scikit-surgeryarucotracker
Documentation

Stephen Thompson

Dec 22, 2021
## Contents

1 scikit-surgeryarucotracker .......................... 3  
1.1 Installing ............................................ 3  
1.2 Using ................................................... 4  
1.3 Developing ............................................ 4  
1.4 Licensing and copyright ............................. 4  
1.5 Acknowledgements .................................... 5  
2 ARuCo Tracking ........................................... 7  
3 Register detected image points (2D) to model points (3D) ........................................ 9  
4 Manage rigid bodies consisting of multiple tags .................................................. 11  
Index ......................................................... 13
Source code is available on GitHub.
scikit-surgeryarucotracker

Author: Stephen Thompson

scikit-surgeryarucotracker provides a simple Python interface between OpenCV’s ARuCo marker tracking libraries and other Python packages designed around scikit-surgerytrackers. It allows you to treat an object tracked using ARuCo markers in the same way as an object tracked using other tracking hardware (e.g. aruco - scikit-surgerynditracker).

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

scikit-surgeryarucotracker is tested with Python 3.6 and may support other Python versions.

1.1 Installing

```bash
pip install scikit-surgeryarucotracker
```
1.2 Using

Configuration is done using Python libraries. Tracking data is returned in NumPy arrays.

```python
from sksurgeryarucotracker.arucotracker import ArUcoTracker
config = {
    "video source" : 0
}
tracker = ArUcoTracker(config)

tracker.start_tracking()
print(tracker.get_frame())
tracker.stop_tracking()
tracker.close()
```

1.3 Developing

1.3.1 Cloning

You can clone the repository using the following command:

```bash
git clone https://github.com/SciKit-Surgery/scikit-surgeryarucotracker
```

1.3.2 Running the tests

You can run the unit tests by installing and running tox:

```bash
pip install tox
tox
```

1.3.3 Contributing

Please see the contributing guidelines.

1.3.4 Useful links

- Source code repository
- Documentation

1.4 Licensing and copyright

Copyright 2019 University College London. scikit-surgeryarucotracker is released under the BSD-3 license. Please see the license file for details.
1.5 Acknowledgements

Supported by Wellcome and EPSRC.
ARuCo Tracking

A class for straightforward tracking with an ARuCo

class sksurgeryarucotracker.arucotracker.ArUcoTracker(configuration)
Bases: sksurgerycore.baseclasses.tracker.SKSBaseTracker

Initialises and Configures the ArUco detector

Parameters configuration – A dictionary containing details of the tracker.
  video source: defaults to 0
  aruco dictionary: defaults to DICT_4X4_50
  marker size: defaults to 50 mm
  camera projection: defaults to None
  camera distortion: defaults to None
  smoothing buffer: specify a buffer over which to average the tracking, defaults to 1
  rigid bodies: a list of rigid bodies to track, each body should have a ‘name’, a ‘filename’ where the tag geometry is defined, and an ‘aruco dictionary’ to use. Additionally we can include ‘tag width’ in mm when the tag has been scaled during printing or is displayed on a mobile phone screen or similar

Raises Exception – ImportError, ValueError

close()
Closes the connection to the Tracker and deletes the tracker device.

  Raises Exception – ValueError

get_frame(frame=None)
Gets a frame of tracking data from the Tracker device.

  Parameters frame – an image to process, if None, we use the OpenCV video source.

  Returns
port_numbers: If tools have been defined port numbers are the tool descriptions. Otherwise port numbers are the aruco tag ID prefixed with aruco

time_stamps : list of timestamps (cpu clock), one per tool

frame_numbers : list of framenumbers (tracker clock) one per tool

tracking : list of 4x4 tracking matrices, rotation and position,

tracking_quality : list the tracking quality, one per tool.

Raises Exception – ValueError

get_tool_descriptions()
Returns tool descriptions

has_capture()
:Returns true if the tracker has it’s own opencv source otherwise false

start_tracking()
Tells the tracking device to start tracking. :raise Exception: ValueError

stop_tracking()
Tells the tracking devices to stop tracking. :raise Exception: ValueError
Register detected image points (2D) to model points (3D)

Classes and functions for 2D to 3D registration

`sksurgeryarucotracker.algorithms.registration_2d3d.estimate_poses_no_calibration(marker_corners, aruco_board)`

Returns tracking data for a camera with no calibration data. x and y are the screen pixel coordinates. z is based on the size of the tag in pixels, there is no rotation. No account is taken of the size of the model marker pattern, so it will be bit flakey.

`sksurgeryarucotracker.algorithms.registration_2d3d.estimate_poses_with_calibration(marker_corners2d, marker_ids, aruco_board, camera_projection_matrix, camera_distortion)`

Estimate the pose of a single tag or a multi-tag rigid body when the camera calibration is known. :param marker_corners2d: a list of 2d marker corners, 1 row per tag, 8 columns per tag.

