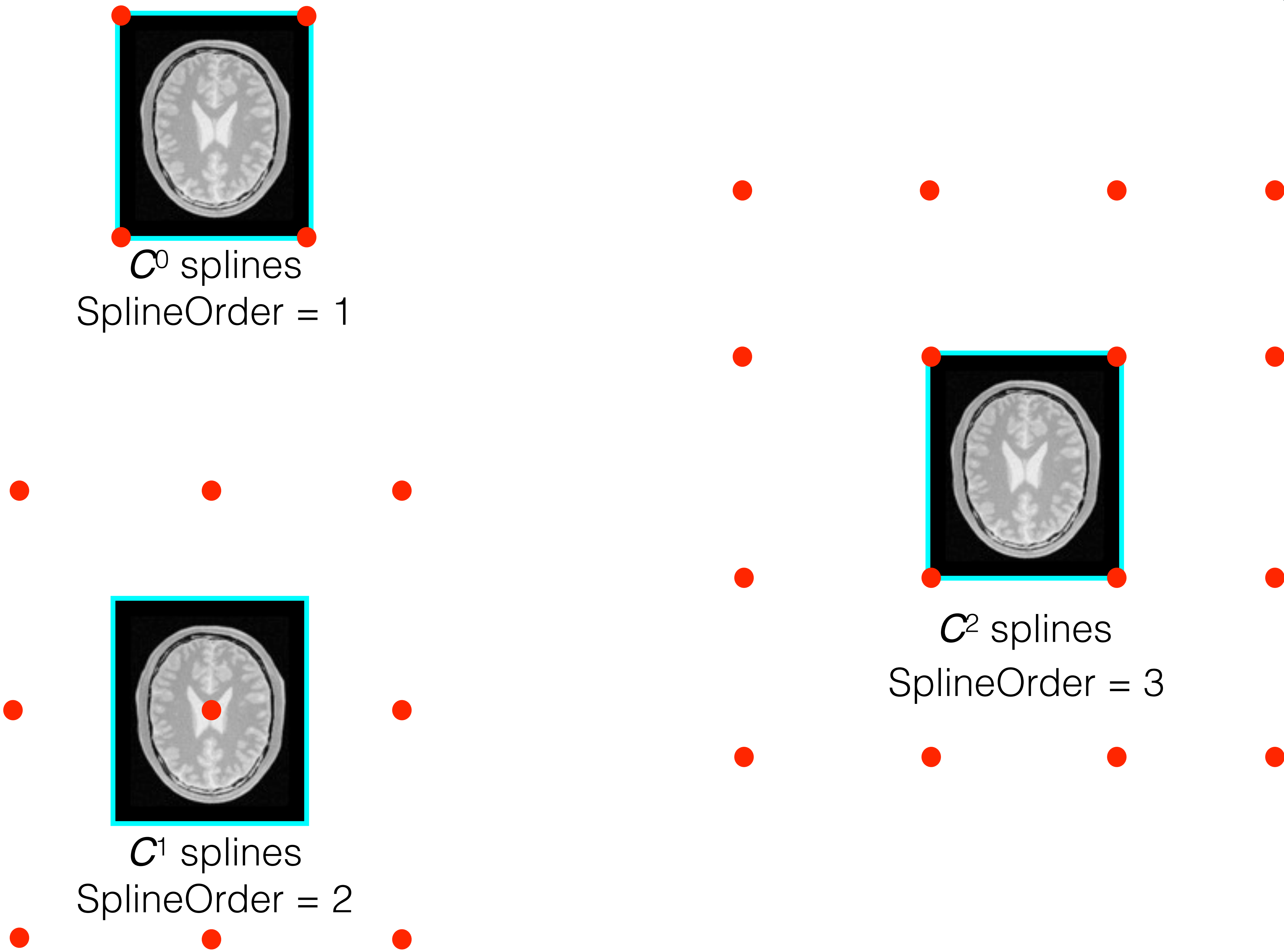
C^0 splines
SplineOrder = 1



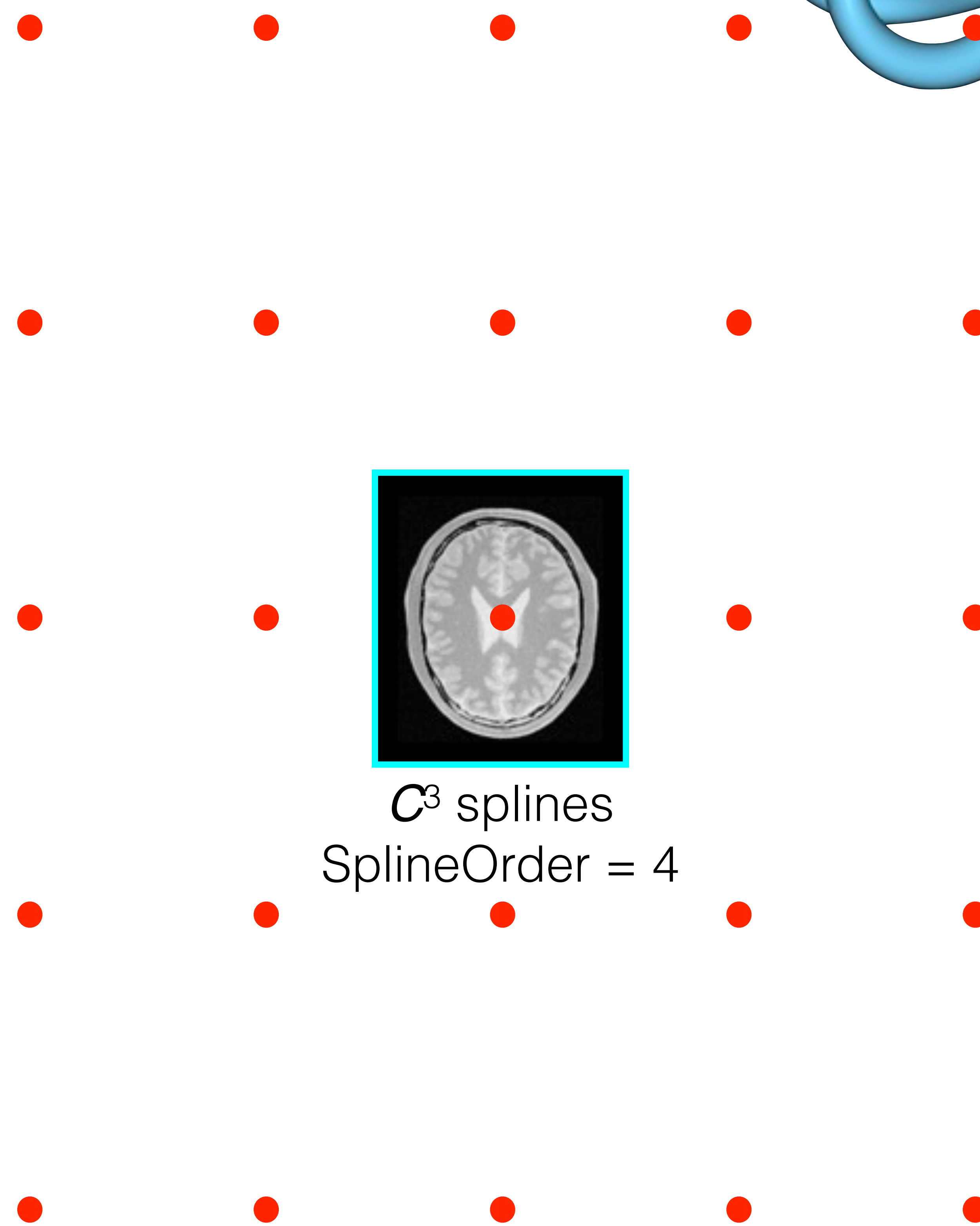
C^1 splines
SplineOrder = 2

C^2 splines
SplineOrder = 3

Single element domains with c.p. grid

