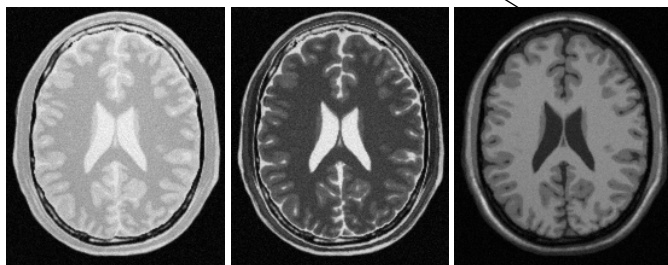
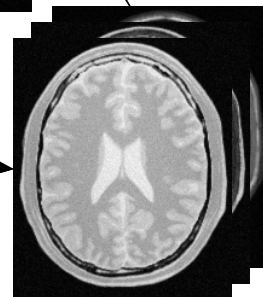


# Creating Images with Multiple Components

## Multiple-Component Images



3 Images of  
1 component  
per pixel



1 Image of  
3 component per pixel

## Image to Vector Image Filter

```
#include "itkImage.h"
#include "itkVectorImage.h"
#include "itkImageFileReader.h"
#include "itkImageFileWriter.h"
#include "itkImageToVectorImageFilter.h"

int main( int argc, char * argv[] )

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

typedef itk::ImageToVectorImageFilter< ImageType > FilterType;

typedef FilterType::OutputImageType OutputImageType;

typedef itk::ImageFileReader< ImageType > ReaderType;
typedef itk::ImageFileWriter< OutputImageType > WriterType;

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

## Image to Vector Image Filter

```
for (unsigned int k = 1; k < argc-1; k++)
{
    ReaderType::Pointer reader = ReaderType::New();
    reader->SetFileName( argv[ k ] );
    reader->Update();
    filter->SetNthInput( k-1, reader->GetOutput() );
}

WriterType::Pointer writer = WriterType::New();
writer->SetInput( filter->GetOutput() );
writer->SetFileName( argv[ argc-1 ] );

writer->Update();
```

## Image to Vector Image Filter

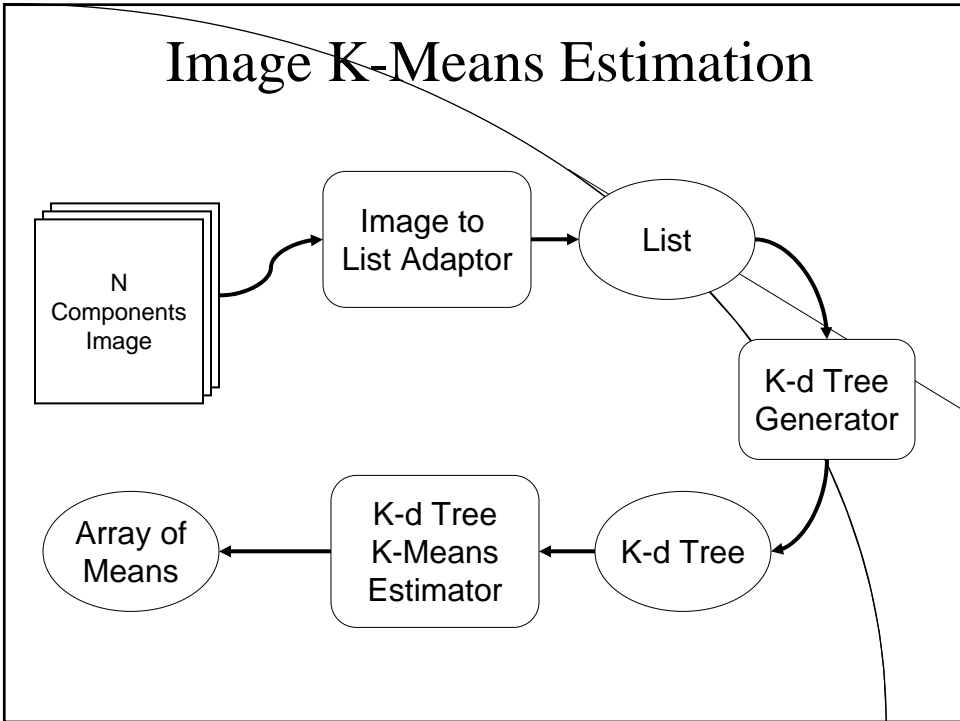
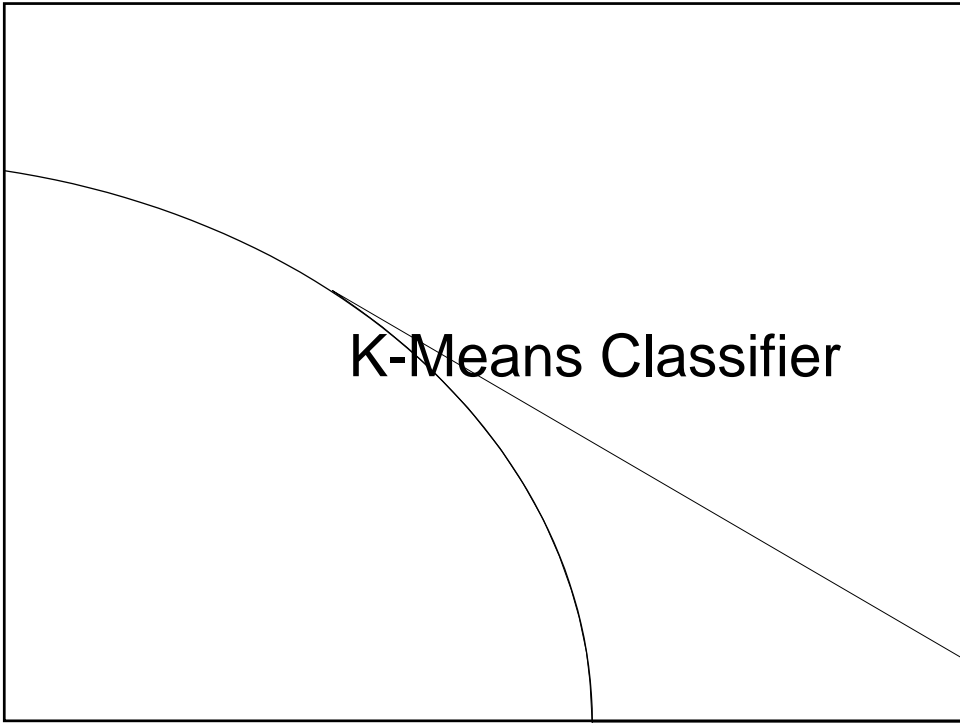
Run Command:

