

Python Documentation

version

April 14, 2020

Contents

Welcome to moseq2-extract's documentation!	1
moseq2-extract	1
moseq2-extract package	1
Subpackages	1
moseq2-extract.moseq2_extract package	1
Subpackages	1
moseq2-extract.moseq2_extract.extract package	1
Submodules	1
Extract - Extract Module	1
Extract - Proc Module	2
Extract - ROI Module	6
Extract - Track Module	6
moseq2-extract.moseq2_extract.helpers package	8
Submodules	8
Helpers - Data Module	8
Helpers - Extract Module	10
Helpers - Wrappers Module	11
moseq2-extract.moseq2_extract.io package	12
Submodules	12
IO - Image Module	12
IO - Video Module	13
moseq2-extract.moseq2_extract.tests.integration_tests package	15
Submodules	15
Integration Tests - Test CLI Module	15
Integration Tests - Test GUI Module	15
moseq2-extract.moseq2_extract.tests.unit_tests package	16
Submodules	16
Unit Tests - Test Extract-Proc Module	16
Unit Tests - Test Extract-ROI Module	16
Unit Tests - Test Extract-Track Module	17
Unit Tests - Test Helper-Data Module	17
Unit Tests - Test Helper-Extract Module	17
Unit Tests - Test IO-Image Module	17
Unit Tests - Test IO-Video Module	18
Unit Tests - Test Utilities Module	18
Submodules	18
CLI Module	18
GUI Module	18
Moseq2-Extract Utilities Module	21
Module contents	24

Indices and tables	24
Index	25
Python Module Index	29

Welcome to moseq2-extract's documentation!

moseq2-extract

moseq2-extract package

Subpackages

moseq2-extract.moseq2_extract package

Subpackages

moseq2-extract.moseq2_extract.extract package

Submodules

Extract - Extract Module

```
moseq2_extract.extract.extract.extract_chunk (chunk, use_em_tracker=False, prefilter_space=3,
prefilter_time=None, iters_tail=1, iters_min=0, strel_tail=array([[0, 0, 0, 0, 1, 0, 0, 0, 0], [0, 1, 1, 1, 1, 1, 1, 1, 0], [0, 1,
1, 1, 1, 1, 1, 1, 0], [1, 1, 1, 1, 1, 1, 1, 1, 1], [1, 1, 1, 1, 1, 1, 1, 1, 1], [1, 1, 1, 1, 1, 1, 1, 1, 1], [0, 1, 1, 1, 1, 1, 1, 1, 0], [0,
1, 1, 1, 1, 1, 1, 1, 0], [0, 0, 0, 0, 1, 0, 0, 0, 0]], dtype=uint8), strel_min=array([[1, 1, 1, 1, 1], [1, 1, 1, 1, 1], [1, 1, 1, 1, 1],
[1, 1, 1, 1, 1], [1, 1, 1, 1, 1]], dtype=uint8), min_height=10, max_height=100, mask_threshold=- 20, use_cc=False,
bground=None, roi=None, rho_mean=0, rho_cov=0, tracking_ll_threshold=- 100, tracking_segment=True,
tracking_init_mean=None, tracking_init_cov=None, tracking_init_strel=array([[0, 0, 0, 0, 1, 0, 0, 0, 0], [0, 1, 1, 1, 1, 1, 1, 1, 0], [0, 1, 1, 1, 1, 1, 1, 1, 0], [1, 1, 1, 1, 1, 1, 1, 1, 1], [1, 1, 1, 1, 1, 1, 1, 1, 1], [1, 1, 1, 1, 1, 1, 1, 1, 1], [0, 1, 1, 1, 1,
1, 1, 1, 0], [0, 1, 1, 1, 1, 1, 1, 1, 0], [0, 0, 0, 0, 1, 0, 0, 0, 0]], dtype=uint8), flip_classifier=None, flip_smoothing=51,
frame_dtype='uint8', save_path='/Users/aymanzeine/Desktop/moseq/moseq2-extract/docs/proc',
progress_bar=True, crop_size=80, 80, true_depth=673.1, centroid_hampel_span=5, centroid_hampel_sig=3,
angle_hampel_span=5, angle_hampel_sig=3, model_smoothing_clips=- 300, - 150, tracking_model_init='raw',
verbose=0)
```

This function extracts individual chunks from depth videos. It is called from the `moseq2_extract.helpers.extract` module.

Parameters:

- **chunk** (3d np.ndarray) (*chunk to extract*)
- **use_em_tracker** (bool) (*boolean for whether to extract 2D plane using RANSAC.*)
- **prefilter_space** (tuple) (*spatial kernel size*)
- **prefilter_time** (tuple) (*temporal kernel size*)
- **iters_tail** (int) (*number of filtering iterations on mouse tail*)
- **iters_min** (int) (*minimum tail filtering filter kernel size*)
- **strel_tail** (cv2::StructuringElement - Ellipse) (*filtering kernel size to filter out mouse tail.*)
- **strel_min** (cv2::StructuringElement - Rectangle) (*filtering kernel size to filter mouse body in cable recording cases.*)
- **min_height** (int) (*minimum (mm) distance of mouse to floor.*)
- **max_height** (int) (*maximum (mm) distance of mouse to floor.*)
- **mask_threshold** (int)
- **use_cc** (bool) (*boolean to use connected components in cv2 structuring elements*)
- **bground** (np.ndarray) (*numpy array represented previously computed background*)
- **roi** (np.ndarray) (*numpy array represented previously computed roi*)
- **rho_mean** (int)
- **rho_cov** (int)
- **tracking_ll_threshold** (int)
- **tracking_segment** (bool) (*boolean for whether to use EM mouse tracking for cable recording cases.*)
- **tracking_init_mean** (float)
- **tracking_init_cov** (float)
- **tracking_init_strel** (cv2::StructuringElement - Ellipse)
- **flip_classifier** (str) (*path to pre-selected flip classifier.*)
- **flip_smoothing** (int) (*amount of smoothing to use for flip classifier.*)
- **frame_dtype** (str)
- **save_path** ((str):)
- **progress_bar** (bool)
- **crop_size** (tuple) (*size of the cropped mouse image.*)
- **true_depth** (float) (*previously computed detected true depth value.*)
- **centroid_hampel_span** (int)
- **centroid_hampel_sig** (int)
- **angle_hampel_span** (int)
- **angle_hampel_sig** (int)
- **model_smoothing_clips** (tuple)
- **tracking_model_init** (str)
- **verbose** (bool)

Returns: results

Return type: (np.ndarray) - extracted RGB video chunk to be written to file.

`moseq2_extract.extract.proc.apply_roi` (frames, roi)
Apply ROI to data, consider adding constraints (e.g. `mod32==0`).

Parameters:

- **frames (3d np.ndarray)** (*input frames to apply ROI.*)
- **roi (2d np.ndarray)** (*selected ROI to extract from input images.*)

Returns: **cropped_frames (3d np.ndarray)**

Return type: Frames cropped around ROI Bounding Box.

`moseq2_extract.extract.proc.clean_frames` (frames, prefilter_space=3, prefilter_time=None, strel_tail=array([[0, 0, 0, 1, 0, 0, 0], [0, 1, 1, 1, 1, 1, 0], [1, 1, 1, 1, 1, 1, 1], [1, 1, 1, 1, 1, 1, 1], [1, 1, 1, 1, 1, 1, 1], [0, 1, 1, 1, 1, 1, 0], [0, 0, 0, 1, 0, 0, 0]], dtype=uint8), iters_tail=None, frame_dtype='uint8', strel_min=array([[1, 1, 1, 1, 1], [1, 1, 1, 1, 1], [1, 1, 1, 1, 1], [1, 1, 1, 1, 1]], dtype=uint8), iters_min=None, progress_bar=True, gui=False, verbose=0)

Simple filtering, median filter and morphological opening.

Parameters:

- **frames (3d np.ndarray)** (*Frames (nframes x r x c) to filter.*)
- **prefilter_space (tuple)** (*kernel size for spatial filtering*)
- **prefilter_time (tuple)** (*kernel size for temporal filtering*)
- **strel_tail (cv2.StructuringElement)** (*Element for tail filtering.*)
- **iters_tail (int)** (*number of iterations to run opening*)
- **frame_dtype (str)** (*frame encodings*)
- **strel_min (int)** (*minimum kernel size*)
- **iters_min (int)** (*minimum number of filtering iterations*)
- **progress_bar (bool)** (*display progress bar*)
- **gui (bool)** (*indicate GUI is executing function*)
- **verbose (bool)** (*display progress*)

Returns: **filtered_frames (3d np array)**

Return type: frame x r x c

`moseq2_extract.extract.proc.compute_scalars` (frames, track_features, min_height=10, max_height=100, true_depth=673.1)
Computes scalars.

Parameters:

- **frames (3d np.ndarray)** (*frames x r x c, uncropped mouse*)
- **track_features (dict)** (*dictionary with tracking variables (centroid and orientation)*)
- **min_height (float)** (*minimum height of the mouse*)
- **max_height (float)** (*maximum height of the mouse*)
- **true_depth (float)** (*detected true depth*)

Returns: **features (dict)**

Return type: dictionary of scalars

`moseq2_extract.extract.proc.crop_and_rotate_frames` (frames, features, crop_size=80, 80, progress_bar=True, gui=False, verbose=0)
Crops mouse from image and orients it s.t it is always facing east.

