scikit-surgerysurfacematch Documentation

Matt Clarkson

Oct 27, 2020

Contents

1 Features **1**
- 1.1 scikit-surgerysurfacematch . . . 1
- 1.2 Algorithms . . . 3
- 1.3 Interfaces . . . 9
- 1.4 Processing Pipelines . . . 10

Index **15**

CHAPTER 1

Features

- Base classes (pure virtual interfaces), for video segmentation, stereo reconstruction, rigid registration / pose estimation.
- A base class to handle rectification properly, and the right coordinate transformation, to save you the trouble.
- Stereo reconstruction classes based on Stoyanov MICCAI 2010, and OpenCV SGBM reconstruction, using above interface, and both allowing for optional masking.
- Rigid registration using PCL's ICP implementation, which is wrapped in scikit-surgerypclcpp
- Rigid registration using GoICP, which is wrapped in scikit-surgerygoicp
- A pipeline to combine the above, segment a video pair, do reconstruction, and register to a 3D model, where each part can then be swapped with whatever implementation you want, as long as you implement the right interface.
- A pipeline to take multiple stereo video snapshots, do surface reconstruction, mosaic them together, and then register to a 3D model. Again, each main component (video segmentation, surface reconstruction, rigid registration) is swappable. Inspired by: [Xiaohui Zhang's](https://doi.org/10.1007/s11548-019-01974-6) method.

Source code is avaialble on GitHub.

1.1 scikit-surgerysurfacematch

Author: Matt Clarkson

scikit-surgerysurfacematch is part of the SNAPPY software project, developed at the Wellcome EPSRC Centre for Interventional and Surgical Sciences, part of University College London (UCL).

scikit-surgerysurfacematch supports Python 3.6 - 3.8

scikit-surgerysurfacematch contains algorithms that are useful in stereo reconstruction from video images, and matching to a pre-operative 3D model, represented as a point cloud.

1.1.1 Features

- Base classes (pure virtual interfaces), for video segmentation, stereo reconstruction, rigid registration / pose estimation.
- A base class to handle rectification properly, and the right coordinate transformation, to save you the trouble.
- Stereo reconstruction classes based on Stoyanov MICCAI 2010, and OpenCV SGBM reconstruction, using above interface, and both allowing for optional masking.
- Rigid registration using PCL's ICP implementation, which is wrapped in scikit-surgerypclcpp
- Rigid registration using GoICP, which is wrapped in scikit-surgerygoicp
- A pipeline to combine the above, segment a video pair, do reconstruction, and register to a 3D model, where each part can then be swapped with whatever implementation you want, as long as you implement the right interface.
- A pipeline to take multiple stereo video snapshots, do surface reconstruction, mosaic them together, and then register to a 3D model. Again, each main component (video segmentation, surface reconstruction, rigid registration) is swappable. Inspired by: [Xiaohui Zhang's](https://doi.org/10.1007/s11548-019-01974-6) method.

1.1.2 Developing

Cloning

You can clone the repository using the following command:

```
git clone https://github.com/UCL/scikit-surgerysurfacematch
```

Running tests

Pytest is used for running unit tests:

```
pip install pytest
python -m pytest
```

Linting

This code conforms to the PEP8 standard. Pylint can be used to analyse the code:

```
pip install pylint
pylint --rcfile=tests/pylintrc sksurgerysurfacematch
```

1.1.3 Installing

You can pip install directly from the repository as follows:

```
pip install git+https://github.com/UCL/scikit-surgerysurfacematch
```

Contributing

Please see the contributing guidelines.

Useful links

- Source code repository
- Documentation

1.1.4 Licensing and copyright

1.1.5 Acknowledgements

Supported by Wellcome and EPSRC.

