Developer manual - EVE software

Contents

EVE is a software platform developed for the analysis of single-molecule imaging data captured by event-based sensors. The software is methodically divided into three integral modules, each serving a distinct purpose in the data analysis pipeline for event-based single-molecule data.
This document is for developers who want to add functionality to E~~ve~~VE. E~~ve~~VE is a highly open framework and can easily be expanded upon. Expandability of E~~ve~~VE is possible in the following routines (with more detailed information following):

- **Candidate Finding**
  Routines involved in finding which events belong to a single molecule localization/PSF
  Input: All events (possibly filtered by polarity), settings, function arguments
  Output: Found candidates, metadata
- **Candidate Fitting**
  Routines involved in fitting ~~all~~ the events of each candidate cluster ~~that belong to a single~~ to determine the x-, y-, (z-) and t-coordinates of each single molecule~~localization/PSF~~. ~~Heavily uses~~Fitting relies on using 'Event distributions' (below)
  Input: All found candidates, settings, function arguments
  Output: Localizations (x,y,(z,)t-coordinates), metadata
- **Event distributions**
  Classes to create varying distributions from the events
  Input: Events, settings, function arguments
  Output: Histogram classes with certain arguments
- **Post-processing**
  Routines involved in post-processing ~~of~~ the localization data, either for further filtering~~,~~ or ~~for~~ data quantification or for calculating quantitative metrics~~ calculation~~
  ~~——~~**Input**: Localizations, candidates, settings, function arguments
  Output: (Possibly changed) localizations, metadata
- **Visualization**~~**Visualisation**~~
  Routines ~~involving~~that ~~visualization~~visualize ~~visualisation of the~~ a current localization list
  Input: Localizations, settings, function arguments
  Output: 2d array containing image data, scale of the image
- **Candidate preview**
  Routines that visualize~~involving visualisation of~~ individual candidates for user inspection
  Input: Candidates, localizations, all events, settings, function arguments
  Output: None (updated figure)

Commented [EU(1]: maybe rather x-, y-, (z-) and t-coordinates?

Commented [EU(2]: I would say that events have/follow a distribution. So rather "classes to select the distribution of events to be fitted, e.g. all events, first event per pixel etc."?

Commented [KM3R2]: I'm not sure I follow. This manual is not to select the distribution, to create new distributions. You can create any distribution here, also a 2d array of 1s if you feel so inclined.

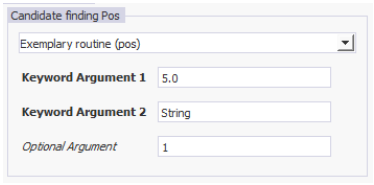
Commented [EU(4]: always histograms?

EEve expandability of EVE

All routines, with the exception of the Event distributions, follow the same method of expandability. One or multiple routines should be written in a .py file, and placed within a sub-folder in the main EveVE GUI folder, or alternatively in the AppData/Local/UniBonn/Eve folder. EveVE will automatically find and add all suitable routines into the GUI. The .py files should start with the following defined structure:

```
def __function_metadata__():
 return {
  "FunctionTitle": {
   "required_kwargs": [
    {"name": " kwarg1", "description": "Some Description", "default": 5.0, type": float, "display_text":
     "Keyword Argument 1"},
    {"name": " kwarg2", "description": "Some other Description", "default":"string", type": str,
     "display_text": "Keyword Argument 2"}
   ],
   "optional_kwargs": [
    {"name": "oarg", "description": "An optional argument", "default":1, "type":int,
     "display_text":"Optional Argument"},
   ],
   "help_string": "This is an exemplary routine.",
    "display_name": "Exemplary Routine"
  }
 }
```

This function would be displayed as such by the EveVE GUI (exemplary for candidate finding on positive events):



The function_metadata function describes important metadata for the function(s) that should be displayed and callable:

- One or multiple functions can be defined with this structure
- The same .py file requires a function called the same as “FunctionTitle”
  The following parameters should be created:
  display_name (optional) provides the function name which is visible for the user
  help_string (optional) provides a description of the function for the user
  required_kwargs (required) defines required keyword arguments that your function expects.
  optional_kwargs (required) defines optional keyword arguments that your function could use.
- For each keyword argument, the following should be provided:
  name (required): the internal name of the keyword argument ~~(required)~~
  display_text (optional): the name visible for the user
  description (optional): a description of the argument that users can see in the EveVE GUI ~~(optional)~~
  default (optional): the default value of this argument ~~(optional)~~

type (optional): the expected type of the input ~~(optional)~~, options are [float, int, str, “fileLoc”]