Parameters:

- **frames (3d np.ndarray)** (*frames to crop and rotate*)
- **features (dict)** (*dict of extracted features, found in result_00.h5 files.*)
- **crop_size (tuple)** (*size of cropped image.*)
- **progress_bar (bool)** (*Display progress bar.*)
- **gui (bool)** (*indicate GUI is executing function*)
- **verbose (bool)** (*display progress*)

Returns: **cropped_frames (3d np.ndarray)**

Return type: Crop and rotated frames.

`moseq2_extract.extract.proc.feature_hampel_filter` (features, centroid_hampel_span=None, centroid_hampel_sig=3, angle_hampel_span=None, angle_hampel_sig=3)

Filters computed extraction features using Hampel Filtering.

Parameters:

- **features (dict)** (*dictionary of video features*)
- **centroid_hampel_span (int)** (*Centroid Hampel Span Filtering Kernel Size*)
- **centroid_hampel_sig (int)** (*Centroid Hampel Signal Filtering Kernel Size*)
- **angle_hampel_span (int)** (*Angle Hampel Span Filtering Kernel Size*)
- **angle_hampel_sig (int)** (*Angle Hampel Span Filtering Kernel Size*)

Returns: **features (dict)**

Return type: filtered version of input dict.

`moseq2_extract.extract.proc.get_bbox` (roi)

Given a binary mask, return an array with the x and y boundaries

Parameters: **roi (2d np.ndarray)** (*ROI boolean mask to calculate bounding box.*)

Returns: **bbox (2d np.ndarray)**

Return type: Bounding Box around ROI

`moseq2_extract.extract.proc.get_bground_im` (frames)

Returns background

Parameters: **frames (3d numpy array)** (*frames x r x c, uncropped mouse*)

Returns: **bground (2d numpy array)**

Return type: r x c, background image

`moseq2_extract.extract.proc.get_bground_im_file` (frames_file, frame_stride=500, med_scale=5, **kwargs)

Returns background from file

Parameters:

- **frames_file (str)** (*path to data with frames*)
- **frame_stride (int)** (*stride size between frames for median bground calculation*)
- **med_scale (int)** (*kernel size for median blur for background images.*)
- **kwargs**

Returns: **bground (2d numpy array)**

Return type: r x c, background image

`moseq2_extract.extract.proc.get_flips` (frames, flip_file=None, smoothing=None)

Predicts frames where mouse orientation is flipped to later correct.

Parameters:

- **frames (3d numpy array)** (*frames x r x c, cropped mouse*)
- **flip_file (str)** (*path to joblib dump of scipy random forest classifier*)
- **smoothing (int)** (*kernel size for median filter smoothing of random forest probabilities*)

Returns: **flips (bool array)**

Return type: true for flips

`moseq2_extract.extract.proc.get_frame_features` (frames, frame_threshold=10, mask=array([], dtype=float64), mask_threshold=-30, use_cc=False, progress_bar=True, gui=False, verbose=0)

Use image moments to compute features of the largest object in the frame

Parameters:

- **frames (3d np.ndarray)** (*input frames*)
- **frame_threshold (int)** (*threshold in mm separating floor from mouse*)
- **mask (3d np.ndarray)** (*input frame mask for parts not to filter.*)
- **mask_threshold (int)** (*threshold to include regions into mask.*)
- **use_cc (bool)** (*Use connected components.*)
- **progress_bar (bool)** (*Display progress bar.*)
- **gui (bool)** (*indicate GUI is executing function*)
- **verbose (bool)** (*display progress*)

Returns: **features (dict of lists)** (*dictionary with simple image features*) **mask (3d np.ndarray)** (*input frame mask.*)

`moseq2_extract.extract.proc.get_largest_cc` (frames, progress_bar=False)

Returns largest connected component blob in image

Parameters:

- **frames (3d numpy array)** (*frames x r x c, uncropped mouse*)
- **progress_bar (bool)** (*display progress bar*)

Returns: **flips (3d bool array)**

Return type: frames x r x c, true where blob was found

`moseq2_extract.extract.proc.get_roi` (depth_image, strel_dilate=array([[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1], [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1], [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1], [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1], [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1], [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1], [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1], [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1], [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1], [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1], [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1], [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1], [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1], [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1], [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]]], dtype=uint8), dilate_iters=1, strel_erode=None, noise_tolerance=30, weights=1, 0.1, 1, overlap_roi=None, gradient_filter=False, gradient_kernel=7, gradient_threshold=3000, fill_holes=True, gui=False, verbose=0, **kwargs)

Get an ROI using RANSAC plane fitting and simple blob features

Parameters:

- **depth_image (2d np.ndarray)** (*Singular depth image frame.*)
- **strel_dilate (cv2.StructuringElement - Rectangle)** (*dilation shape to use.*)
- **dilate_iters (int)** (*number of dilation iterations.*)
- **strel_erode (int)** (*image erosion kernel size.*)
- **noise_tolerance (int)** (*threshold to use for noise filtering.*)
- **weights (tuple)** (*weights describing threshold to accept ROI.*)
- **overlap_roi (np.ndarray)** (*list of ROI boolean arrays to possibly combine.*)
- **gradient_filter (bool)** (*Boolean for whether to use a gradient filter.*)
- **gradient_kernel (tuple)** (*Kernel size of length 2, e.g. (1, 1.5)*)
- **gradient_threshold (int)** (*Threshold for noise gradient filtering*)
- **fill_holes (bool)** (*Boolean to fill any missing regions within the ROI.*)
- **gui (bool)** (*Boolean for whether function is running on GUI.*)
- **verbose (bool)** (*Boolean for whether to display progress*)
- **kwargs**

Returns: **rois (list)** (*list of 2d roi images.*) **roi_plane (2d np.ndarray)** (*computed ROI Plane using RANSAC.*) **bboxes (list)** (*list of computed bounding boxes for each respective ROI.*) **label_im (list)** (*list of scikit-image image properties*) **ranks (list)** (*list of ROI ranks.*) **shape_index (list)** (*list of rank means.*)

`moseq2_extract.extract.proc.im_moment_features` (IM)

Use the method of moments and centralized moments to get image properties.

Parameters: **IM (2d numpy array)** (*depth image*)

Returns: **features (dict)** – centroid, and ellipse axis length

Return type: returns a dictionary with orientation,

`moseq2_extract.extract.proc.model_smoother` (features, ll=None, clips=- 300, - 125)

Spatial feature filtering.

Parameters:

- **features (dict)** (*dictionary of extraction scalar features*)
- **ll (np.array)** (*list of loglikelihoods of pixels in frame*)
- **clips (tuple)** (*tuple to ensure video is indexed properly*)

Returns:

Return type: features (dict) - smoothed version of input features

Extract - ROI Module

`moseq2_extract.extract.roi.plane_fit3` (points)

Fit a plane to 3 points (min number of points for fitting a plane)

Parameters: **points (2d numpy array)** (*each row is a group of points, columns correspond to x,y,z.*)

Returns: **plane (1d numpy array)**

Return type: linear plane fit→ $a*x+b*y+c*z+d$

`moseq2_extract.extract.roi.plane_ransac` (depth_image, depth_range=650, 750, iters=1000, noise_tolerance=30, in_ratio=0.1, progress_bar=True, mask=None, gui=False, verbose=0)

Naive RANSAC implementation for plane fitting

Parameters:

- **depth_image (2d numpy array)** (*hwx, background image to fit plane to*)
- **depth_range (tuple)** (*min/max depth (mm) to consider pixels for plane*)
- **iters (int)** (*number of RANSAC iterations*)
- **noise_tolerance (float)** (*dist. from plane to consider a point an inlier*)
- **in_ratio (float)** (*frac. of points required to consider a plane fit good*)
- **progress_bar (bool)** (*display progress bar*)
- **mask (bool 2d np.array)** (*boolean mask to find region to use*)
- **gui (bool)** (*whether GUI is used.*)
- **verbose (bool)** (*print all information.*)

Returns: **best_plane (1d numpy array)** (*plane fit to data*) **dist (1d numpy array)** (*distance of the calculated coordinates and "best plane"*)

Extract - Track Module

`moseq2_extract.extract.track.em_get_ll` (frames, mean, cov, progress_bar=True)

Returns likelihoods for each frame given tracker parameters

Parameters:

- **frames (3d numpy array)** (*depth frames*)
- **mean (2d numpy array)** (*frames x d, mean estimates*)
- **cov (3d numpy array)** (*frames x d x d, covariance estimates*)
- **progress_bar (bool)** (*use a progress bar*)

Returns: **ll (3d numpy array)**

Return type: frames x rows x columns, log likelihood of all pixels in each frame

`moseq2_extract.extract.track.em_init` (depth_frame, depth_floor, depth_ceiling, init_strel=array([[0, 0, 0, 0, 1, 0, 0, 0, 0], [0, 1, 1, 1, 1, 1, 1, 1, 0], [0, 1, 1, 1, 1, 1, 1, 1, 0], [1, 1, 1, 1, 1, 1, 1, 1, 1], [1, 1, 1, 1, 1, 1, 1, 1, 1], [1, 1, 1, 1, 1, 1, 1, 1, 1], [0, 1, 1, 1, 1, 1, 1, 1, 0], [0, 1, 1, 1, 1, 1, 1, 1, 0], [0, 0, 0, 0, 1, 0, 0, 0, 0]], dtype=uint8), strel_iters=1)
Initialize EM Mask.

