User's Guide to the *u-shape3D* Software Package

User's Guide last updated:
Jan 12, 2024

Publication. The algorithm underlying this software was described in the paper, *Robust and automated detection of subcellular morphological motifs in 3D microscopy images*, published in Nature Methods in 2019.

Overview. The primary purpose of this software package is to detect morphological motifs on 3D surfaces, such as blebs, filopodia, and lamellipodia on cells. However, it can also measure associated features, such as surface curvature and motion. This software is written in Matlab and can be run as either a graphical user interface (GUI) or directly from the command line as scripts. It was mainly developed for use on a cluster and will run in parallel over movie frames and, in some cases, cells. When implemented in a new context, users should verify that both the cell and motif segmentation is reasonable for their data. This software is available under a GNU general public license.

User's Guide. This User's Guide contains an overview of the software, several examples with provided data, and tips for analyzing your own data.

Software. The latest version of the software described here is available at https://github.com/DanuserLab/u-shape3D/

Example data. The sample images, pre-trained motif models, and point spread functions used as examples in this guide can be downloaded from https://cloud.biohpc.swmed.edu/index.php/s/Z9j62w2FCareyJY/download

Memory usage. Processing large 3D images, especially doing so in parallel, can require relatively large amounts of RAM. To run the examples described here on the provided images, we recommend using a system with at least 128 GB of RAM and monitor your memory usage closely. Although this software may work on systems with lower RAM, there is no guarantee, and the user should monitor their memory usage closely.

Software architecture. This software is implemented using an object-oriented framework that has been used for multiple projects published by Gaudenz Danuser's laboratory. Each movie is encapsulated as a MovieData object that contains movie metadata information and links to the raw movie data. Processes act upon and are associated with MovieData objects. Processes can be bundled into packages that track interdependencies between processes that run in serial. Packages are also associated with generic methods for GUI creation. Functions that accumulate data from multiple cells, such as support vector machine training and validation and some data plotting, occur outside of the package framework.

Processes: This package is composed of the following processes:

deconvolution – optionally deconvolves the movie
computeMIP – optionally generates maximum intensity projections of the movie
mesh – creates a triangle mesh representing the cell surface and measures curvature
meshThres – optionally creates a cell mask for validating cell segmentation
surfaceSegment – decomposes the cell surface into convex patches
patchDescribeForMerge – optionally calculates features for merging patches via an SVM
patchMerge – optionally merges patches via an already trained SVM model
patchDescribe – calculates features for patch classification
motifDetect – classifies patches via an already trained SVM model
meshMotion – calculates surface motion
intensity – measures fluorescence intensity near the surface
intensityBlebCompare – compares the outputs of multiple processes
renderMeshTool – plots colored meshes for already run processes

Example 1: Detecting convex motifs (blebs) on one cell using the graphical user interface. This example shows how to detect blebs on one of the three $Kras^{V12}$ labeled MV3 cell images provided as examples. Please follow these steps:

1. Download the code and example images. Put the entire suite of provided code on Matlab's path, for example, by clicking on Matlab's "Set Path" button.
2. To start the graphical user interface, on Matlab's command line type, **`u_quantify`**.
3. Create a new movieData object by clicking on the "New" button. Click the "Add Channel" button, and navigate to the directory of the image you would like to analyze. For this example, go to a cell in the krasMV3 subdirectory in the testData directory. In the Output Path section, click on "Select Path" and add an output path where you would like data to be saved. In the Movie Information section, add a Pixel Size of 160 nm, a Pixel Size Z of 200 nm, and a Time Interval of 60 seconds. Finally, click on the "Save" button. In general, you can add channels one at a time if the channels are in different directories. Alternatively, if the movie's metadata is correct, which it is not in this example, you can use the "Import movie using Bio-Formats" button.
4. From the main Movie Selection screen, run the u-shape3D package by clicking "Continue". Within the package, processes are run by checking the box in front of the process name, clicking the "Setting" button to set any needed settings, and then clicking "Run".
5. Many of the parameters used for this example are default parameters. However, some directories need to be set and some parameters need to be changed. Even if you don't wish to change any parameters for a specific process, before running the process you still need to click on the "Setting" button for that process, and hit the

"Apply" button. The process name will become bold when it can be run. Please set the following parameters,

   a. In "Step 1: Deconvolution", set the PSF path to be the meSpimPSF.mat file within the PSFs directory.
   b. In "Step 3: Mesh", set the Image Gamma to 0.7.
   c. In "Step 9: MotifDetection", set the SVM model paths to SVMcertainBleb1.mat, SVMcertainBleb2.mat, and SVMcertainBleb3.mat in the svmModels subdirectory. (These three models, each generated by a different user, will vote on whether each patch is a bleb.) Also set the remove patches SVM path to SVMcertainRetractionFiber.mat.

