

---

# PORTR Software Manual

Version 1.0.0

Dongjin Kwon

## Contents

<b>1</b>	<b>About the algorithm</b>	<b>2</b>
<b>2</b>	<b>Download</b>	<b>4</b>
2.1	Software License . . . . .	4
2.2	Documentation . . . . .	4
2.3	System Requirements . . . . .	4
2.4	Register for Download . . . . .	4
<b>3</b>	<b>Installation</b>	<b>5</b>
3.1	Build Dependencies . . . . .	5
3.2	Runtime Requirements . . . . .	5
3.3	Build and Installation . . . . .	5
	How to build FLIRT in Windows . . . . .	7
	How to build Tumor Simulator . . . . .	8
	How to build BrainTumorViewer . . . . .	9
<b>4</b>	<b>Manual</b>	<b>11</b>
4.1	PORTR Command and Parameters . . . . .	11
4.2	Output of PORTR . . . . .	13
4.3	Making Initializations Using BrainTumorViewer . . . . .	13
4.4	Auxiliary Commands . . . . .	15
4.5	Running Example . . . . .	15
	Warp Image . . . . .	15
	Normalize Image . . . . .	16
	Resample Image . . . . .	16
	Concatenate Two Deformation Fields . . . . .	16
	Resample Deformation Field . . . . .	17
	Reverse Deformation Field . . . . .	17
<b>5</b>	<b>Changelog</b>	<b>18</b>
<b>6</b>	<b>Publications</b>	<b>19</b>
<b>7</b>	<b>People</b>	<b>19</b>
	<b>Bibliography</b>	<b>19</b>

---

**Pre-Operative and post-Recurrence brain Tumor Registration (PORTR)** [TMI2014] is a software package designed for determining the optimal deformation between pre-operative and post-recurrence scans by finding the minimum of an energy function, which is based on the concept of symmetric registration.

Some typical applications of PORTR include,

- Mapping post-recurrence scans to pre-operative scans;
- Labeling entire brains regions of each scan.

PORTR is implemented as a command-line tool. It is semi-automatic and requires minimal user initializations. Users could use the visual interface called BrainTumorViewer to easily make initializations and a script for the execution. As a results, PORTR will output a label map of each scan, a mapping between two scans, etc.

## 1 About the algorithm

Nonrigid registration of the pre-operative an post-recurrence scans is very challenging due to large deformations, missing correspondences, and inconsistent intensity profiles between the scans. The large deformations and missing correspondences are due to the glioma in the pre-operative scans causing large mass effects as well as the resection cavities and tumor recurrence in the post-recurrence scans, which are acquired several months or years after surgery. The inconsistent intensity profiles result from tissue labeled as edema in the pre-operative scan transforming to healthy tissue in the post-recurrence scan (and vice versa). Thus, corresponding regions can have very different intensity profiles. Figure 1 shows a typical case with the anatomy around the tumor being confounded by resection cavity, tumor recurrences, and edema.

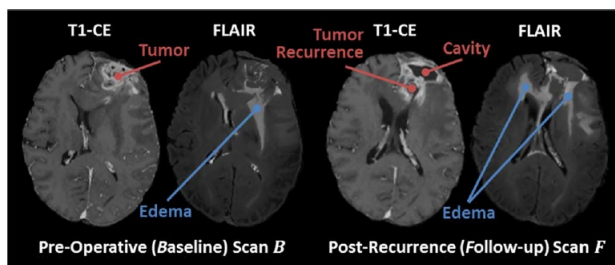


Figure 1: Example of the pre-operative scan and the corresponding post-recurrence scan. Pre-operative scan clearly shows the tumor in the T1-CE scan and edema in FLAIR scan. Edema is also clearly visible in the FLAIR scan of the post-recurrence scan . T1-CE of now shows resection cavity and tumor recurrence.

We solve this problem by the **PORTR**, introduced in [TMI2014]. PORTR determines the optimal deformation between two scans by finding the minimum of an energy function, which is based on the concept of symmetric registration. This energy function is not only comprised of image-based correspondences and smoothness constraints as customary for other registration methods, but also includes pathological information. The pathological information is inferred from the results of two segmenters that are targeted to each scan. Specifically, a new method is developed for segmenting post-recurrence scans, which generally consist of resection cavities after brain surgery and multiple tumor recurrences. For the pre-operative scans, GLISTR [TMI2012] is adapted to outline a single brain glioma which causes a large mass effect on healthy tissue. The resulting segmentations of both scans are a central component in the definition of the image and the shape-based correspondence terms within our symmetric registration framework. Determining the minimum within this framework is difficult as the function contains many local minima. PORTR deals with these difficulties by combining discrete and continuous optimizations. The discrete optimization method finds the optimal solution in a coarse solution space. The continuous optimization method locally improves this solution in a finer solution space.

PORTR requires minimal user input initializing the algorithm. Specifically, for the pre-operative scan, user provides one seed point and its approximate radius and one sample point for each tissue class. For post-recurrence scan, user

provides seed points for abnormal masses and their corresponding radius and one sample point for each tissue class. Users could use the visual interface called BrainTumorViewer to easily mark each point. Another input to PORTR consists of the preprocessed patient scans and a list of probabilistic atlas priors representing a healthy population (we use eve currently). For preprocessing, we coregistered all modalities, corrected MR field inhomogeneity, skull stripping, and scaled intensities to fit  $[0, 255]$ . Then we affinely register the post-recurrence to the pre-operative scans. The output of PORTR consists of a deformation field between pre-operative and post-recurrence scans and a label map corresponding to each scan.

The current implementation of PORTR accepts four MR modalities: T1, T1-CE, T2, and FLAIR for each scan. It segments these images into 9 labels for pre-operative scans: cerebrospinal fluid (CSF), ventricles (VT), gray matter (GM), white matter (WM), vessel (VS), edema (ED), necrosis (NCR), enhancing tumor (TU), and background (BG). For post-recurrence scans, it uses 11 labels: cerebrospinal fluid (CSF), ventricles (VT), gray matter (GM), white matter (WM), vessel (VS), edema (ED), cavity region (CAN), enhanced cavity region (CAE), necrosis for tumor recurrence (RTN), enhanced tumor recurrence (RTE), and background (BG).

