Real-time three-dimensional cell segmentation in large-scale microscopy data of developing embryos

Johannes Stegmaier[1,2,*], Fernando Amat[1], William C. Lemon[1], Katie McDole[1], Yinan Wan[1], George Teodoro[3], Ralf Mikut[2] and Philipp J. Keller[1,*]

[1] *Howard Hughes Medical Institute, Janelia Research Campus*

19700 Helix Drive, Ashburn, VA 20147, USA

[2] *Karlsruhe Institute of Technology, Institute for Applied Computer Science*

Hermann-von-Helmholtz-Platz 1, 76344 Eggenstein-Leopoldshafen, Germany

[3] *University of Brasilia, Department of Computer Science*

Campus Darcy Ribeiro, CEP 70910-900, Brazil

[*]*Correspondence should be addressed to*

J.S. (johannes.stegmaier@kit.edu) or P.J.K. (kellerp@janelia.hhmi.org).

Instructions for using the RACE segmentation framework

Step-by-step protocol, troubleshooting guide and RACE video tutorial

We provide text and video materials that explain and demonstrate how the RACE cell segmentation framework can be applied to new image data, including a detailed step-by-step protocol (**Box 1**), a troubleshooting guide (**Box 2**) and a RACE video tutorial (*https://bitbucket.org/jstegmaier/race/downloads/RACE_VideoTutorial.zip*).

Building the framework from the sources

The C++ implementation of the RACE segmentation algorithm is available from *http://www.bitbucket.org/jstegmaier/race/*. We take advantage of the CMake build tool to generate project files for various operating systems that make it easy to work with the compiler of your choice (*http://www.cmake.org/*). This tool is used to configure the Insight Toolkit (ITK) libraries and to generate the related project files for both ITK and the segmentation executable itself. ITK is freely available for download from *http://www.itk.org/*. Moreover, the Qt libraries are required and can be obtained from *http://qt-project.org/* (specifically, the QtCore and QtWidgets modules are required). Detailed installation instructions for ITK and Qt are provided on the respective webpages. For development and software testing, we used CMake v3.0.0, ITK v4.3 and Qt v5.2 under Windows 7 Professional 64-bit using Microsoft Visual Studio 2012 and its associated C++ compiler. The software has also been successfully compiled and tested under Windows 8.1, Ubuntu 12.04 LTS, 14.04 LTS, Scientific Linux 5, 6 and Mac OS X 10.9.3.

Notes:

- For processing TIFF files larger than 4GB it is necessary to have a BigTIFF-compatible `libtiff` version installed and to enable the ITK CMake flag `ITK_USE_64BIT_IDS` during ITK makefile generation (see *http://bigtiff.org/*). Furthermore, the flag `ITK_USE_REVIEW` needs to be enabled.
- For processing large images, ITK and Qt libraries and executables need to be compiled as 64-bit versions.
- When observing errors related to missing Qt headers or libraries, check if the header and library paths have been properly set by CMake. Otherwise, add `"QTDIR/include/"`, `"QTDIR/include/QtCore/"` to the header search path and `"QTDIR/lib/"`, `"QTDIR/bin/"` to the library search path. Furthermore, confirm that `"qtmain.lib"` and `"Qt5Core.lib"` are listed as additional dependencies in the Visual Studio project linker settings.
- On Windows systems, we recommend installing ITK at most two sub-folders away from the system root. Otherwise path name length limits of the file system may be exceeded.
- For faster compilation of ITK the following options can be disabled: `BUILD_EXAMPLES`, `BUILD_DOCUMENTATION`, `BUILD_SHARED_LIBS` and `BUILD_TESTING`.

- If you would like to use the GPU-accelerated version of the pipeline, the CUDA Toolkit is needed as well (*https://developer.nvidia.com/cuda-toolkit*).

Compiling the sources

After installation and compilation of all prerequisites, it should be possible to compile the application. This can be done with the CMake build tool and a compiler of your choice using the `CMakeLists.txt` located in the folder `PROJECTROOT/Project/CMakeQt5`. Therefore, the source path of CMake has to be set to `PROJECTROOT/Project/CMakeQt5/` and the build path has to be set *e.g.* to `PROJECTROOT/Project/Buildx64/` (folder names are denoted relative to the installation directory). If ITK, Qt or CUDA are not found automatically, make sure to redirect CMake to the paths the respective libraries are located at. After successful Makefile generation using CMake it should be possible to compile the segmentation algorithm, *e.g.* using the `make` command within a Unix terminal or building the generated Visual Studio project files.

