ResMap Manual

`[version 1.95]`

Frédéric Bonnet, Alp Kucukelbir, Frederick Sigworth, Hemant Tagare

June 21, 2018

Contents

1 ResMap

ResMap (Resolution Map) is a software package for computing the local resolution of 3D density maps from electron cryo-microscopy (cryo-EM). ResMap is written in Python and uses NumPy, SciPy, and Matplotlib. ResMap has a Tkinter GUI interface as well as a command-line interface.

This manual describes the installation and operation of ResMap version 1.95, which is the latest version of ResMap as of June 2018. This ResMap version is referred to as ResMap v-1.95 through out this document.

ResMap v-1.95 is a GPU enabled version of ResMap. That is, if you have appropriate GPUs (CUDA 9.0) installed in your computer, then this version of ResMap gives you the option of using GPUs for computation. GPUs give faster results on large volumes. CPU/GPU execution times for typical volumes are given in Section 3.

Of course, you can use this version even if you do not have GPUs. Simply uncheck the Use GPU checkbox in the GUI and ResMap will execute solely on the CPU.

Besides enabling GPU use, ResMap v-1.95 also provides higher numerical precision for the underlying algorithms.

At the time of this release (June 2018), ResMap v-1.95 works on Linux machines and on Mac OSX machines. However, GPU use is only possible on Linux machines. GPU use is not supported on Mac OSX machines. Windows machines are not supported at all.

1.1 What is new in version 1.95? (For users of previous versions)

The most significant changes in ResMap v-1.95 are:

1. ResMap v-1.95 is GPU enabled.
2. A NumPy bug has been identified and eliminated. For additional details, please see Section 10.
3. Steerable basis calculations are now carried out using higher precision QR decompositions.
4. A better noise estimate has been implemented.
5. More extensive benchmarking support is provided for logging execution times for CPU and GPU.
6. The ResMap GUI is modified slightly to account for the above changes The GUI modification is minor. Most users will find the new GUI very similar to the old GUI. Section 5.1 explains the GUI.

The fundamentals of the ResMap algorithm in version 1.95 remains unchanged; it is almost identical to the algorithm in the old version.

1.2 When to enable GPUs

The following rules-of-thumb are useful in determining when to and when not to enable GPU calculation:

1. GPU calculation should not be enabled when ResMap is used on a Mac OSX platform.

2. GPU calculation should not be enabled on a Linux platform if your maps are smaller than $140 \times 140 \times 140$, or if your maps are larger than $700 \times 700 \times 700$. If your maps have a size between $140 \times 140 \times 140$ and $700 \times 700 \times 700$, you may enable GPU usage on a Linux machine.

   For maps smaller than $140 \times 140 \times 140$, the CPU calculation is likely to be just as fast as the GPU (hence GPU need not be used). For maps larger than $700 \times 700 \times 700$, the GPU is likely not to have sufficient memory for the calculation (this upper limit holds for GTX 1080 Ti GPUs, which are typically used with RELION 3.0).

1.3 How to read this manual

Experienced users of ResMap may skip ahead to Section 4 and directly download and install ResMap v-1.95. Other users might benefit from reading Section 2 first to get an idea of how ResMap works and what ResMap expects for input volumes and other parameters.

2 Understanding how ResMap works

2.1 Local resolution

ResMap calculates and reports the local resolution in Angstroms (Å) at every voxel of the input maps. At any voxel, the local resolution is the wavelength of the highest local spatial frequency that is statistically significantly above noise. ResMap calculates local resolutions within a range of resolutions at a given step size, both specified by the user. For example, if the user defines the range as extending from the finest resolution of 2Åto the coarsest resolution of 10Åat a step size of 0.5Å, then for each voxel, ResMap reports one resolution from the set of numbers $[2, 2.5, 3, 3.5, 4, 4.5, \ldots, 9, 0, 9, 5, 10]$ Å.

2.2 The ResMap algorithm

Figure 1 shows a conceptual diagram of ResMap. ResMap expects two gold standard volumes as input. The mean of the two volumes is taken as the signal, while the difference of the two volumes is used to estimate the noise.

The local resolution calculation in ResMap expects noise to be white. But noise in most volumes is not white; therefore, ResMap provides an interactive tool for pre-whitening the volumes. The tool uses the mean and difference volumes to calculate the signal and noise spectrum. The user then interactively selects a frequency range over which to fit the noise spectrum with a noise model. The fitting range extends from a coarse "Elbow resolution" (specified in Åby the user) to the Nyquist frequency (see Fig.1). Having selected a range, ResMap fits a noise model to the noise spectrum in the range. The fitted noise model is then used to pre-whiten the signal and noise. Carefully fitting the noise model so that the pre-whitened noise appears white in the fitted range is critical to ResMap operation. To help the user assess and improve the fit, ResMap also displays the pre-whitened noise spectrum and lets the user change the elbow resolution from its default value. Additionally, the user is also provided with a low resolution boost that can adjust the pre-whitening at coarser than Elbow resolution. The low resolution boost adjustment is usually unnecessary.

Two additional details about pre-whitening are important: First, the user can provide an optional binary mask for the particle as an input (see Fig.1). The mask is used in the pre-whitening calculation. If a mask is not provided, then ResMap creates its own mask. Second, the pre-whitening calculation can be expensive, and to speed up the calculation,

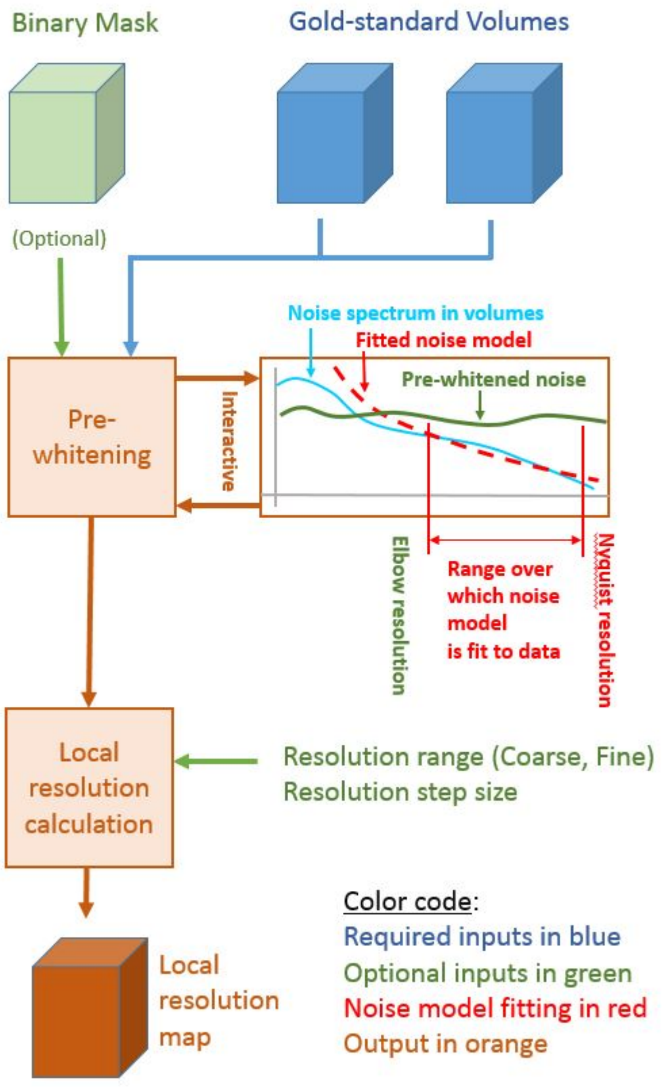


Figure 1: The ResMap algorithm

ResMap subsamples large volumes. The subsampling is done only for pre-whitening. The local resolution is calculated without subsampling.

Important Note: Pre-whitening can fail (i.e. the volumes cannot be pre-whitened) if the volumes are hard filtered at the FSC resolution (hard filtering removes all spectral information beyond the filter cutoff frequency). ResMap cannot be used with such volumes.

Our recommendation for ResMap input volumes is the following: Create gold standard maps without any post-processing. Especially do not post-process the maps by filtering or tight masking or polishing. Then carefully pre-whiten the split maps using the ResMap GUI (see Section 5).

Figures 2-5 shows slices through examples of invalid, valid, and preferred input volumes.

Once the user indicates that the pre-whitening is satisfactory, ResMap proceeds to calculate the local resolution without any user interaction. The ResMap starts from the finest resolution and works towards the coarsest resolution with the step size specified by the user. As mentioned above, for each voxel, ResMap finds the highest resolution at which the signal

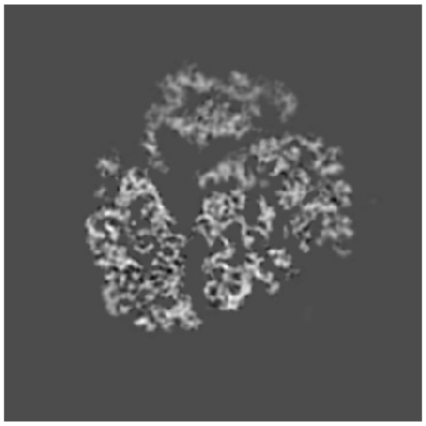


Figure 2: **INVALID:** This map (EMD-2239) has been masked to eliminate the background. ResMap cannot use this map as input.