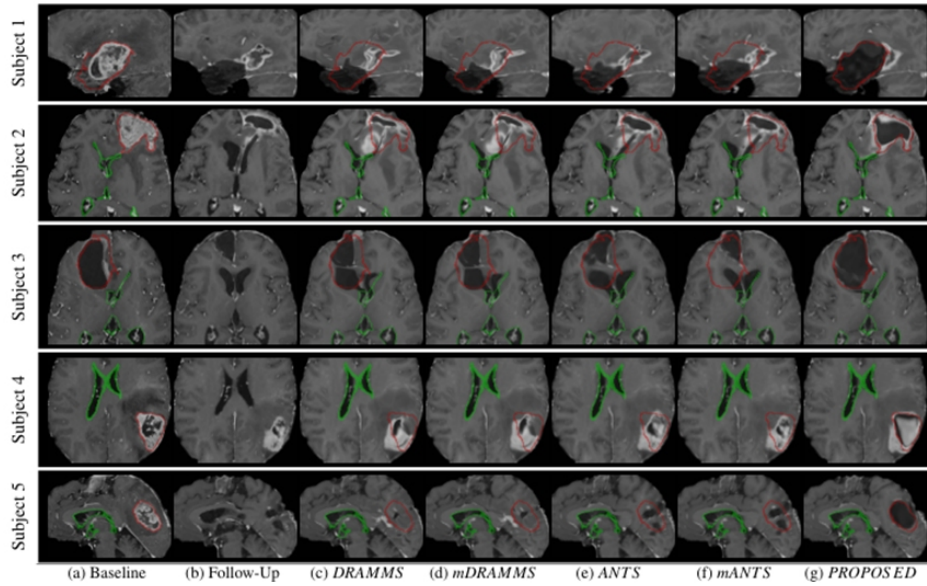


Figure 2: Registration results of follow-up onto baseline scans. In each row, we show T1-CE images of the pre-operative scan (baseline) in (a) and the post-recurrence scan (follow-up) in (b). Images (c)-(g) show the registered post-recurrence scans using DRAMMS, mDRAMMS, ANTS, mANTS, and PORTR, respectively. For baseline and registered scans, boundaries of segmented tumor (red) and ventricles (green) of baseline are overlaid.

In Figure 2, registration results from PORTR are displayed. the aligned follow-up of PORTR much better matches the baseline scan than those of other competing methods. The ventricles of the follow-up scans aligned by PORTR overlap well with the baseline across all examples.

## 2 Download

### 2.1 Software License

The PORTR software is freely available under a BSD-style open source license that is compatible with the Open Source Definition by [The Open Source Initiative](#) and contains no restrictions on use of the software. The full [license](#) text is included with the distribution package and available online.

### 2.2 Documentation

[PORTR Software Manual](#): The software manual of PORTR in PDF.

### 2.3 System Requirements

**Operating System:** Linux, Windows (64 bit)

**Memory Requirement:** 12GB or more.

### 2.4 Register for Download

Please [register online](#) to receive an email with the download links of the software.

## 3 Installation

### 3.1 Build Dependencies

Before PORTR can be build from sources, the following software libraries have to be installed.

Package	Version	Description
CMake ITK	2.8.4 or higher 3.14 or higher	To compile and build PORTR. Use version 2.8.4 or higher. For ITK 3.x, <code>ITK_USE_REVIEW</code> option is required to set <code>ON</code> . For ITK 4.x, there's no mandatory option. In Windows, build ITK using CMake with the Win64 (x64) solution platform of Visual Studio.

**Note:** To build in Windows, use CMake with the Win64 (x64) solution platform of Visual Studio. Our testing environment is Windows 7 64-bit and Visual Studio 2010.

### 3.2 Runtime Requirements

For the successful execution of PORTR, the following software packages have to be installed.

Dependency	Version	Description
FSL (FLIRT)	3.3.11 – 5.0.7	PORTR uses <a href="#">FLIRT</a> for the affine alignment. For Linux and Mac OS, installation packages are provided. In Windows, it could be built from the patched source codes on the <a href="#">Cygwin</a> environment. See the how-to guides <a href="#">How to build FLIRT in Windows</a> .
HOPSPACK	2.0.2	Used for the optimization of the tumor growth model parameters. The multithreaded version is used. <a href="#">Download</a> and install the precompiled binary.
BrainTumorModeling_CoupledSolver	1.2.0	Used for the tumor growth simulation. Note that this package depends on the <a href="#">PETSc</a> library. In Windows, it could be built on the <a href="#">Cygwin</a> environment. For details on the build of this tumor simulator, see the how-to guides <a href="#">How to build Tumor Simulator</a> .
BrainTumorViewer	1.0.0	Used for making initializations of PORTR. It is an <b>optional</b> program and not required for executing PORTR. For details on the build of this viewer, see the how-to guides <a href="#">How to build BrainTumorViewer</a> .

To build `BrainTumorModeling_CoupledSolver`, see [How to build Tumor Simulator](#). To build `FLIRT` in Windows, see [How to build FLIRT in Windows](#). To build `BrainTumorViewer`, see [How to build BrainTumorViewer](#). For `HOPSPACK`, copy the `HOPSPACK_main_threaded` executable to the appropriate directory which could be found by the `PATH` environment variable or PORTR's installation `bin` folder.

### 3.3 Build and Installation

Please follow commands below in a shell/terminal (e.g., [Bash](#)). They will configure and build PORTR using [GNU Make](#) (or Visual Studio in Windows). The main CMake configuration file (`CMakeLists.txt`) is located in the root directory of the package.

**Step 1. Extract source files and create the build directory:**

```
tar xzf portr-${version}-source.tar.gz
mkdir portr-${version}-build; cd portr-${version}-build
```

---

**Note:** In Windows, use the appropriate zip program (e.g., 7-zip) to extract.

---

## Step 2. Run CMake to configure the build tree:

```
ccmake ../portr-${version}-source
```

After the execution of this command and proceed the configure once, you will see a screen like [Figure 3](#).