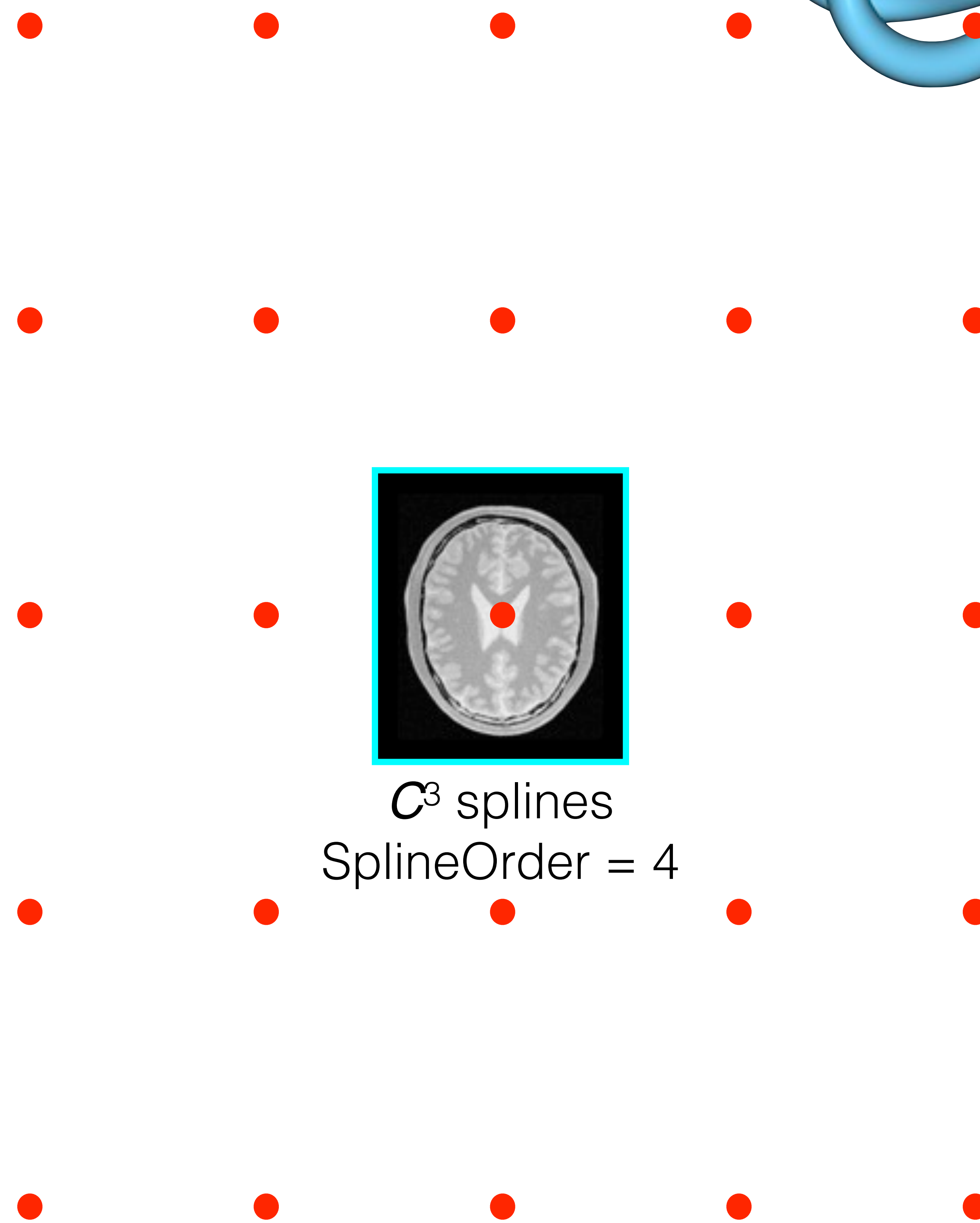


Domain summary



Domain summary

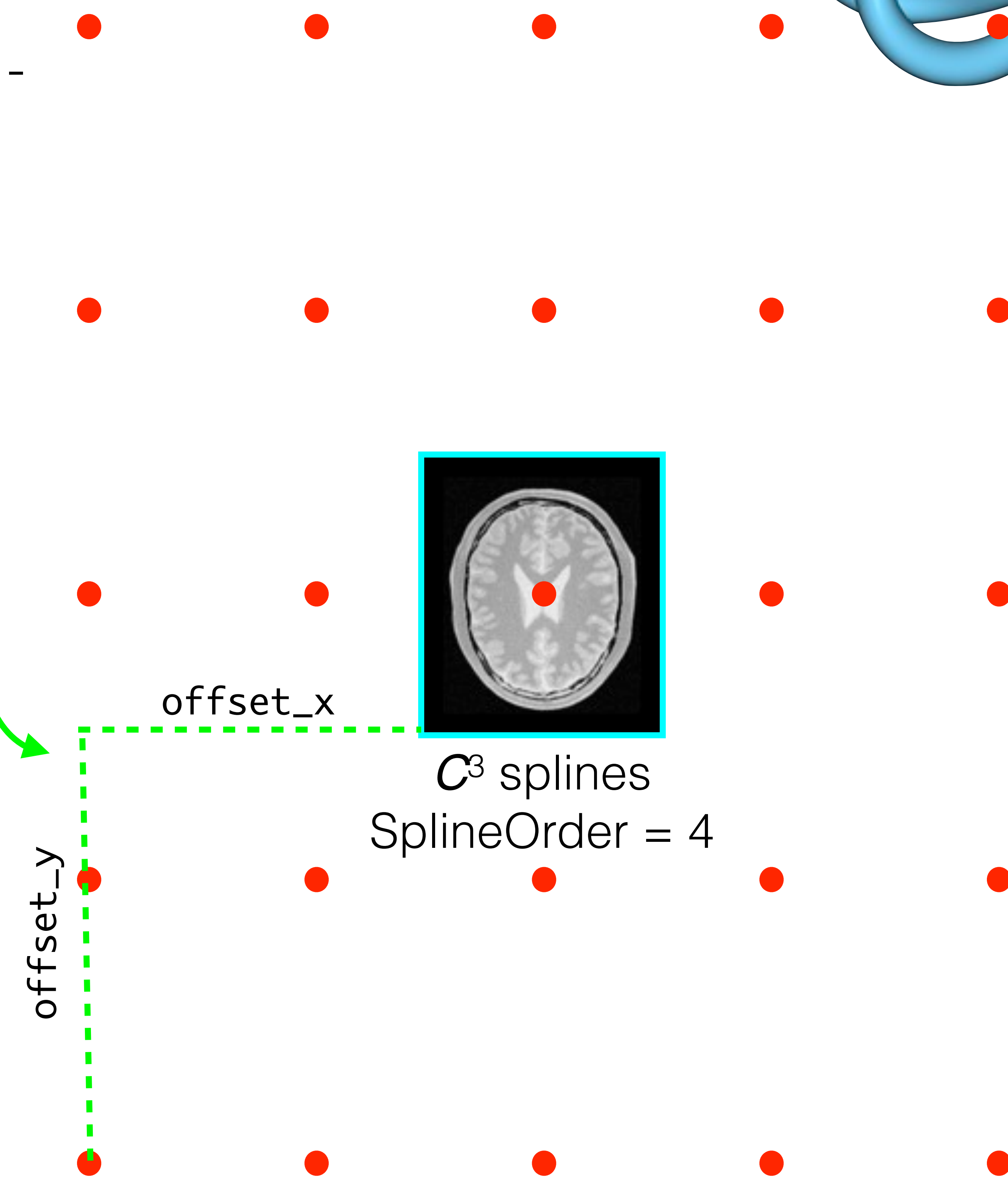
```
meshSize[d] = numberOfControlPoints[d] -
SplineOrder
```