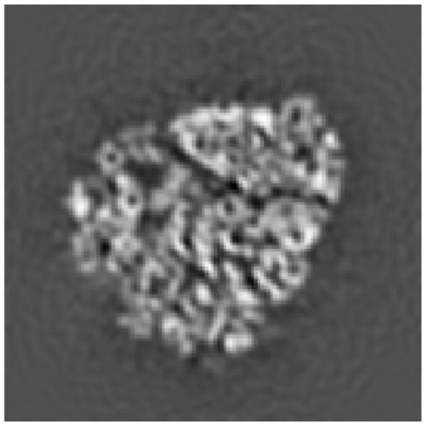


Figure 3: **VALID, BUT NOT IDEAL:** This map (EMD-5562) has been low pass filtered. ResMap will attempt to run, but may fail.

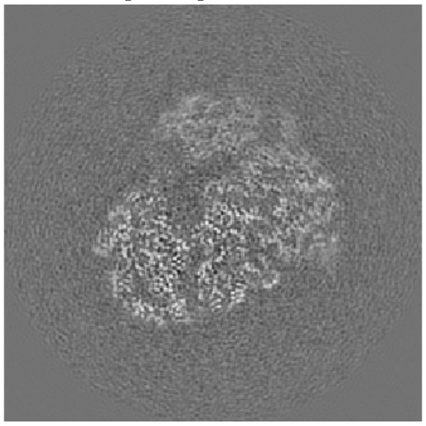


Figure 4: **VALID, BUT NOT IDEAL:** This map (EMD-2275) has been B-factor sharpened. ResMap will attempt to run, but may fail.

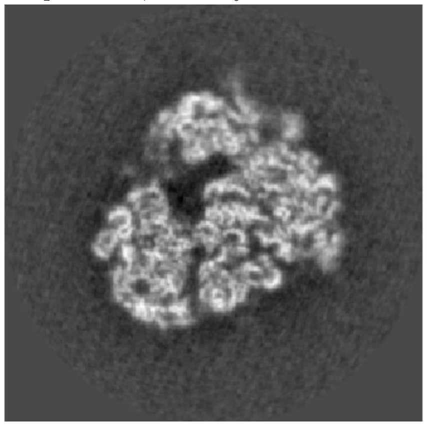


Figure 5: **PREFERRED:** This map has not been post-processed (raw version of EMD-2275). ResMap will run reliably on this map.

in the mean volume exceeds noise according to a statistical test. All of the parameters used in this step (the coarsest resolution, the finest resolution frequency, the resolution step, and the statistical significance level of the test) can be specified by the user.

Once local resolution is calculated, ResMap displays the results and creates an output MRC file containing the local resolution. ResMap also creates 2D and 3D (Chimera) visualizations of the local resolution. In addition, a log file is created to log various execution times and other internal calculations.

While reading the rest of this manual, it is useful to keep in mind that user input to ResMap consists of the following. Additional details about inputs can be found in Section 6.

1. **Input for resolution calculation:** This input specifies how the local resolution calculation is to be carried out. This input includes the two volumes, the optional mask, the finest and coarsest resolutions and resolution step size, and the statistical significance level of the test used to determined whether the signal is sufficiently above noise. This input can be specified via a GUI or via command line. Section 5 explains how.

   Except the filenames of the volumes, ResMap provides defaults for all of these inputs.

2. **Input for pre-whitening:** This input specifies the elbow resolution for the noise model fit, and the low resolution boost. The input can be specified via a GUI, which also allows the user to interactively asses the pre-whitening. Section 5 explains this as well.

   ResMap provides default values for all pre-whitening inputs. However, the user is encouraged to interactively override the default values to improve pre-whitening accuracy.

3 Processing Time

ResMap v-1.95 execution time depends on two factors: 1) Whether GPUs are available and their use is enabled, and 2) On the size of the map.

If GPUs are not used then ResMap executes solely on the CPU. In this mode, maps that are $256 \times 256 \times 256$ or smaller are processed fairly quickly. Larger maps take longer to process.

If GPUs are available and their use is enabled, then the processing is significantly faster, especially for large maps. Figure 6 shows the ResMap v-1.95 processing times with and without the GPU enabled, for several maps downloaded from the EMDataBank. For maps that are smaller than about $140 \times 140 \times 140$, the GPU execution time is comparable to CPU execution time, but for larger maps, the GPU execution time is significantly smaller. We recommend using the GPU for maps bigger than $140 \times 140 \times 140$. For smaller maps, CPU execution is preferable.

Table 1 shows CPU and GPU execution times for some example volumes downloaded from the EMDataBank.

4 Downloading and Installing

Binaries and kernel library for Linux are available on SourceForge:

https://sourceforge.net/projects/resmap-latest

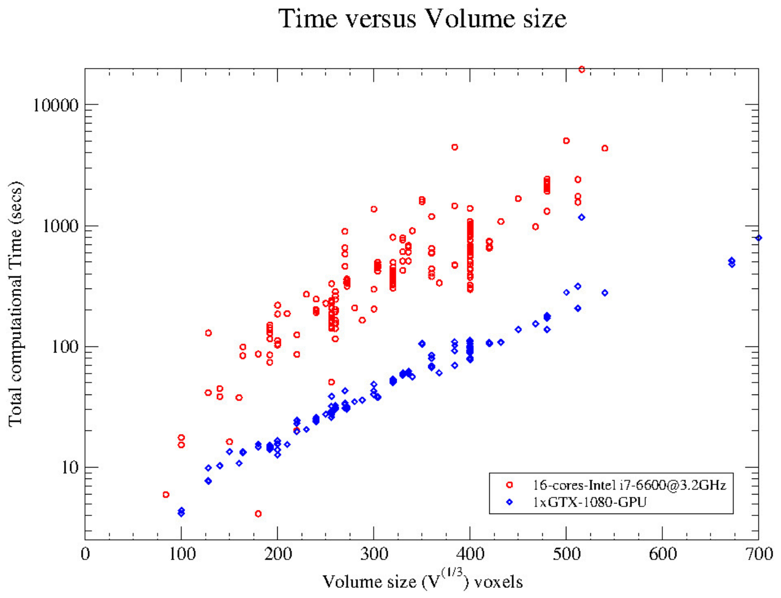


Figure 6: Computational time in secs vs. Volume size (map size) on CPU and GPU

Table 1: Execution time in secs. for various volumes sizes on CPU and GPU. All measurements are done on the same machine. The CPU computation times are for a a mother board equipped with 1xIntel i7–6900K at 3.20 GHz CPUs and is computed over 16 cores. GPU computation times are for a GTX-1080 GPU (2560 cuda cores) with 8GB DDR5.

		Total Exec time (secs)		Speedup
EMD–map	Volume Size	GPU	CPU	$\frac{\text{CPU}}{\text{GPU}}$
8481	$192 \times 192 \times 192$	14.6278	82.8978	5.66
8731	$256 \times 256 \times 256$	25.9995	65.6009	2.52
8384	$400 \times 400 \times 400$	100.1526	917.048	9.15
8763	$672 \times 672 \times 672$	478.1429	2927.3600	6.12

Click the download green button and SourceForge should automatically provide you with the latest binary for your platform.

Alternately, check the "Project Activity" console which shows several tar balls and library files for GPU use. The previous release (v-1.90) is also available there.

ResMap binaries have been packaged using PyInstaller[1] and have been tested on:

- **Mac:** 10.6+ (Binary prepared on 10.13.5 High Sierra)
- **Linux:** Fedora: 14 and 26, Ubuntu: 16.04x64, 17.04x64, 17.10x64 and 18.04x64; Mint: 18.2x64(cinnamon); OpenSuse: TumbleWeed-x64; CentOS 6+.

The Linux tar ball, as well as the download page, has several versions to cover many Linux distributions:

```
ResMap-1.95-cuda9.0-Linux16.04x64
ResMap-1.95-cuda9.0-Linux17.04x64
ResMap-1.95-cuda9.0-Linux17.10x64
ResMap-1.95-cuda9.0-Linux18.04x64
```

[1]http://www.pyinstaller.org/

These have been compiled on different Ubuntu versions, namely (16.04, 17.04, 17.10 and 18.04). Versions 17.10 and 18.04 are not backward compatible with lower versions, but version 16.04 is forward compatible with 18.04. The binaries are compatible across all Linux distributions (SUSE, RedHat, Debians). Just pick the tar ball that is appropriate to your system.