---

**Note:** In Windows, use the CMake GUI instead and set the generator as Visual Studio (Win64).

---

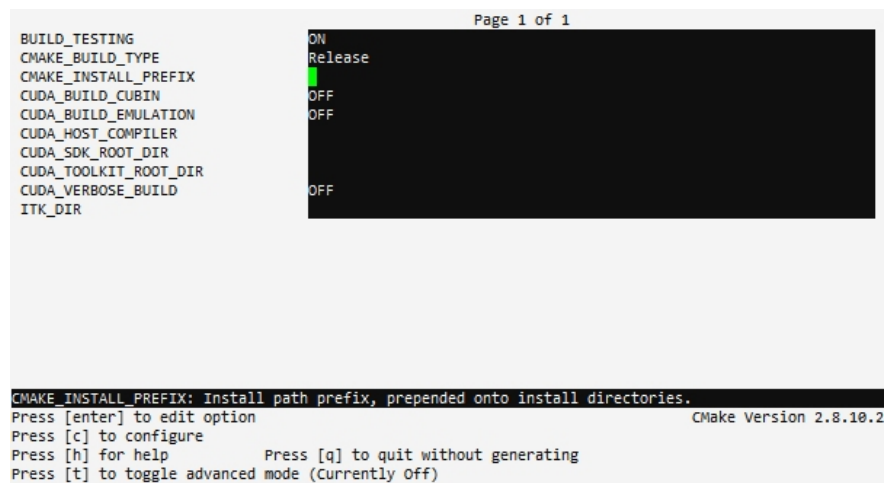


Figure 3: Configuring PORTR installation using ccmake.

In the CMake interface, follow these steps:

2.1. Change `CMAKE_INSTALL_PREFIX` to the folder you want to install PORTR into. This folder should be outside the `portr-${version}-source` folder. Make sure you have the **write** access to this folder. Change `CMAKE_BUILD_TYPE` (`CMAKE_CONFIGURATION_TYPES` in Windows) to `Release`. For CUDA options, leave as it is. Change `ITK_DIR` to the folder that ITK is installed if it fails to find it.

2.2. Keep pressing letter `c` on your keyboard until option `g` is available/displayed on the screen.

2.3. Then press `g` on your keyboard to generate the makefiles and to quit this ccmake window.

---

**Note:** In the CMake GUI, `c` and `g` correspond to `Configure` and `Generate` buttons.

---

## Step 3. Build:

```
make
```

---

**Note:** In Windows, launch the solution file of Visual Studio, select `Release` in the solution configurations (and confirm `x64` in the solution platforms), and then perform the rebuild solution.

---

## Step 4. Test (optional):

```
make test
```

**Note:** In Windows, build the `RUN_TESTS` project. To perform tests, the `BUILD_TESTING` option in the CMake configuration is required to set `ON`.

---

In case of failing tests, re-run the tests, but this time by executing `CTest` directly with the `-v` option to enable verbose output and redirect the output to a text file:

```
ctest -V >& portr-test.log
```

And send the file `portr-test.log` as attachment of the issue report to `sbia-software` at `uphs.upenn.edu`.

### Step 5. Install:

```
make install
```

---

**Note:** In Windows, build the `INSTALL` project.

---

Upon the success of the above compilation and build process, PORTR is installed into the directory specified by the `CMAKE_INSTALL_PREFIX` (set during build configuration in Step 2). The PORTR *Command-line Tools* are located in the `bin` subdirectory. Also, confirm the following programs could be found in the path or PORTR's `bin` folder: `flirt`, `convert_xfm`, `HOPSPACK_main_threaded`, and `ForwardSolverDiffusion`. If not, check *Runtime Requirements* again.

## How to build FLIRT in Windows

**FLIRT** is a part of **FSL** and it is not distributed in Windows. Here, we describe how to build **FLIRT** from the source code and run **FLIRT** in the command line in Windows. As we compile **FLIRT** on the **Cygwin** environment, install **Cygwin** before starting (confirm `gcc` version 4 is installed). After that, download FSL 3.3.11 source codes from [here](#), and copy source code patch file located in `/lib/fsl-3.3.11-sources-flirt-cygwin.patch` of the package.

then follow the steps below in the **Cygwin** terminal.

### Step 1. Extract source files:

```
tar xzf fsl-3.3.11-sources.tar.gz
```

### Step 2. Apply patch:

```
patch -d fsl -p1 < fsl-3.3.11-sources-flirt-cygwin.patch
```

---

**Note:** This patch only works on the source codes relevant to **FLIRT**. Other **FSL** modules may need additional patches.

---

### Step 3. Change to the `fsl` folder and set environmental variables:

```
cd fsl
export FSLDIR=$PWD
export FSLDEVDIR=${FSLDIR}
export FSLCONFDIR=${FSLDIR}/config
export FSLMACHTYPE=`gcc -dumpmachine`-gcc`gcc -dumpversion`
```

Make sure `gcc` is linked to the `gcc-4` binary.

### Step 4. Copy config file:

```
mkdir ./config/`gcc -dumpmachine`-gcc`gcc -dumpversion`
cp ./config/i686-pc-cygwin-gcc3.4.4/* ./config/`gcc -dumpmachine`-gcc`gcc -dumpversion`
```

### Step 5. Build:

```
./build newmat utils znzlib niftiio fslcio miscmaths newimage flirt
```

## Step 6. Install

Upon the success of the above build process, copy `fsl/bin/flirt.exe` and `fsl/bin/convert_xfm.exe` executables into the Cygwin's `bin` folder (we assume it located in `C:\cygwin\bin`). Also, add the Cygwin's `bin` folder in the system path if it is not included:

```
setx Path "%Path%;C:\cygwin\bin" /m
```

If users don't want to modify the path variable, copy `fsl/bin/flirt.exe` and `fsl/bin/convert_xfm.exe` executables into PORTR's installation `bin` folder (where `PORTR.exe` is installed). Also, copy `cygwin1.dll` and its dependent dlls (in our case, `cyggcc_s-1.dll`, `cyggfortran-3.dll`, `cygICE-6.dll`, `cygSM-6.dll`, `cygstdc++-6.dll`, `cyguid-1.dll`, `cygX11-6.dll`, `cygXau-6.dll`, `cygxcb-1.dll`, `cygXdmp-6.dll`, `cygXt-6.dll`) into that folder. Before running `flirt` in the command line, the following environmental variable is needed to be set.:

```
set FSLOUTPUTTYPE=NIFTI_GZ
```

## How to build Tumor Simulator

The following steps describe the procedure to build and install the brain tumor simulation software used in PORTR. Please refer also to the **README** file of the [BrainTumorModeling\\_CoupledSolver](#) package for more information. In Windows, install [Cygwin](#) before starting.