```
ConvertImagesToVectorImage  
  brainweb165a10f17_Slice94.png  
  brainweb1e1a10f20_Slice94.png  
  brainweb1e5a10f17_Slice94.png  
  brainwebSlice94.vtk
```

In ParaView:

Load the brainwebSlice94.vtk image  
Use the Calculator to verify the components

Exercise 36a



## Image K-Means Estimation

```
#include "itkKdTree.h"
#include "itkKdTreeBasedKmeansEstimator.h"
#include "itkWeigthedCentroidKdTreeGenerator.h"

#include "itkImageToListAdaptor.h"

#include "itkImageFileReader.h"
#include "itkImage.h"

int main( int argc, char * argv[] )
{

typedef itk::Vector< unsigned char, 3 > PixelType;

typedef itk::Image< PixelType, 2 > ImageType;
```

## Image K-Means Estimation

```
typedef itk::Statistics::ImageToListAdaptor< ImageType > AdaptorType;

AdaptorType::Pointer adaptor = AdaptorType::New();

adaptor->SetImage( reader->GetOutput() );

typedef itk::Statistics::WeightedCentroidKdTreeGenerator<
        AdaptorType > TreeGeneratorType;

TreeGeneratorType::Pointer treeGenerator = TreeGeneratorType::New();

treeGenerator->SetSample( adaptor );
treeGenerator->SetBucketSize( 16 );
treeGenerator->Update();
```

## Image K-Means Estimation

```
typedef TreeGeneratorType::KdTreeType   TreeType;

typedef itk::Statistics::KdTreeBasedKmeansEstimator< TreeType >
                                           EstimatorType;

EstimatorType::Pointer estimator = EstimatorType::New();

EstimatorType::ParametersType   initialMeans( NumberOfClasses *
                                               NumberOfComponents );

estimator->SetParameters( initialMeans );

estimator->SetKdTree( treeGenerator->GetOutput() );
estimator->SetMaximumIteration( 200 );
estimator->SetCentroidPositionChangesThreshold(0.0);
estimator->StartOptimization();

EstimatorType::ParametersType estimatedMeans =
    estimator->GetParameters();
```

## Image K-Means Estimation