Parameters:

- **depth_frame (2d numpy array)** (*depth frame to initialize mask with.*)
- **depth_floor (float)** (*distance from camera to bucket floor.*)
- **depth_ceiling (float)** (*max depth value.*)
- **init_strel (cv2.structuringElement)** (*structuring Element to compute mask.*)
- **strel_iters (int)** (*number of EM iterations.*)

Returns: **mouse_mask (2d numpy array)**

Return type: mask of depth frame.

`moseq2_extract.extract.track.em_iter` (data, mean, cov, lamd=0.1, epsilon=0.1, max_iter=25)
Single iteration of EM tracker

Parameters:

- **data (3d numpy array)** (*nx3, x, y, z coordinates to use*)
- **mean (1d numpy array)** (*dx1, current mean estimate*)
- **cov (2d numpy array)** (*dxd, current covariance estimate*)
- **lamdb (float)** (*constant to add to diagonal of covariance matrix*)
- **epsilon (float)** (*tolerance on change in likelihood to terminate iteration*)
- **max_iter (int)** (*maximum number of EM iterations*)

Returns: **mean (1d numpy array)** (*updated mean*) **cov (2d numpy array)** (*updated covariance*)

`moseq2_extract.extract.track.em_tracking` (frames, raw_frames, segment=True, ll_threshold=- 30, rho_mean=0, rho_cov=0, depth_floor=10, depth_ceiling=100, progress_bar=True, init_mean=None, init_cov=None, init_frames=10, init_method='raw', init_strel=array([[0, 0, 0, 0, 1, 0, 0, 0, 0], [0, 1, 1, 1, 1, 1, 1, 1, 0], [0, 1, 1, 1, 1, 1, 1, 1, 1], [1, 1, 1, 1, 1, 1, 1, 1, 1], [1, 1, 1, 1, 1, 1, 1, 1, 1], [1, 1, 1, 1, 1, 1, 1, 1, 1], [0, 1, 1, 1, 1, 1, 1, 1, 0], [0, 1, 1, 1, 1, 1, 1, 1, 1], [0, 0, 0, 0, 1, 0, 0, 0, 0]], dtype=uint8))

Naive tracker, use EM update rules to follow a 3D Gaussian
around the room.

Parameters:

- **frames (3d numpy array)** (*filtered frames - nframes x r x c.*)
- **raw_frames (3d numpy array)** (*chunk to track mouse in.*)
- **segment (bool)** (*use only the largest blob for em updates*)
- **ll_threshold (float)** (*threshold on log likelihood for segmentation*)
- **rho_mean (float)** (*smoothing parameter for the mean*)
- **rho_cov (float)** (*smoothing parameter for the covariance*)
- **depth_floor (float)** (*height in mm for separating mouse from floor*)
- **depth_ceiling (float)** (*max height in mm for mouse from floor.*)
- **progress_bar (bool)** (*display progress bar.*)
- **init_mean (np.ndarray)** (*array of initial frame pixel means.*)
- **init_cov (np.ndarray)** (*array of initial frame pixel covariances.*)
- **init_frames (int)** (*number of frames to include in the init calculation*)
- **init_method (str)** (*mode in which to process inputs*)
- **init_strel (cv2.structuringElement)** (*structuring Element to compute mask.*)

Returns: **model_parameters (dict)**

Return type: mean and covariance estimates for each frame

moseq2-extract.moseq2_extract.helpers package

Submodules

Helpers - Data Module

`moseq2_extract.helpers.data.build_manifest` (loaded, format, snake_case=True)
`aggregate_results()` Helper Function. Builds a manifest file used to contain extraction result metadata from h5 and yaml files.

Parameters:

- **loaded (list of dicts)** (*list of dicts containing loaded h5 data.*)
- **format (str)** (*filename format indicating the new name for the metadata files in the aggregate_results dir.*)
- **snake_case (bool)** (*whether to save the files using snake_case*)

Returns: **manifest (dict)**

Return type: dictionary of extraction metadata.

`moseq2_extract.helpers.data.clean_dict` (dct)
Standardizes types of dict value.

Parameters: **dct (dict)** (*dict object with mixed type value objects.*)

Returns: **dct (dict)**

Return type: dict object with list value objects.

`moseq2_extract.helpers.data.copy_manifest_results` (manifest, output_dir)
Copies all consolidated manifest results to their respective output files.

Parameters:

- **manifest (dict)** (*manifest dictionary containing all extraction h5 metadata to save*)
- **output_dir (str)** (*path to directory where extraction results will be aggregated.*)

Returns:

Return type: None

Welcome to moseq2-extract's documentation!

`moseq2_extract.helpers.data.create_extract_h5` (*f*, *acquisition_metadata*, *config_data*, *status_dict*, *scalars*, *scalars_attrs*, *nframes*, *true_depth*, *roi*, *bground_im*, *first_frame*, *timestamps*, *extract=None*)
Creates h5 file that holds all extracted frames and other metadata (such as scalars).

Parameters:

- **f (h5py.File object)** (*opened h5 file object to write to.*)
- **acquisition_metadata (dict)** (*Dictionary containing extracted session acquisition metadata.*)
- **config_data (dict)** (*dictionary object containing all required extraction parameters. (auto generated)*)
- **status_dict (dict)** (*dictionary that helps indicate if the session has been extracted fully.*)
- **scalars (list)** (*list of computed scalar metadata.*)
- **scalars_attrs (dict)** (*dict of respective computed scalar attributes and descriptions to save.*)
- **nframes (int)** (*number of frames being recorded*)
- **true_depth (float)** (*computed detected true depth*)
- **roi (2d np.ndarray)** (*Computed 2D ROI Image.*)
- **bground_im (2d np.ndarray)** (*Computed 2D Background Image.*)
- **first_frame (2d np.ndarray)** (*Computed 2D First Frame Image.*)
- **timestamps (np.array)** (*Array of session timestamps.*)
- **extract (moseq2_extract.cli.extract function)** (*Used to preseve CLI state parameters in extraction h5.*)

Returns:

Return type: None

`moseq2_extract.helpers.data.get_selected_sessions` (*to_extract*, *extract_all*)
Given user input, the function will return either selected sessions to extract, or all the sessions.

Parameters:

- **to_extract (list)** (*list of paths to sessions to extract*)
- **extract_all (bool)** (*boolean to include all sessions and skip user-input prompt.*)

Returns: *to_extract* (list)

Return type: new list of selected sessions to extract.

`moseq2_extract.helpers.data.h5_to_dict` (*h5file*, *path: str = '/'*) → dict
Loads h5 file and returns dictionary object representing all contained data, given a path within the h5 file.

Parameters:

- **h5file (str or h5py.File)** (*file path to the given h5 file or the h5 file handle*)
- **path (str)** (*path to the base dataset within the h5 file. Default: '/'*)

Returns: *out* (dict)

Return type: a dict with h5 file contents with the same path structure

`moseq2_extract.helpers.data.handle_extract_metadata` (*input_file*, *dirname*, *config_data*, *nframes*)
Extracts metadata from input depth files, either raw or compressed.

Parameters:

- **input_file (str)** (*path to input file to extract*)
- **dirname (str)** (*path to directory where extraction files reside.*)
- **config_data (dict)** (*dictionary object containing all required extraction parameters. (auto generated)*)
- **nframes (int)** (*number of frames to extract.*)

Returns: **metadata_path (str)** (path to respective metadata.json) **timestamp_path (str)** (path to respective depth_ts.txt or similar) **alternate_correct (bool)** (indicator for whether an alternate timestamp file was used) **tar (bool)** (indicator for whether the file is compressed.) **nframes (int)** (number of frames to extract) **first_frame_idx (int)** (index number of first frame in extraction.) **last_frame_idx (int)** (index number of last frame in extraction)

`moseq2_extract.helpers.data.load_h5s` (to_load, snake_case=True)
`aggregate_results()` Helper Function to load h5 files.

Parameters:

- **to_load (list)** (list of paths to h5 files.)
- **snake_case (bool)** (whether to save the files using snake_case)

Returns: **loaded (list)**

Return type: list of loaded h5 dicts.

Helpers - Extract Module

`moseq2_extract.helpers.extract.process_extract_batches` (f, input_file, config_data, bground_im, roi, scalars, frame_batches, first_frame_idx, true_depth, tar, strel_tail, strel_min, output_dir, output_filename)
Compute extracted frames and save them to h5 files and avi files.

Parameters:

- **f (h5py.File)** (opened h5 file to write extracted batches to)
- **input_file (str)** (path to depth file)
- **config_data (dict)** (dictionary containing extraction parameters (autogenerated))
- **bground_im (2d numpy array)** (r x c, background image)
- **roi (2d numpy array)** (r x c, roi image)
- **scalars (list)** (list of keys to scalar attribute values)
- **frame_batches (list)** (list of batches of frames to serially process.)
- **first_frame_idx (int)** (index of starting frame.)
- **true_depth (float)** (computed detected true depth.)
- **tar (bool)** (compressed file indicator.)
- **strel_tail (cv2.StructuringElement)** (Element for tail filtering.)
- **strel_min (int)** (minimum kernel size)
- **output_dir (str)** (path to output directory that contains the extracted data, e.g. (proc/).)
- **output_filename (str)** (name of h5 file containing extraction data, e.g. (results_00).)