Domain summary

`meshSize[d] = numberOfControlPoints[d] - SplineOrder`

`offset[d] = 0.5 * (SplineOrder - 1 ) * gridSpacing[d]`

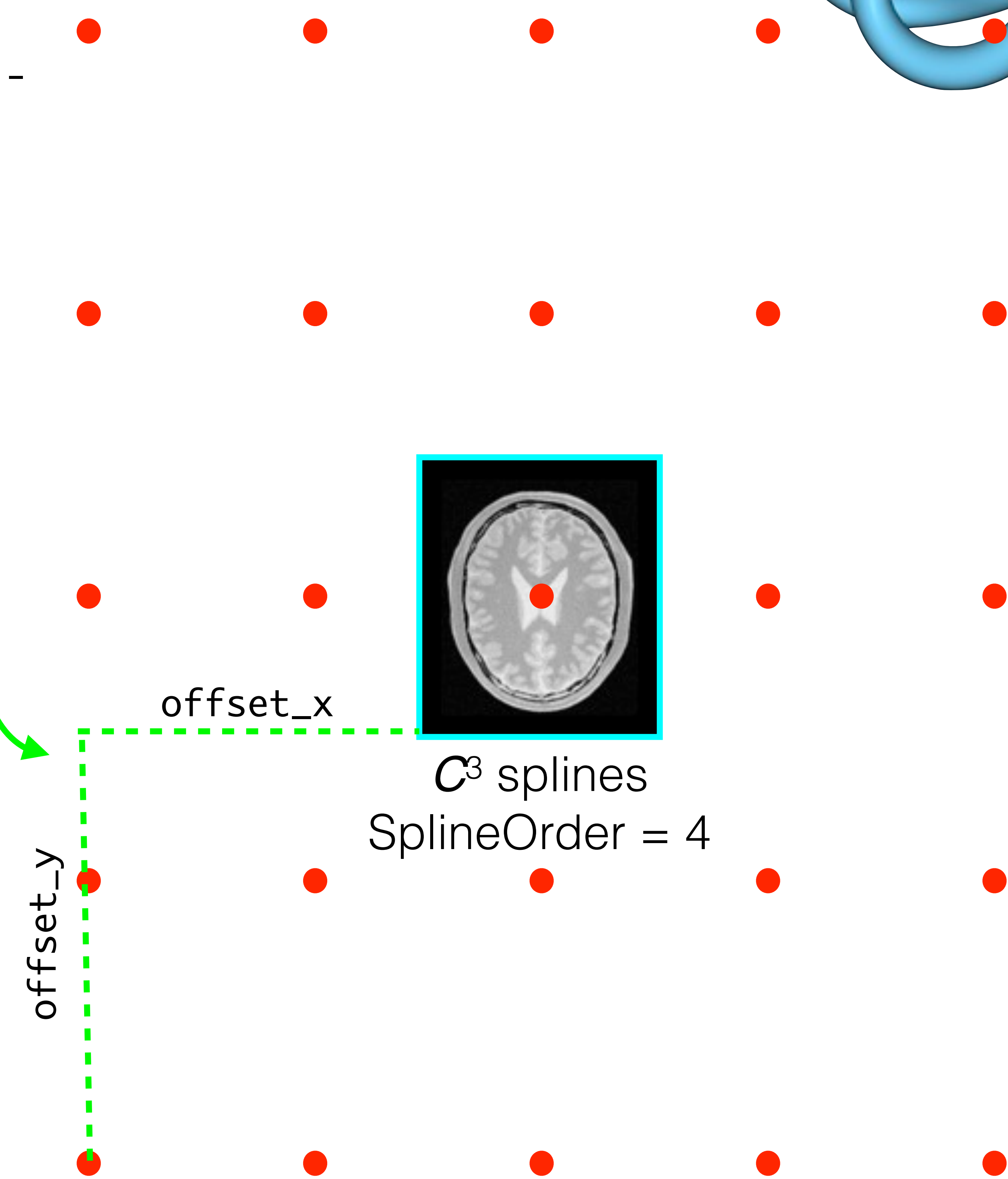


Domain summary

```
meshSize[d] = numberOfControlPoints[d] -
SplineOrder
```

```
offset[d] = 0.5 * (SplineOrder - 1 ) *
gridSpacing[d]
```

control point grid is
centered on the domain



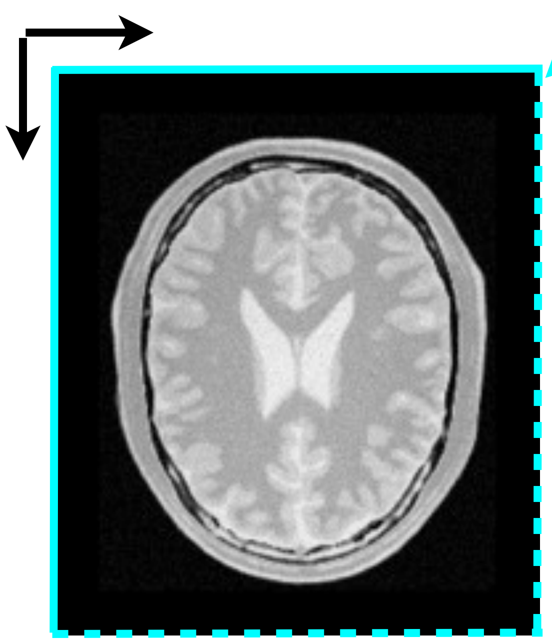
Domain summary

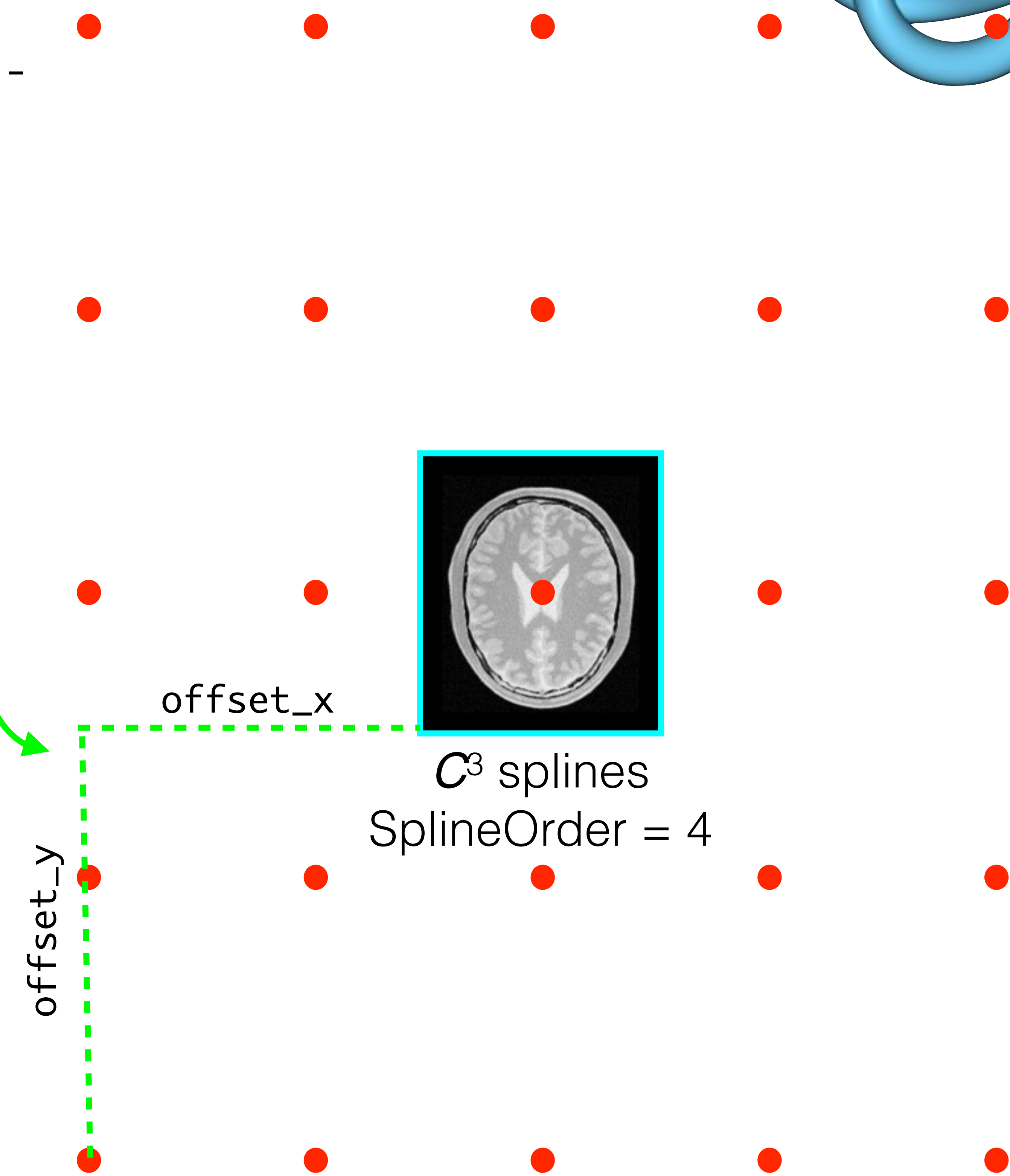
```
meshSize[d] = numberOfControlPoints[d] -
SplineOrder
```

```
offset[d] = 0.5 * (SplineOrder - 1 ) *
gridSpacing[d]
```

control point grid is
centered on the domain

The domain interval is right
open-ended, i.e.





B-spline transform domain changes

Proposed interface change:

```
transform->SetTransformDomainOrigin( origin );
transform->SetTransformDomainPhysicalDimensions( dimensions );
transform->SetTransformDomainDirection( direction );
transform->SetTransformDomainMeshSize( meshSize );
```

This seems to me a much more B-spline-novice friendly interface as the B-spline domain is continuous over a certain finite domain and a typical user can intuit a uniform, rectilinear mesh uniformly placed over that domain of interest (perhaps more FEM-like?).

```
OLD
transform->SetControlPointGridSpacing( something )
transform->SetControlPointGridOrigin( something )
etc
```

B-spline transform initializer

These changes would seemingly obviate the need for a transform initializer except for a “bug” pointed out by Hans Johnson. Since the number of control points in a specific direction directly influences the regularization in the direction, it’s important that the B-spline control point grid is placed relative to the image ***as it resides in physical space***.

B-spline transform initializer

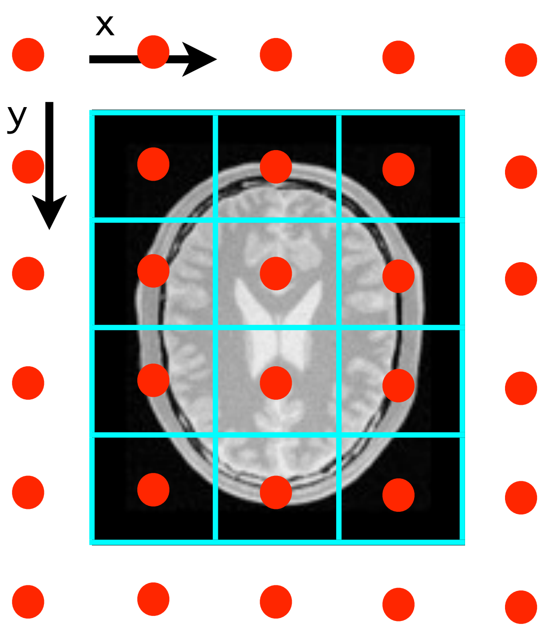
These changes would seemingly obviate the need for a transform initializer except for a “bug” pointed out by Hans Johnson. Since the number of control points in a specific direction directly influences the regularization in the direction, it’s important that the B-spline control point grid is placed relative to the image ***as it resides in physical space***.