```
for ( unsigned int i = 0 ; i < numberOfClasses ; ++i )
{
    std::cout << "cluster[" << i << "]" ";
    std::cout << " estimated mean : ";
    for ( unsigned int j = 0 ; j < NumberOfComponents ; ++j )
    {
        std::cout << " " << estimatedMeans[ i * NumberOfComponents + j ];
    }
    std::cout << std::endl;
}
```

## Vector Image K-Means Estimation

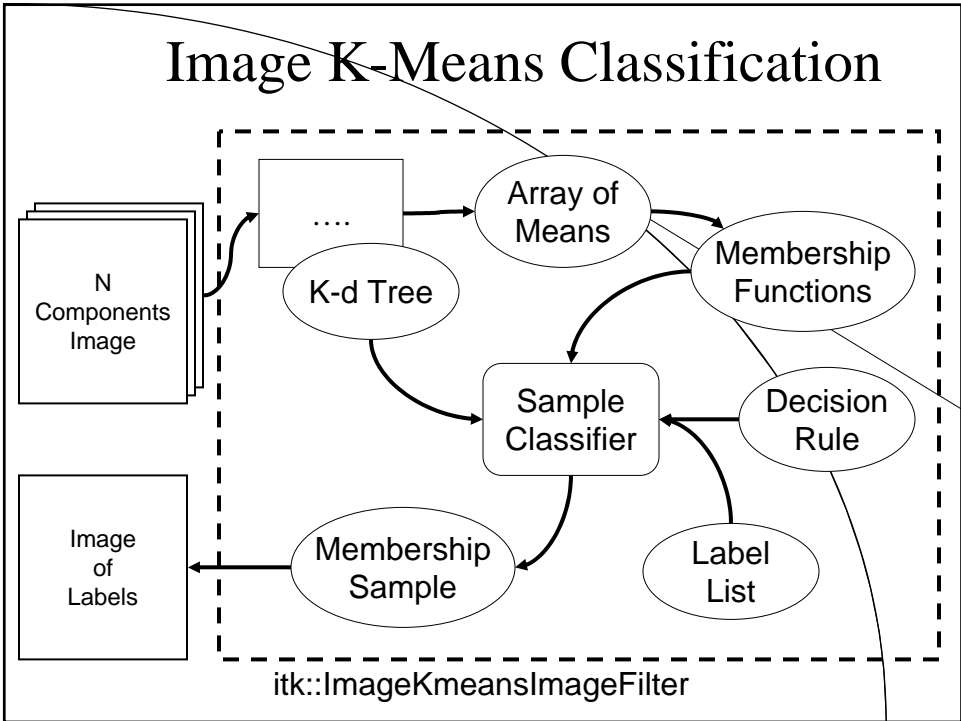
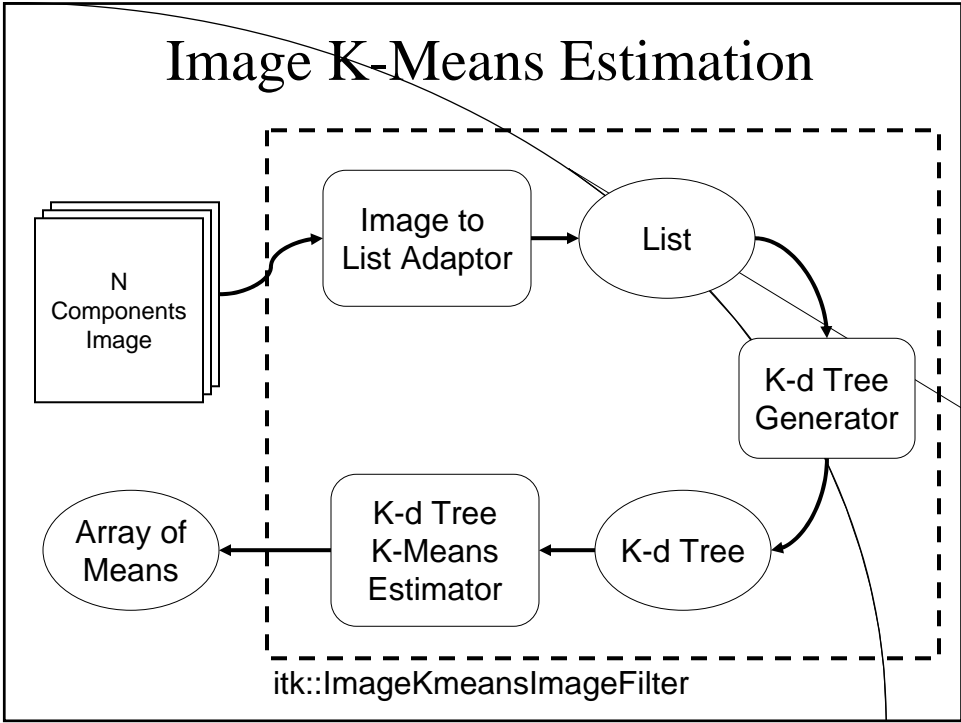
Run Command:

```
ImageKmeansModelEstimator  
brainwebSlice94.vtk 4
```

Output:

Locations of the Means  
in the N-Components Space

Exercise 36b



## Image K-Means Classification

```
#include "itkImage.h"
#include "itkImageFileReader.h"
#include "itkImageFileWriter.h"
#include "itkImageKmeansImageFilter.h"

int main( int argc, char * argv[] )
{

typedef itk::Vector< unsigned char, 3 > PixelType;

typedef itk::Image< PixelType, 2 > ImageType;
```

## Image K-Means Classification

```
typedef itk:: ImageKmeansImageFilter< ImageType > KMeansFilterType;
KMeansFilterType::Pointer kmeansFilter = KMeansFilterType::New();
kmeansFilter->SetInput( reader->GetOutput() );
kmeansFilter->SetUseNonContiguousLabels( useNonContiguousLabels );

KMeansFilterType::RealPixelType initialMean;
initialMean.Fill(0.0); // N-Components array

for( unsigned int cc = 0; cc < numberOfClasses; ++cc )
{
kmeansFilter->AddClassWithInitialMean( initialMean );
}
```

## Image K-Means Classification

```
typedef KMeansFilterType::OutputImageType OutputImageType;
typedef itk::ImageFileWriter< OutputImageType > WriterType;

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

writer->SetInput( kmeansFilter->GetOutput() );

writer->Update();

KMeansFilterType::ParametersType estimatedMeans =
    kmeansFilter->GetFinalMeans();

// Array of NumberOfClasses X NumberOfComponents
```

## Image K-Means Classification

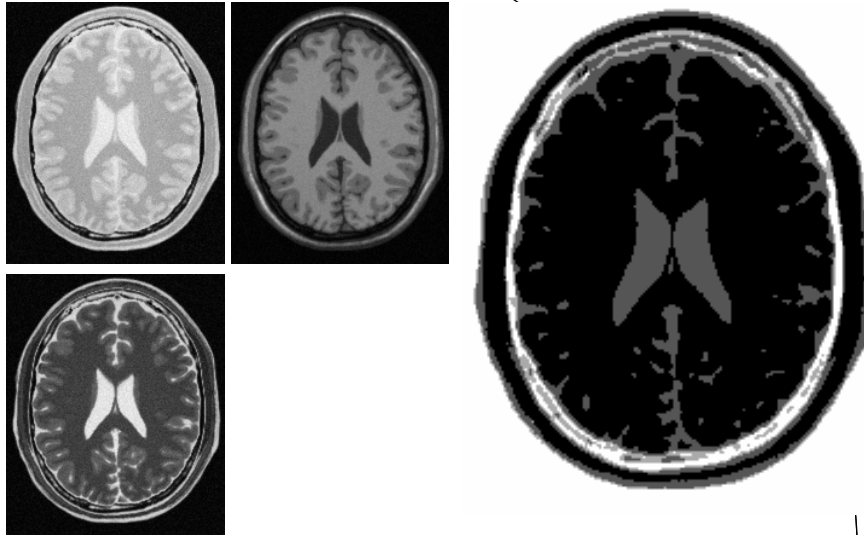
Run Command:

```
ImageKmeansModelClassifier
  brainwebSlice94.vtk labelsImage.vtk 4 0
```

Output:

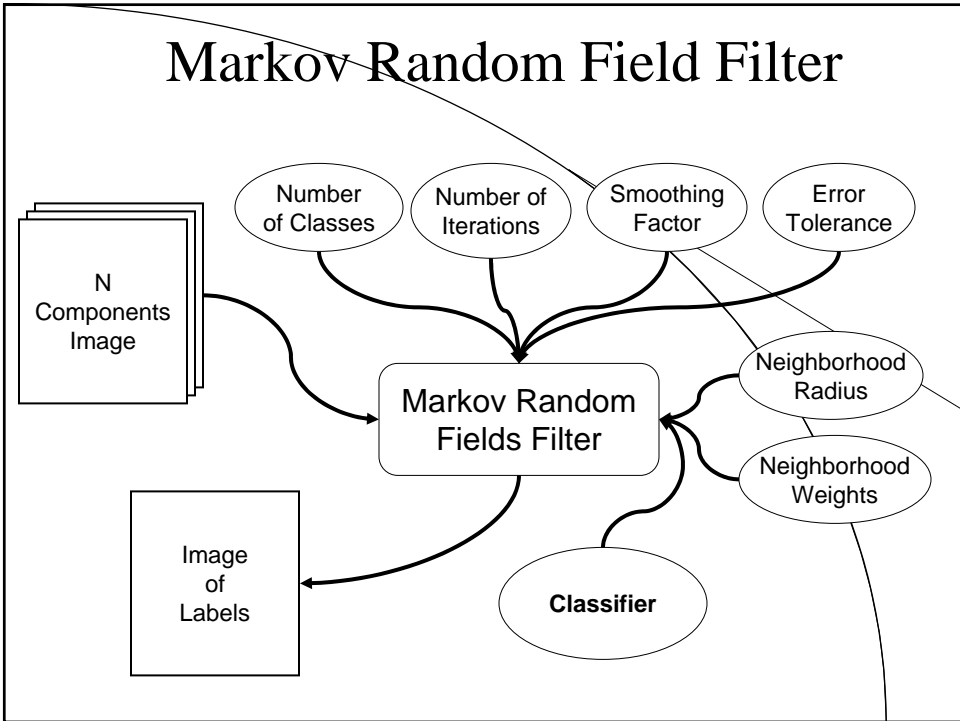
```
Locations of the Means in the N-components space
Image of labels associated to classes
```

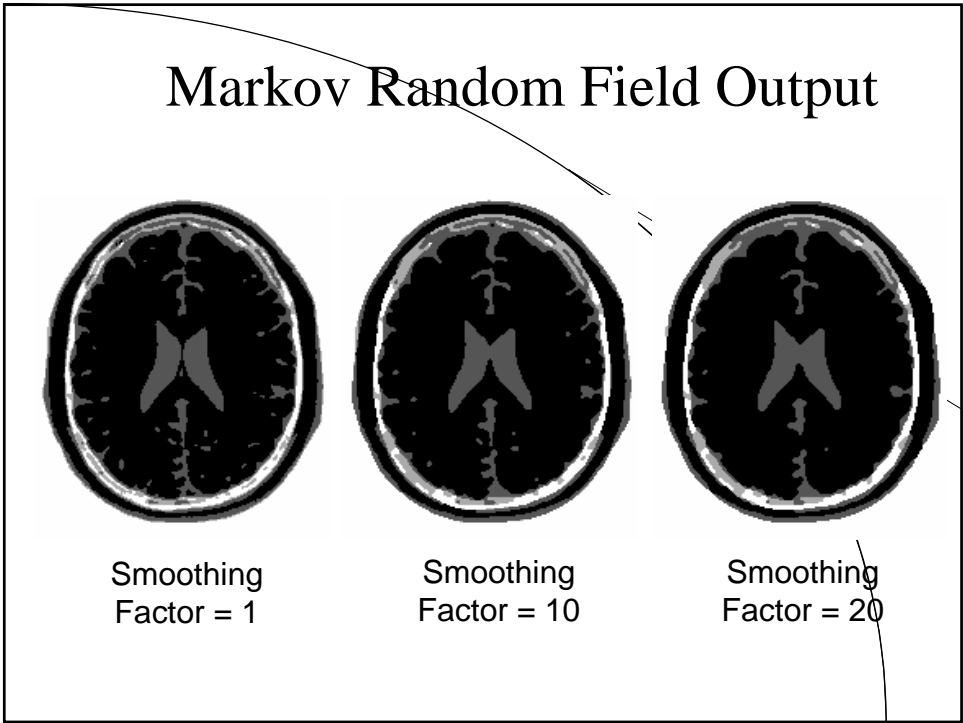
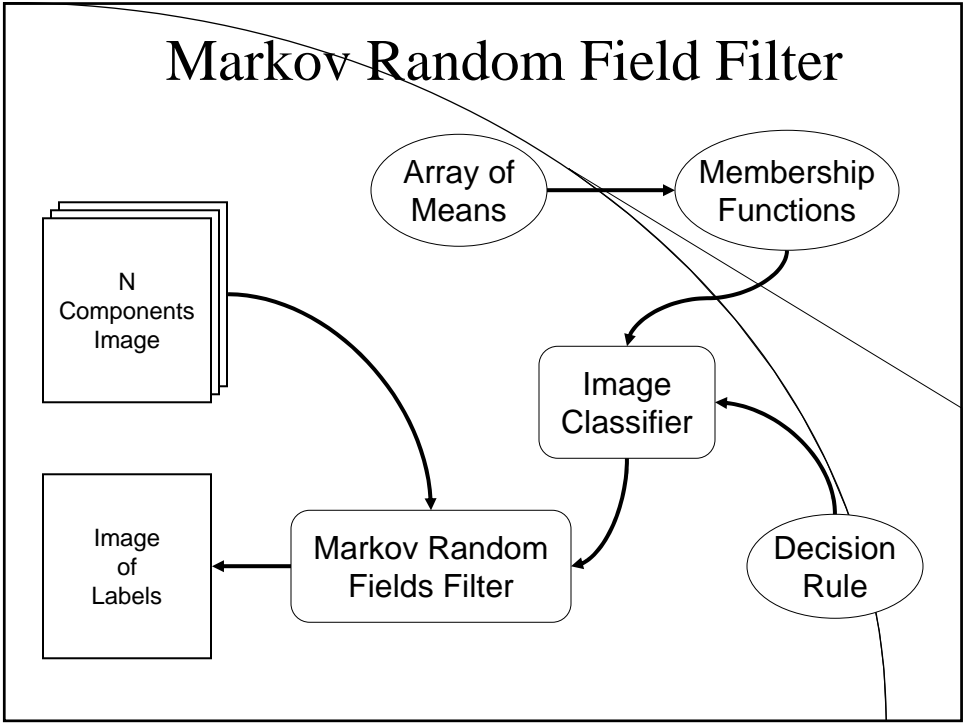
## K-Means Classification Result



Exercise 36c

# Markov Random Fields





## Markov Random Field Classification