Returns: **video_pipe (bool)**

Return type: boolean for whether function is done writing to video file.

`moseq2_extract.helpers.extract.run_local_extract` (to_extract, params, prefix, skip_extracted, output_directory)

Runs the extract command on given list of sessions to extract on local platform.

Parameters:

- **to_extract (list)** (list of paths to files to extract)
- **params (dict)** (dictionary of ROI metadata from config file.)
- **prefix (str)** (prefix to CLI extraction command.)
- **skip_extracted (bool)** (Whether to skip already extracted session.)
- **output_directory (str)** (path to preferred output directory.)

Returns:

Return type: None

`moseq2_extract.helpers.extract.run_slurm_extract` (`to_extract`, `params`, `partition`, `prefix`, `escape_path`, `skip_extracted`, `output_directory`)

Runs the extract command on given list of sessions to extract on SLURM platform.

Parameters:

- **to_extract (list)** (*list of paths to files to extract*)
- **params (dict)** (*dictionary of ROI metadata from config file.*)
- **partition (str)** (*name of slurm partition to use*)
- **prefix (str)** (*prefix to CLI extraction command.*)
- **escape_path (function)** (*gets path to return to original base directory*)
- **skip_extracted (bool)** (*Whether to skip already extracted session.*)
- **output_directory (str)** (*path to preferred output directory.*)

Returns:

Return type: None

Helpers - Wrappers Module

`moseq2_extract.helpers.wrappers.copy_h5_metadata_to_yaml_wrapper` (`input_dir`, `h5_metadata_path`)

Copy's user specified metadata from h5path to a yaml file.

Parameters:

- **input_dir (str)** (*path to directory containing h5 files*)
- **h5_metadata_path (str)** (*path within h5 to desired metadata to copy to yaml.*)

Returns:

Return type: None

`moseq2_extract.helpers.wrappers.extract_wrapper` (`input_file`, `output_dir`, `config_data`, `num_frames=None`, `skip=False`, `extract=None`, `gui=False`)

Wrapper function to run extract function for both GUI and CLI.

Parameters:

- **input_file (str)** (*path to depth file*)
- **output_dir (str)** (*path to directory to save results in.*)
- **config_data (dict)** (*dictionary containing extraction parameters.*)
- **num_frames (int)** (*number of frames to extract. All if None.*)
- **skip (bool)** (*indicates whether to skip file if already extracted*)
- **extract (function)** (*extraction function state (Only passed by CLI)*)
- **gui (bool)** (*indicates if GUI is running.*)

Returns: `output_dir (str)`

Return type: path to directory containing extraction (only if `gui==True`)

`moseq2_extract.helpers.wrappers.flip_file_wrapper` (`config_file`, `output_dir`, `selected_flip=1`, `gui=False`)

Wrapper function to download and save flip classifiers.

Parameters:

- **config_file (str)** (*path to config file*)
- **output_dir (str)** (*path to directory to save classifier in.*)
- **selected_flip (int)** (*index of desired flip classifier.*)
- **gui (bool)** (*indicates if the GUI is running.*)

Returns:

Return type: None

Welcome to moseq2-extract's documentation!

`moseq2_extract.helpers.wrappers.generate_index_wrapper` (input_dir, pca_file, output_file, filter, all_uuids)

Generates index file containing a summary of all extracted sessions.

Parameters:

- **input_dir (str)** (*directory to search for extracted sessions.*)
- **pca_file (str)** (*path to pca_scores file.*)
- **output_file (str)** (*preferred name of the index file.*)
- **filter (list)** (*list of metadata keys to conditionally filter.*)
- **all_uuids (list)** (*list of all session uuids.*)

Returns: **output_file (str)**

Return type: path to index file.

`moseq2_extract.helpers.wrappers.get_roi_wrapper` (input_file, config_data, output_dir=None, output_directory=None, gui=False, extract_helper=False)

Wrapper function to compute ROI given depth file.

Parameters:

- **input_file (str)** (*path to depth file.*)
- **config_data (dict)** (*dictionary of ROI extraction parameters.*)
- **output_dir (str)** (*path to desired directory to save results in.*)
- **output_directory (str)** (*GUI optional secondary external save directory path*)
- **gui (bool)** (*indicate whether GUI is running.*)
- **extract_helper (bool)** (*indicate whether this is being run independently or by extract function*)

Returns: *if gui* – output_dir (str): path to saved ROI results *elif extract_helper* – roi (2d array): ROI image to plot in GUI bground_im (2d array): Background image to plot in GUI first_frame (2d array): First frame image to plot in GUI

moseq2-extract.moseq2_extract.io package

Submodules

IO - Image Module

`moseq2_extract.io.image.read_image` (filename, dtype='uint16', scale=True, scale_key='scale_factor')

Load image data, possibly with scale factor...

filename (str): path to file to write to.

image (2d numpy array): image to write scale (bool): indicates whether to scale image scale_key (str): indicates scale factor.

image (2d np array): loaded image

`moseq2_extract.io.image.write_image` (filename, image, scale=True, scale_factor=None, dtype='uint16', metadata={}, compress=0)

Save image data, possibly with scale factor for easy display.

Parameters:

- **filename (str)** (*path to file to write to.*)
- **image (2d numpy array)** (*image to write*)
- **scale (bool)** (*indicates whether to scale image*)
- **scale_factor (int)** (*factor by which to scale image*)
- **dtype (str)** (*array data type*)
- **metadata (dict)** (*dictionary object that contains scaling info*)
- **compress (int)** (*image compression level*)

IO - Video Module

`moseq2_extract.io.video.convert_mkv_to_avi` (filename)

Converts Azure MKV video file format to AVI.

Parameters: **filename (str)** path to mkv file to convert

Returns: **outpath (str)**

Return type: path to converted AVI video file.

`moseq2_extract.io.video.get_movie_info` (filename, frame_dims=512, 424, bit_depth=16)

Returns dict of movie metadata.

Parameters:

- **filename (str)** (*path to video file*)
- **frame_dims (tuple)** (*video dimensions*)
- **bit_depth (int)** (*integer indicating data type encoding*)

Returns: **metadata (dict)**

Return type: dictionary containing video file metadata

`moseq2_extract.io.video.get_raw_info` (filename, bit_depth=16, frame_dims=512, 424)

Gets info from a raw data file with specified frame dimensions and bit depth.

Parameters:

- **filename (string)** (*name of raw data file*)
- **bit_depth (int)** (*bits per pixel (default: 16)*)
- **frame_dims (tuple)** (*wxh or hwx of each frame*)

Returns: **file_info (dict)**

Return type: dictionary containing depth file metadata

`moseq2_extract.io.video.get_video_info` (filename)

Get dimensions of data compressed using ffv1, along with duration via ffmpeg.

Parameters: **filename (string)** (*name of file*)

Returns: **(dict)**

Return type: dictionary containing video file metadata

`moseq2_extract.io.video.load_movie_data` (filename, frames=None, frame_dims=512, 424, bit_depth=16, **kwargs)

Reads in frames

`moseq2_extract.io.video.read_frames` (filename, frames=range(0, 0), threads=6, fps=30, pixel_format='gray16le', frame_size=None, slices=24, sliceCRC=1, get_cmd=False)

Reads in frames from the .nut/.avi file using a pipe from ffmpeg.

Parameters:

- **filename (str)** (*filename to get frames from*)
- **frames (list or 1d numpy array)** (*list of frames to grab*)
- **threads (int)** (*number of threads to use for decode*)
- **fps (int)** (*frame rate of camera in Hz*)
- **pixel_format (str)** (*ffmpeg pixel format of data*)
- **frame_size (str)** (*wxh frame size in pixels*)
- **slices (int)** (*number of slices to use for decode*)
- **sliceCRC (int)** (*check integrity of slices*)
- **get_cmd (bool)** (*indicates whether function should return ffmpeg command (instead of executing).)*)

Returns: video (3d numpy array)

Return type: frames x h x w

`moseq2_extract.io.video.read_frames_raw` (filename, frames=None, frame_dims=512, 424, bit_depth=16, dtype='<i2', tar_object=None)

Reads in data from raw binary file.

Parameters:

- **filename (string)** (*name of raw data file*)
- **frames (list or range)** (*frames to extract*)
- **frame_dims (tuple)** (*wxh of frames in pixels*)
- **bit_depth (int)** (*bits per pixel (default: 16)*)
- **tar_object (tarfile.TarFile)** (*TarFile object, used for loading data directly from tgz*)

Returns: chunk (numpy ndarray)

Return type: nframes x h x w

`moseq2_extract.io.video.write_frames` (filename, frames, threads=6, fps=30, pixel_format='gray16le', codec='ffv1', close_pipe=True, pipe=None, slices=24, sliceCRC=1, frame_size=None, get_cmd=False, verbose=0)

Write frames to avi file using the ffv1 lossless encoder

Parameters:

- **filename (str)** (*path to file to write to.*)
- **frames (np.ndarray)** (*frames to write*)
- **threads (int)** (*number of threads to write video*)
- **fps (int)** (*frames per second*)
- **pixel_format (str)** (*format video color scheme*)
- **codec (str)** (*ffmpeg encoding-writer method to use*)
- **close_pipe (bool)** (*indicates to close the open pipe to video when done writing.*)
- **pipe (subProcess.Pipe)** (*pipe to currently open video file.*)
- **slices (int)** (*number of frame slices to write at a time.*)
- **sliceCRC (int)** (*check integrity of slices*)
- **frame_size (tuple)** (*shape/dimensions of image.*)
- **get_cmd (bool)** (*indicates whether function should return ffmpeg command (instead of executing)*)
- **verbose (bool)** (*output progress.*)

Returns: pipe (subProcess.Pipe)

Return type: indicates whether video writing is complete.

