# OsiriX Segmentation Plugin

## Systems Development Project (SEP) Final Presentation

**July 28, 2009**

Brian Jensen

Computer Aided Medical Procedures (CAMP),
Technische Universität München, Germany

Department of Nuclear Medicine
University Hospital Rechts der Isar, Germany

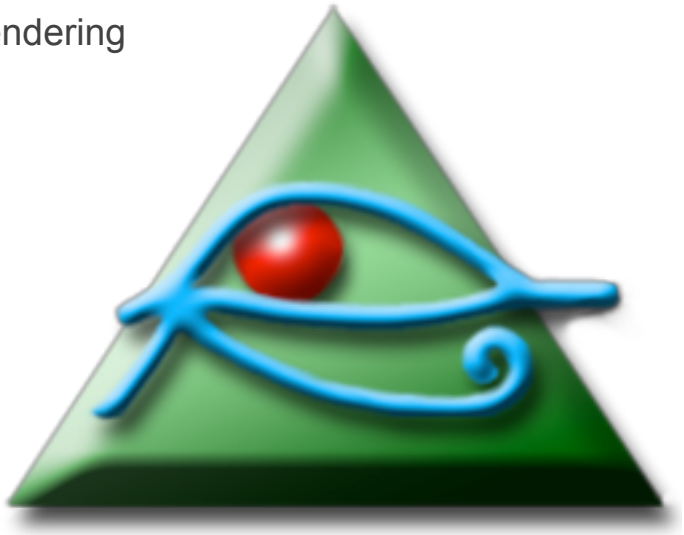
# Contents

# Introduction

- Systems Development Project in Computer Science
- Development of a segmentation plugin for the Osirix DICOM viewer
  - Familiarization with OsiriX and Mac Objective-C
  - Implementation of an appropriate user interface
  - Implementation of selected semiautomatic segmentation methods

# OsiriX Background Information

- DICOM image viewer for OS X
- Open source
- Mostly implemented in Objective-C
  - Uses several open source frameworks (ITK, VTK, DCM)
- Extendable plugin architecture
- Advanced visualization capabilities
  - 3D MIP and volume rendering
  - Fused images



# Project Goals

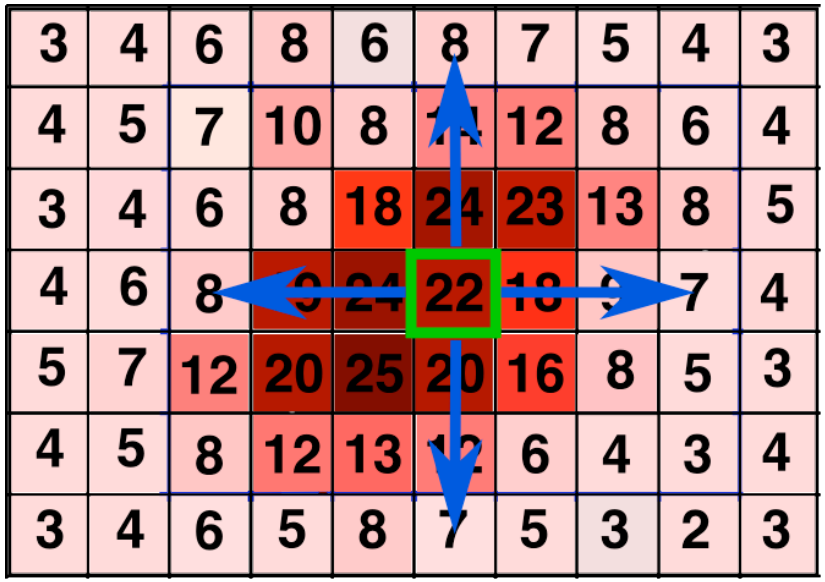
OsiriX lacks segmentation capabilities for fused images (PET/CT, PET/MR)

Implement a plugin that offers a few semiautomatic segmentation algorithms using the PET / SPECT data from a fused image

- Should be able to easily switch between algorithms
- Seamless integration with OsiriX workflow
- Should provides methods for easily adding additional algorithms

