cmrep Documentation

Release 1.0.0

Paul A. Yushkevich

Dec 03, 2021

Contents:

1 About CM-Rep **1**

2 Compiling CM-Rep **3**
- 2.1 Required Packages . . . 3
- 2.2 Building CM-Rep . . . 5

3 CM-Rep QuickStart **7**
- 3.1 "Brute Force" CM-Rep Model . . . 7
- 3.2 PDE-Based CM-Rep Model . . . 8

4 Indices and tables **11**

CHAPTER 1

About CM-Rep

CM-Rep is a method for representing 3D biological objects (e.g, lungs, hippocampi) using a deformable model with predefined medial axis geometry. It allows the shape of these objects to be analyzed in terms of features derived from the medial axis, such as regional thickness. For more about the cm-rep methods, see these citations:

- Yushkevich, P.A., Zhang, H. and Gee, J.C., 2006. Continuous medial representation for anatomical structures. IEEE transactions on medical imaging, 25(12), pp.1547-1564. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4016176
- Yushkevich, P.A., 2009. Continuous medial representation of brain structures using the biharmonic PDE. NeuroImage, 45(1), pp.S99-S110. http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2680440/
- Yushkevich, P.A., Zhang, H., Simon, T.J. and Gee, J.C., 2008. Structure-specific statistical mapping of white matter tracts. NeuroImage, 41(2), pp.448-461. http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2519052/
- Pouch, A.M., Tian, S., Takebe, M., Yuan, J., Gorman, R., Cheung, A.T., Wang, H., Jackson, B.M., Gorman, J.H., Gorman, R.C. and Yushkevich, P.A., 2015. Medially constrained deformable modeling for segmentation of branching medial structures: Application to aortic valve segmentation and morphometry. Medical image analysis, 26(1), pp.217-231. http://www.sciencedirect.com/science/article/pii/S1361841515001401
- Yushkevich, P.A., Aly, A., Wang, J., Xie, L., Gorman, R.C., Younes, L. and Pouch, A.M., 2019, June. Diffeomorphic Medial Modeling. In International Conference on Information Processing in Medical Imaging (pp. 208-220). Springer, Cham. https://arxiv.org/pdf/1902.02371

CHAPTER 2

Compiling CM-Rep

These instructions should work (with minor modifications) for MacOS and Linux. I have never tried building on Windows.

2.1 Required Packages

2.1.1 ITK

You will need to download and compile ITK v4.13.2. Use the following flags when compiling:

- `CMAKE_CXX_FLAGS: -Wno-deprecated -std=c++11`
- `CMAKE_C_FLAGS: -Wno-deprecated`

2.1.2 VTK

You will need to download and compile VTK v6.3.0. Use the following flags

- `CMAKE_CXX_FLAGS: -Wno-deprecated -std=c++11`
- `CMAKE_C_FLAGS: -Wno-deprecated`
- `BUILD_SHARED_LIBS: FALSE`
- `VTK_REQUIRED_OBJCXX_FLAGS:`

2.1.3 IPOPT and HSL

These constrained optimization libraries are required for compiling the newer (2013 IPMI and 2019 IPMI) programs, can be omitted for older ones.

- Download IPOPT and HSL libraries as described in https://www.coin-or.org/Ipopt/documentation/

- Recommend downloading the **Full** HSL package (for academic use only, must receive license)
- When compiling HSL on the Mac, I used the following configuration options. Adapt the paths to your system:

```
./configure F77=/usr/local/Cellar/gcc/4.9.2_1/bin/gfortran-4.9 FFLAGS=-
↪fexceptions -m64 -fbackslash F90=/usr/local/Cellar/gcc/4.9.2_1/bin/gfortran-4.9␣
↪FC=/usr/local/Cellar/gcc/4.9.2_1/bin/gfortran-4.9 --prefix=/Users/pauly/tk/
↪ipopt/install_hsl
make
make install
```

- When compiling HSL on recent Linux, the following suffice:

```
./configure --prefix=/home/pauly/tk/ipopt/install_hsl
make
make install
```