### Build PETSc

[BrainTumorModeling\\_CoupledSolver](#) depends on version 2.2.1 of [PETSc](#) library. [PETSc](#) library could be build as follows as a set of static libraries. The advantage of static libraries is that once the application program is compiled and linked, the PETSc files could be removed as they are not required for the execution of the program.

#### Step 1. Download and extract source files:

```
wget http://ftp.mcs.anl.gov/pub/petsc/software_old/v2.2.1.petsc.tar.gz
tar xzf v2.2.1.petsc.tar.gz
cd petsc-2.2.1
```

---

**Note:** In Windows, open the [Cygwin](#) terminal. In Mac OS, use `curl -O` instead of `wget`.

---

#### Step 2. Configure:

```
./config/configure.py --with-matlab=0 --with-shared=0 --with-mpi=0 --with-dynamic=0
--with-debugging=0 COPTFLAGS='-O3'
```

---

**Note:** The name of the compiler could be specified by `--with-cc=` and `--with-fc=` options. To download LAPACK during building, append `--download-f-blas-lapack=1`. If a fortran compiler is not available, try `--with-fc=0 --download-f2cblaslapack`. Although the MPI option is turned off above, it is okay with the MPI option enabled. For the details of the MPI option, see the [installation\\_instructions\\_of\\_PETSc](#).

---

#### Step 2. Build:

```
export PETSC_DIR=$PWD
export PETSC_ARCH=linux-gnu
make BOPT=O
```

---

**Note:** In Windows and Mac OS, change `linux-gnu` for `PETSC_ARCH` to the value corresponding to your system as reported by the `configure.py` script. For example, use `export PETSC_ARCH=cygwin` in the cygwin terminal.

---



## Build ForwardSolverDiffusion

First extract `braintumormodeling_coupled solver-1.2.0-source.tar.gz` into the appropriate directory. The source code could be obtained via the download URL. The download URL for the `BrainTumorModeling_CoupledSolver` software is sent to you per email in response to your software request for PORTR. An email with the download links of the different release versions of this package can be requested by filling [this online form](#) with the appropriate information.

### Step 1. Extract source files:

```
tar xzf braintumormodeling_coupled solver-1.2.0-source.tar.gz
cd braintumormodeling_coupled solver-1.2.0
```

### Step 2. Build ForwardSolverDiffusion:

```
make BOPT=O ForwardSolverDiffusion
```

---

**Note:** Before running `make`, confirm `PETSC_DIR` and `PETSC_ARCH` environmental variables are properly set to the values used above.

---

### Step 3. Install

Upon the success of the above build process, copy the `ForwardSolverDiffusion` executable to the appropriate directory which could be found by the `PATH` environment variable or PORTR's installation `bin` folder.

---

**Note:** In Windows, copy `ForwardSolverDiffusion.exe` into the Cygwin's `bin` folder (we assume it located in `C:\cygwin\bin`). Also, add the Cygwin's `bin` folder in the system path if it is not included by executing `setx Path "%Path%;C:\cygwin\bin" /m` in the command line. If users don't want to modify the path variable, copy `ForwardSolverDiffusion.exe` into the PORTR's installation `bin` folder. Also, copy `cygwin1.dll` and its dependent dlls (in our case, `cyggcc_s-1.dll`, `cyggfortran-3.dll`, `cygICE-6.dll`, `cygSM-6.dll`, `cygstdc++-6.dll`, `cyguuid-1.dll`, `cygX11-6.dll`, `cygXau-6.dll`, `cygxcb-1.dll`, `cygXdmp-6.dll`, `cygXt-6.dll`) along with `ForwardSolverDiffusion.exe`. These dlls could be found in the Cygwin's `bin` folder.

---

## How to build BrainTumorViewer

The following steps describe the procedure to build and install `BrainTumorViewer` used for making initializations of PORTR.

### Build Dependencies

Before `BrainTumorViewer` can be build from sources, the following software libraries have to be installed.

Package	Version	Description
<a href="#">CMake</a>	2.8.4 or higher	To compile and build <code>BrainTumorViewer</code> . Use version 2.8.4 or higher.
<a href="#">Qt</a>	4.8	The main GUI interface for <code>BrainTumorViewer</code> . Download and install the precompiled binary.
<a href="#">ITK</a>	3.14 or higher	For ITK 3.x or 4.x, there's no mandatory option.
<a href="#">VTK</a>	5.6.1 – 5.10.1	The VTK is required <code>VTK_USE_QT</code> option set to <code>ON</code> . Install Qt before setting VTK.

---

**Note:** To build in Windows, use CMake with the Win32 or Win64 (x64) solution platform of Visual Studio. The selected solution platform is needed to match with dependent libraries.

---

## Build and Installation

The package of BrainTumorViewer could be obtained via the download URL. The download URL for the BrainTumorViewer software is sent to you per email in response to your software request for PORTR. An email with the download links of the different release versions of this package can be requested by filling [this online form](#) with the appropriate information.

Please follow commands below in a shell/terminal (e.g., [Bash](#)). They will configure and build BrainTumorViewer using [GNU Make](#). The main CMake configuration file (CMakeLists.txt) is located in the root directory of the package.