1.2 Algorithms

1.2.1 ICP

PCL ICP implementation of RigidRegistration interface.

class sksurgerysurfacematch.algorithms.pcl_icp_registration.**RigidRegistration**(*max_iterations: int = 100*, *max_corresponder float = 1*, *transformation_epsilon: float = 0.0001*, *fitness_epsilon: float = 0.0001*, *use_lm_icp: bool = True*)

Bases: sksurgerysurfacematch.interfaces.rigid_registration.RigidRegistration

Class that uses PCL implementation of ICP to register fixed/moving clouds.

register(*moving_cloud: numpy.ndarray*, *fixed_cloud: numpy.ndarray*)

Uses PCL library, wrapped in scikit-surgerypclcpp.

Parameters

- **moving_cloud** – [Nx3] source/moving point cloud.
- **fixed_cloud** – [Mx3] target/fixed point cloud.

Returns [4x4] transformation matrix, moving-to-fixed space.

1.2.2 GoICP

Go ICP implementation of RigidRegistration interface.

class sksurgerysurfacematch.algorithms.goicp_registration.**RigidRegistration**(*dt_size: int = 200, dt_factor: float = 2.0, normalise: bool = True, num_moving_points: int = 1000, rotation_limits=[-45, 45], trans_limits=[-0.5, 0.5]*)

Bases: sksurgerysurfacematch.interfaces.rigid_registration.RigidRegistration

Class that uses GoICP implementation to register fixed/moving clouds.

Parameters

- **dt_size** (*int, optional*) – Distance transform size, defaults to 200
- **dt_factor** (*float, optional*) – Distance transform scale factor, defaults to 2.0
- **normalise** (*bool, optional*) – Normalsie data before GoICP (recommended), the resultant transform is given back in 'normal' space, i.e. the noramlisation is reversed, defaults to True
- **num_moving_points** (*int, optional*) – How many points to sample from moving point cloud (fewer is faster), defaults to 1000
- **rotation_limits** (*list, optional*) – Limits on how much the moving cloud can rotate to find a good solution, defaults to [-45, 45]
- **trans_limits** (*list, optional*) – Limits on how much the moving cloud can translate to find a good solution. If normalise=True, this translation is in [-1 1] space, defaults to [-0.5, 0.5]

register(*moving_cloud: numpy.ndarray*, *fixed_cloud: numpy.ndarray*) → numpy.ndarray

Uses GoICP library, wrapped in scikit-surgerygoicp.

Parameters

- **fixed_cloud** – [Nx3] fixed point cloud.
- **moving_cloud** – [Mx3] moving point cloud.
- **normalise** – If true, data will be centred around 0 and normalised.
- **num_moving_points** – How many points to sample from moving cloud if 0, use all points

Returns [4x4] transformation matrix, moving-to-fixed space.

sksurgerysurfacematch.algorithms.goicp_registration.**create_scaling_matrix**(*scale: float*) → numpy.ndarray

Create a scaling matrix, with the same value in each axis.

sksurgerysurfacematch.algorithms.goicp_registration.**create_translation_matrix**(*translate: numpy.ndarray*) → numpy.ndarray

Create translation matrix from 3x1 translation vector.

sksurgerysurfacematch.algorithms.goicp_registration.**demean_and_normalise**(*points_a: numpy.ndarray*, *points_b: numpy.ndarray*)

Independently centre each point cloud around 0,0,0, then normalise both to [-1,1].

Parameters

- **points_a** (*np.ndarray*) – 1st point cloud
- **points_b** (*np.ndarray*) – 2nd point cloud

Returns normalised points clouds, scale factor & translations

sksurgerysurfacematch.algorithms.goicp_registration.**numpy_to_POINT3D_array**(*numpy_pointcloud*)

Covert numpy array to POINT3D array suitable for GoICP algorithm.

sksurgerysurfacematch.algorithms.goicp_registration.**set_rotnode**(*limits_degrees*) → sksurgery-goicp-python.ROTNODE