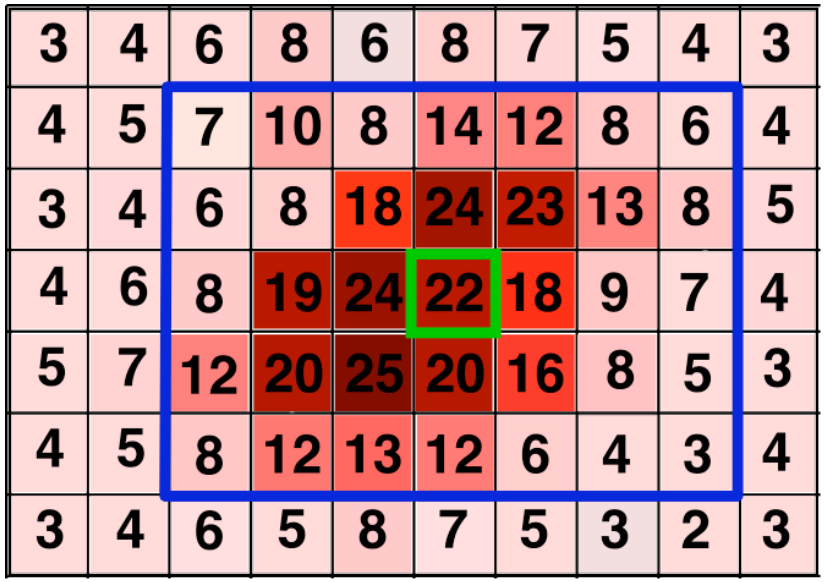
# Segmentation Overview

- Plugin offers four different semiautomatic segmentation algorithms
- All algorithms are region growing variants
  - Region growing algorithms start with a user specified seed point
  - Segmented region is “grown” by including neighboring voxels that are determined to be connected
  - All implementations differ in their inclusion criterion



# Connected Thresholding 1

- Connected thresholding uses upper and lower thresholds to determine inclusion
  - After seed point selection the surrounding region is searched for the maximum value
  - Upper threshold is set to this value
  - Lower threshold is set to a percentage of the upper threshold (here 45%)



# Connected Thresholding Example

- Segmentation example result using connected thresholding

Cut off percent: 45%

Upper threshold: 25

Lower threshold: 11.25

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 3 | 4 | 6 | 8 | 6 | 8 | 7 | 5 | 4 | 3 |
| 4 | 5 | 7 | 10 | 8 | 14 | 12 | 8 | 6 | 4 |
| 3 | 4 | 6 | 8 | 18 | 24 | 23 | 13 | 8 | 5 |
| 4 | 6 | 8 | 19 | 24 | 22 | 18 | 9 | 7 | 4 |
| 5 | 7 | 12 | 20 | 25 | 20 | 16 | 8 | 5 | 3 |
| 4 | 5 | 8 | 12 | 13 | 12 | 6 | 4 | 3 | 4 |
| 3 | 4 | 6 | 5 | 8 | 7 | 5 | 3 | 2 | 3 |

- Seed Point
- Segmented Region

# Neighborhood Connected Thresholding

- Same principal as connected thresholding with one major difference
  - Voxel is only included if all of its neighbors are also within the threshold interval
- Segmentation example result with neighborhood connected thresholding

Cut off percent: 45%

Upper threshold: 25

Lower threshold: 11.25

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 3 | 4 | 6 | 8 | 6 | 8 | 7 | 5 | 4 | 3 |
| 4 | 5 | 7 | 10 | 8 | 14 | 12 | 8 | 6 | 4 |
| 3 | 4 | 6 | 8 | 18 | 24 | 23 | 13 | 8 | 5 |
| 4 | 6 | 8 | 19 | 24 | 22 | 18 | 9 | 7 | 4 |
| 5 | 7 | 12 | 20 | 25 | 20 | 16 | 8 | 5 | 3 |
| 4 | 5 | 8 | 12 | 13 | 12 | 6 | 4 | 3 | 4 |
| 3 | 4 | 6 | 5 | 8 | 7 | 5 | 3 | 2 | 3 |