Welcome to moseq2-extract's documentation!

```
moseq2_extract.io.video.write_frames_preview(filename, frames=array([], dtype=float64), threads=6,
fps=30, pixel_format='rgb24', codec='h264', slices=24, slicecrc=1, frame_size=None, depth_min=0, depth_max=80,
get_cmd=False, cmap='jet', pipe=None, close_pipe=True, frame_range=None)
```

Writes out a false-colored mp4 video.

Parameters:

- **filename (str)** (*path to file to write to.*)
- **frames (np.ndarray)** (*frames to write*)
- **threads (int)** (*number of threads to write video*)
- **fps (int)** (*frames per second*)
- **pixel_format (str)** (*format video color scheme*)
- **codec (str)** (*ffmpeg encoding-writer method to use*)
- **slices (int)** (*number of frame slices to write at a time.*)
- **sliceCRC (int)** (*check integrity of slices*)
- **frame_size (tuple)** (*shape/dimensions of image.*)
- **depth_min (int)** (*minimum mouse depth from floor in (mm)*)
- **depth_max (int)** (*maximum mouse depth from floor in (mm)*)
- **get_cmd (bool)** (*indicates whether function should return ffmpeg command (instead of executing)*)
- **cmap (str)** (*color map to use.*)
- **pipe (subProcess.Pipe)** (*pipe to currently open video file.*)
- **close_pipe (bool)** (*indicates to close the open pipe to video when done writing.*)
- **frame_range (range())** (*frame indices to write on video*)

Returns: **pipe (subProcess.Pipe)**

Return type: indicates whether video writing is complete.

moseq2-extract.moseq2_extract.tests.integration_tests package

Submodules

Integration Tests - Test CLI Module

```
class moseq2_extract.tests.integration_tests.test_cli.CLITests (methodName='runTest')
```

Bases: `unittest.case.TestCase`

`test_convert_raw_to_avi ()`

`test_copy_slice ()`

`test_download_flip_file ()`

`test_extract ()`

`test_find_roi ()`

`test_generate_config ()`

```
moseq2_extract.tests.integration_tests.test_cli.write_fake_movie (data_path)
```

Integration Tests - Test GUI Module

Welcome to moseq2-extract's documentation!

```
class moseq2_extract.tests.integration_tests.test_gui.GUITests (methodName='runTest')
    Bases: unittest.case.TestCase

    progress_vars = {'base_dir': './', 'config_file': 'TBD', 'crowd_dir': 'TBD', 'index_file': 'TBD', 'model_path': 'TBD',
    'pca_dirname': 'TBD', 'plot_path': 'TBD', 'scores_filename': 'TBD', 'scores_path': 'TBD', 'train_data_dir': 'TBD'}

    test_aggregate_results_command ()

    test_check_progress ()

    test_download_flip_file_command ()

    test_extract_command ()

    test_extract_found_sessions ()

    test_find_roi_command ()

    test_generate_config_command ()

    test_generate_index_command ()

    test_get_found_sessions ()

    test_restore_progress_vars ()

    test_sample_extract_command ()

    test_update_progress ()

    test_view_extractions ()
```

moseq2-extract.moseq2_extract.tests.unit_tests package

Submodules

Unit Tests - Test Extract-Proc Module

```
class moseq2_extract.tests.unit_tests.test_extract_proc.TestExtractProc
(methodName='runTest')
    Bases: unittest.case.TestCase

    test_clean_frames ()

    test_compute_scalars ()

    test_crop_and_rotate ()

    test_get_frame_features ()

    test_get_largest_cc ()

    test_get_roi ()

    moseq2_extract.tests.unit_tests.test_extract_proc.script_loc (request)
```

Unit Tests - Test Extract-ROI Module

Welcome to moseq2-extract's documentation!

```
class moseq2_extract.tests.unit_tests.test_extract_roi.TestExtractROI
(methodName='runTest')
    Bases: unittest.case.TestCase

    test_plane_fit3 ()

    test_plane_ransac ()
```

Unit Tests - Test Extract-Track Module

```
class moseq2_extract.tests.unit_tests.test_extract_track.TestEMTracking
(methodName='runTest')
    Bases: unittest.case.TestCase

    test_em_get_ll ()

    test_em_tracking ()

moseq2_extract.tests.unit_tests.test_extract_track.make_fake_movie ()
```

Unit Tests - Test Helper-Data Module

```
class moseq2_extract.tests.unit_tests.test_helper_data.TestHelperData
(methodName='runTest')
    Bases: unittest.case.TestCase

    test_build_manifest ()

    test_copy_manifest_results ()

    test_extract_h5 ()

    test_load_h5s ()

    test_selected_sessions ()
```

Unit Tests - Test Helper-Extract Module

```
class moseq2_extract.tests.unit_tests.test_helper_extract.TestHelperExtract
(methodName='runTest')
    Bases: unittest.case.TestCase

    run_local_extract ()

    test_process_extract_batches ()

    test_run_slurm_extract ()
```

Unit Tests - Test IO-Image Module

```
class moseq2_extract.tests.unit_tests.test_io_image.TestImageIO (methodName='runTest')
    Bases: unittest.case.TestCase

    test_read_image ()

    test_write_image ()
```

Unit Tests - Test IO-Video Module

```
class moseq2_extract.tests.unit_tests.test_io_video.TestVideoIO (methodName='runTest')
    Bases: unittest.case.TestCase

    test_ffv1 ()

    test_get_movie_info ()

    test_get_raw_info ()

    test_load_movie_data ()

    test_read_frames_raw ()

    test_write_frames_preview ()
```

Unit Tests - Test Utilities Module

```
class moseq2_extract.tests.unit_tests.test_util.testExtractUtils (methodName='runTest')
    Bases: unittest.case.TestCase

    test_click_param_annot ()

    test_gen_batch_sequence ()

    test_load_metadata ()

    test_load_timestamps ()

    test_save_dict_contents_to_h5 ()

    test_scalar_attributes ()

    test_select_strel ()
```

Submodules

CLI Module

```
moseq2_extract.cli.new_init (self, *args, **kwargs)
```

GUI Module

```
moseq2_extract.gui.aggregate_extract_results_command (input_dir, format, output_dir,
output_directory=None)
```

Finds all extracted h5, yaml and avi files and copies them all to a new directory relabeled with their respective session names. Also generates the index file.

Parameters:

- **input_dir (str)** (path to base directory to recursively search for h5s)
- **format (str)** (filename format for info to include in filenames)
- **output_dir (str)** (path to directory to save all aggregated results)
- **output_directory (str)** (alternate path to save results)

Returns: **indexpath (str)**

Return type: path to newly generated index file.

`moseq2_extract.gui.check_progress` (base_dir, progress_filepath, output_directory=None)
Checks whether progress file exists and prompts user input on whether to overwrite, load old, or generate a new one.

Parameters:

- **base_dir (str)** (path to directory to create/find progress file)
- **progress_filepath (str)** (path to progress filename)
- **output_directory (str)** (optional alternative output directory path.)

Returns:

Return type: All restored variables or None.

`moseq2_extract.gui.download_flip_command` (output_dir, config_file="", selection=1)
Downloads flip classifier and saves its path in the inputted config file

Parameters:

- **output_dir (str)** (path to output directory to save flip classifier)
- **config_file (str)** (path to config file)
- **selection (int)** (index of which flip file to download (default is Adult male C57 classifier))

Returns:

Return type: None

`moseq2_extract.gui.extract_command` (input_file, output_dir, config_file, num_frames=None, skip=False)
Command to extract a full depth file

Parameters:

- **input_file (str)** (path to depthfile)
- **output_dir (str)** (path to output directory)
- **config_file (str)** (path to config file)
- **num_frames (int)** (number of frames to extract. All if None.)
- **skip (bool)** (skip already extracted file.)

Returns:

Return type: None

`moseq2_extract.gui.extract_found_sessions` (input_dir, config_file, ext, extract_all=True, skip_extracted=False, output_directory=None)
Searches for all depth files within input_directory with selected extension

Parameters:

- **input_dir (str)** (path to directory containing all session folders)
- **config_file (str)** (path to config file)
- **ext (str)** (file extension to search for)
- **extract_all (bool)** (if True, auto searches for all sessions, else, prompts user to select sessions individually.)
- **skip_extracted (bool)** (indicates whether to skip already extracted session.)
- **output_directory (str)** (optional alternative output_directory.)

