`BSplineTransformInitializer`

```
BSplineTransform
itk::simple::BSplineTransformInitializer ( const Image &                 image1,
                                           const std::vector< uint32_t > & transformDomainMeshSize = std::vector< uint32_t >(3, 1u),
                                           unsigned int                   order = 3u
                                         )
```

SetMetricAs

Self & **SetMetricAsANTSNeighborhoodCorrelation** (unsigned int radius)
Use normalized cross correlation using a small neighborhood for each voxel between two images, with speed optimizations for dense registration. More...

Self & **SetMetricAsCorrelation** ()
Use negative normalized cross correlation image metric. More...

Self & **SetMetricAsDemons** (double intensityDifferenceThreshold=0.001)
Use demons image metric. More...

Self & **SetMetricAsJointHistogramMutualInformation** (unsigned int numberOfHistogramBins=20, double varianceForJointPDFSmoothing=1.5)
Use mutual information between two images. More...

Self & **SetMetricAsMattesMutualInformation** (unsigned int numberOfHistogramBins=50)
Use the mutual information between two images to be registered using the method of Mattes et al. More...

Self & **SetMetricAsMeanSquares** ()
Use negative means squares image metric. More...

Self & **SetMetricFixedMask** (const **Image** &binaryMask)
Set an image mask in order to restrict the sampled points for the metric. More...

Self & **SetMetricMovingMask** (const **Image** &binaryMask)
Set an image mask in order to restrict the sampled points for the metric in the moving image space. More...

Self & **SetMetricSamplingStrategy** (**MetricSamplingStrategyType** strategy)
Set sampling strategy for sample generation. More...

SetOptimizerAs

Self & **SetOptimizerAsAmoeba** (double simplexDelta, unsigned int numberOfIterations, double parametersConvergenceTolerance=1e-8, double functionConvergenceTolerance=1e-4, bool withRestarts=false)
Set optimizer to Nelder-Mead downhill simplex algorithm. More...

Self & **SetOptimizerAsConjugateGradientLineSearch** (double learningRate, unsigned int numberOfIterations, double convergenceMinimumValue=1e-6, unsigned int convergenceWindowSize=10, double lineSearchLowerLimit=0, double lineSearchUpperLimit=5.0, double lineSearchEpsilon=0.01, unsigned int lineSearchMaximumIterations=20, EstimateLearningRateType estimateLearningRate=Once, double maximumStepSizeInPhysicalUnits=0.0)
Conjugate gradient descent optimizer with a golden section line search for nonlinear optimization. More...

Self & **SetOptimizerAsExhaustive** (const std::vector< unsigned int > &numberOfSteps, double stepLength=1.0)
Set the optimizer to sample the metric at regular steps. More...

Self & **SetOptimizerAsGradientDescent** (double learningRate, unsigned int numberOfIterations, double convergenceMinimumValue=1e-6, unsigned int convergenceWindowSize=10, EstimateLearningRateType estimateLearningRate=Once, double maximumStepSizeInPhysicalUnits=0.0)
Gradient descent optimizer. More...

Self & **SetOptimizerAsGradientDescentLineSearch** (double learningRate, unsigned int numberOfIterations, double convergenceMinimumValue=1e-6, unsigned int convergenceWindowSize=10, double lineSearchLowerLimit=0, double lineSearchUpperLimit=5.0, double lineSearchEpsilon=0.01, unsigned int lineSearchMaximumIterations=20, EstimateLearningRateType estimateLearningRate=Once, double maximumStepSizeInPhysicalUnits=0.0)
Gradient descent optimizer with a golden section line search. More...

Self & **SetOptimizerAsLBFGSB** (double gradientConvergenceTolerance=1e-5, unsigned int numberOfIterations=500, unsigned int maximumNumberOfCorrections=5, unsigned int maximumNumberOfFunctionEvaluations=2000, double costFunctionConvergenceFactor=1e+7, double lowerBound=std::numeric_limits< double >::min(), double upperBound=std::numeric_limits< double >::max(), bool trace=false)
Limited memory Broyden Fletcher Goldfarb Shannon minimization with simple bounds. More...

Self & **SetOptimizerAsOnePlusOneEvolutionary** (unsigned int numberOfIterations=100, double epsilon=1.5e-4, double initialRadius=1.01, double growthFactor=-1.0, double shrinkFactor=-1.0, unsigned int seed=itkWallClock)
1+1 evolutionary optimizer strategy. More...

Self & **SetOptimizerAsPowell** (unsigned int numberOfIterations=100, unsigned int maximumLineIterations=100, double stepLength=1, double stepTolerance=1e-6, double valueTolerance=1e-6)
Powell optimization using Brent line search. More...

Self & **SetOptimizerAsRegularStepGradientDescent** (double learningRate, double minStep, unsigned int numberOfIterations, double relaxationFactor=0.5, double gradientMagnitudeTolerance=1e-4, EstimateLearningRateType estimateLearningRate=Never, double maximumStepSizeInPhysicalUnits=0.0)
Regular Step Gradient descent optimizer. More...

Self & **SetOptimizerScales** (const std::vector< double > &scales)
Manually set per parameter weighting for the transform parameters. More...

Self & **SetOptimizerScalesFromIndexShift** (unsigned int centralRegionRadius=5, double smallParameterVariation=0.01)
Estimate scales from maximum voxel shift in index space cause by parameter change. More...

Self & **SetOptimizerScalesFromJacobian** (unsigned int centralRegionRadius=5)
Estimate scales from Jacobian norms. More...