Example: current behavior of the initializer

B-spline transform initializer

These changes would seemingly obviate the need for a transform initializer except for a “bug” pointed out by Hans Johnson. Since the number of control points in a specific direction directly influences the regularization in the direction, it’s important that the B-spline control point grid is placed relative to the image ***as it resides in physical space***.

Example: current behavior of the initializer

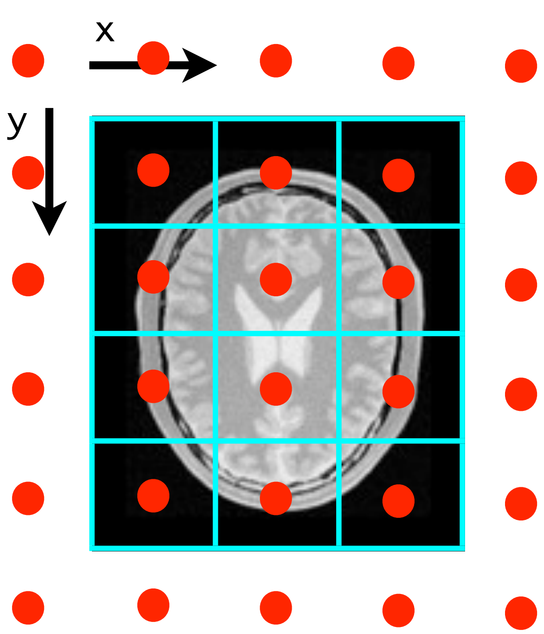


3x4 mesh, quadratic splines, isotropic regularization

B-spline transform initializer

These changes would seemingly obviate the need for a transform initializer except for a “bug” pointed out by Hans Johnson. Since the number of control points in a specific direction directly influences the regularization in the direction, it’s important that the B-spline control point grid is placed relative to the image ***as it resides in physical space***.

Example: current behavior of the initializer



`3x4 mesh, quadratic splines, isotropic regularization`

`itkPermuteAxesImageFilter`

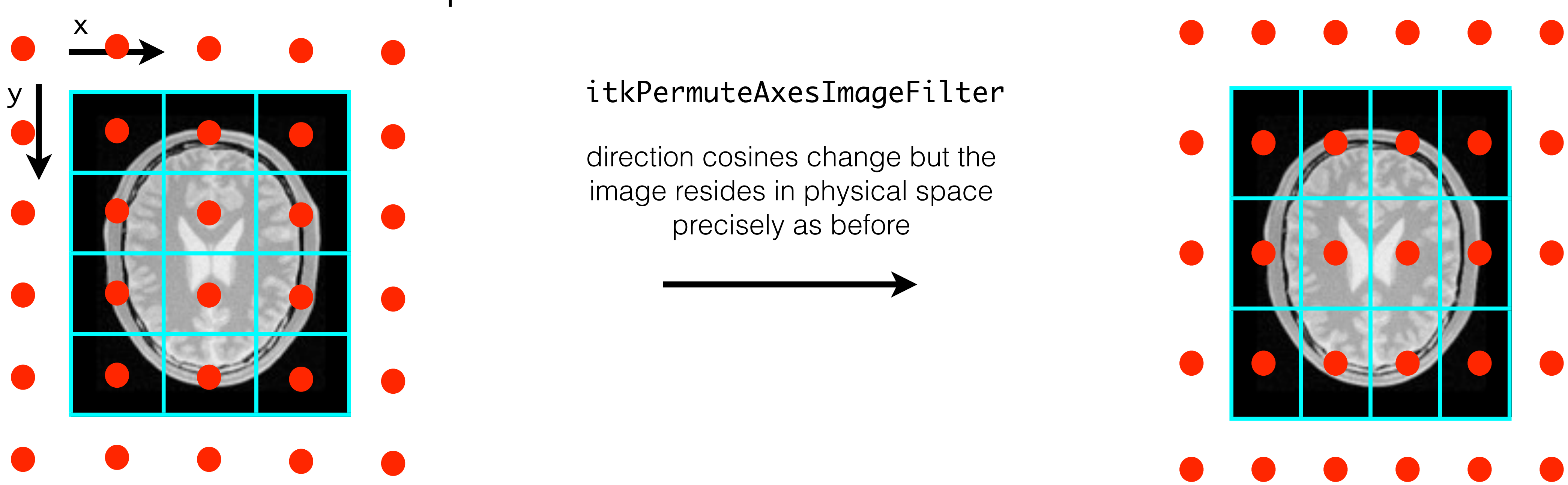
direction cosines change but the image resides in physical space precisely as before

→

B-spline transform initializer

These changes would seemingly obviate the need for a transform initializer except for a "bug" pointed out by Hans Johnson. Since the number of control points in a specific direction directly influences the regularization in the direction, it's important that the B-spline control point grid is placed relative to the image ***as it resides in physical space***.

Example: current behavior of the initializer



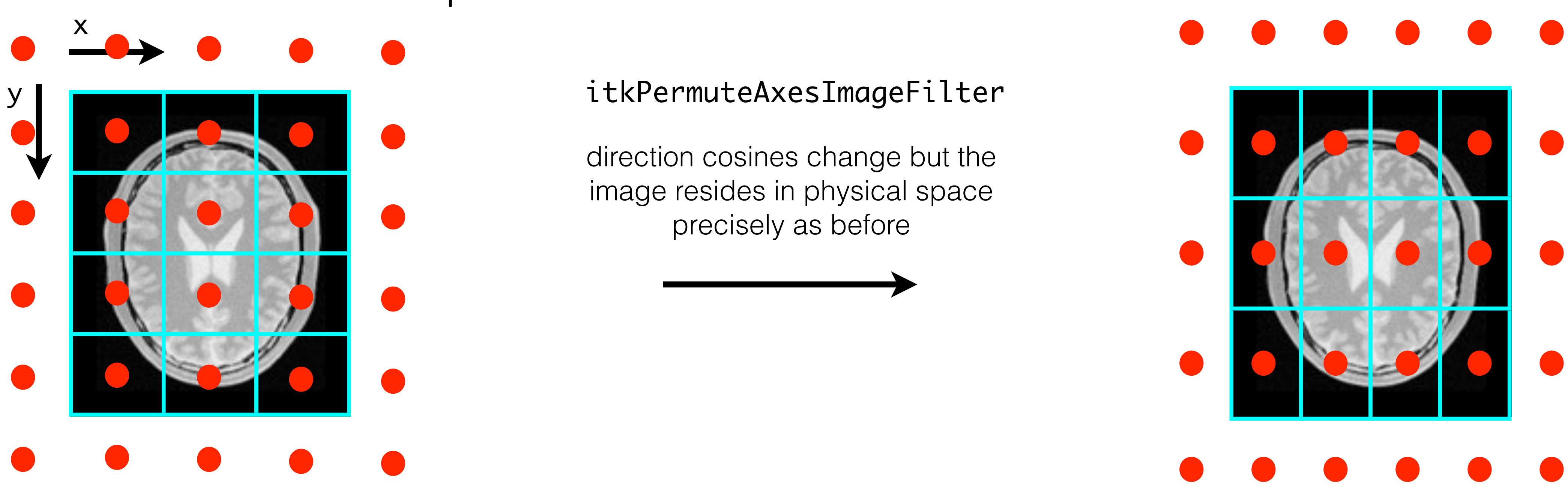
`3x4 mesh, quadratic splines, isotropic regularization`

`4x3 mesh, quadratic splines, anisotropic regularization`

B-spline transform initializer

These changes would seemingly obviate the need for a transform initializer except for a “bug” pointed out by Hans Johnson. Since the number of control points in a specific direction directly influences the regularization in the direction, it’s important that the B-spline control point grid is placed relative to the image ***as it resides in physical space***.