GPU kernels have been compiled using CUDA-9.0, and are built for the following systems:

- **Linux:** Ubuntu 17.04x64(CPU: Intel Core i7-6900K @ 3.20GHz×16) and GeForce GTX 1080 with 8GB-DDR5.
- **Linux kernel:**

```
    == [Operating System: Linux] ==
    == [Release          : 4.10.0-42-generic] ==
    == [Kernel         : Linux-4.10.0-42-generic-x86_64-with-Ubuntu-17.04-zesty] ==
    == [Processor type   : x86_64] ==
    == [CPU cores count  : 16] ==
```

GPU kernels have been tested on Kepler, Maxwell, and Pascal achitectures. Pascal give the best computational performance.

NOTE: Users may need to change the permissions of the downloaded binary to launch the program. To do so, please execute the following command from the terminal,

```
chmod a+x ResMap-1.95-distrib
```

where you should replace “ResMap-1.95-distrib” with the name of the file that you downloaded.

Finally, please check that you have the following installed on your system, especially if you wish to run ResMap from the source code:

- **Python:** (3.5.2+) [**ResMap version 1.95 is not compatible with Python 2.X**]
- **NumPy:** (Mint: 1.13.3+, Ubuntu-17.04: 1.13.1+, OpenSuse: 1.13.3+, Fedora-26+: 1.13.3+, MacOsX: 1.14.2+)
- **SciPy:** (Mint: 0.19.1+, Ubuntu-17.04: 0.19.1+, OpenSuse: 0.19.1+, Fedora-26+: 0.19.1+, MacOsX: 1.0.1+)
- **Matplotlib:** (OpenSuse: 2.0.2+, same for other Linux, MacOsX: 2.2.2+)
- **tkinter:** (under python3) (MacOsX: 8.5)

You can check the version your current numerical and scientific libraries as follows:

```
user@host:> python3
Python 3.5.3 (default, Sep 14 2017, 22:58:41)
[GCC 6.3.0 20170406] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy
>>> numpy.version.version
'1.13.1'
>>> import scipy
>>> scipy.version.version
'0.19.1'
>>> import matplotlib
>>> matplotlib.__version__
'2.0.2'
>>> import tkinter
>>> tkinter.TkVersion
8.6
>>>
```

4.1 The GPU libraries

The GPU libraries are pre-compiled and provided:

```
ResMap_krnl-cuda-V9.0.102-sm30_gpu.so
ResMap_krnl-cuda-V9.0.102-sm35_gpu.so
                ...
ResMap_krnl-cuda-V9.0.102-sm60_gpu.so
```

These files are only to be used with a Linux system. ResMap requires that one of these files is chosen as the GPU library if GPU Use is enabled. This is explained in detail in the next section (read Sec. 5.1).

5 Using ResMap

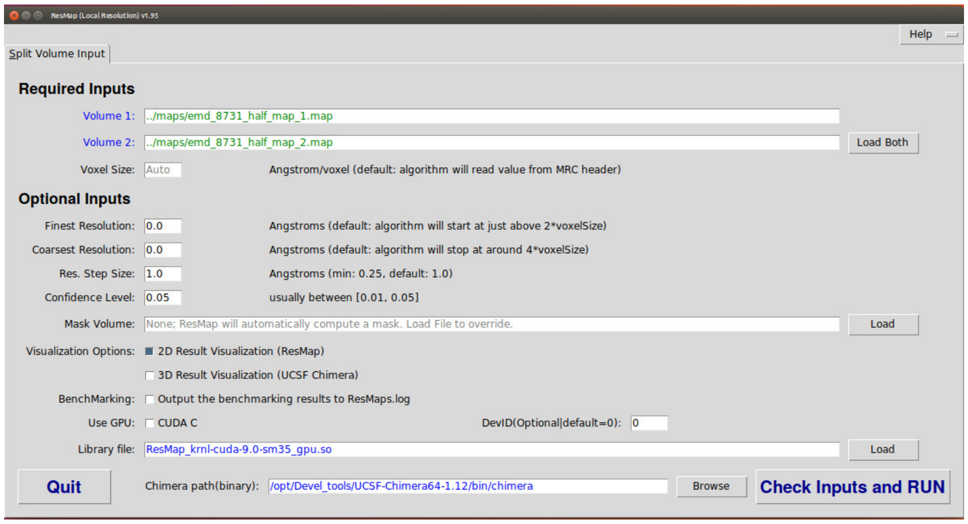
ResMap v-1.95 can be used in GUI mode or in command line mode. Most users are likely to use ResMap in the GUI mode (at least initially), and we begin by describing that mode. The descriptions given below provide sufficient details of ResMap inputs and settings to get started. Complete details are provided in Section 6.

5.1 Using ResMap in GUI mode

To run ResMap v-1.95 in GUI mode, in Mac and Linux, execute the following command from the terminal

```
./ResMap-1.95-distrib
```

ResMap should launch a graphical user interface (GUI) similar to Figure 7. The first time you launch ResMap, it may take a while for the GUI to appear. Subsequent launches should be faster.



Required Inputs

Volume 1: ../maps/emd_8731_half_map_1.map

Volume 2: ../maps/emd_8731_half_map_2.map

Voxel Size: Auto   Angstrom/voxel (default: algorithm will read value from MRC header)

Optional Inputs

Finest Resolution: 0.0   Angstroms (default: algorithm will start at just above 2*voxelSize)

Coarsest Resolution: 0.0   Angstroms (default: algorithm will stop at around 4*voxelSize)

Res. Step Size: 1.0   Angstroms (min: 0.25, default: 1.0)

Confidence Level: 0.05   usually between [0.01, 0.05]

Mask Volume: None; ResMap will automatically compute a mask. Load File to override.

Visualization Options: ■ 2D Result Visualization (ResMap)

☐ 3D Result Visualization (UCSF Chimera)

BenchMarking: ☐ Output the benchmarking results to ResMaps.log

Use GPU: ☐ CUDA C   DevID(Optional|default=0): 0

Library file: ResMap_krnl-cuda-9.0-sm35_gpu.so

Chimera path(binary): /opt/Devel_tools/UCSF-Chimera64-1.12/bin/chimera

Figure 7: ResMap Graphical User Interface (here shown for Ubuntu-17.04x64)

The ResMap v-1.95 GUI is very similar to the GUI from previous versions. However, it contains only one tab (for split maps). The tab for processing a single map has been eliminated.

The workflow for using ResMap with the GUI proceeds according to the following steps:

Step 1. Specify Inputs: The initial ResMap GUI (figure 7) allows you to specify inputs. The inputs are grouped according to whether they are required or optional. Optional inputs have default values.

Required Inputs: Volume 1 and Volume 2 are filenames of the two gold standard volumes. You can select the volumes using the **Load Both** button (you may select both files at the same time). Both volumes must be the same structure calculated by splitting the input into two independent sets. The volumes must be strictly cubic maps, that is of size $n_x \times n_y \times n_z$ where $n_x = n_y = n_z$. If this condition is not met, then ResMap will terminate with return code -1.

Optional Inputs: These are inputs for which ResMap provides default values (as indicated in the GUI). But you may change the values if you like. A brief description of the optional inputs is given here. Section 6 gives more details.

As mentioned in Section 2, ResMap calculates local resolution by iteratively seeking voxels with resolutions from the finest resolution, called **Finest Res.** in the GUI, to the coarsest resolution, called **Coarsest Res.** in the GUI, in steps of **Step size**. The default values of these parameters can be changed manually in the GUI. You may also provide a binary mask for your volume. The mask file can be loaded using the **Load** button to the right of the **Mask Volume** field. A mask file is not necessary for ResMap to work. ResMap attempts to calculate its own mask if you do not provide one.

The 2D Results Visualization(ResMap) checkbox is set to true by default, this shows 2D rendering of the output. The output is described below in detail.

There is also the option for a 3D visualization output via UCSF Chimera. This produces an animation of the local resolution. The animation will (should) launch automatically using an auto–generated chimera input script. If the 3D result Visualization checkbox is selected, then you must specify the path to the Chimera executable binary file either by typing it directly or using Browse button next to the **Chimera Path (binary)** field.

The **Benchmarking** option provides extensive benchmarking (e.g., execution time for various steps in ResMap). Most users do not need this option and should unselect it.

The **Use GPU** checkbox enables the use of GPU devices. If this box is unchecked, then ResMap runs solely on the CPU. Checking this box enables ResMap to use GPUs. Additional information is also required in order to use GPUs. This information is as follows:

1. Required: Different types of GPUs have different computational architectures, and ResMap GPU libraries are separately compiled for the different architectures. According to NVIDIA convention, GPU architectures are named as *sm_xy*, e.g. *sm_*30. The names are such that if $x_1y_1 \leq x_2y_2$, then all capabilities of $sm\_x_1y_1$ are included in those of $sm\_x_2y_2$. Table 2 (adapted from the CUDA Toolkit Documentation) shows the GPU type and the corresponding *sm_* name. If your GPU belongs to a particular row in the table, then any library compiled for the *sm_* number in that row, or above that row, can be used. For example, GTX 1080 Ti GPUs have a Pascal architecture, and thus any of the libaries compiled for *sm_*62, *sm_*61, *sm_*60, *sm_*53, *sm_*52, *sm_*50, *sm_*35, *sm_*32, and *sm_*30 can be used with it.