- Seed Point
- Segmented Region

# Confidence Connected

- Confidence connected uses dynamic thresholding for including voxels into the region
  - Thresholds are set to a confidence interval around the average of the current region
  - $I(X) \in [m - f\sigma, m + f\sigma]$ where *m* is the average in the region, $\sigma$ is the standard deviation, and *f* a constant factor specified by the user
- Algorithm starts with an initial radius size that is used to calculate $\sigma$ and *m* for the thresholds
- Once the region growing is completed, $\sigma$ and *m* are recalculated with the values from the new region and the process is repeated
- Algorithm terminates once the user specified number of iterations have run

# Confidence Connected Example 1

Example with 2 iterations, an initial radius of 1, and a constant factor *f* of 1.5

Iteration 1

$\sigma$ = 3.4

m = 21.1

Interval:

[16, 26]

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 3 | 4 | 6 | 8 | 6 | 8 | 7 | 5 | 4 | 3 |
| 4 | 5 | 7 | 10 | 8 | 14 | 12 | 8 | 6 | 4 |
| 3 | 4 | 6 | 8 | 18 | 24 | 23 | 13 | 8 | 5 |
| 4 | 6 | 8 | 19 | 24 | 22 | 18 | 9 | 7 | 4 |
| 5 | 7 | 12 | 20 | 25 | 20 | 16 | 8 | 5 | 3 |
| 4 | 5 | 8 | 12 | 13 | 12 | 6 | 4 | 3 | 4 |
| 3 | 4 | 6 | 5 | 8 | 7 | 5 | 3 | 2 | 3 |

- Seed Point
- Initial Radius

# Confidence Connected Example 1

Iteration 2

$\sigma$ = 3.2

m = 20.8

Interval:

[16, 26]

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 3 | 4 | 6 | 8 | 6 | 8 | 7 | 5 | 4 | 3 |
| 4 | 5 | 7 | 10 | 8 | 14 | 12 | 8 | 6 | 4 |
| 3 | 4 | 6 | 8 | 18 | 24 | 23 | 13 | 8 | 5 |
| 4 | 6 | 8 | 19 | 24 | 22 | 18 | 9 | 7 | 4 |
| 5 | 7 | 12 | 20 | 25 | 20 | 16 | 8 | 5 | 3 |
| 4 | 5 | 8 | 12 | 13 | 12 | 6 | 4 | 3 | 4 |
| 3 | 4 | 6 | 5 | 8 | 7 | 5 | 3 | 2 | 3 |

- Seed Point
- Segmentation result from the previous iteration

# Confidence Connected Example 1

Example 1 segmentation result

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 3 | 4 | 6 | 8 | 6 | 8 | 7 | 5 | 4 | 3 |
| 4 | 5 | 7 | 10 | 8 | 14 | 12 | 8 | 6 | 4 |
| 3 | 4 | 6 | 8 | 18 | 24 | 23 | 13 | 8 | 5 |
| 4 | 6 | 8 | 19 | 24 | 22 | 18 | 9 | 7 | 4 |
| 5 | 7 | 12 | 20 | 25 | 20 | 16 | 8 | 5 | 3 |
| 4 | 5 | 8 | 12 | 13 | 12 | 6 | 4 | 3 | 4 |
| 3 | 4 | 6 | 5 | 8 | 7 | 5 | 3 | 2 | 3 |

- Seed Point
- Segmented Region

# Confidence Connected Example 2

Second Example with 2 iterations, initial radius of 1, and a constant factor of 2.5

Iteration 1

$\sigma = 8.5$

m = 21.1

Interval:

[13, 30]

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 3 | 4 | 6 | 8 | 6 | 8 | 7 | 5 | 4 | 3 |
| 4 | 5 | 7 | 10 | 8 | 14 | 12 | 8 | 6 | 4 |
| 3 | 4 | 6 | 8 | 18 | 24 | 23 | 13 | 8 | 5 |
| 4 | 6 | 8 | 19 | 24 | 22 | 18 | 9 | 7 | 4 |
| 5 | 7 | 12 | 20 | 25 | 20 | 16 | 8 | 5 | 3 |
| 4 | 5 | 8 | 12 | 13 | 12 | 6 | 4 | 3 | 4 |
| 3 | 4 | 6 | 5 | 8 | 7 | 5 | 3 | 2 | 3 |

