ALMA-TDA Manual

Paul Rosen, University of South Florida
Bei Wang, University of Utah
Betsy Mills, San Jose State University
Chris R. Johnson, University of Utah
Jeff Kern, National Radio Astronomy Observatory

Contact: Paul Rosen
E-Mail: prosen@usf.edu
Office Phone: 813-974-2282
4202 E. Fowler Avenue, ENB 118
Tampa, FL 33620

1 Brief Overview

ALMA-TDA is a persistent homology based engine for noise removal in ALMA data cubes. ALMA-TDA uses a data structure known as the contour tree to summarize and simplify the data. Figure 1 shows an example of the process. ALMA-TDA takes a scalar function from a data cube, computes a *topological skeleton* in the form of a contour tree, and simplifies that skeleton and the scalar field. The feature removal *only* impacts regions that connect pairs of critical points. In this way, ALMA-TDA provides the minimum perturbation of the field required to remove unwanted features. For a technical description of the process, please see the documents provided at: http://alma-tda.cspaul.com.

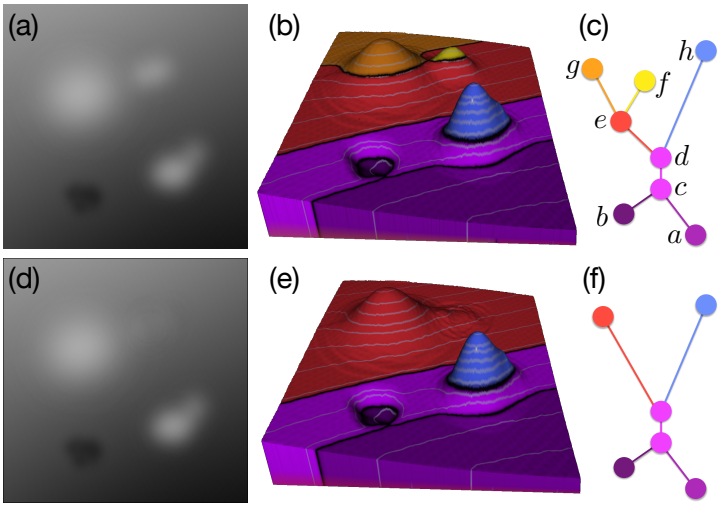

Figure 1: (a) An image of a 2D scalar function before simplification. (b) 3D height map of the contours corresponding to the scalar function shown in (a). (c) The contour tree structures that capture the features (i.e., relationships among local minima, local maxima, and saddle points). (d)-(f): The image, 3D height map, and the contour tree after simplifying the features.

2 Getting Started

The latest version of the software may be downloaded from: https://github.com/SCIInstitute/ALMA-TDA/releases.

2.1 Prerequisites

Most required libraries are prepackaged within ALMA-TDA. The only requirement is that the system have Java SE Runtime Environment 7 or higher installed (see http://www.oracle.com/technetwork/java/javase/overview/index.html for the latest version).

2.2 Running the software

From the command-line, type:

```
$ java -jar ALMA-TDA-YY.MM.DD.jar
```

where YY, MM, and DD are the release year, month, and date, respectively. As of writing this document this would be `ALMA-TDA-17.04.04.jar`.

This command will produce some licensing information and a list of options discussed in the next section.

3 Command-line Interface

Upon running the software, the following list of options appears:

```
Usage: java -jar ALMA-TDA.jar [options] input_file

  Required:
    input_file       --- Input file in fits format.

  Options:
    interactive      --- Places the application into interactive mode.
    dim=DIM_TYPE     --- Type of contour tree to create. Options: 2D or 3D. (default: 2D)
    x=RANGE          --- Range of pixels in x direction. Valid options included single
                         values and inclusive/exclusive ranges. (default: width of image)
    y=RANGE          --- Range of pixels in y direction. (default: height of image)
    z=RANGE          --- Range of pixels in z direction. (default: depth of image)
    simplify=AMOUNT  --- Maximum persistence to simplify. (default: 0, no simplification)
    output=FILE      --- The place to save the results. (default: not saved)

  Example:
    java -jar ALMA-TDA.jar x=[0,512) y=[0,512) z=0 output=output.fits input.fits
```

Required Parameters

- **`input_file`** — This option is the path to the input file. The software only takes FITS files. The software has been tested with 2D, 3D, and 4D (but only for the first volume) files. Specifying this parameter only will run the default configuration.