The “fileLoc” value indicates a file which can be found by the user

Detailed information on input/output data of EVE

All data has these two input variables:

settings: named dictionary with (advanced) settings.
kwargs: dictionary with named entries of the function (as defined in *__function_metadata__()*)

Candidate Finding

Function definition
*def function(npy_array, settings,**kwargs):*
return candidates, performance_metadata

Input
npy_array: numpy.ndarray with one entry for each event. Each entry has dtype([('x', '<u2'), ('y', '<u2'), ('p', '<i2'), ('t', '<i8')]) structure, with x/y in pixels, p either 0 or 1 (negative or positive), and t in microseconds

Output
candidates: dictionary where each entry is a candidate. Each entry should have three named sub-entries:

events: pandas DataFrame with N-by-4 array, array names x,y,t,p (same units as input, N being the number of events in this cluster).

N_events: Number of events

cluster_size: [size_x, size_y, size_t] of the cluster (in [pixel, pixel, microsecond] units)

performance_metadata: string with details on the performance. Will be stored in the metadata.txt output.

Candidate Fitting

For the candidate fitting, the *__function_metadata__()* needs to be expanded to provide information about the dist_kwarg and time_kwarg. These structures contain information about which XY distribution and Time distribution can be selected by the user. ~~In the *__function_metadata__()* structure, additionally to what is shown above, there should also be information about the dist_kwarg and time_kwarg structure, which allows the user to choose an XY+Time distribution.~~ If these are not defined, an XYT-combined fitting is ran (which should result in XY ~~á~~and time fitting results). In an XY+Time distribution, the candidate fitting routine should only provide the XY fitting result, since the Time distribution is handled independently. Please look at the following examples for implementation details:

Example for XY+Time: GaussianFitting

Example for XYT: Radial_Symmetry – RadialSym3D.

Function definition
*def function(candidate_dic, settings,**kwargs):*
return localizations, fit_info

Input
candidate_dic: see output from candidate finding

Output
localizations: Pandas Dataframe of localizations corresponding to input clusters.
fit_info: string with info of metadata. Will be stored in the metadata.txt output. Should at least have columns with names 'candidate_id', 'x', 'y', 'p', ['t'], in units integer, pixel, pixel, 0/1, microseconds, respectively. t is not required if an independent Time distribution fitting is used (see above). Can also have more columns as wanted, with nomenclature normally following 'del_x' for uncertainty in x. Commonly also 'fit_info' column can be used to report on incomplete/wrong fits. Each candidate should have one entry.
fit_info: string with info of metadata. Will be stored in the metadata.txt output.

Post-processing

Function definition
*def function(localizations, findingResult, settings,**kwargs):*
return localizations, metadata

Input
localizations: See Candidate Fitting 'localizations' output.
findingResult: See Candidate Finding 'candidates' output.

Output
localizations: See Candidate Fitting 'localizations' output. Practically, should be a filtered/amended list of the original localizations.
metadata: string with info of metadata. Will be shown in the Run info GUI tab.

Visualization~~Visualisation~~

Function definition
*def function(resultArray, settings,**kwargs):*
return image, scale

Input
resultArray: See Candidate Fitting 'localizations' output. Uses the currently found results as in Eve (i.e. could be adapted via Post-processing).

Output
image: numpy.ndarray of pixel-values of the resulted image. Will be displayed in the 'Visualization~~Visualisation~~' tab
scale: float value of pixel-to-micrometer size (e.g. value of 0.01 means 0.01 micrometer per pixel, or 10 nm per pixel). Used to set the scale_bar in the 'Visualization~~Visualisation~~' tab.

Candidate preview

Function definition
*def function(findingResult, fittingResult, previewEvents, figure, settings,**kwargs):*
return None

Input
findingResult: See Candidate Finding ‘candidates’ output. The information of a single candidate is provided.
fittingResult: See Candidate Fitting ‘localizations’ output. The information of a single localization is provided.
previewEvents: Unused
figure: Matplotlib Figure object. Should be addressed by e.g. performing *ax = figure.add_subplot(111); ax.bar(…).* figure.show() does not have to be called.

Output
None. Expected that *figure* is updated properly.

Event distributions

These ~~e~~Event distributions follow a different expandability method, and cannot be adapted from the AppData folder, but only from changing the EventDistributions/eventDistributions.py file in the E~~ve~~VE installation folder.

Each ~~e~~Event distribution is defined by a class (e.g. *class Hist1d_t()*). These classes should have a *__call__(self, events, **kwargs)* function, which should return the wanted distribution and bin edge positions.

Please use the ~~Look at~~ the existing classes in *eventDistributions.py* for detailed info.

Temporal Fitting

Same structure as Event distributions, only for fitting time distributions.