- Seed Point
- Initial Radius

# Confidence Connected Example 2

Iteration 2

$\sigma$ = 12.5

m = 19.2

Interval:

[6.7, 31.7]

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 3 | 4 | 6 | 8 | 6 | 8 | 7 | 5 | 4 | 3 |
| 4 | 5 | 7 | 10 | 8 | 14 | 12 | 8 | 6 | 4 |
| 3 | 4 | 6 | 8 | 18 | 24 | 23 | 13 | 8 | 5 |
| 4 | 6 | 8 | 19 | 24 | 22 | 18 | 9 | 7 | 4 |
| 5 | 7 | 12 | 20 | 25 | 20 | 16 | 8 | 5 | 3 |
| 4 | 5 | 8 | 12 | 13 | 12 | 6 | 4 | 3 | 4 |
| 3 | 4 | 6 | 5 | 8 | 7 | 5 | 3 | 2 | 3 |

- Seed Point
- Segmentation result from previous iteration

# Confidence Connected Example 2

Second example segmentation result

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 3 | 4 | 6 | 8 | 6 | 8 | 7 | 5 | 4 | 3 |
| 4 | 5 | 7 | 10 | 8 | 14 | 12 | 8 | 6 | 4 |
| 3 | 4 | 6 | 8 | 18 | 24 | 23 | 13 | 8 | 5 |
| 4 | 6 | 8 | 19 | 24 | 22 | 18 | 9 | 7 | 4 |
| 5 | 7 | 12 | 20 | 25 | 20 | 16 | 8 | 5 | 3 |
| 4 | 5 | 8 | 12 | 13 | 12 | 6 | 4 | 3 | 4 |
| 3 | 4 | 6 | 5 | 8 | 7 | 5 | 3 | 2 | 3 |

- Seed Point
- Segmented Region

# Gradient Magnitude Thresholding

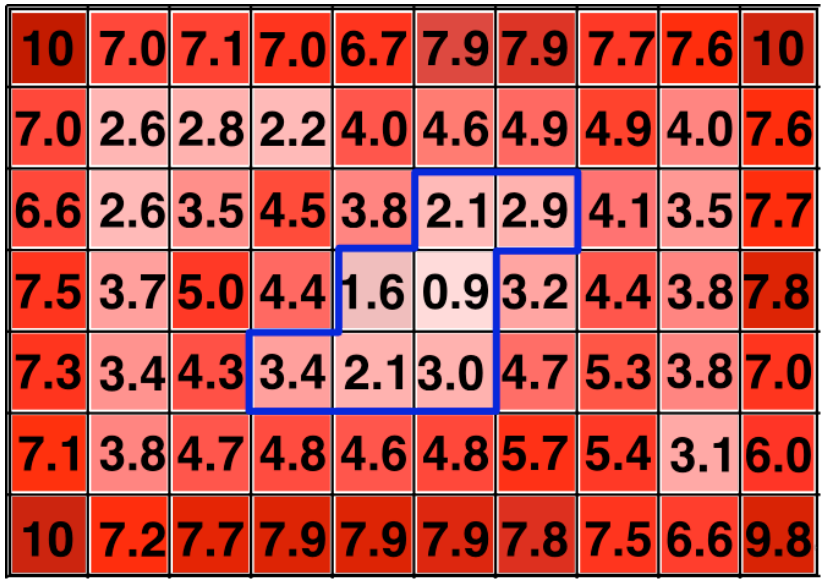
- Use image gradient magnitude for region inclusion criterion
- Tumor regions exhibit large intensity drop off at their borders
  - Only include voxels whose gradient is below a user specified threshold are included into the region
- Image gradient estimated with 3-D Sobel operator