Setup a ROTNODE with upper/lower rotation limits

sksurgerysurfacematch.algorithms.goicp_registration.**set_transnode**(*trans_limits*) → sksurgery-goicp-python.TRANSNODE

Setup a TRANSNODE with upper/lower limits

1.2.3 Stereo Recon Base Class

Base class for surface reconstruction on already rectified images.

class sksurgerysurfacematch.algorithms.reconstructor_with_rectified_images.**StereoReconstruc**

Bases: sksurgerysurfacematch.interfaces.stereo_reconstructor.StereoReconstructor

Base class for those stereo reconstruction methods that work specifically from rectified images. This class handles rectification and the necessary coordinate transformations. Note: The client calls the reconstruct() method which requires undistorted images, which are NOT already rectified. It's THIS class that does the rectification for you, and calls through to the _compute_disparity() method that derived classes must implement.

Constructor creates some member variables, so this class becomes statefull. You call reconstruct() once, and then you can call extract multiple times with different masks to pull out different subsets of data.

Parameters

- **lower_disparity_multiplier** – min=median - (this * std), default = 2.0.
- **upper_disparity_multiplier** – max=median + (this * std), default = 2.0.
- **apha** – OpenCV alpha parameter, default = 0.

extract(*left_mask: numpy.ndarray*)

Extracts the actual point cloud. This is a separate method, so that you can reconstruct once using reconstruct(), and then call this extract method with multiple masks, without incurring the cost of multiple calls to the reconstruction algorithm, which may be expensive. :param left_mask: mask image, single channel, same size as left_image :return: [Nx6] point cloud where the 6 columns are x, y, z in left camera space, followed by r, g, b colours.

reconstruct(*left_image: numpy.ndarray*, *left_camera_matrix: numpy.ndarray*, *right_image: numpy.ndarray*, *right_camera_matrix: numpy.ndarray*, *left_to_right_rmat: numpy.ndarray*, *left_to_right_tvec: numpy.ndarray*, *left_mask: numpy.ndarray = None*)

Implementation of stereo surface reconstruction that takes undistorted images, rectifies them, asks derived classes to compute a disparity map on the rectified images, and then sorts out extracting points and their colours.

Camera parameters are those obtained from OpenCV.

Parameters

- **left_image** – undistorted left image, BGR
- **left_camera_matrix** – [3x3] camera matrix
- **right_image** – undistorted right image, BGR
- **right_camera_matrix** – [3x3] camera matrix
- **left_to_right_rmat** – [3x3] rotation matrix

- **left_to_right_tvec** – [3x1] translation vector
- **left_mask** – mask image, single channel, same size as left_image

Returns [Nx6] point cloud where the 6 columns

are x, y, z in left camera space, followed by r, g, b colours.

1.2.4 SGBM Stereo Recon

Surface reconstruction using OpenCV's SGBM reconstruction

class sksurgerysurfacematch.algorithms.sgbm_reconstructor.**SGBMReconstructor**(*min_disparity=16*, *num_disparities=112*, *block_size=3*, *p_1=360*, *p_2=1440*, *disp_12_max_diff=0*, *uniqueness_ratio=0*, *speckle_window_size*, *speckle_range=0*)

Bases: sksurgerysurfacematch.algorithms.reconstructor_with_rectified_images.StereoReconstructorWithRectifiedImages

Constructor. See OpenCV StereoSGBM for parameter comments.

1.2.5 Stoyanov Stereo Recon

Surface reconstruction using Stoyanov MICCAI 2010 paper.

class sksurgerysurfacematch.algorithms.stoyanov_reconstructor.**StoyanovReconstructor**(*use_hartl*

Bases: sksurgerysurfacematch.interfaces.stereo_reconstructor.StereoReconstructor

Constructor.

reconstruct(*left_image: numpy.ndarray*, *left_camera_matrix: numpy.ndarray*, *right_image: numpy.ndarray*, *right_camera_matrix: numpy.ndarray*, *left_to_right_rmat: numpy.ndarray*, *left_to_right_tvec: numpy.ndarray*, *left_mask: numpy.ndarray = None*)

Implementation of dense stereo surface reconstruction using Dan Stoyanov's MICCAI 2010 method.