Example: current behavior of the initializer



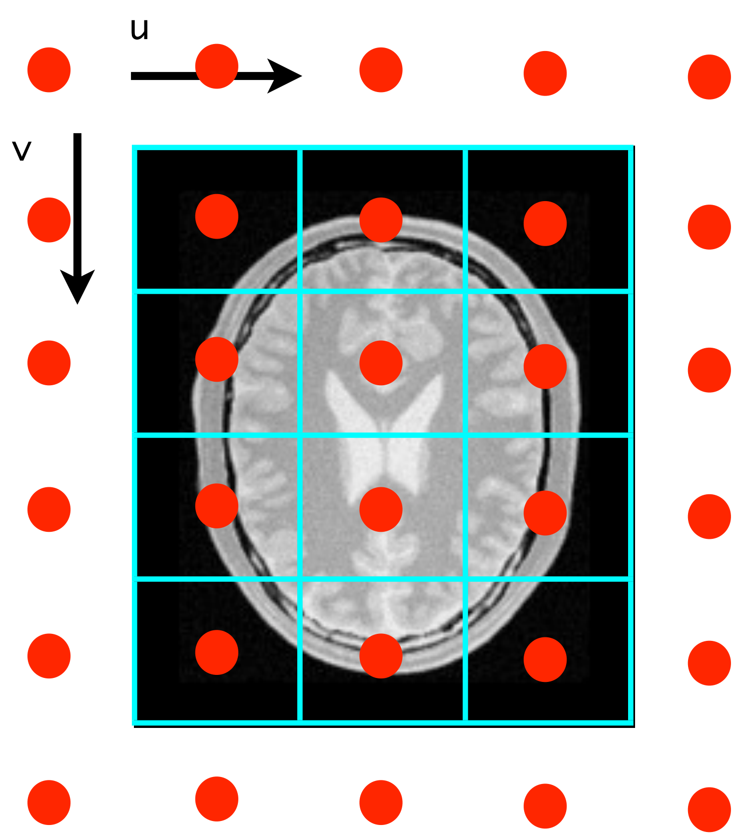
`3x4 mesh, quadratic splines, isotropic regularization`

`4x3 mesh, quadratic splines, anisotropic regularization`

To remedy this, we have introduced a new B-spline transform initializer which takes the n-D fixed image and calculates its location and pose based on the location of its 2^D corners.

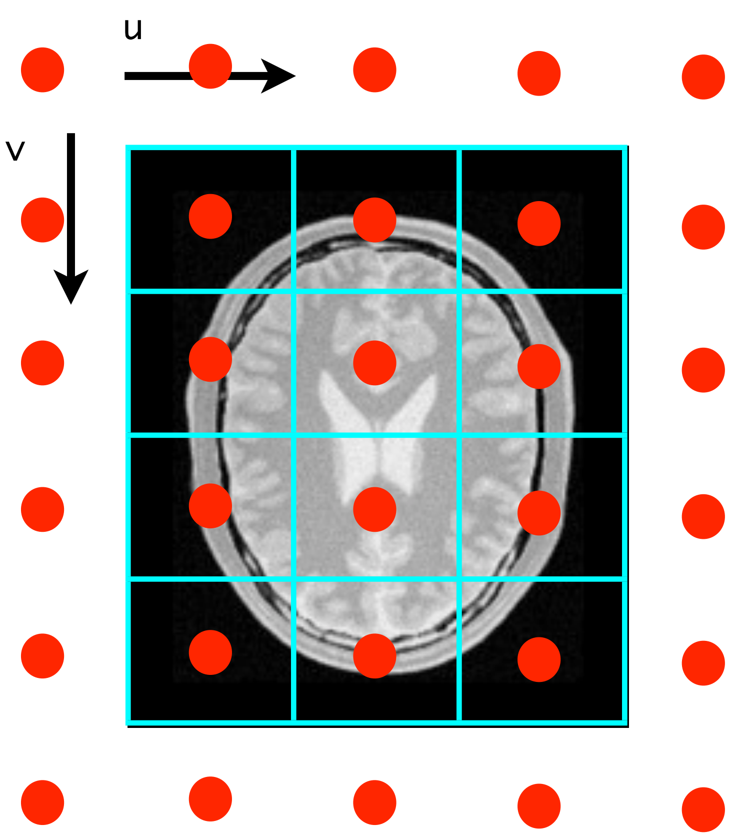
B-spline deformation field transform

One of the other B-spline related issues we are hoping to address is the time it takes to perform image registration. Currently, various sampling strategies and large memory overhead are used to trim down the time it takes to perform registration. The following is another possibility.



B-spline deformation field transform

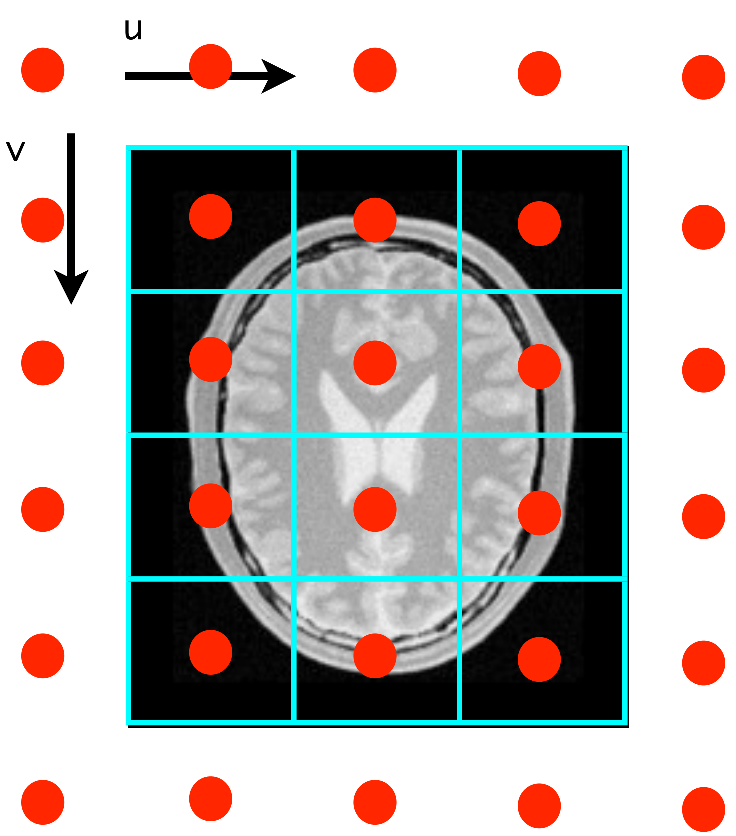
One of the other B-spline related issues we are hoping to address is the time it takes to perform image registration. Currently, various sampling strategies and large memory overhead are used to trim down the time it takes to perform registration. The following is another possibility.



This is how the B-spline transform is currently evaluated via the transform point function:

B-spline deformation field transform

One of the other B-spline related issues we are hoping to address is the time it takes to perform image registration. Currently, various sampling strategies and large memory overhead are used to trim down the time it takes to perform registration. The following is another possibility.

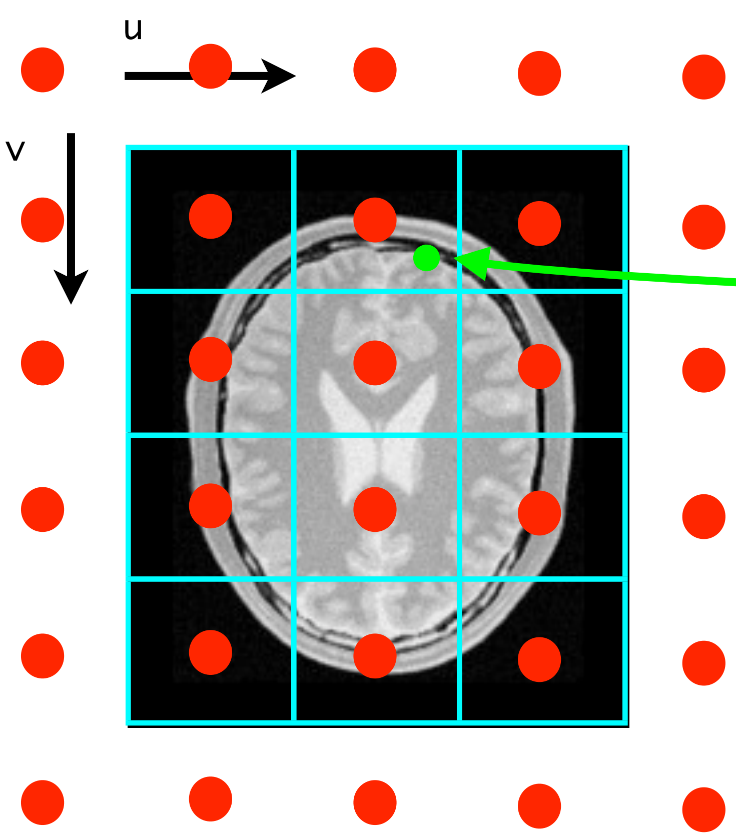


This is how the B-spline transform is currently evaluated via the transform point function:

```
outputPoint = transform->TransformPoint( inputPoint );
```

B-spline deformation field transform

One of the other B-spline related issues we are hoping to address is the time it takes to perform image registration. Currently, various sampling strategies and large memory overhead are used to trim down the time it takes to perform registration. The following is another possibility.