### Step 1. Extract source files and create the build directory:

```
tar xzf braintumorviewer-${version}-source.tar.gz
mkdir braintumorviewer-${version}-build
cd braintumorviewer-${version}-build
```

---

**Note:** In Windows, use the appropriate zip program (e.g., [7-zip](#)) to extract.

---

### Step 2. Run CMake to configure the build tree:

```
cmake ../braintumorviewer-${version}-source
```

In the CMake interface, follow these steps:

2.1. Change CMAKE\_INSTALL\_PREFIX to the folder you want to install BrainTumorViewer into. This folder should be outside the braintumorviewer-\${version}-source folder. Make sure you have the **write** access to this folder. Change CMAKE\_BUILD\_TYPE (CMAKE\_CONFIGURATION\_TYPES in Windows) to Release. Locate ITK\_DIR, QT\_QMAKE\_EXECUTABLE, and VTK\_DIR to appropriate folders.

2.2. Keep pressing letter `c` on your keyboard until option `g` is available/displayed on the screen.

2.3. Then press `g` on your keyboard to generate the makefiles and to quit this cmake window.

---

**Note:** In the CMake GUI, `c` and `g` correspond to `Configure` and `Generate` buttons.

---

### Step 3. Build and Install:

```
make
make install
```

---

**Note:** In Windows, launch the solution file of Visual Studio, select `Release` in the solution configurations, and then perform the rebuild solution. For installation, build the `INSTALL` project.

---

Upon the success of the above compilation and build process, BrainTumorViewer is installed into the directory specified by the CMAKE\_INSTALL\_PREFIX (set during build configuration in step 3).

## 4 Manual

### 4.1 PORTR Command and Parameters

The main command of PORTR which segments the subject scan and registers the atlas to the subject scan is named PORTR. The simplest use is:

```
PORTR --scan0_list <scan0_list_file>      --scan2_list <scan2_list_file>
      --scan0_seed <scan0_seed_file>      --scan2_seed <scan2_seed_file>
      --scan0_point <scan0_point_file>    --scan2_point <scan2_point_file>
      --atlas_folder <atlas_folder>      --outputdir <output_folder>
```

The parameters used above are **mandatory**. The Scan 0 represents the pre-operative (baseline) scan and Scan 2 represents the post-recurrence (follow-up) scan. For image format, **NIfTI-1** (\*.nii or \*.nii.gz) format is required. For memory requirements, it depends on the size of the atlas and the number of thread used. For the eve atlas with using single thread, 12GB memory is recommended. For 3 threads in same configuration, 15GB memory is recommended. The absolute path is required for each path in the parameter. The meaning of each parameter is:

```
--scan0_list, -sl0 <file>      Text file listing Scan 0, one per line.
--scan2_list, -sl2 <file>      Text file listing Scan 2, one per line.
                                It is recommended to use T1, T1-CE, T2, and FLAIR
                                patient scans as input.

--scan0_seed, -ss0 <file>      Tumor seed information for Scan 0.
--scan2_seed, -ss2 <file>      Tumor seed information for Scan 2.
                                The format of the seed file is:

                                <n: number of seeds>
                                <x1> <y1> <z1> <d1>
                                ...
                                <xn> <yn> <zn> <dn>

                                where <xi> <yi> <zi> is the RAS coordinate of ith seed and
                                <di> is the approximated diameter of ith tumor.

--scan0_point, -sp0 <file>      One sample point for each class of Scan 0.
                                This option overrides -sm0.
--scan2_point, -sp2 <file>      One sample point for each class of Scan 2.
                                This option overrides -sm2.
                                The format of the point file is:

                                <Class 1>
                                <x1> <y1> <z1>
                                ...
                                <Class n>
                                <xn> <yn> <zn>

                                where <Class i> is the name of ith tissue class and
                                <xi> <yi> <zi> is the RAS coordinate of ith tissue class.

--atlas_folder, -af <dir>      The directory path for the atlas.
--outputdir, -d <dir>          The directory path for output files.
```

Each parameter can be specified as a long name starting with -- or a short name starting with -. For pre-operative scan (Scan 0), only one tumor seed should be placed. This tumor is assumed to be resected by surgery and the region corresponds to the cavity in the post-recurrence scan (Scan 2). In Scan 2, 1st tumor seed should be placed on the center of this cavity. The other tumor seeds correspond to each center of tumor recurrences. So the typical tumor seed file for Scan 0 contains:

```
1
x y z d
```

where  $\langle x \rangle$   $\langle y \rangle$   $\langle z \rangle$  is the RAS coordinate of center of the tumor to be resected and  $\langle d \rangle$  is its approximated diameter. If there're two tumor recurrences in Scan 2, the tumor seed file for Scan 2 contains:

```
3
xc yc zc dc
x1 y1 z1 d1
x2 y2 z2 d2
```

where  $\langle xc \rangle$   $\langle yc \rangle$   $\langle zc \rangle$  is the RAS coordinate of the center of cavity and  $\langle dc \rangle$  is its approximated diameter, and  $\langle xi \rangle$   $\langle yi \rangle$   $\langle zi \rangle$  is the RAS coordinate of  $i$ th center of tumor recurrence and  $\langle di \rangle$  is its approximated diameter.

For each tissue class in the pre-operative scan (Scan 0), the label index and value are defined as follows:

Label	Description	Index	Value
BG	Background	0	0
CSF	Cerebrospinal Fluid	1	10
VT	Ventricles	2	50
GM	Gray Matter	3	150
WM	White Matter	4	250
VS	Vessel	5	25
ED	Edema	6	100
NCR	Necrosis	7	175
TU	Enhancing Tumor	8	200

For each tissue class in the post-recurrence scan (Scan 2), the label index and value are defined as follows:

Label	Description	Index	Value
BG	Background	0	0
CSF	Cerebrospinal Fluid	1	10
VT	Ventricles	2	50
GM	Gray Matter	3	150
WM	White Matter	4	250
VS	Vessel	5	25
ED	Edema	6	100
CAN	Cavity	7	175
CAE	Cavity (Enhanced)	8	200
RTN	Tumor Recurrence	9	210
RTE	Tumor Recurrence (Enhanced)	10	220

Users can specify only one of CAN and CAE (or RTN and RTE), if those regions have homogeneous intensity. The point file contains the location of one sample point for each tissue class. The typical point file for Scan 0 contains:

```
CSF
x1 y1 z1
VT
x2 y2 z2
GM
x3 y3 z3
WM
x4 y4 z4
VS
x5 y5 z5
ED
x6 y6 z6
NCR
x7 y7 z7
TU
x8 y8 z8
```

where  $\langle xi \rangle$   $\langle yi \rangle$   $\langle zi \rangle$  is the RAS coordinate of one sample point for  $i$ th tissue class. The point for BG doesn't need to be specified. The point file for Scan 2 could be specified similarly using labeled defined for Scan 2.