Camera parameters are those obtained from OpenCV.

Parameters

- **left_image** – undistorted left image, BGR
- **left_camera_matrix** – [3x3] camera matrix
- **right_image** – undistorted right image, BGR
- **right_camera_matrix** – [3x3] camera matrix
- **left_to_right_rmat** – [3x1] rotation matrix
- **left_to_right_tvec** – [3x1] translation vector
- **left_mask** – mask image, single channel, same size as left_image

Returns [Nx6] point cloud where the 6 columns

are x, y, z in left camera space, and r, g, b, colors.

1.3 Interfaces

1.3.1 Rigid Registration

Base class (pure virtual interface) for rigid registration.

class sksurgerysurfacematch.interfaces.rigid_registration.**RigidRegistration**
Bases: object

Base class for classes that can rigidly register (align), two point clouds.

register(*source_cloud: numpy.ndarray*, *target_cloud: numpy.ndarray*)
A derived class must implement this.

Parameters

- **source_cloud** – [Nx3] fixed point cloud.
- **target_cloud** – [Mx3] moving point cloud.

Returns residual, [4x4] transformation matrix, moving-to-fixed space.

1.3.2 Stereo Reconstruction

Base class (pure virtual interface) for classes that do stereo recon.

class sksurgerysurfacematch.interfaces.stereo_reconstructor.**StereoReconstructor**
Bases: object

Base class for stereo reconstruction algorithms. Clients call the reconstruct() method, passing in undistorted images. The output is an [Nx6] array where the N rows are each point, and the 6 columns are x, y, z, r, g, b.

reconstruct(*left_image: numpy.ndarray*, *left_camera_matrix: numpy.ndarray*, *right_image: numpy.ndarray*, *right_camera_matrix: numpy.ndarray*, *left_to_right_rmat: numpy.ndarray*, *left_to_right_tvec: numpy.ndarray*, *left_mask: numpy.ndarray = None*)
A derived class must implement this.

Camera parameters are those obtained from OpenCV.

Parameters

- **left_image** – left image, BGR
- **left_camera_matrix** – [3x3] camera matrix
- **right_image** – right image, BGR
- **right_camera_matrix** – [3x3] camera matrix
- **left_to_right_rmat** – [3x3] rotation matrix
- **left_to_right_tvec** – [3x1] translation vector
- **left_mask** – mask image, single channel, same size as left_image

Returns [Nx6] point cloud in left camera space, where N is the number

of points, and 6 columns are x,y,z,r,g,b.

1.3.3 Video Segmentation

Base class (pure virtual interface) for classes to do video segmentation

class sksurgerysurfacematch.interfaces.video_segmentor.**VideoSegmentor**
Bases: object

Base class for classes that can segment a video image into a binary mask. For example, a deep network that can produce a mask of background=0, foreground=255.

segment(*image: numpy.ndarray*)
A derived class must implement this.

Parameters **image** – image, BGR

Returns image, same size as input, 1 channel, uchar, [0-255].

1.4 Processing Pipelines

1.4.1 Register Point Cloud To Stereo Images

Pipeline to register 3D point cloud to 2D stereo video

class sksurgerysurfacematch.pipelines.register_cloud_to_stereo_reconstruction.**Register3DToS**

Bases: object

Class for single-shot, registration of 3D point cloud to stereo video. Uses Dependency Injection for each pluggable component.

Parameters

- **video_segmentor** – Optional class to pre-segment the video.
- **surface_reconstructor** – Mandatory class to do reconstruction.
- **rigid_registration** – Mandatory class to perform rigid alignment.
- **left_camera_matrix** – [3x3] camera matrix.
- **right_camera_matrix** – [3x3] camera matrix.
- **left_to_right_rmat** – [3x3] left-to-right rotation matrix.
- **left_to_right_tvec** – [1x3] left-to-right translation vector.