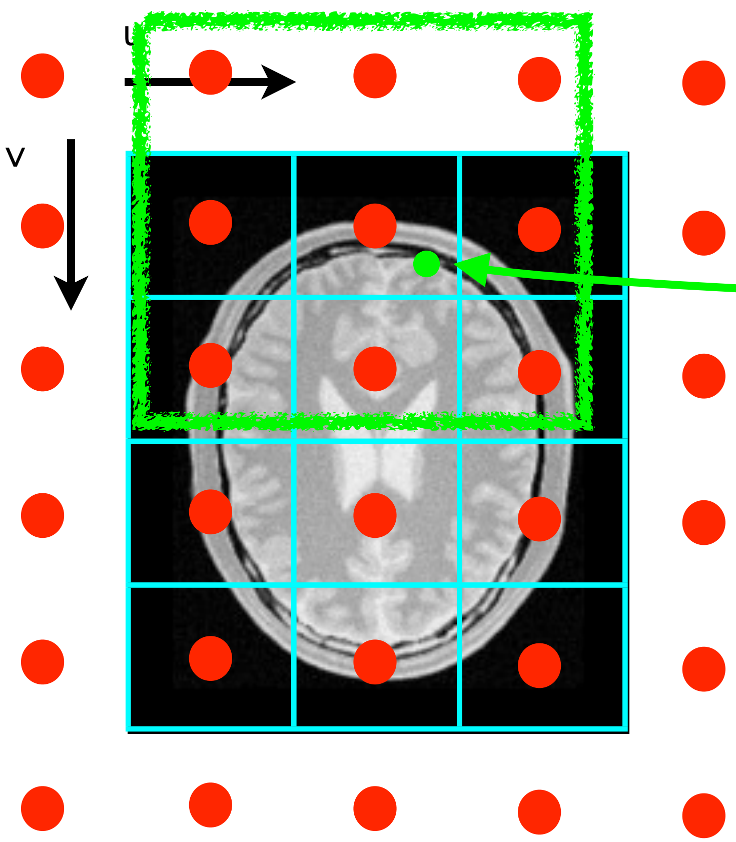


This is how the B-spline transform is currently evaluated via the transform point function:

```
outputPoint = transform->TransformPoint( inputPoint );
```

B-spline deformation field transform

One of the other B-spline related issues we are hoping to address is the time it takes to perform image registration. Currently, various sampling strategies and large memory overhead are used to trim down the time it takes to perform registration. The following is another possibility.



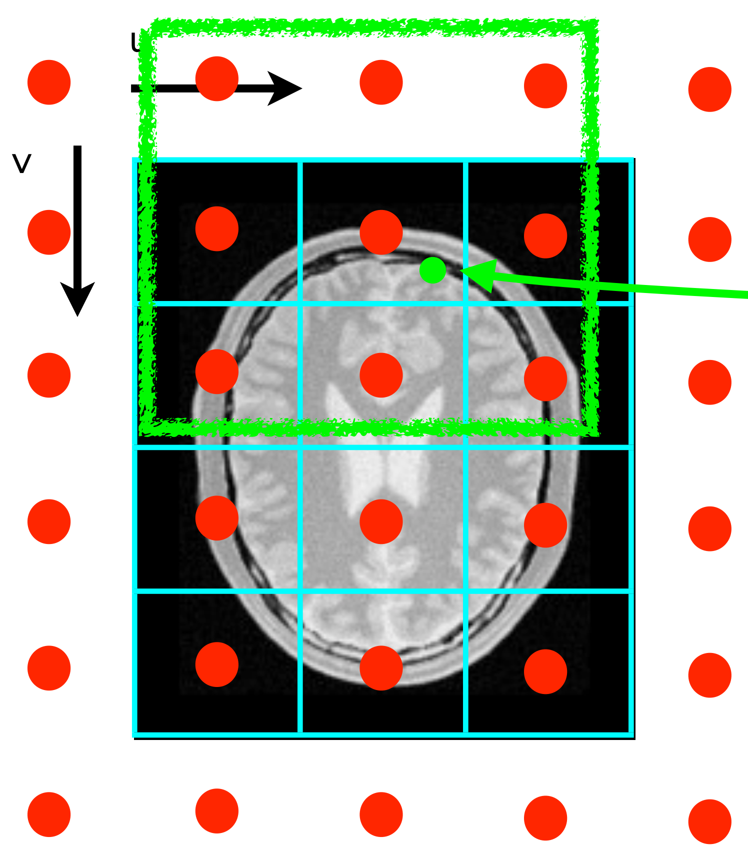
This is how the B-spline transform is currently evaluated via the transform point function:

```
outputPoint = transform->TransformPoint( inputPoint );
```

1. The set of neighborhood control points is determined. For this example, since this is using quadratic splines in 2-D, the total number of control points is `(SplineOrder + 1)^D = 9` non-zero weight control point contributions.

B-spline deformation field transform

One of the other B-spline related issues we are hoping to address is the time it takes to perform image registration. Currently, various sampling strategies and large memory overhead are used to trim down the time it takes to perform registration. The following is another possibility.



This is how the B-spline transform is currently evaluated via the transform point function:

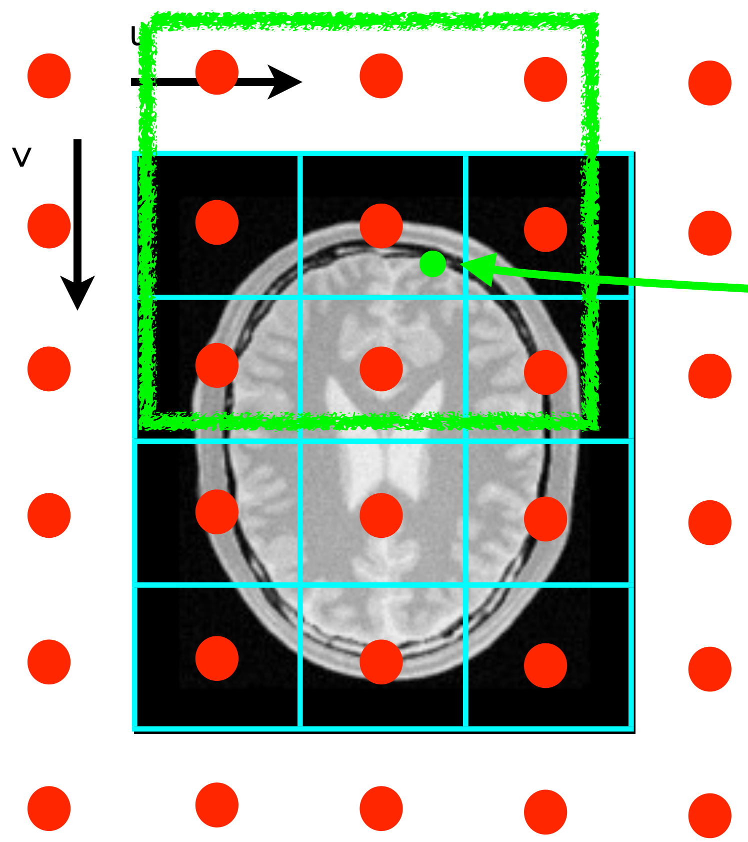
```
outputPoint = transform->TransformPoint( inputPoint );
```

1. The set of neighborhood control points is determined. For this example, since this is using quadratic splines in 2-D, the total number of control points is `(SplineOrder + 1)^D = 9` non-zero weight control point contributions.

2. The weighted sum of the control point values (which are determined based on the location of the input point) is returned to give the output point.

B-spline deformation field transform

One of the other B-spline related issues we are hoping to address is the time it takes to perform image registration. Currently, various sampling strategies and large memory overhead are used to trim down the time it takes to perform registration. The following is another possibility.



This is how the B-spline transform is currently evaluated via the transform point function:

```
outputPoint = transform->TransformPoint( inputPoint );
```

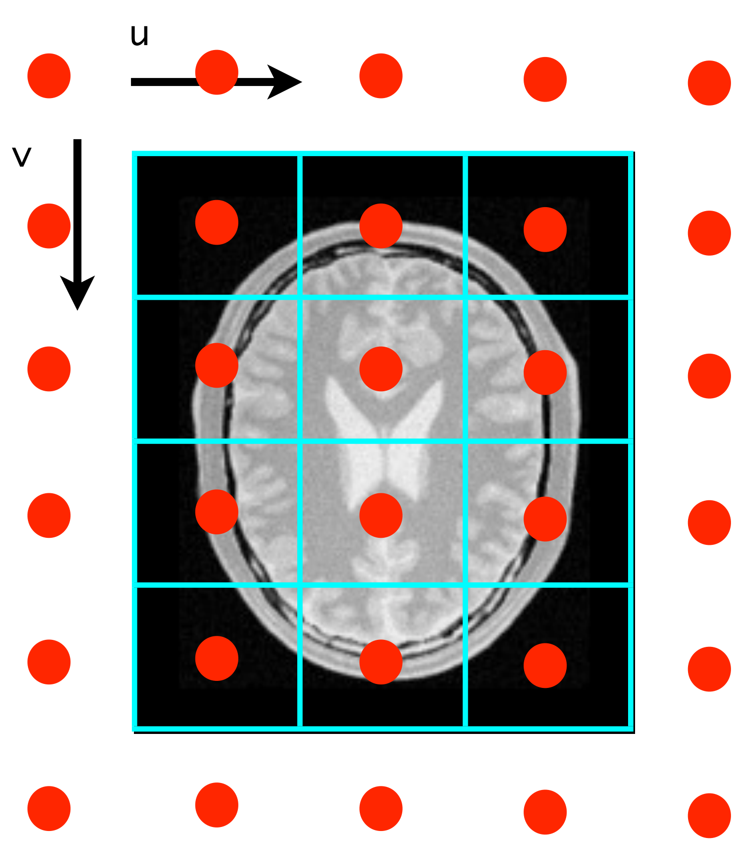
1. The set of neighborhood control points is determined. For this example, since this is using quadratic splines in 2-D, the total number of control points is `(SplineOrder + 1)^D = 9` non-zero weight control point contributions.