Optional Parameters

- **`dim=DIM_TYPE`** — This option determines if processing occurs on a slice-by-slice basis (`dim=2D`) or if the entire volume (`dim=3D`) is processed. We recommend 2D processing (see Section 5). As such, 2D is the default.
- **`x=RANGE`** — This option determines range of pixels to be processed in the x- or horizontal direction. Valid input include fixed values, as well as inclusive/exclusive ranges. Examples include `x=64`, `x=[0,200]`, `x=(64,128]`, `x=[128,512)`, etc. Be sure when specifying a range that NO spaces are included in the text. The default range is the entire width of the image.
- **`y=RANGE`** — This option determines range of pixels to be processed in the y- or vertical direction. See `x=RANGE` parameter for valid options. The default range is the entire height of the images.
- **`z=RANGE`** — This option determines range of pixels to be processed in the z-, depth, velocity, or spectral direction. See `x=RANGE` parameter for valid input. The default range include all slices.
- **`simplify=AMOUNT`** — This option determines the level of simplification to apply. Intuitively, a feature with ranges less than or equal to the simplification level will be removed from the output image. The default value is 0 (i.e., no simplification).
- **`output=FILE`** — The path to where results should be saved. This output will be a FITS file. The results will include the entire input cube with the selected region modified. The default option is that the results will not be saved.

- **`interactive`** — This option enables the interactive visualization mode. This mode is helpful for parameter selection, when you're unsure of what to use. This interface is discussed in Section 4.

4 Interactive Mode

We provide an interactive environment to explore and select parameters in the software. Starting the this mode only requires adding the `interactive` option to the command line.

```
$ java -jar ALMA-TDA-17.04.04.jar interactive path_to_file.fits
```

If no input file path is included, a file selection dialog box will appear.

The visualization initially opens to the interface seen in Figure 2 (top). The interface includes only minimal required capabilities. The main window, **A**, shows visualizations related to the loaded data cube. The controls component, **B**, provides options for processing the data. The controls are placed in groups, numbered for steps 1-5.

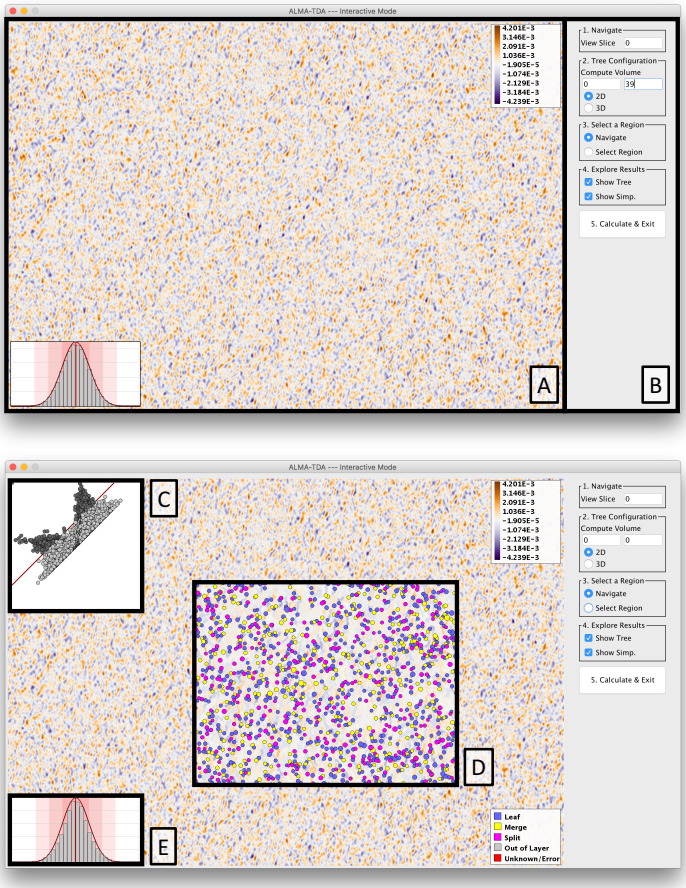

Figure 2: Top: Initial view of interactive mode. Bottom: Visualizations shown as the user selects a region of interest and the contour tree is calculated (on the back end).

4.1 Interaction Process

Though the use of our software requires some explanation, we strive to make it as simple to use as possible. Part of this effort is providing a simple and intuitive five step approach.