```
#include "itkImage.h"
#include "itkImageFileReader.h"
#include "itkImageFileWriter.h"

#include "itkMRFFilter.h"
#include "itkDistanceToCentroidMembershipFunction.h"
#include "itkMinimumDecisionRule.h"
#include "itkImageClassifierBase.h"

int main( int argc, char * argv[] )
{

typedef itk::Vector< unsigned char, 3 > PixelType;
typedef itk::Image< PixelType, 2 > ImageType;
```

## Markov Random Field Classification

```
typedef itk:: MRFFilter< ImageType > MarkovFilterType;

MarkovFilterType::Pointer markovFilter = MarkovFilterType::New();

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

markovFilter->SetNumberOfClasses( numberOfClasses );
markovFilter->SetMaximumNumberOfIterations( maxNumberIterations );
markovFilter->SetSmoothingFactor( smoothingFactor );
markovFilter->SetErrorTolerance( 1e-7 );
markovFilter->SetNeighborhoodRadius( 1 );
markovFilter->SetMRFNeighborhoodWeight( weights );

markovFilter->Classifier( classifier );

writer->SetInput( markovFilter->GetOutput() );
```

## Markov Random Field Classification

```
typedef itk:: ImageClassifierBase< ImageType,  
                                LabelImageType >  ClassifierType;  
  
ClassifierType::Pointer classifier = ClassifierType::New();  
  
classifier->SetDecisionRule( reader->GetOutput() );  
  
typedef itk:: Statistics:: DistanceToCentroidMembershipFunction<  
                                PixelType >  MembershipFunctionType;  
  
MembershipFunctionType::Pointer membershipFunction;  
    MembershipFunctionType ::New();  
  
for( unsigned int cc = 0; cc < numberOfClasses; ++cc )  
{  
    membershipFunction = MembershipFunctionType ::New();  
    membershipFunction ->SetCentroid( classMean[cc] );  
    classifier->AddMembershipFunction( membershipFunction );  
}
```

## Markov Random Fields

Run Command:

```
ImageMarkovRandomField  
brainwebSlice94.vtk labelsImage.vtk  
outputLabelsImage.vtk 50 1  
4 means....
```

Output:

Image of labels smoothed

Termination criteria:

(max num iterations or convergence)