6. Run steps 1 through 5, 8, and 9. For convex motif detection, such as the bleb detection in this example, you do not need to run steps 6 and 7. The Matlab command line will show the code's progress. The command line
7. To display meshes colored by motif detection or another output, click on the "Results" button for the corresponding process, or run step 13.

Example 2: Detecting convex motifs (blebs) on three cells using the graphical user interface.

1. Run u_quantify.m and then make or load movieData objects for all three Kras-labeled example images. For each cell, create a unique directory for saving analyses. See example 1.3 for more information about creating movieData objects. If you're already in the u-shape3D section of u_quantify, go to File and then Open to open a new movie.
2. From the main Movie Selection screen, run the u-shape3D package. You can navigate between movies using the arrows at the top of the screen. Run steps 1-5, and 8-9 for all three movies. Note that the Setting section of each process has a check box you can use to apply settings to all movies. At the bottom of each movie's screen, there are also check boxes to check/uncheck processes for all three movies and run all movies.

Example 3: Detecting convex motifs (blebs) on three cells using the provided script.

1. If you haven't already done so, put the software on Matlab's path, for example, by using Matlab's "Set Path" button.
2. Open runKrasBlebSegment.m, the script that analyzes the three provided Kras-labeled cells, by typing `edit runKrasBlebSegment` on Matlab's command line.
3. In the set directories section of the m-file, set the paths for
   a. imageDirectory – the directory of the three provided kras-labeled cells. These cells are assumed to be in the subdirectories Cell1, Cell2, and Cell3, within imageDirectory.
   b. motifModelDirectory – the directory of example pre-trained SVM models
   c. psfDirectory – the directory with the provided meSPIM PSF
   d. saveDirectory – the directory where outputs will be saved
4. Save the m-file and run it by typing `runKrasBlebSegment` on Matlab's command line. Note that a pool for parallel processing will likely open unless you previously configured Matlab to not automatically open pools. If you would like to manually control the parallel processing, then open a parallel pool before running the m-file by right clicking on the array of rectangles on the lower left-hand corner of the main Matlab window.

5. The plotIntensityBlebCompare function, which is called at the end of the script, will make several different kinds of plots that compare kras intensity and blebs across all three cells.

Example 4: Training your own support vector machine. Train your own SVM for bleb detection using the three kras-labeled example images.

1. Run the script described in Example 3 at least through the surfaceSegment and patchDescribe processes.
2. To train your own bleb detector, open the function runKrasTrain.m by typing `edit runKrasTrain` on Matlab's command line.
3. In the Set Directories section at the top of the file, enter the two directories. The analysisDirectory is the directory where the output of the analysis performed in Example 3 is saved. The motifModelDirectory, is the directory where the SVM model that you will generate will be saved. If you intend to use this model to detect blebs, it is simplest to save it in the directory with the example pre-trained SVMs.
4. Run the function by typing `runKrasTrain` on Matlab's command line. Follow the on screen directions to click on patches that are certainly blebs and patches that are certainly not blebs. (Note that the clicking functionality uses aspects of Matlab's built-in figure interface, which changes from version to version. In 2018 versions, the rotate and zoom buttons are hidden on the upper right hand-side of the figure unless you hover over the cell, and to select points after rotating or zooming, you must click on the also hidden rectangular Data Tips button. For these reasons, earlier versions of Matlab may be easier to use. However, in some pre-2016 versions, clicking on the mesh will select the point on the sight line on the back of the mesh, and in other versions you must first rotate the cell before you can zoom.) If you would like to generate training data over multiple sessions, in the m-file, set p.mode to 'continue' rather than 'restart'. You will then be shown only those cells that you haven't previously clicked on.
5. To detect blebs using your newly trained model, in runKrasBlebSegment.m set p.motifDetect.svmPath to be the path of your newly trained model by uncommenting out the line in the Override Default Parameters section. Rerun runKrasBlebSegment.