Application example

Once code generation is complete and the executable has been successfully built, the program can be started via the Unix terminal application with the call `./XPIWIT < input.txt` or in the Windows command prompt using `XPIWIT.exe < input.txt`, where `input.txt` is a text file that determines (1) the input and output parameters for the executable, (2) an XML pipeline to process and (3) additional parameters. Note that the executable itself is called without parameters but all inputs are piped either directly or using a file. The application expects information about the following input parameters in the `input.txt` text file:

```
--output PROJECTROOT/Example/Results/
--input 0, PROJECTROOT/Example/Data/Membrane/Drosophila_c=00_t=0010.tif, 3, float
--input 1, PROJECTROOT/Example/Data/Nuclei/Drosophila_c=02_t=0010.tif, 3, float
--xml PROJECTROOT/Example/membraneSegmentationDrosophilaMS.xml
--seed 0
--lockfile off
--subfolder filterid, filtername
--outputformat imagename, filtername
--end
```

The most important parameters are the output path (line 1), the input paths (lines 2 and 3) and the path of the XML file (line 4). Always use "/" as a folder separator instead of "\", even on Windows systems. The remaining parameters can be left unchanged. The `Example` folder contains three small 3D data sets showing labeled nuclei and membranes in a *Drosophila* embryo. Input files for the compiled executable as well as XML pipeline files for segmentation and for GPU version are also provided. Make sure to adjust the absolute paths within the input files according to the specific location on your disk. If program execution was successful, the specified output folder should contain a log file including processing time, parameters and

pipeline components as well as the resulting image data. All parameters can be adjusted in the file `PROJECTROOT/Example/membraneSegmentationDrosophila*.xml` using a simple text editor. A comprehensive list with available filter options as well as the description of parameters can be requested with the call `XPIWIT.exe --filterlist myfilters.xml` on Windows and `./XPIWIT --filterlist myfilters.xml` on Unix-based systems, respectively.

Box 1 | Step-by-step protocol for the RACE cell segmentation framework

Note: We provide a video tutorial that illustrates the steps described in this protocol using SiMView example image data from an early *Drosophila* embryo (https://bitbucket.org/jstegmaier/race/downloads).

Step	Description
A) Download and install precompiled RACE software package	In order to use the RACE segmentation framework please select one of the supplementary software packages for the operating system of your choice and download the package from *https://bitbucket.org/jstegmaier/race/*. Extract the archive to your local hard drive and start the graphical user interface (GUI) by double-clicking "RACEGUI.exe" (Windows) or by starting the shell script "./RACEGUI.sh" from the terminal (Mac OS X and Ubuntu). These files are located in the sub-folder "Bin". **Note:** Ensure that the path to the executable does not contain spaces or special characters.
B) Specify input data	To process your image data, simply drag and drop a valid single-channel 3D TIFF image stack containing fluorescently labeled cell membranes onto the GUI's first field "Membrane input (File/Folder)". Example image data from an early *Drosophila* embryo are included in the sub-folder "Data" in the provided binary packages. If you additionally acquired image data of fluorescently labeled cell nuclei and would like to use these data for seed detection, drag and drop the corresponding file onto the second field "Nuclei/Seed input (File/Folder)".
C) Specify output folder	In order to define the output folder used to store RACE processing results, drag and drop the desired output folder onto the third field "Output (Folder)". This result folder has to exist prior to RACE execution and write permission to this folder are required. **Note:** Make sure that neither input nor output paths contain any spaces or special characters.
D) Alternative seeding options (optional)	The GUI also allows using seed points derived from an external tracking algorithm such as TGMM (Amat et al., 2014) or manually corrected seed points stored in a CATMAID database (Saalfeld et al., 2009). Use the buttons *Import TGMM Seeds* or *Import CATMAID Seeds* if you would like to import seed points from either TGMM or CATMAID projects, respectively. RACEGUI automatically converts the seed points to a valid CSV seed file that can be used for seeding in RACE. Alternatively, you can also use your own seed points stored in a CSV-based format using the ";"-delimiter. In this format, each row contains information about one seed point: the first column provides the unique identifier of the seed point and columns 3-5 provide the seed location in image pixel coordinates. If you have such a CSV file, simply drag and drop it onto the second field "Nuclei/Seed input (File/Folder)" instead of a nucleus image and set the seeding mode to "CSV".
E) Select RACE implementation	Choose the version of the RACE algorithm you would like to use by selecting the seeding type (Membrane, Nuclei or CSV) and possible acceleration options (ITK (default), NScale, CUDA or NScale+CUDA). Note that a CUDA-enabled device is required for the GPU-accelerated (CUDA) version of RACE.
F) Tune RACE parameters for optimal segmentation quality	Several parameters in the cell segmentation pipeline need to be set correctly to ensure that the algorithm produces optimal results. Most of the framework's parameters represent the microscope's optical configuration or can be directly derived from basic prior knowledge about the investigated biological model system. The image-dependent intensity thresholds can be visually determined, *e.g.* by using Fiji (Schindelin et al., 2012). For debugging and parameter adjustment, it is advisable to write images produced by intermediate processing steps to disk and tune parameters step-by-step in the order outlined below. In the following paragraphs we describe each of the parameters and include strategies for determining their optimal settings. All parameters are adjustable through the XML configuration files (**Box 2**).

