MCVF Documentation

Contents

Module `mcvf`

Implementation of Motion-Compensated Video Filtering

Sub-modules

Module `mcvf.core`

Core MCVF components

Classes

Class `Video`

```
class Video(
    fname: str = None
)
```

A sequence of frames read from a file

Methods

Method `apply_filter`

```
def apply_filter(
    self,
    filter: mcvf.filters.Filter
)
```

Parse all frames through a given filter instance

Parameters

`filter` : **filters.Filter** A filter instance to parse frames

Method `load_from_file`

```
def load_from_file(
    self,
    fname: str
)
```

Load a video from filesystem

Parameters

`fname` : **str** The name of the file to load

Method `play`

```
def play(
    self
)
```

Instantiate an OpenCV window and display all frames one by one

Method `save_to_file`

```
def save_to_file(
    self,
    fname: str,
    fps: int
)
```

Write the sequence of frames to a MP4 file

Parameters

`fname` **: str** The name of the file to write
`fps` **: int** How many frames per second to encode in the destination file

Module `mcvf.filters`

Video filters with support for motion-compensation

Classes

Class `Filter`

```
class Filter
```

Base class for video filters

Descendants

- mcvf.filters.GaussianFilter
- mcvf.filters.MCFilter
- mcvf.filters.MFDrawerFilter

Methods

Method `filter_frames`

```
def filter_frames(
    self,
    frames: list
) -> Iterable[numpy.ndarray]
```

Parse the given list of frames and return a new list of filtered ones

Parameters

`frames` **: list[np.ndaray]** A list of frames (as NumPy arrays) to filter

Returns

```
frames : list[np.ndarray]
    A list of filtered frames
```

Class `GaussianFilter`

```
class GaussianFilter
```

Low-Pass Gaussian blur

Ancestors (in MRO)

- mcvf.filters.Filter

Methods

Method `filter_frames`

```
def filter_frames(
    self,
    frames: list
) -> Iterable[numpy.ndarray]
```

Apply a gaussian blur with 5x5 kernel

Parameters

`frames` **: list[np.ndaray]** A list of frames (as NumPy arrays) to filter

Returns

```
frames : list[np.ndarray]
    A list of filtered frames
```

Class `MCDarkenFilter`

```
class MCDarkenFilter(
    block_size: int
)
```

Motion-Compensated darkening filter

Parameters

`block_size` **: int** The size in pixel of the blocks in which the frames will be subdivided
`motion_threshold` **: int** The motion vector strength above which there will be considered to be movement

Ancestors (in MRO)

- mcvf.filters.MCFilter
- mcvf.filters.Filter

Methods

Method `filter_frames`

```
def filter_frames(
    self,
    frames: list
) -> Iterable[numpy.ndarray]
```

Darken the frame areas where motion is not present

Parameters

`frames` **: list[np.ndaray]** A list of frames (as NumPy arrays) to filter

Returns

```
frames : list[np.ndarray]
    A list of filtered frames
```

Class `MCFilter`

```
class MCFilter(
    block_size: int
)
```

Base class for motion-compensated video filters

Parameters

`block_size` **: int** The size in pixel of the blocks in which the frames will be subdivided
`motion_threshold` **: int** The motion vector strength above which there will be considered to be movement

Ancestors (in MRO)

- mcvf.filters.Filter

Descendants

- mcvf.filters.MCDarkenFilter
- mcvf.filters.MCGaussianFilter

Methods

Method `filter_frames`

```
def filter_frames(
    self,
    frames: list
) -> Iterable[numpy.ndarray]
```

Parse the given list of frames contextually with a Motion Field and return a new list of filtered ones

Parameters

`frames` **: list[np.ndaray]** A list of frames (as NumPy arrays) to filter

Returns

```
frames : list[np.ndarray]
    A list of filtered frames
```

Class `MCGaussianFilter`

```
class MCGaussianFilter(
    block_size: int
)
```

Motion-Compensated Low-Pass gaussian blur

Parameters

`block_size` **: int** The size in pixel of the blocks in which the frames will be subdivided
`motion_threshold` **: int** The motion vector strength above which there will be considered to be movement

Ancestors (in MRO)

- mcvf.filters.MCFilter
- mcvf.filters.Filter

Methods

Method `filter_frames`

```
def filter_frames(
    self,
    frames: list
) -> Iterable[numpy.ndarray]
```

Apply a 5x5 gaussian blur to the frames where motion is not present

Parameters

`frames` **: list[np.ndaray]** A list of frames (as NumPy arrays) to filter

Returns

```
frames : list[np.ndarray]
    A list of filtered frames
```

Class `MFDrawerFilter`

```
class MFDrawerFilter(
    block_size: int
)
```

A drawer filter to render motion vectors onto each frame

Attributes

`block_size` **: int** The size in pixel of the blocks in which the frames will be subdivided

Parameters

`block_size` **: int** The size in pixel of the blocks in which the frames will be subdivided

Ancestors (in MRO)

- mcvf.filters.Filter

Methods

Method `filter_frames`

```
def filter_frames(
    self,
    frames: list
) -> Iterable[numpy.ndarray]
```

Overlay a needle diagram to each frame showing its motion field

Parameters

`frames` **: list[np.ndaray]** A list of frames (as NumPy arrays) to filter

Returns

```
frames : list[np.ndarray]
    A list of filtered frames
```

Module `mcvf.motion_estimation`

Motion estimation utilities

Classes

Class `BBME`

```
class BBME(
    frames: list,
    block_size: int = 16,
    window_size: int = 5,
    algorithm: str = 'EBBME'
)
```

A Block-Based Motion Estimator

Attributes

`block_size` **: int** The size in pixels of each block in which frames are subdivided
`window_size` **: int** How many neighboring blocks are searched for a match when estimating motion
`algorithm` **: str** The algorithm being used to detect motion

Parameters

`frames` **: list[np.ndaray]** A list of frames to process (as NumPy arrays)
`block_size` **: int** The size in pixels of each block in which frames are subdivided
`window_size` **: int** How many neighboring blocks are searched for a match when estimating motion
`algorithm` **: str** Which algorithm to use to detect motion

Raises

ValueError If the requested algorithm is not available

Methods

Method `calculate_motion_field`

```
def calculate_motion_field(
    self
)
```

Iterate all frames to estimate a motion field for each one

Returns

`motion_field` **: list[np.ndaray]** A list of motion fields, one for each frame after the first

Class `MotionVector`

```
class MotionVector(
    origin_x: int,
    origin_y: int,
    target_x: int,
    target_y: int
)
```

A Motion Vector describing a detected movement

Attributes

`origin_x` **: int** The X coordinate from where the vector originates
`origin_y` **: int** The Y coordinate from where the vector originates
`target_x` **: int** The X coordinate where the vector ends
`target_y` **: int** The Y coordinate where the vector ends

Parameters

`origin_x` **: int** The X coordinate from where the vector originates
`origin_y` **: int** The Y coordinate from where the vector originates
`target_x` **: int** The X coordinate where the vector ends
`target_y` **: int** The Y coordinate where the vector ends

Generated by *pdoc* 0.10.0 (https://pdoc3.github.io).