- **left_mask** – a static mask to apply to stereo reconstruction.
- **z_range** – [min range, max range] to limit reconstructed points.
- **radius_removal** – [radius, number] to reject points with too few neighbours
- **voxel_reduction** – [vx, vy, vz] parameters for PCL Voxel Grid reduction.

register(*reference_cloud: numpy.ndarray*, *left_image: numpy.ndarray*, *right_image: numpy.ndarray*, *initial_ref2recon: numpy.ndarray = None*) → Tuple[numpy.ndarray, numpy.ndarray, numpy.ndarray, numpy.ndarray]

Main method to do a single 3D cloud to 2D stereo video registration.

Camera calibration parameters are in OpenCV format.

Parameters

- **reference_cloud** – [Nx3] points, each row, x,y,z, e.g. from CT/MR.
- **left_image** – undistorted, BGR image
- **right_image** – undistorted, BGR image
- **initial_ref2recon** – [4x4] of initial rigid transform.

Returns residual, [4x4] transform, of reference_cloud to left camera space, [Mx3] downsampled xyz points and [Mx6] reconstructed point cloud, as [x, y, z, r, g, b] rows.

1.4.2 Register Point Cloud To Mosaic

Pipeline to register 3D point cloud to mosaic'ed surface reconstruction.

class sksurgerysurfacematch.pipelines.register_cloud_to_stereo_mosaic.**Register3DToMosaicedS**

Bases: object

Class to register a point cloud to a series of surfaces derived from stereo video, and stitched together.

Uses Dependency Injection for each pluggable component.

Parameters

- **video_segmentor** – Optional class to pre-segment the video.
- **surface_reconstructor** – Mandatory class to do reconstruction.
- **rigid_registration** – Mandatory class to perform rigid alignment.
- **left_camera_matrix** – [3x3] camera matrix.

- **right_camera_matrix** – [3x3] camera matrix.
- **left_to_right_rmat** – [3x3] left-to-right rotation matrix.
- **left_to_right_tvec** – [1x3] left-to-right translation vector.
- **min_number_of_keypoints** – Number of keypoints to use for matching.
- **max_fre_threshold** – maximum FRE when stitching frames together.
- **left_mask** – a static mask to apply to stereo reconstruction.
- **z_range** – [min range, max range] to limit reconstructed points.
- **radius_removal** – [radius, number] to reject points with too few neighbours
- **voxel_reduction** – [vx, vy, vz] parameters for PCL

Voxel Grid reduction.

grab(*left_image: numpy.ndarray*, *right_image: numpy.ndarray*)

Call this repeatedly to grab a surface and use ORM key points to match previous reconstruction to the current frame.

Parameters

- **left_image** – undistorted, BGR image
- **right_image** – undistorted, BGR image

register(*point_cloud: numpy.ndarray*, *initial_transform: numpy.ndarray = None*)

Registers a point cloud to the internal mosaicc'ed reconstruction.

Parameters

- **point_cloud** – [Nx3] points, each row, x,y,z, e.g. from CT/MR.
- **initial_transform** – [4x4] of initial rigid transform.

Returns residual, [4x4] transform, of point_cloud to left camera space,

and [Mx6] reconstructed point cloud, as [x, y, z, r, g, b] rows.

reset()

Reset's internal data members, so that you can start accumulating data again.