Box 1 | Step-by-step protocol for the RACE cell segmentation framework (continued)

Step	Description
G) Configure microscope- and specimen-dependent parameters	**(1)** The *image sampling ratio, i.e.* the ratio of axial vs. lateral voxel size in the image data, is directly defined by the volume acquisition settings and the microscope's detection system. For instance, if the lateral voxel size is 0.4 µm and the axial voxel size is 2.0 µm, enter a value of 5. **(2)** The minimum value of the *radius range for iterative morphological closing (MinRadius, MaxRadius)* should usually be set to 1 and the maximum value should be set to the minimum cell radius. For most data sets used in this work we used radii $r \in \{1, 2, 3, 4\}$. **(3)** The *minimum seed size (MinSeedArea)* and *maximum 2D segment size (MaxSegmentArea)* parameters can be used to exclude small seeds comprising only a few pixels and large 2D background segments, respectively. Furthermore, the *cell volume boundary (MinVolume, MaxVolume)* guides decisions of the fusion heuristic based on prior information about minimum and maximum cell volumes expected for the given specimen. To take advantage of this feature, simply measure a few representative cell volumes at the lower and the upper end of the spectrum and adjust volume constraints accordingly. **Note:** *MinVolume* and *MaxVolume* are only considered if the optional parameter *SSH Fusion Heuristic* is turned on.
H) Configure intensity-dependent parameters	**(1)** For optimal performance of the membrane-based and the nucleus-based seed detection, the respective *binary threshold (MS, NS)* and *H-maxima level* have to be adjusted such that the detected seeds are sufficiently well separated (see also RACE video tutorial). The threshold should be set such that individual cells are clearly distinguishable. The H-maxima of the squared Euclidean distance map are used to split erroneously connected cells. **(2)** To optimize the balance of over- vs. under-segmentation of the slice-based segmentation in low-contrast regions, the *morphological watershed level* has to be properly adjusted. All local minima below the specified level will be ignored, *i.e.* the level should be set slightly below the minimum intensity of membrane structures that need to be split by a watershed in the membrane-enhanced, iteratively-closed membrane image (see also RACE video tutorial). This is a core parameter of the algorithm and typically varies from sample to sample since it relates to absolute intensity levels.
I) Configure optional parameters (optional)	The remaining parameters and optional fusion heuristics can be tweaked and further optimized, but the default values should generally produce reasonable results. While searching for optimal parameter settings it is helpful to use random labels for final segmentation and write intermediate results to disk in order to investigate the effect of parameter changes. We also recommend using relatively small image regions in this process, such that updated segmentation results are obtained almost instantaneously. The most important optional parameters are: **(1)** The parameters for *Hessian-based membrane enhancement (HessianToObjectnessFilter)* control the regularization scale of the Hessian calculation (σ), the weight of the Frobenius norm of the Hessian (γ) and the influence of the ratio of the two largest eigenvalues (β). We used $\sigma = 2.0$, $\gamma = 0.1$ and $\beta = 1.0$ for all experiments presented in this study. **(2)** The *2D median radius* depends on the noise level of the images. A fixed 5x5 kernel size produced good results across all data sets examined in this study. **(3)** The *standard deviation σ of the LoG filter* used for the nucleus-based seed detection can be directly determined from the equation $r = \text{sqrt}(2) \cdot \sigma$, where r is the nucleus radius in pixels (Lowe, 2004). **Note:** Parameters 1-3 usually do not need to be changed and are thus not exposed in the GUI. However, they can be adjusted in the XML template files as described in **Box 2**.
J) Execute RACE using the current settings	After all parameters have been set, click on the *Run* button to start processing. The progress is displayed in the command line window. If you would like to adjust parameters after inspecting the results of a processing run, simply apply changes as needed and run the application again to obtain new results. Finally, use button *Open Result Folder* to view the final results or button *Quit* to close the application.