   The **Library file** text field is used to specify the GPU library suitable for your GPU. Since the library compiled for *sm_*35 can be used with all GPUs listed in Table 2, it is set as the default library. Other libraries can be chosen from one of the precompiled library files mentioned in Section 4.1. The library file can be chosen by clicking on the load button, navigating to the directory where the downloaded files are saved and choosing the appropriate file.

Table 2: GPU achitecture and sm_ numbers

Arch. Name	*sm*_ numbers
Kepler	sm_30, sm_32, sm_35
Maxwell	sm_50, sm_52, sm_53
Pascal	sm_60, sm_61, sm_62
Volta	sm_70

There is no distinct performance gain in selecting an *sm* of 60 versus a 35, for example. The output and computation will be the same.

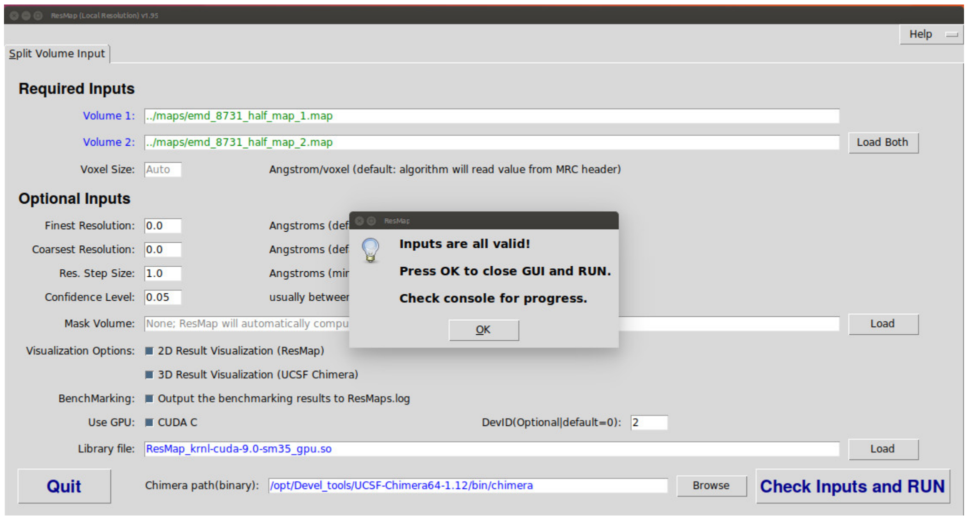
2. Optional: The **DevId**: For computers that have multiple GPUs (e.g. RELION 3.0 computers), the GPUs are identified by device ids. The device ids are numerical and are identified as $0, 1, 2, \cdots$. ResMap uses only a single GPU and the **DevId** box in the GUI allows you to specify the GPU device that ResMap should use. The **default DevId is** 0, but if you have multiple GPUs and want ResMap to use another device, then enter its device number in the DevID box.

Check Inputs and RUN: Once all the inputs are entered, this button checks that the inputs are ok, after which a pop-up appears indicating that "inputs are all valid". See Figure 8. Clicking the **OK** button on this pop-up closes this GUI and opens the pre-whitening GUI.

If there is a problem with reading the volumes (e.g. files are missing, or file names have a typo) a warning pop-up appears (figure 9). Clicking the **OK** keeps the base GUI, allowing you to change the file names.

Similary if the is a problem with the Chimera executable path name, a pop-up appears (Figure 11). As above, clicking the **OK** keeps the base GUI, allowing you to change the path name. You may, of course, unselect the **3D Result Visualization (UCSF Chimera)** box to disable the search for the Chimera executable. Similarly for the library file.

Quit: This button terminates ResMap.



Required Inputs

Volume 1: ../maps/emd_8731_half_map_1.map

Volume 2: ../maps/emd_8731_half_map_2.map

Voxel Size: Auto Angstrom/voxel (default: algorithm will read value from MRC header)

Optional Inputs

Finest Resolution: 0.0 Angstroms (def

Coarsest Resolution: 0.0 Angstroms (def

Res. Step Size: 1.0 Angstroms (mir

Confidence Level: 0.05 usually betweer

Mask Volume: None; ResMap will automatically compu

Visualization Options: 2D Result Visualization (ResMap)

3D Result Visualization (UCSF Chimera)

BenchMarking: Output the benchmarking results to ResMaps.log

Use GPU: CUDA C DevID(Optional|default=0): 2

Library file: ResMap_krnl-cuda-9.0-sm35_gpu.so

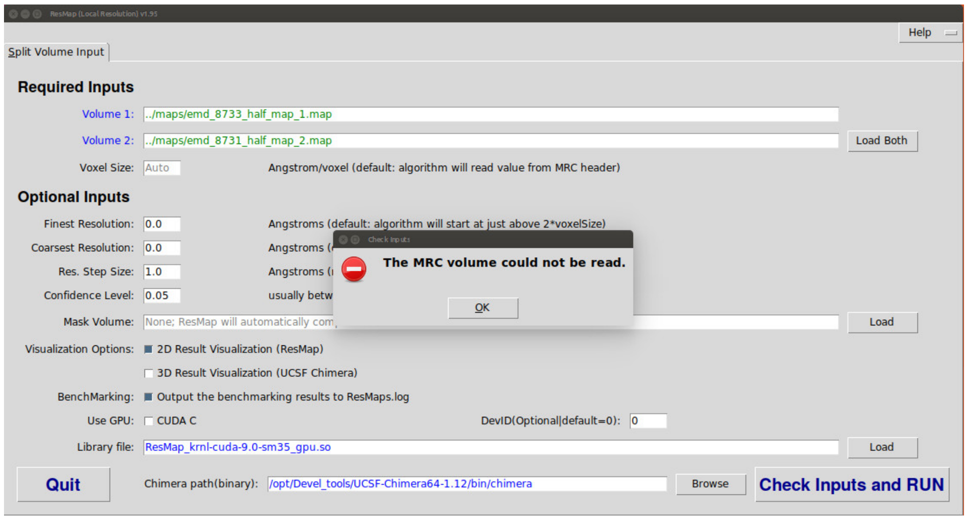
Chimera path(binary): /opt/Devel_tools/UCSF-Chimera64-1.12/bin/chimera

Inputs are all valid!

Press OK to close GUI and RUN.

Check console for progress.

Figure 8: ResMap Graphical User Interface when all the inputs have been entered correctly (shown for Ubuntu-17.04x64 here)



ResMap (Local Resolution) v1.95

Split Volume Input

Required Inputs

Volume 1: ../maps/emd_8733_half_map_1.map

Volume 2: ../maps/emd_8731_half_map_2.map

Voxel Size: Auto — Angstrom/voxel (default: algorithm will read value from MRC header)

Optional Inputs

Finest Resolution: 0.0 — Angstroms (default: algorithm will start at just above 2*voxelSize)

Coarsest Resolution: 0.0 — Angstroms (

Res. Step Size: 1.0 — Angstroms (

Confidence Level: 0.05 — usually betw

Mask Volume: None; ResMap will automatically com

Visualization Options: [✓ 2D Result Visualization (ResMap)] 3D Result Visualization (UCSF Chimera)

BenchMarking: [✓ Output the benchmarking results to ResMaps.log]

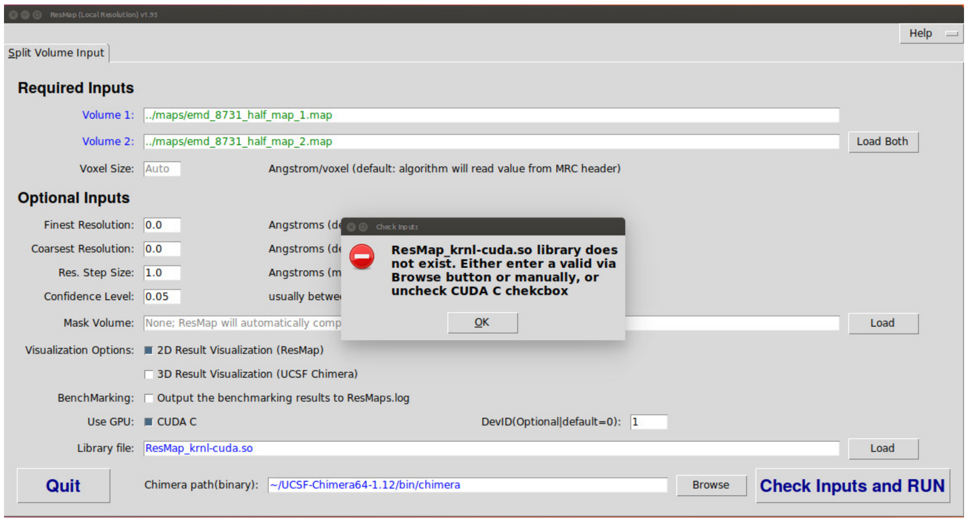
Use GPU: CUDA C — DevID(Optional|default=0): 0

Library file: ResMap_krnl-cuda-9.0-sm35_gpu.so

Chimera path(binary): /opt/Devel_tools/UCSF-Chimera64-1.12/bin/chimera

Check Inputs: **The MRC volume could not be read.**

Figure 9: ResMap pop-up when input files cannot be read.