Returns:

Return type: None

`moseq2_extract.gui.find_roi_command` (input_dir, config_file, exts=['dat', 'mkv', 'avi'], output_directory=None)
Computes ROI files given depth file

Parameters:

- **input_dir (str)** (*path to directory containing depth file*)
- **config_file (str)** (*path to config file*)
- **exts (list)** (*list of supported extensions*)
- **output_directory (str)** (*alternate output path*)

Returns: **images (list of 2d arrays)** (*list of 2d array images to graph in Notebook.*) **filenames (list)** (*list of paths to respective image paths*)

`moseq2_extract.gui.generate_config_command` (output_file)
Generates configuration file to use throughout pipeline.

Parameters: **output_file (str)** (*path to saved config file.*)

Returns: **(str)**

Return type: status message.

`moseq2_extract.gui.generate_index_command` (input_dir, pca_file, output_file, filter, all_uuids)
Generates Index File based on aggregated sessions

Parameters:

- **input_dir (str)** (*path to aggregated_results/ dir*)
- **pca_file (str)** (*path to pca file*)
- **output_file (str)** (*index file name*)
- **filter (list)** (*keys to filter through*)
- **all_uuids (list)** (*all extracted session uuids*)

Returns: **output_file (str)**

Return type: path to index file.

`moseq2_extract.gui.get_found_sessions` (data_dir="", exts=['dat', 'mkv', 'avi'])
Find all depth recording sessions (with given extensions) to work on given base directory.

Parameters:

- **data_dir (str)** (*path to directory containing all session folders*)
- **exts (list)** (*list of depth file extensions to search for*)

Returns: **data_dir (str)** (*path to base_dir to save in progress file*) **found_sessions (int)** (*number of found sessions with given extensions*)

`moseq2_extract.gui.restore_progress_vars` (progress_file)
Restore all saved progress variables to Jupyter Notebook.

Parameters: **progress_file (str)** (*path to progress file*)

Returns:

Return type: All progress file variables

`moseq2_extract.gui.sample_extract_command` (input_dir, config_file, nframes, output_directory=None, exts=['dat', 'mkv', 'avi'])

Test extract command to extract a subset of the video.

Parameters:

- **input_dir (str)** (*path to directory containing depth file to extract*)
- **config_file (str)** (*path to config file*)
- **nframes (int)** (*number of frames to extract*)
- **output_directory (str)** (*path to alternative directory*)
- **exts (list)** (*list of supported depth file extensions.*)

Returns: **output_dir (str)**

Return type: path to directory containing sample extraction results.

`moseq2_extract.gui.update_progress` (progress_file, varK, varV)
Updates progress file with new notebook variable

Parameters:

- **progress_file (str)** (*path to progress file*)
- **varK (str)** (*key in progress file to update*)
- **varV (str)** (*updated value to write*)

Returns:

Return type: None

`moseq2_extract.gui.view_extraction` (extractions)

Prompts user to select which extracted video(s) to preview.

Parameters: **extractions (list)** (*list of paths to all extracted avi videos.*)

Returns: **extractions (list)**

Return type: list of selected extractions.

Moseq2-Extract Utilities Module

`moseq2_extract.util.build_path` (keys: dict, format_string: str, snake_case=True) → str

Produce a new file name using keys collected from extraction h5 files. The format string must be using python's formatting specification, i.e. '{subject_name}_{session_name}'.

Parameters:

- **keys (dict)** (*dictionary specifying which keys used to produce the new file name*)
- **format_string (str)** (*the string to reformat using the keys dictionary*)
- **snake_case (bool)** (*whether to save the files with snake_case*)

Returns: **out (str)**

Return type: a newly formatted filename useable with any operating system

`moseq2_extract.util.camel_to_snake` (s)

Converts CamelCase to snake_case

Parameters: **s (str)** (*CamelCase string to convert to snake_case.*)

Returns: **(str)**

Return type: string in snake_case

`moseq2_extract.util.clean_file_str` (file_str: str, replace_with: str = '-') → str

Removes invalid characters for a file name from a string.

Parameters:

- **file_str (str)** (*filename substring to replace*)
- **replace_with (str)** (*value to replace str with*)

Returns: **out (str)**

Return type: cleaned file string

`moseq2_extract.util.click_param_annot` (click_cmd)

Given a click.Command instance, return a dict that maps option names to help strings. Currently skips click.Arguments, as they do not have help strings.

Parameters: **click_cmd (click.Command)** (*command to introspect*)

Returns: **annotations (dict)**

Return type: click.Option.human_readable_name as keys; click.Option.help as values

`moseq2_extract.util.command_with_config` (config_file_param_name)

`moseq2_extract.util.convert_pxs_to_mm` (coords, resolution=512, 424, field_of_view=70.6, 60, true_depth=673.1)

Converts x, y coordinates in pixel space to mm. #
<http://stackoverflow.com/questions/17832238/kinect-intrinsic-parameters-from-field-of-view/18199938#18199938>
<http://www.imaginativeuniversal.com/blog/post/2014/03/05/quick-reference-kinect-1-vs-kinect-2.aspx>
<http://smeenk.com/kinect-field-of-view-comparison/>

Parameters:

- **coords (list)** (*list of x,y pixel coordinates*)
- **resolution (tuple)** (*image dimensions*)
- **field_of_view (tuple)** (*width and height scaling params*)
- **true_depth (float)** (*detected true depth*)

Returns: **new_coords (list)**

Return type: x,y coordinates in mm

`moseq2_extract.util.convert_raw_to_avi_function` (input_file, chunk_size=2000, fps=30, delete=False, threads=3)

Converts depth file to avi file.

Parameters:

- **input_file (str)** (*path to depth file*)
- **chunk_size (int)** (*size of chunks to process at a time*)
- **fps (int)** (*frames per second*)
- **delete (bool)** (*whether to delete original depth file*)
- **threads (int)** (*number of threads to write video.*)

Returns:

Return type: None

`moseq2_extract.util.escape_path` (path)

Given current path, will return a path to return to original base directory. (Used in recursive h5 search, etc.)

Parameters: **path (str)** (*path to current working dir*)

Returns: **path (str)**

Return type: path to original base_dir

`moseq2_extract.util.gen_batch_sequence` (nframes, chunk_size, overlap, offset=0)

Generates batches used to chunk videos prior to extraction.

Parameters:

- **nframes (int)** (*total number of frames*)
- **chunk_size (int)** (*desired chunk size*)
- **overlap (int)** (*number of overlapping frames*)
- **offset (int)** (*frame offset*)

Returns:

Return type: Yields list of batches

`moseq2_extract.util.h5_to_dict` (h5file, path) → dict

Loads h5 contents to dictionary object.

Parameters:

- **h5file (str or h5py.File)** (*file path to the given h5 file or the h5 file handle*)
- **path (str)** (*path to the base dataset within the h5 file*)

Returns: **out (dict)**

Return type: a dict with h5 file contents with the same path structure

`moseq2_extract.util.load_metadata` (metadata_file)

Loads metadata.

Parameters: **metadata_file (str)** (*path to metadata file*)

Returns:

Return type: metadata (dict)

`moseq2_extract.util.load_textdata` (data_file, dtype=<class 'numpy.float32'>)

Loads timestamp from txt/csv file

Parameters:

- **data_file (str)** (*path to timestamp file*)
- **dtype (dtype)** (*data type of timestamps*)

Returns: **data (np.ndarray)** (*timestamp data*) **timestamps (np.array)** (*time stamp keynames.*)

`moseq2_extract.util.load_timestamps(timestamp_file, col=0)`

Read timestamps from space delimited text file.

Parameters:

- **timestamp_file (str)** (*path to timestamp file*)
- **col (int)** (*column in ts file read.*)

Returns: **ts (list)**

Return type: list of timestamps

`moseq2_extract.util.mouse_threshold_filter(h5file, thresh=0)`

Filters frames in h5 files by threshold value

Parameters:

- **h5file (str)** (*path to h5 file*)
- **thresh (int)** (*threshold at which to apply filter*)

Returns: **(np boolean array)**

Return type: array of regions to include after threshold filter.

`moseq2_extract.util.read_yaml(yaml_file)`

Reads yaml file into dict object

Parameters: **yaml_file (str)** (*path to yaml file*)

Returns: **return_dict (dict)**

Return type: dict of yaml contents

`moseq2_extract.util.recursive_find_h5s`

(`root_dir='/Users/aymanzeine/Desktop/moseq/moseq2-extract/docs'`, `ext='.h5'`, `yaml_string='{ }.yaml'`)

Recursively find h5 files, along with yaml files with the same basename

Parameters:

- **root_dir (str)** (*path to base directory to begin recursive search in.*)
- **ext (str)** (*extension to search for*)
- **yaml_string (str)** (*string for filename formatting when saving data*)

Returns: **h5s (list)** (*list of found h5 files*) **dicts (list)** (*list of found metadata files*) **yamls (list)** (*list of found yaml files*)

`moseq2_extract.util.recursive_find_unextracted_dirs`

(`root_dir='/Users/aymanzeine/Desktop/moseq/moseq2-extract/docs'`, `session_pattern='session_\\d+\\.(?:tgz|tar\\.gz)'`, `filename='.dat'`, `yaml_path='proc/results_00.yaml'`, `metadata_path='metadata.json'`, `skip_checks=True`)

Recursively find unextracted (or incompletely extracted) directories

Parameters:

- **root_dir (os Path-like)** (*path to base directory to start recursive search from.*)
- **session_pattern (str)** (*folder name pattern to search for*)
- **filename (str)** (*file extension to search for*)
- **yaml_path (str)** (*path to respective extracted metadata*)
- **metadata_path (str)** (*path to relative metadata.json files*)
- **skip_checks (bool)** (*indicates whether to check if the files exist at the given relative paths*)

Returns: **proc_dirs (list)**

Return type: list of paths to each unextracted session's proc/ directory

`moseq2_extract.util.save_dict_contents_to_h5(h5, dic, root='/', annotations=None)`

Save an dict to an h5 file, mounting at root. Keys are mapped to group names recursively.