The following parameters are **optional**:

<code>--workingdir, -w &lt;dir&gt;</code>	The directory path for intermediate files.
<code>--align_scan, -as &lt;int&gt;</code>	1 if align Scan 2 to Scan 0 before registration, 0 do nothing.
<code>--atlas_template, -at &lt;file&gt;</code>	The atlas template file.
<code>--atlas_prior, -ap &lt;file&gt;</code>	The list file for atlas priors.
<code>--atlas_mask, -am &lt;file&gt;</code>	The atlas mask file.
<code>--atlas_label, -al &lt;file&gt;</code>	The atlas label map.
<code>--num_omp_threads, -not &lt;int&gt;</code>	The number of OpenMP threads.
<code>--num_itk_threads, -nit &lt;int&gt;</code>	The number of ITK threads.
<code>--num_hop_threads, -nht &lt;int&gt;</code>	The number of HOPSPACK threads.
<code>--verbose, -v &lt;int&gt;</code>	if greater than 0, print log messages: 1 (default), 2 (detailed).

The following parameters are **obsolete**:

<code>--scan0_mean, -sm0 &lt;file&gt;</code>	Initial mean values for Scan 0.
<code>--scan2_mean, -sm2 &lt;file&gt;</code>	Initial mean values for Scan 2.

## 4.2 Output of PORTR

The following output files are generated by PORTR in the `<output_folder>` specified by the `--outputdir` or `-d` parameter:

<code>scan0_label_map.nii.gz</code>	Obtained segmentation label map for Scan 0.
<code>scan2_label_map.nii.gz</code>	Obtained segmentation label map for Scan 2.
<code>scan0_to_scan2.*</code>	Deformation field mapping Scan 0 to Scan 2.
<code>scan2_to_scan0.*</code>	Deformation field mapping Scan 2 to Scan 0.
<code>scan0_posterior*</code>	Obtained posterior map for each tissue class of Scan 0.
<code>scan2_posterior*</code>	Obtained posterior map for each tissue class of Scan 2.
<code>scan0_prior*</code>	Obtained prior map for each tissue class of Scan 0.
<code>scan2_prior*</code>	Obtained prior map for each tissue class of Scan 2.
<code>log.txt</code>	Log file.

To map an image in Scan 2 (post-recurrence) space to Scan 0 (pre-operative) space, warp the image using `scan0_to_scan2.mhd`. `scan0_posterior*`, `scan2_posterior*`, `scan0_prior*`, and `scan2_prior*` for each tissue class are appended with the label index defined in the above table. Also, the label value of each class used in `scan0_label_map.nii.gz` and `scan2_label_map.nii.gz` is defined above.

## 4.3 Making Initializations Using BrainTumorViewer

Using [BrainTumorViewer](#), making initial tumor seeds and tissue points could be easily done. Assuming co-registered scans for brain tumor MRI are given, open all modalities using `File > Open Image(s)` menu or drag and drop images to the viewer. Users can navigate each modality and roughly figure out the location of the tumor and the edema. In the `Tumor` tab as shown in [Figure 4](#) and [Figure 5](#), select `PORTR Pre` for the pre-operative scan (Scan 0) (see [Figure 4](#)) or `PORTR Post` for the post-operative scan (Scan 2) (see [Figure 5](#)) in `Type of Initializations`. To add a point for tumor seed, click one item in the `Tumor Seed Points` list and then click (move cross hair to) an approximate center of each tumor mass and push `< space >`. To add a radius for this mass, click its approximate boundary and push `< shift > + < space >`. After completing, the list can be saved using the `Save` button. In the `Tissue Points` list, each row represent one sample location for the corresponding tissue type. To add a point, select one tissue type and then click one sample location of that tissue on the view and push `< space >`.

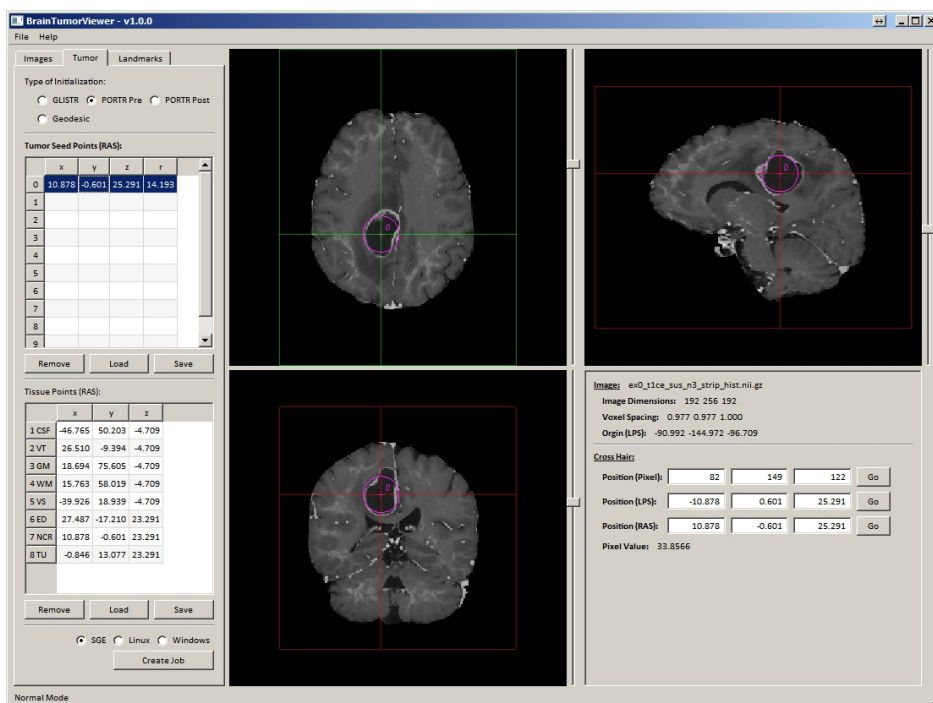


Figure 4: An example screenshot for Scan 0 of BrainTumorViewer.