- Before compiling IPOPT, enter the `ThirdParty` directory and run `get.XXX` scripts in the directories `Blas`, `Lapack`, `ASL`, `Mumps` and `Metis`
- When compiling IPOPT on the Mac, I used the following command line:

```
./configure --prefix=/Users/pauly/tk/ipopt/install64_so F77=/usr/local/Cellar/gcc/
↪4.9.2_1/bin/gfortran FFLAGS=-fexceptions -m64 -fbackslash CFLAGS=-fno-common -
↪no-cpp-precomp -fexceptions -arch x86_64 -m64 CFLAGS=-fno-common -no-cpp-
↪precomp -fexceptions -arch x86_64 -m64 CXXFLAGS=-fno-common -no-cpp-precomp -
↪fexceptions -arch x86_64 -m64 --with-hsl=/Users/pauly/tk/ipopt/install_hsl/lib/
↪libcoinhsl.a
make
make install
```

- For Linux:

```
./configure --prefix=/home/pauly/tk/ipopt/install --with-hsl=/home/pauly/tk/ipopt/
↪install_hsl
make
make install
```

2.1.4 TETGEN

- Download from http://wias-berlin.de/software/tetgen/
- Build using `CMake`

2.1.5 NLOPT

This optimization library is needed for the IPMI 2019 method only * Download from https://nlopt.readthedocs.io/en/latest/ * Build using CMake

2.1.6 PARDISO

Sparse solver, needed for PDE-based cm-rep programs (2005 IPMI, 2008 NeuroImage) * Download from https://www.pardiso-project.org/ * Get license and follow instructions for how to use it * The easiest is to put the library into `/usr/local/lib`

2.2 Building CM-Rep

- Create a build directory and run ccmake there
- Set the following flags to `ON`:
  - `USE_IPOPT` (recommended, use for 2013 and 2019 boundary-first methods)
  - `USE_HSL` (recommended, use for 2013 and 2019 boundary-first methods)
  - `USE_TETGEN`
  - `USE_NLOPT` (recommended, use for 2019 boundary-first method)
  - `USE_PARDISO` (optional, use for 2006 and 2008 PDE-based method)
- Set the necessary library and include paths:
  - `ITK_DIR` (point to ITK build directory)
  - `VTK_DIR` (point to VTK build directory)
  - `IPOPT_LIBRARY` (e.g., `/home/pauly/tk/ipopt/install/lib/libipopt.so`)
  - `IPOPT_INCLUDE_DIR` (e.g., `/home/pauly/tk/ipopt/install/include/coin`)
  - `IPOPT_HSL_LIBRARY` (e.g., `/home/pauly/tk/ipopt/install_hsl/libcoinhsl.so`)
  - `IPOPT_GFORTRAN_LIB` (point to system `gfortran`, e.g., `/usr/lib/gcc/x86_64-linux-gnu/4.9/libgfortran.so`)
  - `IPOPT_BLAS_LIB` (point to BLAS, e.g., `/home/pauly/tk/ipopt/install/lib/libcoinblas.so`)
  - `TETGEN_LIBRARY:` (e.g., `/home/pauly/tk/tetgen/build/libtet.a`)
  - `TETGEN_INCLUDE_DIR:` (e.g., `/home/pauly/tk/tetgen`)
  - `NLOPT_LIBRARIES:` (e.g., `/home/pauly/tk/nlopt/build/lib/libnlopt.so`)
  - `NLOPT_INCLUDE_DIRS:` (e.g., `/home/pauly/tk/nlopt/include`)
- When using PARDISO:
  - `PARDISO_LIB` (point to the shared library)
  - `LAPACK_LIB` (point to the system Lapack)
  - `GOMP_LIB` (point to the system GOMP: GNU OpenMP library)
- Set compilation flags:
  - `CMAKE_CXX_FLAGS`: set to match those for ITK/VTK, i.e., `-Wno-deprecated -std=c++11`
  - `CMAKE_C_FLAGS`: set to match those for ITK/VTK, i.e., `-Wno-deprecated`
- Run `make`

CHAPTER 3

CM-Rep QuickStart

This page uses test data included with the cm-rep package to run simple examples. There are different sections for different cm-rep models.

In all examples, we assume you are running in the directory where you compiled cm-rep and the source code directory is `$CMREP_SRC`.