Parameters:

- **h5 (h5py.File instance)** (*h5py.file object to operate on*)
- **dic (dict)** (*dictionary of data to write*)
- **root (string)** (*group on which to add additional groups and datasets*)
- **annotations (dict)** (*annotation data to add to corresponding h5 datasets. Should contain same keys as dic.*)

Returns:

Return type: None

`moseq2_extract.util.scalar_attributes()`

Gets scalar attributes

Returns: **attributes (dict)**

Return type: collection of metadata keys and descriptions.

`moseq2_extract.util.select_strel (string='e', size=10, 10)`

Returns structuring element of specified shape.

Parameters:

- **string (str)** (*indicates whether to use ellipse or rectangle*)
- **size (tuple)** (*size of structuring element*)

Returns:

Return type: strel (cv2.StructuringElement)

`moseq2_extract.util.strided_app (a, L, S)`

from <https://stackoverflow.com/questions/40084931/taking-subarrays-from-numpy-array-with-given-stride-steps/40085052#40085052> # dang this is fast!

Parameters:

- **a (np.ndarray)** - array to get subarrays from.
- **L (int)** - Window Length
- **S (int)** - Stride size

Returns:

Return type: (np.ndarray) - array of subarrays at stride S.

`moseq2_extract.util.time_str_for_filename (time_str: str) → str`

Process the time string supplied by moseq to be used in a filename. This removes colons, milliseconds, and timezones.

Parameters: **time_str (str)** (*time str to format*)

Returns: **out (str)**

Return type: formatted timestamp str

Module contents

Indices and tables

- `genindex`
- `modindex`
- `search`

Index

A

aggregate_extract_results_command() (in module moseq2_extract.gui)

apply_roi() (in module moseq2_extract.extract.proc)

B

build_manifest() (in module moseq2_extract.helpers.data)

build_path() (in module moseq2_extract.util)

C

camel_to_snake() (in module moseq2_extract.util)

check_progress() (in module moseq2_extract.gui)

clean_dict() (in module moseq2_extract.helpers.data)

clean_file_str() (in module moseq2_extract.util)

clean_frames() (in module moseq2_extract.extract.proc)

click_param_annot() (in module moseq2_extract.util)

CLITests (class in moseq2_extract.tests.integration_tests.test_cli)

command_with_config() (in module moseq2_extract.util)

compute_scalars() (in module moseq2_extract.extract.proc)

convert_mkv_to_avi() (in module moseq2_extract.io.video)

convert_pxs_to_mm() (in module moseq2_extract.util)

convert_raw_to_avi_function() (in module moseq2_extract.util)

copy_h5_metadata_to_yaml_wrapper() (in module moseq2_extract.helpers.wrappers)

copy_manifest_results() (in module moseq2_extract.helpers.data)

create_extract_h5() (in module moseq2_extract.helpers.data)

crop_and_rotate_frames() (in module moseq2_extract.extract.proc)

D

download_flip_command() (in module moseq2_extract.gui)

E

em_get_ll() (in module moseq2_extract.extract.track)

em_init() (in module moseq2_extract.extract.track)

em_iter() (in module moseq2_extract.extract.track)

em_tracking() (in module moseq2_extract.extract.track)

escape_path() (in module moseq2_extract.util)

extract_chunk() (in module moseq2_extract.extract.extract)

extract_command() (in module moseq2_extract.gui)

extract_found_sessions() (in module moseq2_extract.gui)

extract_wrapper() (in module moseq2_extract.helpers.wrappers)

F

feature_hampel_filter() (in module moseq2_extract.extract.proc)

find_roi_command() (in module moseq2_extract.gui)

flip_file_wrapper() (in module moseq2_extract.helpers.wrappers)

G

gen_batch_sequence() (in module moseq2_extract.util)

generate_config_command() (in module moseq2_extract.gui)

generate_index_command() (in module moseq2_extract.gui)

generate_index_wrapper() (in module moseq2_extract.helpers.wrappers)

get_bbox() (in module moseq2_extract.extract.proc)

get_bground_im() (in module moseq2_extract.extract.proc)

get_bground_im_file() (in module moseq2_extract.extract.proc)

get_flips() (in module moseq2_extract.extract.proc)

get_found_sessions() (in module moseq2_extract.gui)

get_frame_features() (in module moseq2_extract.extract.proc)

get_largest_cc() (in module moseq2_extract.extract.proc)

get_movie_info() (in module moseq2_extract.io.video)

get_raw_info() (in module moseq2_extract.io.video)

get_roi() (in module moseq2_extract.extract.proc)

get_roi_wrapper() (in module moseq2_extract.helpers.wrappers)

get_selected_sessions() (in module moseq2_extract.helpers.data)

get_video_info() (in module moseq2_extract.io.video)

GUITests (class in moseq2_extract.tests.integration_tests.test_gui)

H

[h5_to_dict\(\)](#) (in module [moseq2_extract.helpers.data](#))
(in module [moseq2_extract.util](#))

[handle_extract_metadata\(\)](#) (in module [moseq2_extract.helpers.data](#))

I

[im_moment_features\(\)](#) (in module [moseq2_extract.extract.proc](#))

L

[load_h5s\(\)](#) (in module [moseq2_extract.helpers.data](#))

[load_metadata\(\)](#) (in module [moseq2_extract.util](#))

[load_movie_data\(\)](#) (in module [moseq2_extract.io.video](#))

[load_textdata\(\)](#) (in module [moseq2_extract.util](#))

[load_timestamps\(\)](#) (in module [moseq2_extract.util](#))

M

[make_fake_movie\(\)](#) (in module [moseq2_extract.tests.unit_tests.test_extract_track](#))

[model_smoother\(\)](#) (in module [moseq2_extract.extract.proc](#))

module

[moseq2_extract.cli](#)

[moseq2_extract.extract.extract](#)

[moseq2_extract.extract.proc](#)

[moseq2_extract.extract.roi](#)

[moseq2_extract.extract.track](#)

[moseq2_extract.gui](#)

[moseq2_extract.helpers.data](#)

[moseq2_extract.helpers.extract](#)

[moseq2_extract.helpers.wrappers](#)

[moseq2_extract.io.image](#)

[moseq2_extract.io.video](#)

[moseq2_extract.tests.integration_tests.test_cli](#)

[moseq2_extract.tests.integration_tests.test_gui](#)

[moseq2_extract.tests.unit_tests.test_extract_proc](#)

[moseq2_extract.tests.unit_tests.test_extract_roi](#)

[moseq2_extract.tests.unit_tests.test_extract_track](#)

[moseq2_extract.tests.unit_tests.test_helper_data](#)

[moseq2_extract.tests.unit_tests.test_helper_extract](#)

[moseq2_extract.tests.unit_tests.test_io_image](#)

[moseq2_extract.tests.unit_tests.test_io_video](#)

[moseq2_extract.tests.unit_tests.test_util](#)

[moseq2_extract.util](#)

[moseq2_extract.cli](#)

[module](#)

[moseq2_extract.extract.extract](#)

[module](#)

[moseq2_extract.extract.proc](#)

[module](#)

[moseq2_extract.extract.roi](#)

[module](#)

[moseq2_extract.extract.track](#)

[module](#)

[moseq2_extract.gui](#)

[module](#)

[moseq2_extract.helpers.data](#)

[module](#)

[moseq2_extract.helpers.extract](#)

[module](#)

[moseq2_extract.helpers.wrappers](#)

[module](#)

[moseq2_extract.io.image](#)

[module](#)

[moseq2_extract.io.video](#)

[module](#)

[moseq2_extract.tests.integration_tests.test_cli](#)

[module](#)

[moseq2_extract.tests.integration_tests.test_gui](#)

[module](#)

[moseq2_extract.tests.unit_tests.test_extract_proc](#)

[module](#)

[moseq2_extract.tests.unit_tests.test_extract_roi](#)

[module](#)

[moseq2_extract.tests.unit_tests.test_extract_track](#)

[module](#)

[moseq2_extract.tests.unit_tests.test_helper_data](#)

[module](#)

[moseq2_extract.tests.unit_tests.test_helper_extract](#)

[module](#)

[moseq2_extract.tests.unit_tests.test_io_image](#)

[module](#)

[moseq2_extract.tests.unit_tests.test_io_video](#)

[module](#)

[moseq2_extract.tests.unit_tests.test_util](#)

[module](#)

[moseq2_extract.util](#)

[module](#)

[mouse_threshold_filter\(\)](#) (in module [moseq2_extract.util](#))

N

`new_init()` (in module `moseq2_extract.cli`)

P

`plane_fit3()` (in module `moseq2_extract.extract.roi`)

`plane_ransac()` (in module `moseq2_extract.extract.roi`)