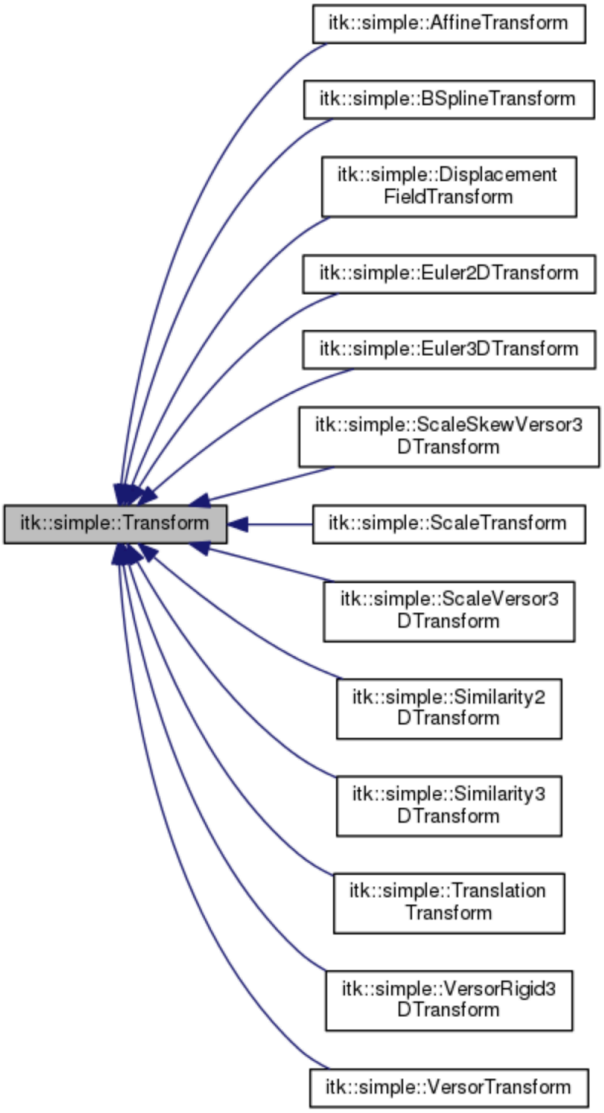
Self & **SetOptimizerScalesFromPhysicalShift** (unsigned int centralRegionRadius=5, double smallParameterVariation=0.01)
Estimating scales of transform parameters a step sizes, from the maximum voxel shift in physical space caused by a parameter change. More...

```
SetOptimizerScalesFromPhysicalShift(
self,
centralRegionRadius=5,
smallParameterVariation=0.01):
Estimating scales of transform parameters a step sizes, from the
maximum voxel shift in physical space caused by a parameter change.
```

We could also use Regular Step Gradient descent optimizer.

```
enum OptimizerScalesType {
        Manual,
        Jacobian,
        IndexShift,
        PhysicalShift
      }
```

```
SetInitialTransform(
self,
transform,
inPlace=True):
```



This transform is applied before the MovingInitialTransform, to map from the virtual image domain to the moving image domain.

If the inPlace flag is explicitly false, then the ITK registration will internally make a copy, and the transform will not be accessible during registration. Otherwise, the accessible InitialTransform value will be the same object used during registration, and will have a modified value upon completion.

For each transformation we can set the following parameters:

```
enum MetricType {
  ANTSNeighborhoodCorrelation,
  Correlation,
  Demons,
  JointHistogramMutualInformation,
  MeanSquares,
  MattesMutualInformation
}
enum OptimizerScalesType {
  Manual,
  Jacobian,
  IndexShift,
  PhysicalShift
}
enum OptimizerType {
  ConjugateGradientLineSearch,
  RegularStepGradientDescent,
  GradientDescent,
  GradientDescentLineSearch,
  LBFGSB,
  Exhaustive,
  Amoeba,
  Powell,
  OnePlusOneEvolutionary
}
```

```
SetInterpolator(
self,
Interpolator):
```

sitkNearestNeighbor
sitkLinear
sitkBSpline
sitkGaussian
sitkLabelGaussian
sitkHammingWindowedSinc
sitkCosineWindowedSinc
sitkWelchWindowedSinc
sitkLanczosWindowedSinc
sitkBlackmanWindowedSinc

SetShrinkFactorsPerLevel

template<typename TFixedImage , typename TMovingImage , typename TOutputTransform = Transform<double, TFixedImage::ImageDimension, TFixedImage::ImageDimension>, typename TVirtualImage = TFixedImage, typename TPointSet = PointSet<unsigned int, TFixedImage::ImageDimension>>

void itk::ImageRegistrationMethodv4< TFixedImage, TMovingImage, TOutputTransform, TVirtualImage, TPointSet >::SetShrinkFactorsPerLevel (ShrinkFactorsArrayType **factors**) inline

Set the shrink factors for each level where each level has a constant shrink factor for each dimension. For example, input to the function of factors = [4,2,1] will shrink the image in every dimension by 4 the first level, then by 2 at the second level, then the original resolution for the final level (uses the `itkShrinkImageFilter`).

Definition at line **341** of file **itkImageRegistrationMethodv4.h**.

References **itk::FixedArray< TValue, VLength >::Fill()**, and **itk::Array< TValue >::Size()**.

SetSmoothingSigmasPerLevel

template<typename TFixedImage , typename TMovingImage , typename TOutputTransform >

virtual void itk::ImageRegistrationMethodv4< TFixedImage, TMovingImage, TOutputTransform >::SetSmoothingSigmasPerLevel (SmoothingSigmasArrayType **_arg**) virtual

Set/Get the smoothing sigmas for each level. At each resolution level, a gaussian smoothing filter (specifically, the `itkDiscreteGaussianImageFilter`) is applied. Sigma values are specified according to the option `m_SmoothingSigmasAreSpecifiedInPhysicalUnits`.