2. The weighted sum of the control point values (which are determined based on the location of the input point) is returned to give the output point.

Most commonly people use cubic B-splines in 3-D which means that for *each* input point, one has to calculate **4x4x4=64** B-spline weights! Storing the weights is currently used but this is a huge memory cost alternative.

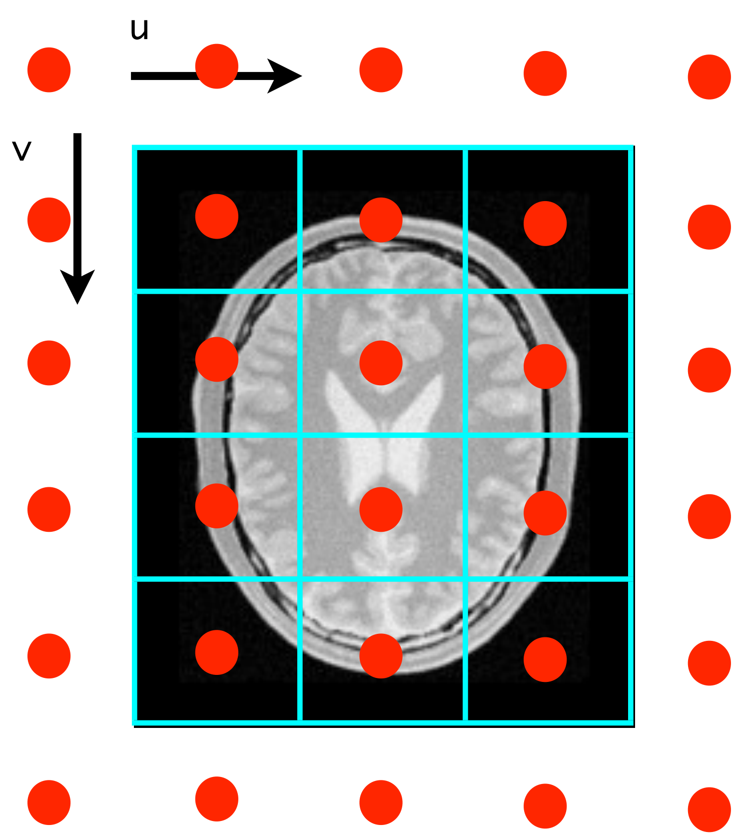
B-spline deformation field transform

Fortunately there is a faster way which we currently use in ANTs for our DMFFD image registration. It is also used in our previous ITK contributions (itkBSplineScatteredDataPointSetToImageFilter, itkBSplineControlPointImageFilter). We do not have to resort to a large memory overhead and we can also sample the metric at each voxel in the entire image with speeds comparable to a Demon's type implementation.



B-spline deformation field transform

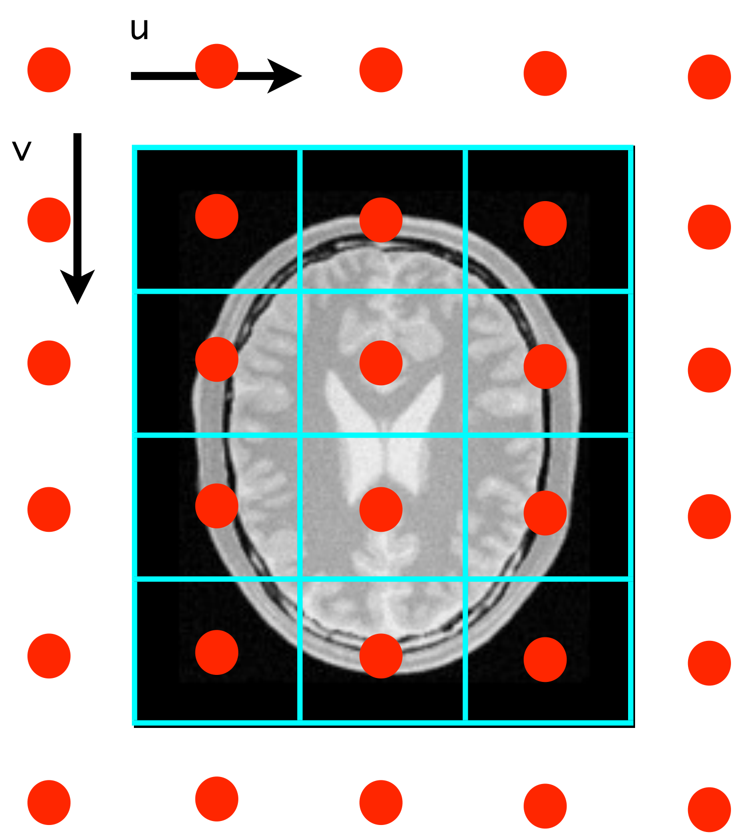
Fortunately there is a faster way which we currently use in ANTs for our DMFFD image registration. It is also used in our previous ITK contributions (itkBSplineScatteredDataPointSetToImageFilter, itkBSplineControlPointImageFilter). We do not have to resort to a large memory overhead and we can also sample the metric at each voxel in the entire image with speeds comparable to a Demon's type implementation.



Suppose instead of evaluating individual points, we sample the transform in its entirety as an `itkImageIterator` would traverse the image. To see what would that give us, we look at the actual B-spline equation.

B-spline deformation field transform

Fortunately there is a faster way which we currently use in ANTs for our DMFFD image registration. It is also used in our previous ITK contributions (itkBSplineScatteredDataPointSetToImageFilter, itkBSplineControlPointImageFilter). We do not have to resort to a large memory overhead and we can also sample the metric at each voxel in the entire image with speeds comparable to a Demon's type implementation.

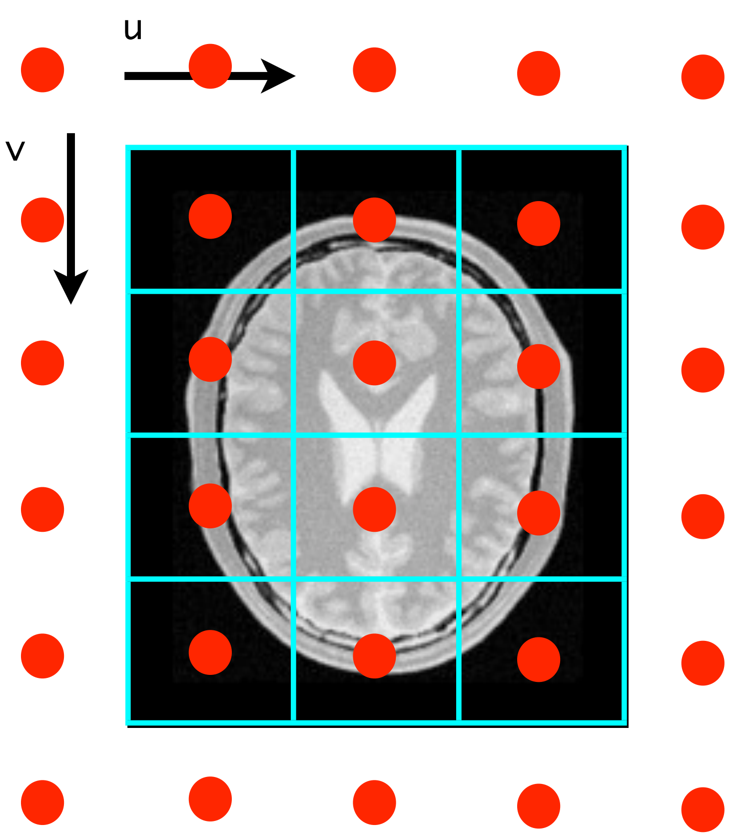


Suppose instead of evaluating individual points, we sample the transform in its entirety as an `itkImageIterator` would traverse the image. To see what would that give us, we look at the actual B-spline equation.

$$\mathcal{T} = \sum_{i=1}^{I} \sum_{j=1}^{J} \phi_{i,j} B_j(v) B_i(u)$$

B-spline deformation field transform

Fortunately there is a faster way which we currently use in ANTs for our DMFFD image registration. It is also used in our previous ITK contributions (itkBSplineScatteredDataPointSetToImageFilter, itkBSplineControlPointImageFilter). We do not have to resort to a large memory overhead and we can also sample the metric at each voxel in the entire image with speeds comparable to a Demon's type implementation.