Parameters

- `model_points` – Matched list of of corresponding model points, 1 row per tag, 15 columns per tag: corner points and centre point
- `camera_projection_matrix` – a 3x3 camera projection matrix
- `camera_distortion` – camera distortion vector

:return: a tracking rotation, translation and a quality
CHAPTER 4

Manage rigid bodies consisting of multiple tags

Classes and functions for maintaining ArUco rigid bodies

```python
class sksurgeryarucotracker.algorithms.rigid_bodies.ArUcoRigidBody(rigid_body_name):
    Bases: object

    Class to handle the loading and registering of ArUco Rigid Bodies

    add_single_tag(tag_size, marker_id, dictionary)
    We can use this to track single ArUco tags rather than patterns as long as we know the tag size in mm
    Param tag size in mm
    Param marker id/

    get_dictionary_name()
    returns the name of the aruco dictionary in use

    get_pose(camera_projection_matrix, camera_distortion)
    Estimate the pose of the rigid body, with or without camera calibration
    Param camera_projection_matrix 3x3 projection matrix. If None we estimate pose based on pattern size
    Param 1x5 camera distortion vector

    load_3d_points(filename, dictionaryname)
    Loads the 3D point geometry from a file
    Parameters filename – Path of file containing tag data

    reset_2d_points()
    Clears 2D point lists.

    scale_3d_tags(measured_pattern_width)
    We can scale the tag, which is very useful if you’ve got the tag on your mobile phone.
    Parameters measured_pattern_width – Width of the tag in mm

    set_2d_points(two_d_points, tag_ids)
    takes a list of 2 points, and if the id’s match 3D points, assigns them to a list of 2d points
```
Parameters

- **two_d_points** – array of marker corners, 4 for each tag
- **tag_ids** – id for each tag

**Returns** tag ids for any assigned tags

class sksurgeryarucotracker.algorithms.rigid_bodies.TwoDTags
    Bases: object

Stores two linked arrays, one of tag IDs and the other 2D points

**append_tag** *(tag_id, points)*
    Adds a tag to the two point list :param tag_id: The id of the tag :param points: 4 points defining the tag corners

sksurgeryarucotracker.algorithms.rigid_bodies.configure_rigid_bodies *(configuration)*
    reads configuration and creates a list of rigid bodies together with a list of dictionaries used.

sksurgeryarucotracker.algorithms.rigid_bodies.load_board_from_file *(filename, dictionary=16)*
    loads marker pattern from filename. :return: an aruco.board :raise ValueError: If the file does not have 16 or 13 columns

sksurgeryarucotracker.algorithms.rigid_bodies.scale_tags *(board, measured_pattern_width)*
    We can scale the tag on a board, which is very useful if you've got the tag on your mobile phone.

    **Param** the board to scale
    **Parameters** measured_pattern_width – Width of the tag in mm

sksurgeryarucotracker.algorithms.rigid_bodies.single_tag_board *(tag_size, marker_id, dictionary=aruco_Dictionary 0x7f005e58fb10)*
    Create a board consisting of a single ArUco tag

    **Param** tag size in mm
    **Param** marker id
    **Param** dictionary to use
Index

A

add_single_tag() (sksurgeryaruco-tracker.algorithms.rigid_bodies.ArUcoRigidBody method), 11
append_tag() (sksurgeryaruco-tracker.algorithms.rigid_bodies.TwoDTags method), 12
ArUcoRigidBody (class in sksurgeryaruco-tracker.algorithms.rigid_bodies), 11
ArUcoTracker (class in sksurgeryaruco-tracker.arucotracker), 7

C

close() (sksurgeryaruco-tracker.arucotracker.ArUcoTracker method), 7
configure_rigid_bodies() (in module sksurgeryaruco-tracker.algorithms.rigid_bodies), 12

E

estimate_poses_no_calibration() (in module sksurgeryaruco-tracker.algorithms.registration_2d3d), 9
estimate_poses_with_calibration() (in module sksurgeryaruco-tracker.algorithms.registration_2d3d), 9

G

get_dictionary_name() (sksurgeryaruco-tracker.algorithms.rigid_bodies.ArUcoRigidBody method), 11
get_frame() (sksurgeryaruco-tracker.arucotracker.ArUcoTracker method), 7
get_pose() (sksurgeryaruco-tracker.algorithms.rigid_bodies.ArUcoRigidBody method), 11
get_tool_descriptions() (sksurgeryaruco-tracker.arucotracker.ArUcoTracker method), 8

H

has_capture() (sksurgeryaruco-tracker.arucotracker.ArUcoTracker method), 8

L

load_3d_points() (sksurgeryaruco-tracker.algorithms.rigid_bodies.ArUcoRigidBody method), 11
load_board_from_file() (in module sksurgeryaruco-tracker.algorithms.rigid_bodies), 12

R

reset_2d_points() (sksurgeryaruco-tracker.algorithms.rigid_bodies.ArUcoRigidBody method), 11

S

scale_3d_tags() (sksurgeryaruco-tracker.algorithms.rigid_bodies.ArUcoRigidBody method), 11
scale_tags() (in module sksurgeryaruco-tracker.algorithms.rigid_bodies), 12
set_2d_points() (sksurgeryaruco-tracker.algorithms.rigid_bodies.ArUcoRigidBody method), 11
single_tag_board() (in module sksurgeryaruco-tracker.algorithms.rigid_bodies), 12
sksurgeryarucotracker.algorithms.registration_2d3d (module), 9
sksurgeryarucotracker.algorithms.rigid_bodies (module), 11
sksurgeryarucotracker.arucotracker (module), 7
start_tracking() (sksurgeryaruco-tracker.arucotracker.ArUcoTracker method), 8
stop_tracking() (sksurgeryaruco-tracker.arucotracker.ArUcoTracker method), 8

T

TwoDTags (class in sksurgeryaruco-tracker.algorithms.rigid_bodies), 12