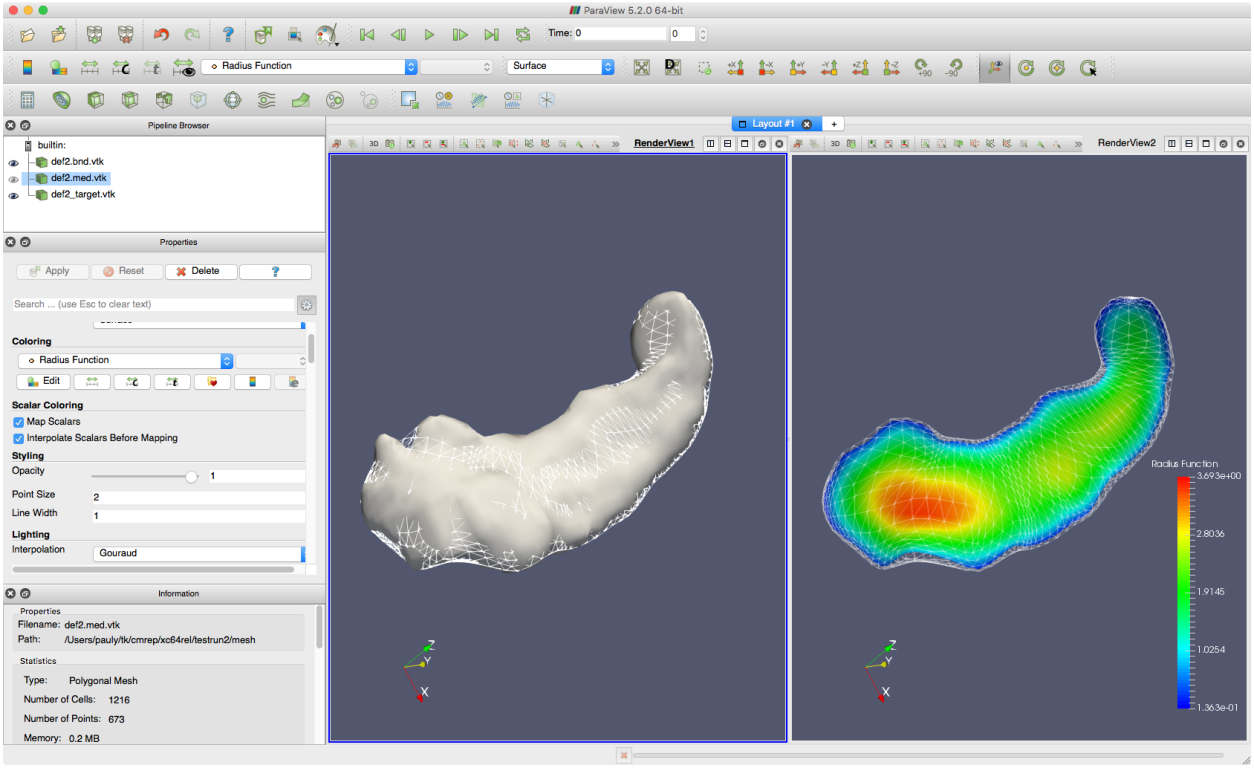
3.1 "Brute Force" CM-Rep Model

This model (circa 2006, used in 2008 NeuroImage tract-specific analysis paper) handles medial geometry constraints using soft penalties. It does not require any PDE solvers. It is also relatively fast.

This example will fit a cm-rep model of the hippocampus to a binary image:

```
./cmrep_fit $CMREP_SRC/testing/t001_param.txt \
  $CMREP_SRC/testing/t001_template_brute.cmrep \
  $CMREP_SRC/testing/t001_img_binary.nii.gz \
  test_brute
```

Fitting is done in four stages (align by moments, affine, coarse deformable, fine deformable). Output cm-rep models are in `test_brute/cmrep` and full-resolution medial and boundary meshes are in `test_brute/mesh`. Below the fitting is visualized in ParaView.