Box 2 | Troubleshooting tips

Problem/Question	Solution
What is the best fluorescent marker strategy for my application?	We designed the RACE framework to allow multiple fluorescent marker choices for seed-based fusion of 2D cell segments to complete 3D cell shapes. RACE is capable of automatically extracting seeds either directly from the image data of the cell membrane marker (using an inverted version of the enhanced cell membrane image), but it can optionally also take advantage of a cell nucleus marker. If image data of cell nuclei are available, we usually recommend using this information for seed-based fusion, as the sparseness and blob-like appearance of cell nuclei generally simplifies correct cell identification and localization. For "hollow" specimens with strong auto-fluorescence contribution from interior regions, such as early-stage *Drosophila* embryos with auto-fluorescent yolk, a seeding approach based on cell nucleus markers helps eliminate false positive detections in background regions and directly increases the precision of the algorithm. However, images of cell nuclei should only be acquired and used for seed detection if the imaging speed of the microscope is sufficiently high. Specifically, cell movements during sequential two-color imaging need to be small enough to ensure that segmented nuclei correctly and unambiguously match the spatial domain occupied by the corresponding cell in the membrane channel. Moreover, the nuclei seed detection algorithm can only produce high-quality seeds if spatial resolution and image quality allow for a clear distinction between individual nuclei. This requirement was not fulfilled in our mouse data set, which exhibits very densely packed cell nuclei and for which it was therefore advantageous to extract seed points from the membrane images instead.
How can I improve the seed quality to optimize segmentation results?	Since seed quality directly influences cell shape segmentation quality, we developed several automated mechanisms for eliminating or counteracting erroneous seeding and also implemented infrastructure supporting manual data curation. In order to efficiently suppress false-positive seed points, we introduced quality filters for rejecting small seeds and preventing redundant seeding. In many cases, the use of an additional nucleus channel image can substantially improve seeding quality (see above). For applications that require completely error-free results, we furthermore provide an interface based on CATMAID (Saalfeld et al., 2009) that allows rapid manual correction of remaining seed errors.
Morphological closing seems to be a bottleneck. Can this be improved?	Processing time may increase significantly if the maximum radius is set to an unnecessarily large value, due to the large 3D neighborhood used for the morphological operations. Optionally, the closing operation can also be restricted to 2D masks. If cell sizes are relatively homogeneous, using only the maximum radius might also be sufficient to reduce processing time.
How can I reduce over-segmentation errors?	Similar to MARS and EDGE4D, we implemented two segment fusion heuristics that can partially compensate for over-segmentation errors resulting from incorrect seeds. These heuristics are based on morphological criteria, specific estimates of the physiological range of cell sizes, and can be quickly evaluated by manual inspection of a small number of representative cells. Although very effective when cell sizes are relatively uniform across the specimen, these heuristics can fail when cell shapes and cell volumes vary substantially. An example of such a case is the mouse embryo data set included in our performance analysis. In future releases of RACE, fusion heuristics could be further improved for data sets with very diverse cell morphologies by including more general features, such as intensity profiles, or classification-based approaches.
I would like to perform a large job on a computer cluster.	We decoupled the RACE processing module from its graphical user interface (GUI), which makes it straightforward to execute and parametrize RACE in a headless mode suitable for computer clusters. We also provide platform-independent project files and the RACE source code, which allows recompilation of RACE if a specific desired target operating system is not covered by the executables included in our software repository.

Box 2 | Troubleshooting tips (continued)

Problem/Question	Solution
Is it possible to reduce the memory footprint of RACE?	Even with the algorithmic optimizations already incorporated in the RACE framework, memory requirements can in principle still become a bottleneck when analyzing extremely large image volumes. Therefore, we further optimized memory requirements in the GPU-accelerated version of our pipeline and achieved a 9-fold improvement over the CPU-optimized implementation of RACE. In addition, our GPU-accelerated implementation greatly speeds up some of the most computation-intensive operations performed by RACE. We therefore generally recommend using the GPU-accelerated implementation of RACE if a conventional CUDA-enabled graphics card is available.
My output path does not contain image data after processing is finished.	Please follow these steps to investigate possible sources of this problem: **(1)** Make sure that the name of the path RACE has been extracted to does not contain any spaces or special characters. **(2)** Make sure that you have write permissions for the output folder you selected and that this folder already exists prior to executing RACE. **(3)** Existing results in the output folder will simply be overwritten. Thus, to preserve previous processing results, make sure to select a different output folder or move your temporary results to a different location if you wish to keep them.
I would like to access one of the non-standard parameters of the RACE algorithm.	If you wish to modify parameters that are not directly accessible via the GUI, you can edit the XML template files provided in the folder "PROJECTROOT/Bin/templates/". Simply open and edit one of the "*Template.xml" files with a text editor of your choice and adjust parameters as needed. In addition, we provide RACE project files for XPIWIT, a new software tool that enables easy graphical adjustment of all RACE parameters and furthermore facilitates customization of the pipeline, if needed.
How do I adapt or recompile RACE?	We provide executables for Windows, Mac OS X and Ubuntu. If the operating system of your choice is not among these or if you need to customize specific parts of the pipeline, you can simply compile our framework from the provided sources as detailed above. The modular design of the RACE framework furthermore facilitates replacing, optimizing or customizing individual software components if required for specific applications.