ResMap (Local Resolution) v1.95

Split Volume Input

Required Inputs

Volume 1: ../maps/emd_8731_half_map_1.map

Volume 2: ../maps/emd_8731_half_map_2.map

Voxel Size: Auto — Angstrom/voxel (default: algorithm will read value from MRC header)

Optional Inputs

Finest Resolution: 0.0 — Angstroms (de

Coarsest Resolution: 0.0 — Angstroms (de

Res. Step Size: 1.0 — Angstroms (m

Confidence Level: 0.05 — usually betwe

Mask Volume: None; ResMap will automatically comp

Visualization Options: [✓ 2D Result Visualization (ResMap)] 3D Result Visualization (UCSF Chimera)

BenchMarking: Output the benchmarking results to ResMaps.log

Use GPU: [✓ CUDA C] — DevID(Optional|default=0): 1

Library file: ResMap_krnl-cuda.so

Chimera path(binary): ~/UCSF-Chimera64-1.12/bin/chimera

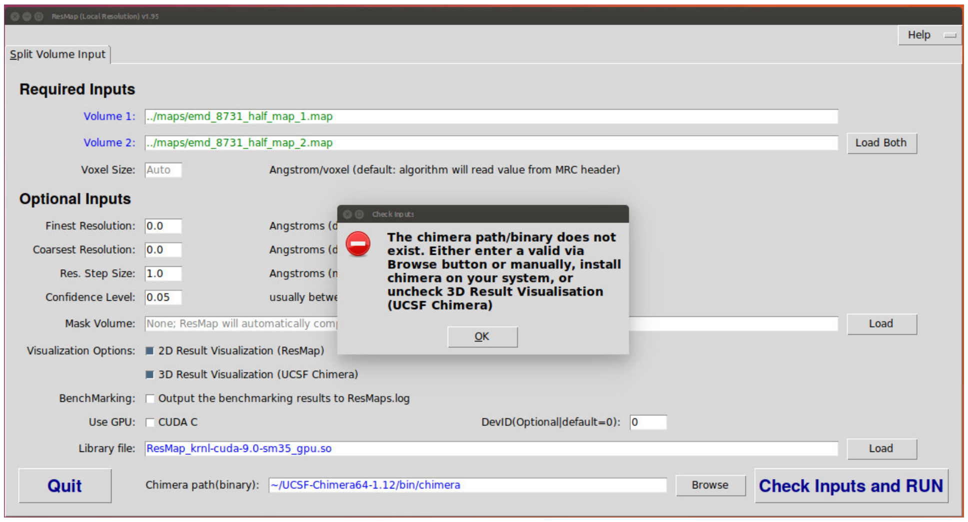
Check Inputs: **ResMap_krnl-cuda.so library does not exist. Either enter a valid via Browse button or manually, or uncheck CUDA C chekcbox**

Figure 10: ResMap Graphical User Interface when input files cannot be read for the kernel library.

Step 2. Pre-whiten the Volumes: The ResMap pre-whitening tool GUI is shown in figure 12. The GUI serves three purposes: to show the signal and noise spectrum in the input volumes, to interactively fit a noise model to the noise spectrum, and to display the pre-whitened volumes and spectra.

The top left subplot of the GUI shows the following spectra in a log scale:

1. The signal power spectrum of the input volume [Blue curve] (estimated as the power spectrum of the mean of the two volumes)
2. The power spectrum of the noise [Cyan curve] (estimated from the difference of the two volumes)
3. A fit of the noise model to the noise spectrum [Dotted Red curve], between elbow resolution 10Å (default) and Nyquist resolution. This fit is used to pre-whiten the volumes.
4. The pre-whitened signal power spectrum [Blue curve].



Split Volume Input

Help

Required Inputs

Volume 1: ../maps/emd_8731_half_map_1.map

Volume 2: ../maps/emd_8731_half_map_2.map Load Both

Voxel Size: Auto Angstrom/voxel (default: algorithm will read value from MRC header)

Optional Inputs

Finest Resolution: 0.0 Angstroms (d

Coarsest Resolution: 0.0 Angstroms (d

Res. Step Size: 1.0 Angstroms (n

Confidence Level: 0.05 usually betwe

Mask Volume: None; ResMap will automatically comp Load

Visualization Options: ■ 2D Result Visualization (ResMap)

■ 3D Result Visualization (UCSF Chimera)

BenchMarking: □ Output the benchmarking results to ResMaps.log

Use GPU: □ CUDA C DevID(Optional|default=0): 0

Library file: ResMap_krnl-cuda-9.0-sm35_gpu.so Load

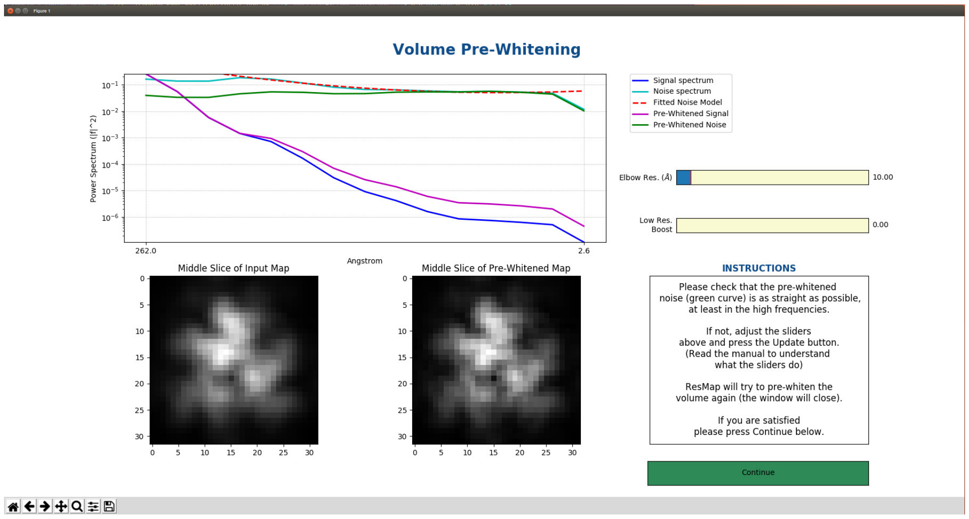
Quit Chimera path(binary): ~/UCSF-Chimera64-1.12/bin/chimera Browse Check Inputs and RUN

Check Inputs

The chimera path/binary does not exist. Either enter a valid via Browse button or manually, install chimera on your system, or uncheck 3D Result Visualisation (UCSF Chimera)

OK

Figure 11: ResMap Graphical User Interface when input files cannot be read.



Please check that the pre-whitened noise (green curve) is as straight as possible, at least in the high frequencies.

If not, adjust the sliders above and press the Update button. (Read the manual to understand what the sliders do)

ResMap will try to pre-whiten the volume again (the window will close).

If you are satisfied please press Continue below.

Figure 12: ResMap Pre-Whitening Interface as seen on a reasonably sized volume. The passband has been set adjusted to 14.41Å to obtain a flat Green curve.

5. The pre-whitened noise power spectrum [Green curve].

The legend, displayed to the right of the plot in the GUI, labels the curves. The GUI also displays a slice through the center of the volume before and after pre-whitening.

NOTE: It is critically important that pre-whitened noise spectrum be flat. That is, the green curve in the above plot should be as flat and as horizontal as possible, especially in the range of Angstroms that you expect the local resolution to lie in.

ResMap makes an automatic attempt at pre-whitening. But you can interactively adjust and improve the pre-whitening. The adjustment is done using the two sliders on the right labeled **Elbow Res.** and **Low Res. Boost** (see Section 2 for explanation). The elbow slider adjusts the elbow resolution. The Low Res. Boost slider gives you some control over pre-whitening at lower than elbow resolution. Increasing the slider value boosts the low resolution noise pre-whitening.

As the sliders are adjusted, ResMap recalculates and displays all the spectra. You are encouraged to experiment with the sliders till the pre-whitened noise spectra (the green curve) is as flat as possible between the local resolution limits that you are interested in.