`process_extract_batches()` (in module `moseq2_extract.helpers.extract`)

`progress_vars` (`moseq2_extract.tests.integration_tests.test_gui.GUITests` attribute)

R

`read_frames()` (in module `moseq2_extract.io.video`)

`read_frames_raw()` (in module `moseq2_extract.io.video`)

`read_image()` (in module `moseq2_extract.io.image`)

`read_yaml()` (in module `moseq2_extract.util`)

`recursive_find_h5s()` (in module `moseq2_extract.util`)

`recursive_find_unextracted_dirs()` (in module `moseq2_extract.util`)

`restore_progress_vars()` (in module `moseq2_extract.gui`)

`run_local_extract()` (in module `moseq2_extract.helpers.extract`)

(`moseq2_extract.tests.unit_tests.test_helper_extract.TestHelperExtract` method)

`run_slurm_extract()` (in module `moseq2_extract.helpers.extract`)

S

`sample_extract_command()` (in module `moseq2_extract.gui`)

`save_dict_contents_to_h5()` (in module `moseq2_extract.util`)

`scalar_attributes()` (in module `moseq2_extract.util`)

`script_loc()` (in module `moseq2_extract.tests.unit_tests.test_extract_proc`)

`select_strel()` (in module `moseq2_extract.util`)

`strided_app()` (in module `moseq2_extract.util`)

T

`test_aggregate_results_command()` (`moseq2_extract.tests.integration_tests.test_gui.GUITests` method)

`test_build_manifest()` (`moseq2_extract.tests.unit_tests.test_helper_data.TestHelperData` method)

`test_check_progress()` (`moseq2_extract.tests.integration_tests.test_gui.GUITests` method)

`test_clean_frames()` (`moseq2_extract.tests.unit_tests.test_extract_proc.TestExtractProc` method)

`test_click_param_annot()` (`moseq2_extract.tests.unit_tests.test_util.testExtractUtils` method)

`test_compute_scalars()` (`moseq2_extract.tests.unit_tests.test_extract_proc.TestExtractProc` method)

`test_convert_raw_to_avi()` (`moseq2_extract.tests.integration_tests.test_cli.CLITests` method)

`test_copy_manifest_results()` (`moseq2_extract.tests.unit_tests.test_helper_data.TestHelperData` method)

`test_copy_slice()` (`moseq2_extract.tests.integration_tests.test_cli.CLITests` method)

`test_crop_and_rotate()` (`moseq2_extract.tests.unit_tests.test_extract_proc.TestExtractProc` method)

`test_download_flip_file()` (`moseq2_extract.tests.integration_tests.test_cli.CLITests` method)

`test_download_flip_file_command()` (`moseq2_extract.tests.integration_tests.test_gui.GUITests` method)

`test_em_get_ll()` (`moseq2_extract.tests.unit_tests.test_extract_track.TestEMTracking` method)

`test_em_tracking()` (`moseq2_extract.tests.unit_tests.test_extract_track.TestEMTracking` method)

`test_extract()` (`moseq2_extract.tests.integration_tests.test_cli.CLITests` method)

`test_extract_command()` (`moseq2_extract.tests.integration_tests.test_gui.GUITests` method)

`test_extract_found_sessions()` (`moseq2_extract.tests.integration_tests.test_gui.GUITests` method)

`test_extract_h5()` (`moseq2_extract.tests.unit_tests.test_helper_data.TestHelperData` method)

`test_ffv1()` (`moseq2_extract.tests.unit_tests.test_io_video.TestVideoIO` method)

`test_find_roi()` (`moseq2_extract.tests.integration_tests.test_cli.CLITests` method)

`test_find_roi_command()` (`moseq2_extract.tests.integration_tests.test_gui.GUITests` method)

`test_gen_batch_sequence()` (`moseq2_extract.tests.unit_tests.test_util.testExtractUtils` method)

`test_generate_config()` (`moseq2_extract.tests.integration_tests.test_cli.CLITests` method)

`test_generate_config_command()` (`moseq2_extract.tests.integration_tests.test_gui.GUITests` method)

`test_generate_index_command()` (`moseq2_extract.tests.integration_tests.test_gui.GUITests` method)

`test_get_found_sessions()` (`moseq2_extract.tests.integration_tests.test_gui.GUITests` method)

`test_get_frame_features()` (`moseq2_extract.tests.unit_tests.test_extract_proc.TestExtractProc` method)

`test_get_largest_cc()` (`moseq2_extract.tests.unit_tests.test_extract_proc.TestExtractProc` method)

test_get_movie_info() (moseq2_extract.tests.unit_tests.test_io_video.TestVideoIO method)

test_get_raw_info() (moseq2_extract.tests.unit_tests.test_io_video.TestVideoIO method)

test_get_roi() (moseq2_extract.tests.unit_tests.test_extract_proc.TestExtractProc method)

test_load_h5s() (moseq2_extract.tests.unit_tests.test_helper_data.TestHelperData method)

test_load_metadata() (moseq2_extract.tests.unit_tests.test_util.testExtractUtils method)

test_load_movie_data() (moseq2_extract.tests.unit_tests.test_io_video.TestVideoIO method)

test_load_timestamps() (moseq2_extract.tests.unit_tests.test_util.testExtractUtils method)

test_plane_fit3() (moseq2_extract.tests.unit_tests.test_extract_roi.TestExtractROI method)

test_plane_ransac() (moseq2_extract.tests.unit_tests.test_extract_roi.TestExtractROI method)

test_process_extract_batches() (moseq2_extract.tests.unit_tests.test_helper_extract.TestHelperExtract method)

test_read_frames_raw() (moseq2_extract.tests.unit_tests.test_io_video.TestVideoIO method)

test_read_image() (moseq2_extract.tests.unit_tests.test_io_image.TestImageIO method)

test_restore_progress_vars() (moseq2_extract.tests.integration_tests.test_gui.GUITests method)

test_run_slurm_extract() (moseq2_extract.tests.unit_tests.test_helper_extract.TestHelperExtract method)

test_sample_extract_command() (moseq2_extract.tests.integration_tests.test_gui.GUITests method)

test_save_dict_contents_to_h5() (moseq2_extract.tests.unit_tests.test_util.testExtractUtils method)

test_scalar_attributes() (moseq2_extract.tests.unit_tests.test_util.testExtractUtils method)

test_select_strel() (moseq2_extract.tests.unit_tests.test_util.testExtractUtils method)

test_selected_sessions() (moseq2_extract.tests.unit_tests.test_helper_data.TestHelperData method)

test_update_progress() (moseq2_extract.tests.integration_tests.test_gui.GUITests method)

test_view_extractions() (moseq2_extract.tests.integration_tests.test_gui.GUITests method)

test_write_frames_preview() (moseq2_extract.tests.unit_tests.test_io_video.TestVideoIO method)

test_write_image() (moseq2_extract.tests.unit_tests.test_io_image.TestImageIO method)

TestEMTracking (class in moseq2_extract.tests.unit_tests.test_extract_track)

TestExtractProc (class in moseq2_extract.tests.unit_tests.test_extract_proc)

TestExtractROI (class in moseq2_extract.tests.unit_tests.test_extract_roi)

testExtractUtils (class in moseq2_extract.tests.unit_tests.test_util)

TestHelperData (class in moseq2_extract.tests.unit_tests.test_helper_data)

TestHelperExtract (class in moseq2_extract.tests.unit_tests.test_helper_extract)

TestImageIO (class in moseq2_extract.tests.unit_tests.test_io_image)

TestVideoIO (class in moseq2_extract.tests.unit_tests.test_io_video)

time_str_for_filename() (in module moseq2_extract.util)

U

update_progress() (in module moseq2_extract.gui)

V

view_extraction() (in module moseq2_extract.gui)

W

write_fake_movie() (in module moseq2_extract.tests.integration_tests.test_cli)

write_frames() (in module moseq2_extract.io.video)

write_frames_preview() (in module moseq2_extract.io.video)

write_image() (in module moseq2_extract.io.image)

Python Module Index

m

- [moseq2_extract](#)
- [moseq2_extract.cli](#)
- [moseq2_extract.extract.extract](#)
- [moseq2_extract.extract.proc](#)
- [moseq2_extract.extract.roi](#)
- [moseq2_extract.extract.track](#)
- [moseq2_extract.gui](#)
- [moseq2_extract.helpers.data](#)
- [moseq2_extract.helpers.extract](#)
- [moseq2_extract.helpers.wrappers](#)
- [moseq2_extract.io.image](#)
- [moseq2_extract.io.video](#)
- [moseq2_extract.tests.integration_tests.test_cli](#)
- [moseq2_extract.tests.integration_tests.test_gui](#)
- [moseq2_extract.tests.unit_tests.test_extract_proc](#)
- [moseq2_extract.tests.unit_tests.test_extract_roi](#)
- [moseq2_extract.tests.unit_tests.test_extract_track](#)
- [moseq2_extract.tests.unit_tests.test_helper_data](#)
- [moseq2_extract.tests.unit_tests.test_helper_extract](#)
- [moseq2_extract.tests.unit_tests.test_io_image](#)
- [moseq2_extract.tests.unit_tests.test_io_video](#)
- [moseq2_extract.tests.unit_tests.test_util](#)
- [moseq2_extract.util](#)