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 18 | 23 | 32 | 35 | 40 | 48 | 48 | 36 | 25 | 20 |
| 18 | 12 | 18 | 18 | 46 | 62 | 62 | 47 | 25 | 25 |
| 19 | 13 | 29 | 54 | 63 | 39 | 46 | 48 | 25 | 29 |
| 24 | 23 | 51 | 68 | 31 | 18 | 54 | 52 | 26 | 28 |
| 25 | 22 | 47 | 53 | 39 | 53 | 60 | 44 | 20 | 20 |
| 22 | 23 | 41 | 53 | 62 | 60 | 51 | 31 | 12 | 13 |
| 18 | 24 | 34 | 46 | 50 | 45 | 32 | 19 | 14 | 13 |

- Neighborhood connected thresholding result

# Gradient Magnitude Thresholding

- Gradient magnitudes increase with image intensity values
  - Absolute Intensity values in tumor regions are quite varying
- Weight the gradient against the average intensity value in the Sobel operator's region size




- Neighborhood connected thresholding result

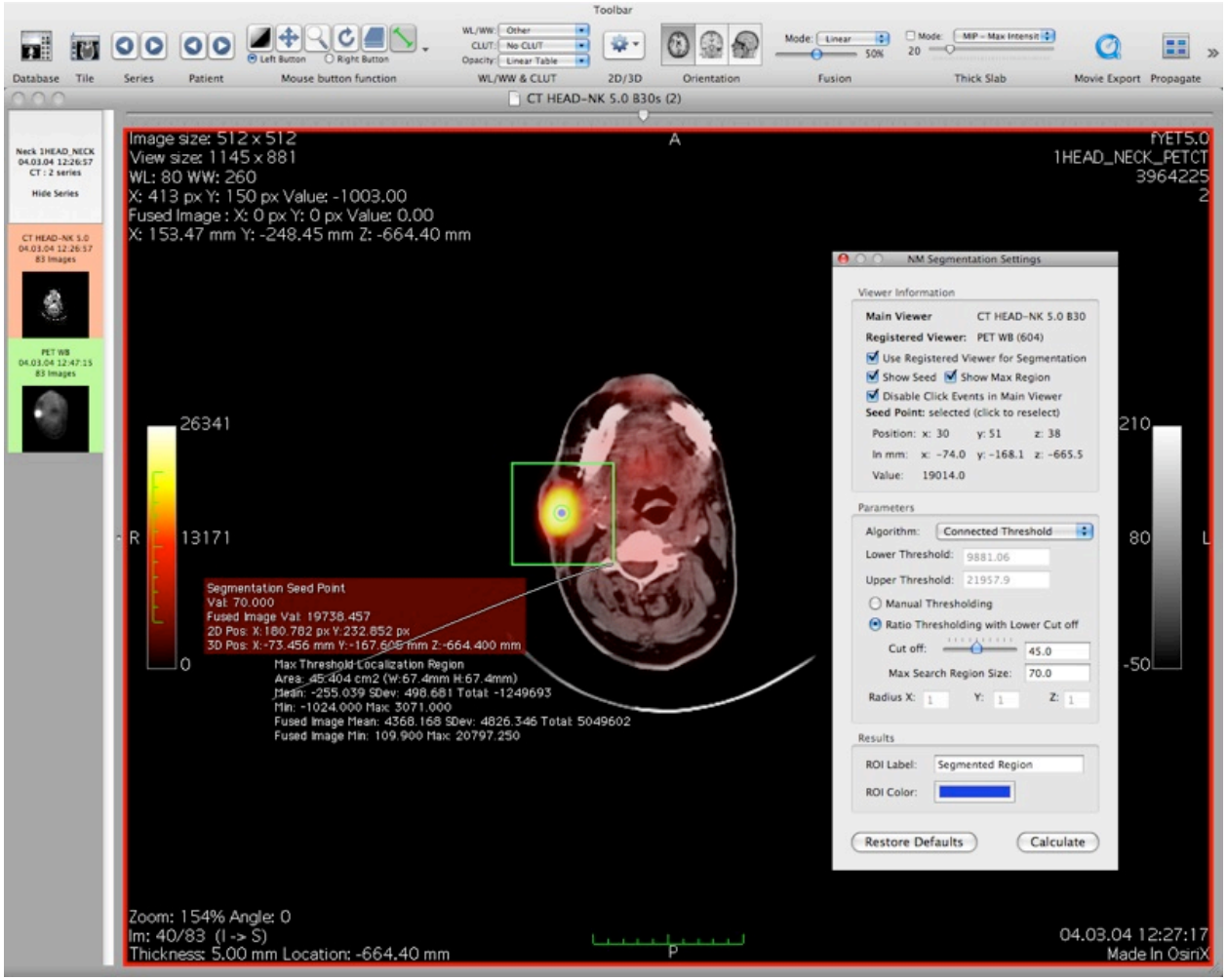
# Gradient Magnitude Thresholding Example