Once pre-whitening is satisfactory, clicking on the green **Continue** button at the bottom right closes the pre-whitening GUI. ResMap proceeds to pre-whiten the input volumes using the last slider values. Slices of the pre-whitened mean volume are then displayed for inspection, as shown in figure 13. Clicking this window close launches Step 3, the local resolution calculation.

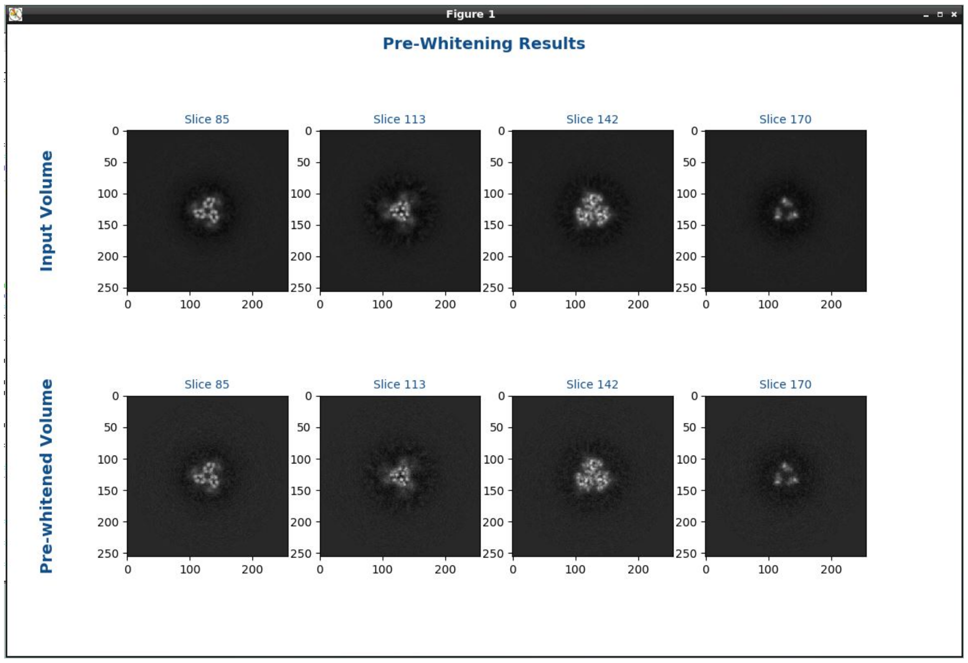


Figure 13: ResMap Pre-whitening inspection GUI. Clicking this GUI close starts local resolution calculation

NOTE: Pre-whitening is not instantaneous after the sliders are moved! It involves a few Fourier transforms of the volumes; please give it some time to re-compute the pre-whitening.

Pre-whitening can be slow for large volumes. To increase speed, ResMap subsamples large volumes in the pre-whitening stage. A console message is displayed, if subsampling is used. Subsampling is used only in the pre-whitening stage, all remaining processing is carried out without subsampling.

If the pre-whitening causes unexpected problems, try setting the Low Res. Boost to 0. Please also notify the authors as we would like to make this tool as robust as possible.

Step 3. Calculate and Visualize the Local Resolution: No user interaction is necessary after the pre-whitening step. ResMap proceeds on its own to calculate local resolution and saves it as an MRC/CCP4 volume file with the filename of the input volume, but with `_resmap.map` appended to the end. Resmap also writes out a UCSF Chimera script for 3D visualization with the same filename as the input volume, but with `_resmap_chimera.cmd` at the end. ResMap also produces a log file, which contains details that are useful for understanding ResMap execution times and intermediate results. Most users need not bother with the log file and can safely delete it after using ResMap.

In addition to the output file, ResMap offers two modes of visualization for the local resolution results: a 2D and a 3D mode. The visualization mode is selected via the **Visualization Options** in the initial ResMap GUI (Fig.7). Both modes can be selected at the same time.

2D mode: In the 2D mode, ResMap displays a histogram of local resolutions (Fig.14), a set of slices through the volume (Fig.16), and the same set of slices rendered in color with the local resolution (Fig.15).

You can save the 2D mode figures in a variety of formats: PNG, TIFF, JPEG, PS, EPS, SVG, PDF, and LaTeX PGF.

3D mode: In the 3D mode ResMap outputs a Chimera script for visualizing the local resolution map. ResMap also offers an option to automatically launch this script into Chimera after ResMap has finished its computations. The Chimera script makes use of Chimera's `Tools > Volume Data > Surface Color` to color the surface of the input map with the local resolution. The script animates a slice as it goes through a low-pass filtered contour of the input volume.

When the automated Chimera script launches, Chimera requests permission to open the script, Fig. 18. Click yes in the bottom left of the warning window to proceed. Then, the sliced volume in chmiera will appear as in Fig. 19.

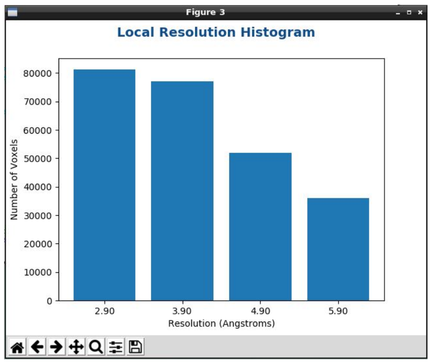


Figure 14: Output histogram from the ResMap computation

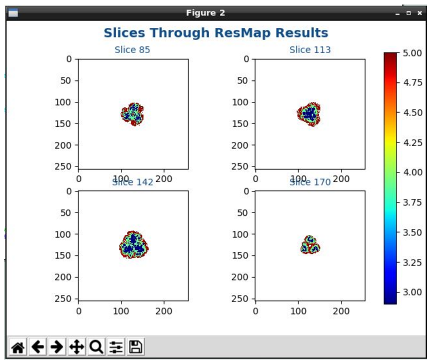


Figure 15: The different slices shown using a gradient scale measure from the ResMap computation.

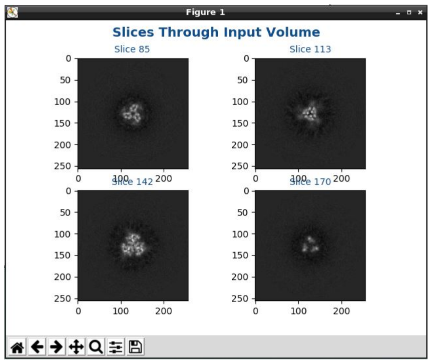
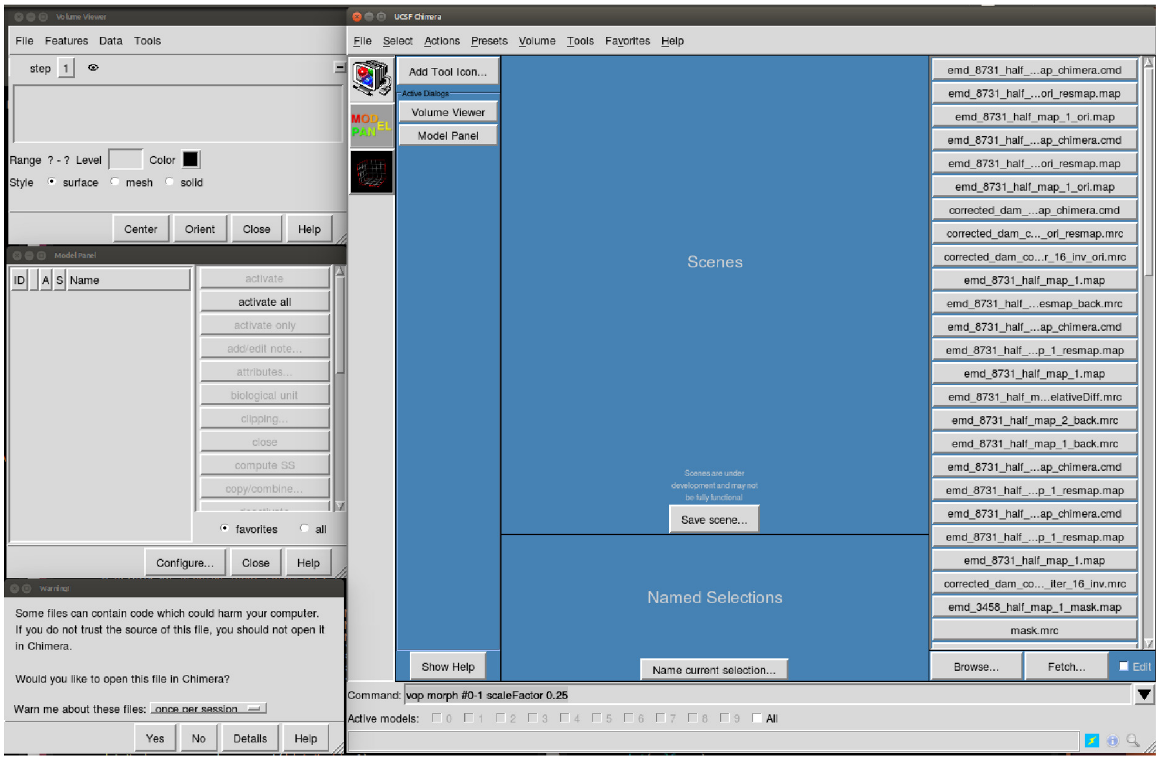


Figure 16: Slices through the volume.