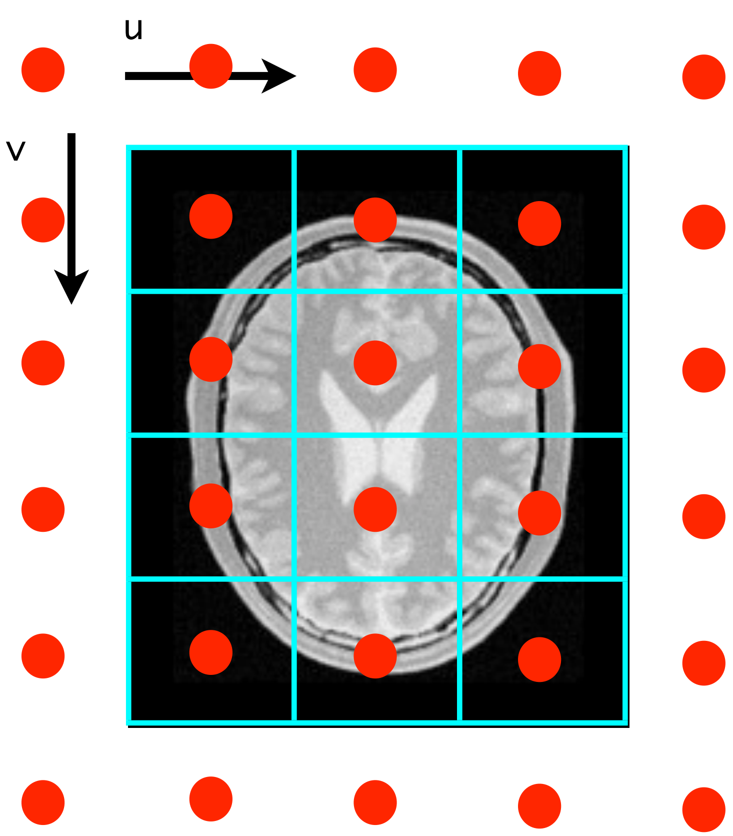
Suppose instead of evaluating individual points, we sample the transform in its entirety as an `itkImageIterator` would traverse the image. To see what would that give us, we look at the actual B-spline equation.

$$\mathcal{T} = \sum_{i=1}^{I} \sum_{j=1}^{J} \phi_{i,j} B_j(v) B_i(u)$$

Since ***v*** is constant along the direction of the first image axis, the weight $B_i(u)$ is also constant so the term in green above can be "collapsed" such that for the entire length of the image sampling where ***v*** is constant, we only have to evaluate a 1-D B-spline with the set of ϕ_i' serving as "pseudo-control" points. This relationship is recursive and extends to ***n***-D. In fact, as mentioned, this is what we do in itkBSplineScatteredDataPointSetToImageFilter. This would also facilitate contributing our own DMFFD-brand of B-spline image registration which is also multi-threaded so Jacobian calculations are much faster.

B-spline deformation field transform

Fortunately there is a faster way which we currently use in ANTs for our DMFFD image registration. It is also used in our previous ITK contributions (itkBSplineScatteredDataPointSetToImageFilter, itkBSplineControlPointImageFilter). We do not have to resort to a large memory overhead and we can also sample the metric at each voxel in the entire image with speeds comparable to a Demon's type implementation.



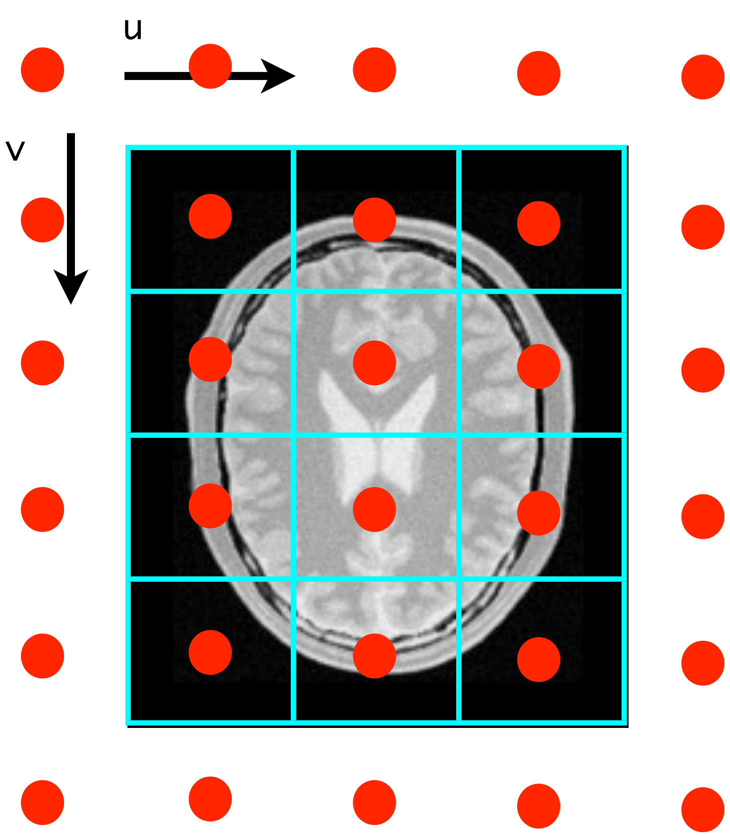
Suppose instead of evaluating individual points, we sample the transform in its entirety as an `itkImageIterator` would traverse the image. To see what would that give us, we look at the actual B-spline equation.

$$\mathcal{T} = \sum_{i=1}^{I} \boxed{\sum_{j=1}^{J} \phi_{i,j} B_j(v)} B_i(u)$$

Since *v* is constant along the direction of the first image axis, the weight $B_i(u)$ is also constant so the term in green above can be "collapsed" such that for the entire length of the image sampling where *v* is constant, we only have to evaluate a 1-D B-spline with the set of ϕ_i' serving as "pseudo-control" points. This relationship is recursive and extends to *n*-D. In fact, as mentioned, this is what we do in itkBSplineScatteredDataPointSetToImageFilter. This would also facilitate contributing our own DMFFD-brand of B-spline image registration which is also multi-threaded so Jacobian calculations are much faster.

B-spline deformation field transform

Fortunately there is a faster way which we currently use in ANTs for our DMFFD image registration. It is also used in our previous ITK contributions (itkBSplineScatteredDataPointSetToImageFilter, itkBSplineControlPointImageFilter). We do not have to resort to a large memory overhead and we can also sample the metric at each voxel in the entire image with speeds comparable to a Demon's type implementation.



Suppose instead of evaluating individual points, we sample the transform in its entirety as an `itkImageIterator` would traverse the image. To see what would that give us, we look at the actual B-spline equation.

$$\mathcal{T} = \sum_{i=1}^{I} \boxed{\phi_i'} B_i(u)$$

Since ***v*** is constant along the direction of the first image axis, the weight ***B$_i$(u)*** is also constant so the term in green above can be "collapsed" such that for the entire length of the image sampling where ***v*** is constant, we only have to evaluate a 1-D B-spline with the set of ϕ_i' serving as "pseudo-control" points. This relationship is recursive and extends to *n*-D. In fact, as mentioned, this is what we do in itkBSplineScatteredDataPointSetToImageFilter. This would also facilitate contributing our own DMFFD-brand of B-spline image registration which is also multi-threaded so Jacobian calculations are much faster.

Other discussion points

Other discussion points

- Still numerous bugs on tracker: some very tricky. some quite trivial.

Other discussion points

- Still numerous bugs on tracker: some very tricky. some quite trivial.
- Time series: worth it to put an individual with experience in video analysis on this?

Other discussion points

- Still numerous bugs on tracker: some very tricky. some quite trivial.
- Time series: worth it to put an individual with experience in video analysis on this?
- Linear registration performance: evaluated ANTs against FSL on relatively difficult data and results were nearly identical. However, we are aware of isolated cases that require more parameter tuning. We agree strongly with Hans.

Other discussion points

- Still numerous bugs on tracker: some very tricky. some quite trivial.

- Time series: worth it to put an individual with experience in video analysis on this?

- Linear registration performance: evaluated ANTs against FSL on relatively difficult data and results were nearly identical. However, we are aware of isolated cases that require more parameter tuning. We agree strongly with Hans.

- Surface analysis and normalization: how much interest?

Other discussion points

- Still numerous bugs on tracker: some very tricky. some quite trivial.
- Time series: worth it to put an individual with experience in video analysis on this?
- Linear registration performance: evaluated ANTs against FSL on relatively difficult data and results were nearly identical. However, we are aware of isolated cases that require more parameter tuning. We agree strongly with Hans.
- Surface analysis and normalization: how much interest?
- Still need to work on DTI. How much interest?

CSV file reader/writer

CSV file reader/writer

A starter project for one of our developers, Shreyas.

CSV file reader/writer

A starter project for one of our developers, Shreyas.

Motivation: read landmarks and help us organize our evaluation work. Generally useful?

CSV file reader/writer

A starter project for one of our developers, Shreyas.

Motivation: read landmarks and help us organize our evaluation work. Generally useful?

Is currently in Gerrit.

Pipeline: 4D Processing + SCCA