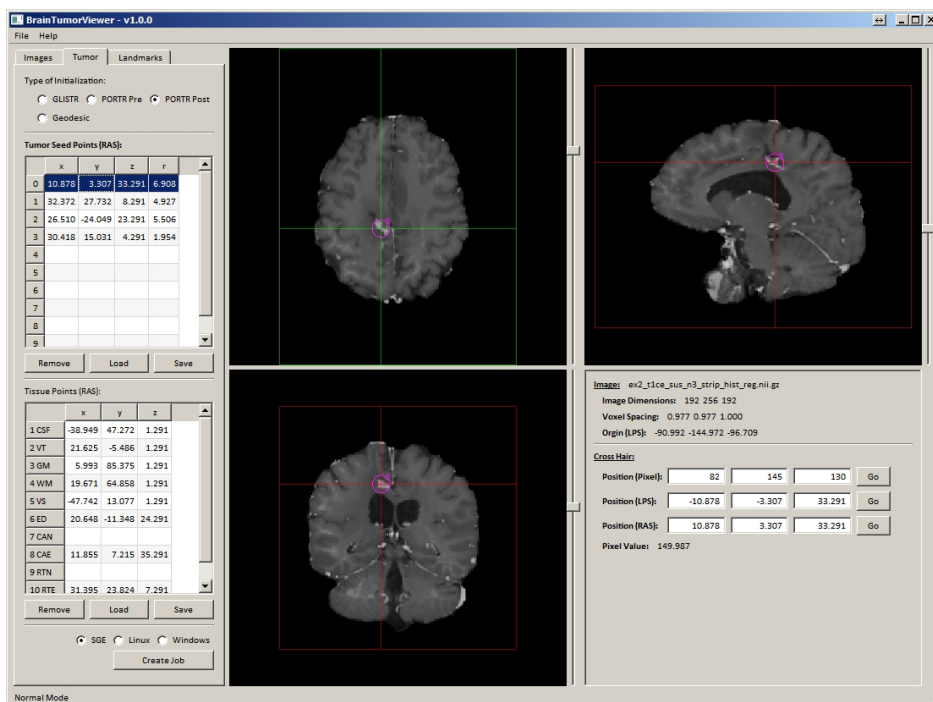


Figure 5: An example screenshot for Scan 2 of BrainTumorViewer.

## 4.4 Auxiliary Commands

Command	Operation
<i>WarpImage{NN}</i>	Warp image
<i>NormalizeImage</i>	Normalize image
<i>ResampleImage</i>	Resample image
<i>ConcatenateFields</i>	Concatenate two deformation fields
<i>ResampleDeformationField</i>	Resample deformation field
<i>ReverseDeformationField</i>	Reverse deformation field
<i>EvaluateQ</i>	Computing the cost function

*EvaluateQ* is the internal procedure and called only by *PORTR*.

## 4.5 Running Example

If we assume `$(install)` is an installation directory of *PORTR*, an example scans and its worked initializations are installed in `$(install)/example/ex0`. The *PORTR* for this example is performed by

```
$(install)/bin/PORTR --scan0_list $(install)/example/ex0/scan0.lst
--scan2_list $(install)/example/ex2/scan2.lst
--scan0_seed $(install)/example/ex0/init_seed.txt
--scan2_seed $(install)/example/ex2/init_seed.txt
--scan0_point $(install)/example/ex0/init_point.txt
--scan2_point $(install)/example/ex2/init_point.txt
--atlas_folder $(install)/atlas9
--outputdir $(install)/example/portr
```

Then, output files are created in `$(install)/example/portr`. To check the intermediate results of the procedure specify the working directory by appending `--workingdir $(install)/example/portr/tmp`. To check more detailed output messages, append `--verbose 2`. To execute *PORTR* using *N* cores in the system, append `--num_omp_threads N`, `--num_itk_threads N`, and `--num_hop_threads N`. If Scan 0 and Scan 2 are not affinely aligned previously, append `--align_scan 1` to align Scan 2 toward Scan 0 using the affine transformation before registration. The running time of this example is about 4 hours using 3 cores on an Intel Core i7 3.4 GHz machine with Windows operating system having 32GB memory and NVidia GeForce GTX 580.

---

**Note:** Before running *PORTR*, confirm *flirt*, *convert\_xfm* of *FSL*, *HOPSPACK\_main\_threaded* of *HOPSPACK*, and *ForwardSolverDiffusion* of *BrainTumorModeling\_CoupledSolver* could be found in the path or located in `$(install)/bin` folder. If not, check *Runtime Requirements* again.

---

## Warp Image

Given the reference image and the deformation field (mapping reference to input spaces), *WarpImage* or *WarpImageNN* warps an input image to the space of the reference image.

```
WarpImage{NN} -i [input_image_file] -r [reference_image_file]
               -o [output_image_file] -d [deformation_field_file]
```

The meaning of each parameter is

```
-i (--input ) [input_image_file]      input (moving) image (input)
-r (--reference) [reference_image_file] reference image (input)
-o (--output ) [output_image_file]    output image (output)
-d (--def_field) [deformation_field_file] reference to input (moving)
                                         def. field (input)
```

*WarpImage* is using the bilinear interpolation, for warping finite value images like label maps, use *WarpImageNN* which uses the nearest neighbor interpolation.