Example of the gradient magnitude thresholding algorithm with a threshold of 4.0

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 3 | 4 | 6 | 8 | 6 | 8 | 7 | 5 | 4 | 3 |
| 4 | 5 | 7 | 10 | 8 | 14 | 12 | 8 | 6 | 4 |
| 3 | 4 | 6 | 8 | 18 | 24 | 23 | 13 | 8 | 5 |
| 4 | 6 | 8 | 19 | 24 | 22 | 18 | 9 | 7 | 4 |
| 5 | 7 | 12 | 20 | 25 | 20 | 16 | 8 | 5 | 3 |
| 4 | 5 | 8 | 12 | 13 | 12 | 6 | 4 | 3 | 4 |
| 3 | 4 | 6 | 5 | 8 | 7 | 5 | 3 | 2 | 3 |

- Seed Point
- Segmented Region

# Demonstration



# Plugin Development Configuration

- Several prerequisites
  - Current copy of the OsiriX source code
  - ITK source code version 3.12 or 3.14:
- Refactor ITK namespace before compiling using refactor script
  - Avoid symbol collisions between OsiriX's and the plugin's ITK version
- Create new plugin project with the OsiriX plugin generator script
- Set symbolic links to the OsiriX source code and plugin's ITK libraries
- Configure search paths in plugin's Xcode project
- Add post build script that links current executable target to the OsiriX plugin directory
- Add OsiriX to executables list
- Build and go!

# Plugin Implementation

- Always use unique names for classes and other plugin resources!
- Use Interface Builder for creating the user interface
  - Bind non volatile elements to the shared user defaults controller
- Extract image volume from current viewer
  - (float*) [ViewerController volumePtr]
- Create ITK Image object using ImportImageFilter
- Catch mouse click events
  - NSNotificationCenter nc = [NSNotificationCenter defaultCenter]; [nc addObserver: self selector: @selector(mouseViewerDown:) name: @"mouseDown" object: nil];
- Convert coordinates between registered and main viewer
  - convertedPoint = [[registeredViewer imageView] ConvertFromGL2GL:myPoint toView:[mainViewer imageView]];

# Plugin Implementation

- Create ROI from segmentation results (**buff** contains volume with segmentation mask)

```
ROI *theNewROI = [[ROI alloc] initWithTexture:buff
   textWidth:buffWidth
   textHeight:buffHeight
   textName:roiText
   positionX:0
   positionY:0
   spacingX:[[[mainViewer imageView] curDCM]
   pixelSpacingX]spacingY:[[[mainViewer imageView] curDCM]
                                pixelSpacingY]
   imageOrigin:NSMakePoint([[[mainViewer imageView] curDCM] originX],
                [[[mainViewer imageView] curDCM] originY])
   ];

[[[mainViewer roiList] objectAtIndex:i] addObject:theNewROI];
```

# Conclusion

- All four segmentation algorithms have different results
- Connected Thresholding offers the best subjective results
  - Thorough evaluation still needed
- Gradient magnitude thresholding still needs improvement
  - Different method for gradient estimation
  - Include border voxels into region
- OsiriX is a very capable environment for image processing plugins
  - Easily create a user interface and visualization for segmentation algorithms

# Any Questions? Ideas?

Thanks for your attention!

# References

1. Luis Ibanez, Will Schroeder, Lydia Ng, Josh Cates, and Insight Consortium. *The ITK Software Guide Second Edition*. Kitware Inc, November 2005.
2. Irwin Sobel. An isotropic 3x3x3 gradient volume gradient operator. August 1996

For more detailed information see:
http://campar.in.tum.de/Students/SepOsiriXSegmentation