Note: Models generated by different users differ. For many applications you may want to simultaneously use multiple models generated by different users.

Example 5: Detecting non-convex motifs (lamellipodia) with a graphical user interface.

1. Download the code and example images. Put the entire suite of software on Matlab's path, for example by clicking on Matlab's "Set Path" button.
2. To start the graphical user interface, on Matlab's command line type, **`u_quantify`**.
3. Create a new MovieData object for the example movie in the lamDendritic directory. This cell is a mature dendritic cell with lamellipodia. Example 1.3 describes how to create a movieData object. Set the Pixel Size to 160.1 nm, the Pixel Size Z to 200 nm, and the Time Interval to 60 seconds.
4. In the u-shape3D package, run steps 1 through 9. Leave the default parameters set as is, except set or override the following parameters:
    a. In "Step 1: Deconvolution", set the PSF path to be the meSpimPSF.mat file within the PSFs directory.

b. In “Step 3: Mesh”, set the mesh mode to be “threeLevelSurface”.
c. In “Step 5: SurfaceSegmentation”, set the bleb mode to be “triangleLosMergeThenLocal”.
d. In “Step 7: PatchMerge”, set the svmPath to be the pre-trained model SVMcertainLamPatchMerge.mat in the svmModels subdirectory.
e. In “Step 8: PatchDescription”, check the usePatchMerge check box.
f. In “Step 9: MotifDetection”, check the usePatchMerge check box and set the svmPath to be the pre-trained model SVMcertainLam.mat.

5. To display meshes colored by motif detection or another output, click on the “Results” button for the corresponding process, or run step 13.

Example 6: Detecting non-convex motifs (lamellipodia) with a script.

1. If you haven’t already done so, put the software on Matlab’s path.
2. Type `edit runDendSegment` on Matlab’s command line to open runDendSgment.
3. In the set directories section of the m-file, set all the paths as described in example 3.3.
4. Save the m-file and run it by typing `runDendSegment` on Matlab’s command line.

Example 7: Using ChimeraX. ChimeraX is a 3D rendering and visualization software package that is free for most users. You can use the u-shape3D package to create colored meshes illustrating features, such as motifs or curvature, for export to ChimeraX.

1. Create a dae file for export using one of the following methods.
   a. In step 13 of the GUI, click on the “Make Collada Dae” check box. A file will be saved at the .dae SavePath specified in the bottom of the window. The Surface Mode parameter controls what feature the mesh is colored by. For example, to color the mesh by motif detection, set the Surface Mode parameter to protrusions.
   b. From a script, use the plotMeshMD() function to generate a Collada Dae file. Both provided scripts show examples of how to do this. Please see the documentation for plotMeshMD for more information.
2. Download ChimeraX from https://www.rbvi.ucsf.edu/chimerax/download.html.
3. Start ChimeraX and then open the dae file by typing, on ChimeraX’s command line at the bottom of the screen, `open` and then the full path to the dae file.
4. To use the lighting conditions shown in the paper, click the apple button at the top of the screen. Surfaces can be rotated via dragging. To save images, click the camera or movie camera button.

Tips for analyzing your own data.

1. **Focus first on cell segmentation.** When analyzing new data, setting up a pipeline that distinguishes the cell from the image background is usually the trickiest part, so first focus on getting this to work. Try out several different sets of parameters and then use the checkCellSegmentation/MeshThres process to check the segmentation. If you’re having trouble selecting parameters, see Supplementary Table 5 of the associated paper, and try picking a set of parameters that matches your motif or microscope. You can also import images segmented by another software program.
2. **Finesse the deconvolution.** Deconvolution is microscope specific. To enhance the quality of you cell segmentation, try using a synthetic or experimental point spread function specific to your microscope. If you already have deconvolved images generated by another software program, you can skip the deconvolution process

here and go straight to the meshing process. You will need to enable the useUndeconvolved checkbox in the Mesh process.

3. **Train your own motif model.** Although the motif models are portable between some cell types and microscopes, you may get better results if you train your own motif model on your images. For robust results, we recommended that multiple models, each trained by a different person, are simultaneously used for detection.
4. **To analyze a lot of data, use the script.** If you have many dozens of cells to analyze the GUI can be cumbersome. Try using the script instead.