## Normalize Image

`NormalizeImage` normalize an input image using the histogram mathing with the given template image. It additionally performs the susan smoothing and the n3 correction if options are selected.

```
NormalizeImage -i [input_image_file] -o [output_image_file]
               -t [template_image_file] -a [template_wm_range_a]
               -b [template_wm_range_b] -s [1 or 0] -n [1 or 0]
```

The meaning of each parameter is

<code>-i</code>	<code>(--input )</code>	<code>[input_image_file]</code>	input image (input)
<code>-o</code>	<code>(--output )</code>	<code>[output_image_file]</code>	output image (output)
<code>-t</code>	<code>(--template )</code>	<code>[template_image_file]</code>	template image (input)
<code>-a</code>	<code>(--template_wm_a)</code>	<code>[template_wm_range_a]</code>	template wm range a, intensity in [a,b] is used for scaling
<code>-b</code>	<code>(--template_wm_b)</code>	<code>[template_wm_range_b]</code>	template wm range b, intensity in [a,b] is used for scaling
<code>-s</code>	<code>(--apply_susan )</code>	<code>[1 or 0]</code>	1 (default) if apply susan smoothing, 0 otherwise
<code>-sb</code>	<code>(--susan_bt )</code>	<code>[bt]</code>	bt for susan
<code>-sd</code>	<code>(--susan_dt )</code>	<code>[dt]</code>	dt for susan
<code>-n</code>	<code>(--apply_n3 )</code>	<code>[1 or 0]</code>	1 (default) if apply n3 bias field correction, 0 otherwise
<code>-sh</code>	<code>(--scale_hist )</code>	<code>[1 or 0]</code>	1 (default) if scale via histogram mathing, 0 scale using maximum value
<code>-sm</code>	<code>(--scale_max )</code>	<code>[max]</code>	max value for scaling
<code>-w</code>	<code>(--apply_window )</code>	<code>[1 or 0]</code>	1 (default) if apply windowing, 0 otherwise
<code>-wa</code>	<code>(--window_a )</code>	<code>[window_a]</code>	window lower percent (default: 0.01)
<code>-wb</code>	<code>(--window_b )</code>	<code>[window_b]</code>	window upper percent (default: 0.09)

## Resample Image

`ResampleImage` scales an input image using ratios in x, y, z axis. It requires the reference image in the resampled space.

```
ResampleImage -i [input_image_file] -o [output_image_file]
               -r [reference_image_file]
               -x [ratio_x] -y [ratio_y] -z [ratio_z]
```

The meaning of each parameter is

<code>-i</code>	<code>(--input )</code>	<code>[input_image_file]</code>	input image file (input)
<code>-o</code>	<code>(--output )</code>	<code>[output_image_file]</code>	output image file (output)
<code>-r</code>	<code>(--reference)</code>	<code>[reference_image_file]</code>	reference image file (input)
<code>-x</code>	<code>(--ratio_x )</code>	<code>[ratio_x]</code>	ratio in x axis (input)
<code>-y</code>	<code>(--ratio_y )</code>	<code>[ratio_y]</code>	ratio in y axis (input)
<code>-z</code>	<code>(--ratio_z )</code>	<code>[ratio_z]</code>	ratio in z axis (input)

## Concatenate Two Deformation Fields

`ConcatenateFields` concatenates two given deformation fields.

```
ConcatenateFields -fi [fix_to_inter_deformation_file]
                  -im [inter_to_mov_deformation_file]
                  -fm [fix_to_mov_deformation_file]
```

The meaning of each parameter is



```
-fi (--fix_to_inter) [fix_to_inter_deformation_file]    fix to inter def. field (input)
-im (--inter_to_mov) [inter_to_mov_deformation_file]   inter to mov def. field (input)
-fm (--fix_to_mov ) [fix_to_mov_deformation_file]      fix to mov def. field (output)
```

## Resample Deformation Field

`ResampleDeformationField` scales an input deformation field using ratios in x, y, z axis. It requires the reference image in the resampled space.

```
ResampleDeformationField -i [input_deformation_file] -o [output_deformation_file]
                        -r [reference_image_file]
                        -x [ratio_x] -y [ratio_y] -z [ratio_z]
```

The meaning of each parameter is

```
-i (--input ) [input_deformation_file]    input def. field (input)
-o (--output ) [output_deformation_file]  output def. field (output)
-r (--reference) [reference_image_file]    reference image (input)
-x (--ratio_x ) [ratio_x]                 ratio in x axis (input)
-y (--ratio_y ) [ratio_y]                 ratio in y axis (input)
-z (--ratio_z ) [ratio_z]                 ratio in z axis (input)
```

## Reverse Deformation Field

`ReverseDeformationField` generates the inverse deformation field from an input deformation field.

```
ReverseDeformationField -i [input_deformation_file] -o [output_deformation_file]
                        -n [iterations_number] -s [stop_value]
```

The meaning of each parameter is

```
-i (--input ) [input_deformation_file]    input def. field (input)
-o (--output ) [output_deformation_file]  output def. field (output)
-n (--num_iter) [number_iterations]       number of iterations (input)
-s (--stop_val) [stop_value]              stop value (input)
```

Higher iterations number give more accurate results. It stops the iteration if the error (in mm) between forward and backward mapping is smaller than specified `stop_val`.

## 5 Changelog

### Version 1.0.0 (Oct 10, 2014)

- First public release of the PORTR software.

## 6 Publications

Please cite [TMI2014] when you used PORTR in your research:

[TMI2014]

[TMI2012]

## 7 People

### Advisor

- Christos Davatzikos ✉
- Kilian M. Pohl

### Software Authors

- Dongjin Kwon ✉
  - Initiated the project.
  - Developed the software.

## References

- [TMI2014] D. Kwon, M. Niethammer, H. Akbari, M. Bilello, C. Davatzikos, and K.M. Pohl, [PORTR: Pre-Operative and Post-Recurrence Brain Tumor Registration](#), IEEE Trans. Med. Imaging 33(3): 651-667 (2014)
- [TMI2012] A. Gooya, K.M. Pohl, M. Bilello, L. Cirillo, G. Biros, E.R. Melhem, C. Davatzikos, [GLISTR: Glioma Image Segmentation and Registration](#), IEEE Trans. Med. Imaging 31(10): 1941-1954 (2012)