Figure 17: ResMap 2D Visualization Results. Map EMD–8731 obtained from the PDBJ database

This completes the description if using ResMap from the GUI.



Range ? - ? Level Color

Style [✓ surface] mesh solid

emd_8731_half_...ap_chimera.cmd
emd_8731_half_...ori_resmap.map
emd_8731_half_map_1_ori.map
emd_8731_half_...ap_chimera.cmd
emd_8731_half_...ori_resmap.map
emd_8731_half_map_1_ori.map
corrected_dam_...ap_chimera.cmd
corrected_dam_c..._ori_resmap.mrc
corrected_dam_co...r_16_inv_ori.mrc
emd_8731_half_map_1.map
emd_8731_half_...esmap_back.mrc
emd_8731_half_...ap_chimera.cmd
emd_8731_half_...p_1_resmap.map
emd_8731_half_map_1.map
emd_8731_half_m...elativeDiff.mrc
emd_8731_half_map_2_back.mrc
emd_8731_half_map_1_back.mrc
emd_8731_half_...ap_chimera.cmd
emd_8731_half_...p_1_resmap.map
emd_8731_half_...ap_chimera.cmd
emd_8731_half_...p_1_resmap.map
emd_8731_half_map_1.map
corrected_dam_co..._iter_16_inv.mrc
emd_3458_half_map_1_mask.map
mask.mrc

Warning!

Some files can contain code which could harm your computer.
If you do not trust the source of this file, you should not open it in Chimera.

Would you like to open this file in Chimera?

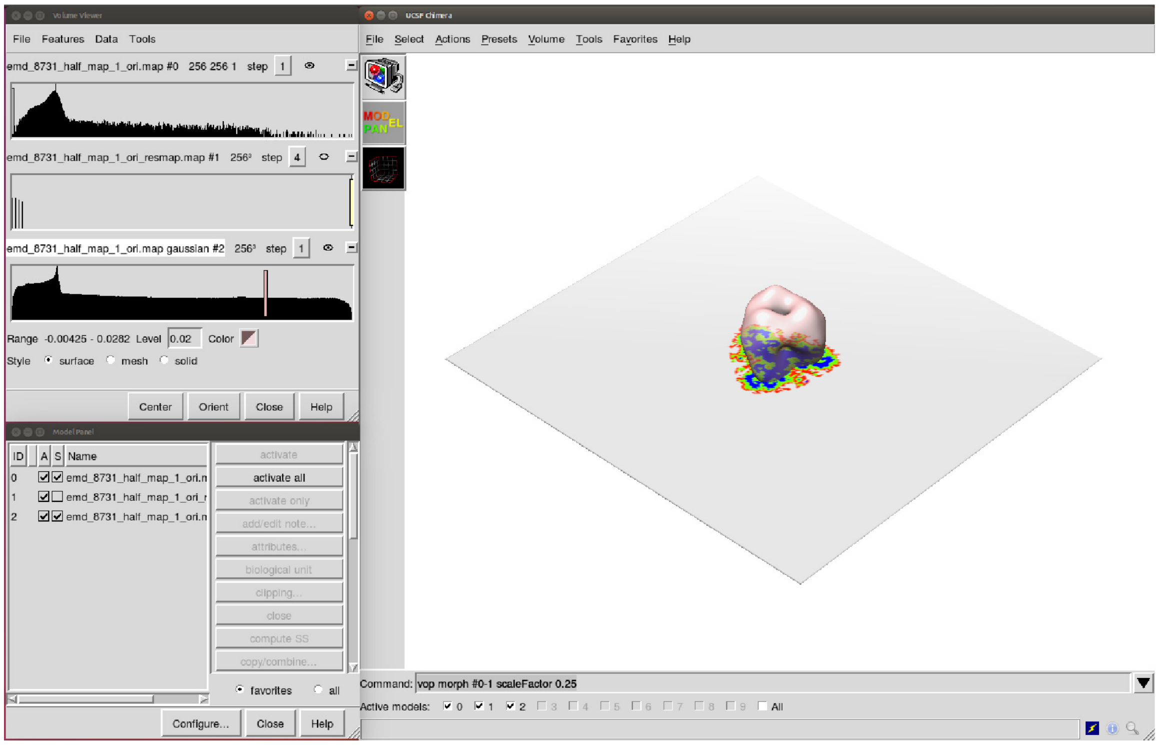
Warn me about these files: once per session

Yes No Details Help

Command: vop morph #0-1 scaleFactor 0.25

Active models: 0 1 2 3 4 5 6 7 8 9 All

Figure 18: Chimera warning message at automated launch, click yes on the warning window.



emd_8731_half_map_1_ori.map #0 256 256 1 step 1

emd_8731_half_map_1_ori_resmap.map #1 256³ step 4

emd_8731_half_map_1_ori.map gaussian #2 256³ step 1

Range -0.00425 - 0.0282 Level 0.02 Color

Style [✓ surface] mesh solid

ID	A	S	Name
0	✓	✓	emd_8731_half_map_1_ori.n
1	✓		emd_8731_half_map_1_ori_r
2	✓	✓	emd_8731_half_map_1_ori.n

Command: vop morph #0-1 scaleFactor 0.25

Active models: ✓ 0 ✓ 1 ✓ 2 3 4 5 6 7 8 9 All

Figure 19: Chimera automated annimation. Here shown Map EMD–8731 obtained from the PDBJ database

5.2 Using ResMap in Command Line Mode

To run the program in command line mode, open a command line prompt or terminal and change your directory to where you downloaded the ResMap binary. The command line interface can be invoked using “ResMap-1.95-distrib-Linux”, which has an help option which looks as follows:

```
  user@host:> python3 ./ResMap-\version-distrib-Linux --help
```

The help option produces a list of command options:

```
Usage:
  ResMap.py [(--noguiSplit INPUT1 INPUT2)] [--vxSize=VXSIZE]
            [--pVal=PVAL]
            [--minRes=MINRES] [--maxRes=MAXRES] [--stepRes=STEPRES]
            [--maskVol=MASKVOL]
            [--vis2D]
            [--launchChimera=CHIMERABIN]
            [--use_gpu=USE_GPU]
            [--set_gpu=SET_GPU]
            [--lib_krnl_gpu=LIB_GPU]
            [--noiseDiag]
            [--doBenchMarking]

NOTE: INPUT(s) is/are mandatory

Arguments:
  INPUTS                        Input volumes in MRC format

Options:
  --noguiSplit                  Run ResMap for Split Volumes in command-line mode.
  --vxSize=VXSIZE               Voxel size of input map (A) [default: 0.0].
  --pVal=PVAL                   P-value for likelihood ratio test [default: 0.05].
  --minRes=MINRES               Minimum resolution (A) [default: 0.0].
  --maxRes=MAXRES               Maximum resolution (A) [default: 0.0].
  --stepRes=STEPRES             Step size (A) [default: 1.0].
  --maskVol=MASKVOL             Mask volume.
  --vis2D                       Output 2D visualization.
  --launchChimera=CHIMERABIN    Launch Chimera after execution with bin Path
                               [default:~/UCSF-Chimera64-1.11.2/bin/chimera].
  --use_gpu=USE_GPU             Uses GPU {yes|no} [default=no].
  --set_gpu=SET_GPU             Sets GPU {(0,1,...) [default=0].
  --lib_krnl_gpu=LIB_GPU        Specifies the library path for the GPU kernels
  --noiseDiag                   Run and show noise diagnostics.
  --doBenchMarking              Run ResMap in BenchMarking mode, if combined with
                                --noguiSplit no GUI will appear.
  --help -h                     Show this help message and exit.
  --version                     Show version.
```

5.2.1 CPU only

ResMap v-1.95 can run in two modes when launched from the command line:

- **Mode 1:** Suppress the main GUI shown in Fig.7 but show other GUIs:

```
    ResMap-\version-distrib --noguiSplit {path to:}/half-map-1.map {path to:}/half-
    map-2.map
```

- **Mode 2:** Suppress all GUIs and popups

```
    ResMap-\version-distrib --doBenchMarking --noguiSplit {path to:}/half-map-1.map
      {path to:}/half-map-2.map
```

  This mode also generates benchmarking.