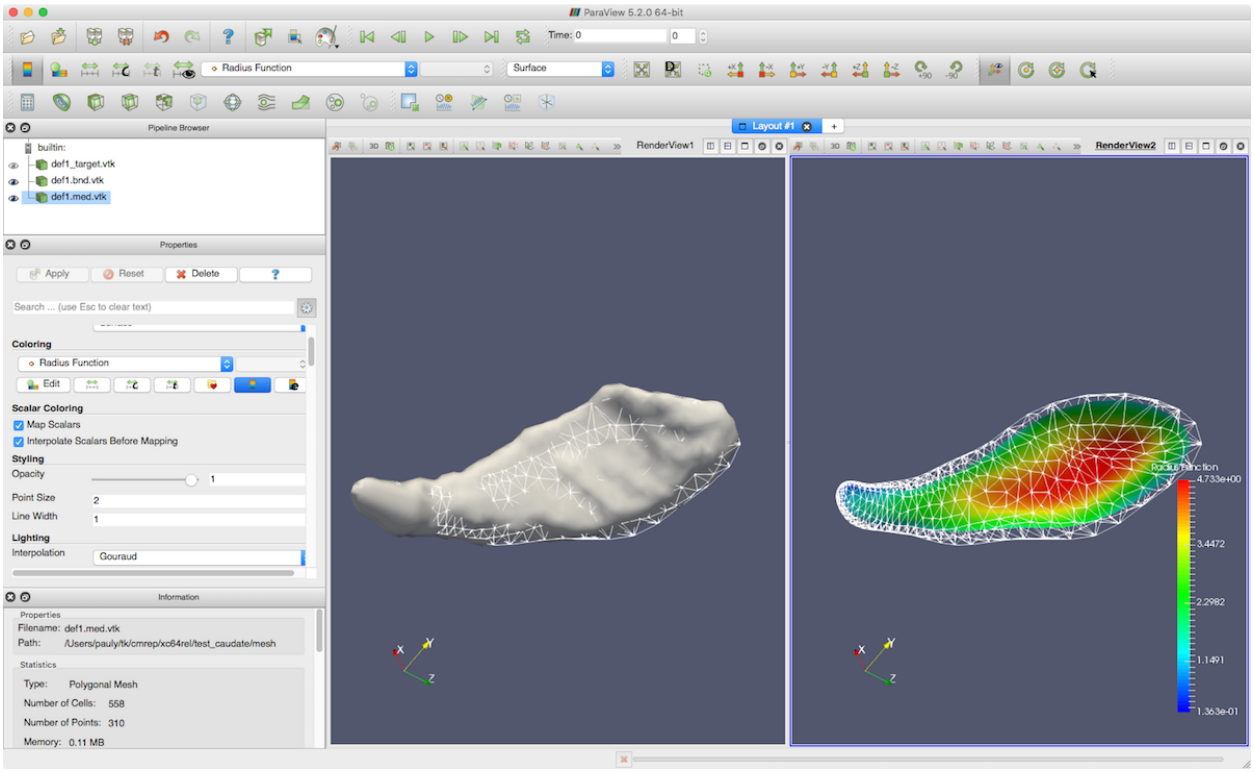
3.2 PDE-Based CM-Rep Model

This model from the 2008 NeuroImage paper uses the biharmonic PDE to implement cm-rep geometric constraints. To use it, you must enable `PARDISO` and have an up to date PARDISO license.

This example will fit a cm-rep model of a caudate to a binary image:

```
./cmrep_fit $CMREP_SRC/testing/caudate_pde_param.txt \
  $CMREP_SRC/testing/caudate_pde_model.cmrep \
  $CMREP_SRC//testing/caudate_target.nii.gz \
  test_caudate
```

Fitting is done in two stages (align by moments, deformable). Output cm-rep models are in `test_brute/cmrep` and full-resolution medial and boundary meshes are in `test_brute/mesh`. Below the fitting is visualized in ParaView.


ParaView 5.2.0 64-bit
Radius Function
Surface
Pipeline Browser
builtin:
def1_target.vtk
def1.bnd.vtk
def1.med.vtk
Properties
Apply
Reset
Delete
Coloring
Radius Function
Edit
Scalar Coloring
Map Scalars
Interpolate Scalars Before Mapping
Styling
Opacity
Point Size
Line Width
Lighting
Interpolation
Gouraud
Information
Filename: def1.med.vtk
Path: /Users/pauly/tk/cmrep/xc64rel/test_caudate/mesh
Type: Polygonal Mesh
Number of Cells: 558
Number of Points: 310
Memory: 0.11 MB
Layout #1
RenderView1
RenderView2
Radius Function
4.733e+00
3.4472
2.2982
1.1491
1.363e-01

CHAPTER 4

Indices and tables

- genindex
- modindex
- search