Step 1: Navigation. You should begin by identifying the general region of interest. This includes translating using the left button and zooming using scroll wheel within the Scalar Field View. The current slice can be changed by modifying the `View Slice` option.

Step 2: Tree Configuration. Next, the `Compute Volume` should be set to the range of slices of interest. Then, the dimension of the computation should be set. The options include `2D`, for 2D computation on 1 or more slices; or `3D` for 3D computation on a volume. We recommend 2D processing (see Section 5).

Step 3: Region Selection. Next, clicking `Select Region` option, enables setting the specific region of interest. Click-drag-release the mouse within the Scalar Field View to select a region. As soon as the mouse is released, computation begins. Depending upon the size of the region selected this could take between a few seconds and a few minutes (see Section 5).

Step 4: Exploration. Once the computation is completed, the you should explore the domain. This includes navigation (translation, zooming, and changing slices) and adjustment of the simplification level. The simplification level can be adjusted by dragging the red bar in the Persistence Diagram (see Section 4.2).

As simplification level is adjusted and the mouse is released, you will see the scalar field updated as well. You can compare the result to the original using the `Show Simp.` button. You can also enable/disable the drawing of critical points with the `Show Tree` button.

Step 5: Compute and Exit. Steps 1-4 may be repeated as many times as necessary, until you are satisfied. Once done, clicking `Compute and Exit`. This will trigger the processing of the data cube and saving of output. Finally, the command required to reproduce the exact results will be printed to the command-line.

4.2 Visual Elements

The visualization is composed of five main visual elements.

Scalar Field View (Figure 2 A). This is fairly standard view of the current slices. It is displayed using a divergent orange/purple colormap (orange for positive, purple for negative). By default, the first slice is selected and viewed by centering on the middle of the domain. The view can be changed by using the left mouse button for translation and scroll wheel for zooming. Different slices can be selected by changing the `View Slice` option in the controls.

Persistence Diagram (Figure 2 C). Once the contour tree is calculated, the data are displayed using a persistence diagram. A persistence diagram is a scatterplot display that places points representing features by their *birth time* (minimum scalar value) horizontally and their *death time* (maximum scalar value) vertically. Most importantly, the distance of that feature point from the diagonal is known as the *persistence* (i.e., importance) of that feature. *Sliding the red bar up and down increases or decreases the level of simplification applied to the scalar field.* Once released, a simplified scalar field is calculated.

Critical Points (Figure 2 D). Within the selected region, the *critical points* are displayed. These points represent the features that will potentially be removed. Ones with higher persistence are large than those with smaller persistence. In other words, when simplifying, the smaller points are removed before the large ones. Each point is colored by its type, local extrema – blue, merge saddle points – yellow, and split saddle points – magenta. For 3D analysis, nodes off layer are colored gray. An error state, red, is provided for nodes that don't cancel. If this occurs, please report it to the developers. This view can be enabled or disabled on demand using the `Show Tree` controls on the right.

Simplified Scalar Field (Figure 2 D). As the level of simplification is adjusted, the scalar field will be simplified and overlaid with the original visualization. This view can be enabled or disabled on demand using the `Show Simp.` controls on the right.

Histogram (Figure 2 E). A histogram is also provided to indicate the distribution of intensity values within the current view. In addition to showing histogram bins, this view shows the mean as a solid red line and ± 3 standard deviations as consecutively lighter red bars. This histogram is modified as the view changes or when the simplification level of the scalar field is adjusted.

5 Recommendations and Best Practice

2D vs. 3D Computation. Our observation has been that due to the uncorrelated noise patterns slice-to-slice, 3D contour trees do not produce particularly interesting results. Instead, we recommend using 2D computations on sets of slices.

Computation Time. Some computations in this process are long for real-time interaction. We recommend selected small subsets of data for interactive mode and expanding those regions during command-line computation.

Quantifying Change in Flux. The design of our scalar field simplification was chosen such that as features are removed, there flux will in general be closer to neutral (i.e., zero). This may have an impact on the total flux change in the domain. This change can be observed using the histogram view. Shifts in the mean indicate changes in overall flux.

6 License Information

7 Acknowledgement

This work was supported by ALMA Development Studies Program under the project title “Feature Extraction and Visualization of ALMA Data Cubes through Topological Data Analysis”.