Chimera can be launched in both modes by adding the launchChimera command line option, for example:

```
ResMap-\version-distrib --doBenchMarking --noguiSplit {path to:}/half-map-1.map {path
    to:}/half-map-2.map --launchChimera={path to}/bin/chimera
```

5.2.2 Using GPU

To enable GPU use, add $use\_gpu = yes$ to the command line:

```
ResMap-\version-distrib --noguiSplit {path to:}/half-map-1.map {path
   to:}/half-map-2.map --use_gpu=yes
```

You can also specify a GPU device id if you like (the default is device id 0) via $set\_gpu = \{0, 1, ...\}$. For example,

```
ResMap-\version-distrib --noguiSplit {path to:}/half-map-1.map {path
   to:}/half-map-2.map --use_gpu=yes --set_gpu=2
```

will use the GPU device with $DevID = 2$.

The default kernel library is the same as in the GUI, that is

`ResMap_krnl-cuda-V9.0.102-sm35_gpu.so`

but it can be specified using the $lib\_krnl\_gpu = file.so$ command, e.g.

```
ResMap-\version-distrib --noguiSplit {path to:}/half-map-1.map {path
   to:}/half-map-2.map --use_gpu=yes --set_gpu=2 --lib_krnl_gpu=ResMap_krnl-cuda-V9
    .0.102-sm52_gpu.so
```

As a final example, the following command line will suppress all GUI display, execute on the GPU with device id 2 using `ResMap_krnl-cuda-V9.0.102-sm52_gpu.so` and launch Chimera upon completion:

```
ResMap-\version-distrib --doBenchMarking --noguiSplit {path to:}/half-map-1.map {path
    to:}/half-map-2.map --launchChimera={path to}/bin/chimera --use_gpu=yes --set_gpu=2
    --lib_krnl_gpu=ResMap_krnl-cuda-V9.0.102-sm52_gpu.so
```

6 ResMap Inputs and Settings: Complete Details

This section gives all of the details of ResMap inputs and settings.

6.1 Input Volume(s)

ResMap v-1.95 requires MRC/CCP4 format volumes for input. In addition ResMap v-1.95 **requires** that:

1. The volumes have the same number of voxels along each dimension.
2. The particle be centered in the volume. (Helical particles are not well supported, yet.)
3. The background not be masked out.

ResMap v-1.95 prefers the volume to be unfiltered. **Hard filtering the volume at FSC is especially discouraged.** Other filtering, e.g. B-factor sharpening, is also discouraged. Please refer to Section 2 for examples of valid and invalid inputs.

6.2 Resolution range and Step Size

ResMap calculates local resolution by iteratively seeking voxels with resolutions from the finest resolution, called **Finest Res.** in the GUI, to the coarsest resolution, called **Coarsest Res.** in the GUI, in steps of **Step size**. The default values of these parameters can be changed manually in the GUI.

Caution : Smaller step size will lead to slower execution. We do not recommend a step size small than 0.5 Å. Users have occasionally informed us that they have used ResMap with step size as low as 0.1 Å. This is not advisable: it is unclear that resolution difference of this magnitude can be reliably estimated from noisy volumes.

6.3 Confidence Level

This is the p-value of the statistical hypothesis test on which ResMap is based on. It is customarily set to 0.05 although you are welcome to reduce it (e.g. 0.01) if you would like to obtain a more conservative result. Empirically, ResMap results are not much affected by the p-value.

6.4 Mask Volume

You may optionally provide a mask for the volume (see the Mask Volume field in the input GUI, Fig.7). ResMap uses the mask as a hint to separate particle from the background (when the signal spectrum is esimated). The mask should be binary and must match the input volumes in size. The mask volume should contain zeros (0) for the background/solvent voxels and some positive value (typically 1) for the particle voxels.

If a mask is not provided, ResMap calculates a mask by low-pass filtering and thresholding the mean input volume.

6.5 ResMap Pre-Whitening

ResMap pre-whitening is described in detail in Section 5.1.

7 Test data provided

This release also contains two volumes test data that you can use to test your installation of ResMap. The two volumes are obtained from the Protein Data Bank[2], and are volumes of EMD–8731 the Influenza hemagglutinin (HA) trimer reconstructed at a 4.2 Angstrom resolution.

[2]`https://pdbj.org/`

Lauching ResMap at the command line (*doBenchMarking* and *launchChimera* being optional) via

```
user@host:> python3 ResMap.py --doBenchMarking --noguiSplit ./test_maps/
     emd_8731_half_map_1.map ./test_maps/emd_8731_half_map_2.map --launchChimera=~/UCSF-
     Chimera64-1.12/bin/chimera
```

gives the following output at the end of the execution:

```
    [Number of voxels assigned in this iteration =  36140] ==
    [We have reached MaxRes = 5.000] ==

  MEAN RESOLUTION in MASK = 4.516(CPU) 4.453(GPU)
MEDIAN RESOLUTION in MASK = 3.900

TOTAL :: [Time elapsed: 0.0 minutes and 48.8524 seconds] ::
 << [Volume: Line 1284: ./ResMap_algorithm.py ---> ResMap_algorithm ---> 256 x 256 x 256
     ---> TOTAL :: 48.8536 secs ---> 0.0 min and 48.8536 secs] >>
RESULT WRITTEN to MRC file: ./test_maps/emd_8731_half_map_1_ori_resmap.map
CHIMERA SCRIPT WRITTEN to: ./test_maps/emd_8731_half_map_1_ori_resmap_chimera.cmd

 == [ATTEMPTING TO LAUNCH CHIMERA... ] ==
    [Searching for Chimera's binaries in path variable] ==
    [Looking in possible known locations first...:] ==
[>>>>                                              ] (7\%, 0.0003 (secs))
 == [ResMap has exited with no problems.] ==
 == [Return code:        0] ==
```

Identical results are obtained via the Graphical User Interface.

The computation time on an Intel Core i7-6900K CPU @ 3.20GHz x 16 is about 48 seconds.

8 Troubleshooting FAQ

The library does not work!

Please contact the authors. But first, please check whether the correct CUDA version (which is 9.0) is installed on the system and that you are using the appropriate *sm_* library for your GPU. Also, make sure that all of the required software has been installed correctly, see Sec. 4. Furthermore, note that for volumes larger than $700 \times 700 \times 700$, the 8 GB-DDR5 devices runs out of memory.

The binary does not work!

Please contact the authors. You may also consider installing Python 3 and the dependencies noted in Section 4 and running ResMap from the source.

The results are over-/under-estimating the resolution!

Typically the median local resolution should be close to the FSC resolution. If that does not hold for your volumes, then please check that the power spectrum after the pre-whitening looks reasonable. Please also check that your input volume satisfies the requirements in 6.1. If the calculated resolution still do not make sense, please contact the authors.

Why can't ResMap do X?

We would love to hear your suggestions! Please contact the authors.

9 Citing and Email

If you use ResMap, we kindly ask that you use the following citation:

A. Kucukelbir, F.J. Sigworth, H.D. Tagare, Quantifying the Local Resolution of Cryo-EM Density Maps, *Nature Methods*, Volume 11, Issue 1, Pages 63-65, 2014.

You may contact the authors at *frederic.bonnet@yale.edu* and *hemant.tagare@yale.edu*

10 Known issues

- Two users (Jian Wang and Alvaro Ortega-Esteban) have brought to our attention a bug in one of the Python NumPy libraries. This bug affects the ResMap test which checks whether the volume is low-pass filtered.

  To elaborate: Because hard-low-pass filtering the volume makes it difficult to estimate the SNR beyond the filter cut-off, ResMap explicitly tests whether the volume is low-pass filtered and sets the search range of local resolutions accordingly. The ResMap test uses a Python NumPy function which has a bug. The test actually works in majority of the cases (even with hard-low-pass filtering), but under some conditions, the NumPy bug yields a wrong result for the test, thereby determining the cut-off frequency incorrectly. This bug has no effect on the local resolutions calculated within the cut-off frequency range. And the bug has absolutely no effect on volumes that are not hard low-pass filtered. Nevertheless, in this version of ResMap, the NumPy function has been replaced with an alternative procedure which is bug-free.

- Using older versions of Numpy and Scipy.

```
    Traceback (most recent call last):
    File "ResMap.py", line 565, in <module>
    noiseDiagnostics = args['--noiseDiag']
    File "./ResMap_algorithm.py", line 211, in ResMap_algorithm
    LPFtest                  = isPowerSpectrumLPF(dataPowerSpectrum)
    File "./ResMap_spectrumTools.py", line 422, in isPowerSpectrumLPF
    if peakInd.any():
    AttributeError: 'list' object has no attribute 'any'
```

  This can be resolved by upgrading to the latest numpy and scipy version typically via pip3 for example. The correct versions of these libraries are given in Sec. 4.

11 Acknowledgment

Jian Wang from the School of Physics (Huazhong University of Science and Technology, Wuhan, China) and Alvaro Ortega-Estebanf from the Centro Nacional De Biotecnologica were responsible for pointing out the NumPy bug. Special thanks to Jian Wang for providing us with a patch for the bug. This research was supported by the grant R01GM114051 from the NIGMS.