Index

C

create_scaling_matrix() (*in module sksurgerysurfacematch.algorithms.goicp_registration*), 6
create_translation_matrix() (*in module sksurgerysurfacematch.algorithms.goicp_registration*), 6

D

demean_and_normalise() (*in module sksurgerysurfacematch.algorithms.goicp_registration*), 6

E

extract() (*sksurgerysurfacematch.algorithms.reconstructor_with_rectified_images.StereoReconstructorWithRectifiedImages method*), 7

G

grab() (*sksurgerysurfacematch.pipelines.register_cloud_to_stereo_mosaic.Register3DToMosaicedStereoVideo method*), 14

N

numpy_to_POINT3D_array() (*in module sksurgerysurfacematch.algorithms.goicp_registration*), 6

R

reconstruct() (*sksurgerysurfacematch.algorithms.reconstructor_with_rectified_images.StereoReconstructorWithRectifiedImages method*), 7
reconstruct() (*sksurgerysurfacematch.algorithms.stoyanov_reconstructor.StoyanovReconstructor method*), 8
reconstruct() (*sksurgerysurfacematch.interfaces.stereo_reconstructor.StereoReconstructor method*), 9
register() (*sksurgerysurfacematch.algorithms.goicp_registration.RigidRegistration method*), 5
register() (*sksurgerysurfacematch.algorithms.pcl_icp_registration.RigidRegistration method*), 4
register() (*sksurgerysurfacematch.interfaces.rigid_registration.RigidRegistration method*), 9
register() (*sksurgerysurfacematch.pipelines.register_cloud_to_stereo_mosaic.Register3DToM method*), 14
register() (*sksurgerysurfacematch.pipelines.register_cloud_to_stereo_reconstruction.Register method*), 12
Register3DToMosaicedStereoVideo (*class in sksurgerysurfacematch.pipelines.register_cloud_to_stereo_mosaic*), 12
Register3DToStereoVideo (*class in sksurgerysurfacematch.pipelines.register_cloud_to_stereo_reconstruction*), 10
reset() (*sksurgerysurfacematch.pipelines.register_cloud_to_stereo_mosaic.Register3DToM method*), 14
RigidRegistration (*class in sksurgerysurfacematch.algorithms.goicp_registration*), 4
RigidRegistration (*class in sksurgerysurfacematch.algorithms.pcl_icp_registration*), 3
RigidRegistration (*class in sksurgerysurfacematch.interfaces.rigid_registration*), 9

S

segment() (*sksurgerysurfacematch.interfaces.video_segmentor.VideoSegmentor method*), 10
set_rotnode() (*in module sksurgerysurfacematch.algorithms.goicp_registration*), 6
set_transnode() (*in module sksurgerysurfacematch.algorithms.goicp_registration*), 6
SGBMReconstructor (*class in sksurgerysurfacematch.algorithms.sgbm_reconstructor*), 8

sksurgerysurfacematch.algorithms.goicp_registration
(*module*), 4
sksurgerysurfacematch.algorithms.pcl_icp_registration
(*module*), 3
sksurgerysurfacematch.algorithms.reconstructor_with_rectified_images
(*module*), 6
sksurgerysurfacematch.algorithms.sgbm_reconstructor
(*module*), 8
sksurgerysurfacematch.algorithms.stoyanov_reconstructor
(*module*), 8
sksurgerysurfacematch.interfaces.rigid_registration
(*module*), 9
sksurgerysurfacematch.interfaces.stereo_reconstructor
(*module*), 9
sksurgerysurfacematch.interfaces.video_segmentor
(*module*), 10
sksurgerysurfacematch.pipelines.register_cloud_to_stereo_mosaic
(*module*), 12
sksurgerysurfacematch.pipelines.register_cloud_to_stereo_reconstruction
(*module*), 10
StereoReconstructor (*class in sksurgerysurfacematch.interfaces.stereo_reconstructor*), 9
StereoReconstructorWithRectifiedImages
(*class in sksurgerysurfacematch.algorithms.reconstructor_with_rectified_images*), 6
StoyanovReconstructor (*class in sksurgerysurfacematch.algorithms.stoyanov_reconstructor*), 8

V

VideoSegmentor (*class in sksurgerysurfacematch.interfaces.video_segmentor*), 10