odometry-evaluation

Release 0.0.1

Luiz Cartolano

May 18, 2020

Table of Contents

This is an API that implements a system to evaluate odometry trajectories.

CHAPTER 1

Overview on How to Run this API

1. Either install a Python IDE or create a Python virtual environment to install the packages required
2. Install packages required

CHAPTER 2

Setup procedure

1. **Configure project environment (Either A. Install Pycharm OR B. Create a Virtual Environment)**

    1. **Install Pycharm (www.jetbrains.com/pycharm/download/) or Sublime (https://www.sublimetext.com/3)**

        - configure pylinter

    2. **Create a Python Virtual Environment**

        - Install virtualenv:

```
sudo pip install virtualenv
```

        - Create virtialenv:

```
virtualenv -p python3 <name of virtualenv>
```

        - Install requirements:

```
pip install -r requirements.txt
```

2. Run app.py

    python run.py path_gt path_pred -v python run.py path_gt path_pred -ate python run.py path_gt path_pred -rpe python run.py path_gt path_pred -v -ate -rpe

CHAPTER 3

Visualize Functions

NAME

visualize

DESCRIPTION

Module that implements methods to plot ground_truth trajectories and estimated for 6D poses.

METHODS

get_xy(pose)

Function that extracts (x,y) positions from a list of 6D poses.

get_seq_start(pose)

Function that extracts the first (x,y) point from a list of 6D poses.

EXAMPLES

gt = np.load('04.npy') # create plot obj plt.clf() # get gt_poses gt_x, gt_y = get_xy(gt) # get sequence start x_start, y_start = get_seq_start(gt) # plot gt plt.scatter(x_start, y_start, label='Sequence Start', color='black') plt.plot(gt_x, gt_y, color='g', label='Ground Truth') # make the adjust for compute just translation # instead of absolute position plt.gca().set_aspect('equal', adjustable='datalim') # show plot plt.legend() plt.show()

`src.visualize.`**`get_seq_start`** (*pose*)

Function that extracts the first (x,y) point from a list of 6D poses.

pose : *nd.array*
List of 6D poses.

x_start : *float*
Start x point.

y_start : *float*
Start y point

`src.visualize.`**`get_xy`** (*pose*)

Function that extracts (x,y) positions from a list of 6D poses.

pose : *nd.array*
List of 6D poses.

x_pose : *list*
List of x positions.

y_pose : *list*
List of y positions.

CHAPTER 4

RPE Functions

NAME

rpe_calc

DESCRIPTION

Module that implements methods to calculate the relative pose error between two trajectories.

METHODS

convert_pose_se3(pose_tst, pose_rot)
Convert a rotation matrix (or euler angles) plus a translation vector into a 4x4 pose representation.

relative_se3(pose_1, pose_2)
Relative pose between two poses (drift).

se3_inverse(pose)
The inverse of a pose.

calc_rpe_pair(Q_i, Q_i_delta, p_i, p_i_delta)
The relative error between GT and Predict.

so3_log(rot_matrix)
Gets the rotation matrix from pose.

calculate_rpe_vector(gt_tst, gt_rot, pred_tst, pred_rot)
Gets a vector of relative errors for all poses.

calc_rpe_error(error_vector, error_type='rotation_angle_deg')
Calculate an specific error from relatives errors.

get_statistics(rpe_vector)
Statistics of a vector.

EXAMPLES

calculate rpe errors vector rpe_vector = calculate_rpe_vector(gt_tst, gt_rot, pred_tst, pred_rot) rpe_error = calc_rpe_error(rpe_vector)
calculate errors statistics statistics = get_statistics(rpe_error)

`src.rpe_calc.`**`calc_rpe_error`** (*error_vector*, *error_type='rotation_angle_deg'*)

Calculate an specific error from relatives errors.

error_vector : *list*
List of relative errors.

error_type : *str*
Type of relative error to compute.

error : *list*
The error asked by user.

src.rpe_calc.**calc_rpe_pair** (*q_i*, *q_i_delta*, *p_i*, *p_i_delta*)
The relative error between GT and Predict.

q_i : *np.array*
The pose at time i.

q_i_delta : *np.array*
The pose at time i + delta.

p_i : *np.array*
The predicted pose at time i.

p_i_delta : *np.array*
The predicted pose at time i + delta.

np.float32
The relative distance between two poses.

src.rpe_calc.**calculate_rpe_vector** (*gt_tst*, *gt_rot*, *pred_tst*, *pred_rot*)
Gets a vector of relative errors for all poses.

gt_tst : *np.array*
The (x,y,z) of the ground truth.

gt_rot : *np.array*
The (theta_x, theta_y, theta_z) of the ground truth.

pred_tst : *np.array*
The (x,y,z) of the predict pose.

pred_rot : *np.array*
The (theta_x, theta_y, theta_z) of the predict pose.

errors : *list*
The list of relative errors.

src.rpe_calc.**convert_pose_se3** (*pose_tst*, *pose_rot*)
Convert a rotation matrix (or euler angles) plus a translation vector into a 4x4 pose representation.

pose_tst : *np.array*
The (x,y,z) of pose.

pose_rot : *np.array*
The (theta_x, theta_y, theta_z) of pose.

np.array (4x4)
The pose 4x4 matrix.

src.rpe_calc.**get_statistics** (*rpe_vector*)
Statistics of a vector.

rpe_vector : *list*
List of errors.

dict
Dict with statistics of a list.

src.rpe_calc.**relative_se3** (*pose_1*, *pose_2*)
Relative pose between two poses (drift).

pose_1 : *np.array*
The first pose.

pose_2 : *np.array*
The second pose.

np.float32
The relative transformation pose_1^{⁻1} * pose_2.

src.rpe_calc.**se3_inverse** (*pose*)
The inverse of a pose.

pose : *np.array*
The pose.

np.float32
The inverted pose.

src.rpe_calc.**so3_log** (*rot_matrix*)
Gets the rotation vector from rotation matrix.

rot_matrix : *np.array*
The rotation matrix.

np.float32
The error angle.

CHAPTER 5

ATE Functions

NAME

ate_calc

DESCRIPTION

Module that implements methods to calculate the absolute trajectory error between two trajectories.

METHODS

compute_ate(gt_tst, pred_tst)

Calculate the absolute trajectory error between two poses. Based on LearnerLee - KITTI_odometry_evaluation_tool repository.

compute_ate_horn(gt_tst, pred_tst)

Calculate the absolute trajectory error between two poses. Based on Horn align method. Explained by https://vision.in.tum.de/.

ate_xyz(alignment_error)

Calculate the statistics for separate axis.

EXAMPLES

get translational attrs gt_tst = [v for v in gt_poses[:, 3:6]] pred_tst = [v for v in pred_poses[:, 3:]] alignment_error, trans_error = compute_ate_horn(gt_tst, pred_tst) statistics = ate_xyz(alignment_error)

`src.ate_calc.`**`ate_xyz`** (*alignment_error*)

Calculate the statistics for separate axis.

gt_tst : *list*
List of ground truth poses.

pred_tst : *list*
List of predict poses.

alignment_error : *np.array (3xn)*
A matrix of errors by axis.

trans_error : *list*
The sum of error by row.

`src.ate_calc.`**`compute_ate`** (*gt_tst*, *pred_tst*)

Calculate the absolute trajectory error between two poses. Based on LearnerLee - KITTI_odometry_evaluation_tool repository.

gt_tst : *list*
List of ground truth poses.

pred_tst : *list*
List of predict poses.

alignment_error : *np.array (nx3)*
A matrix of errors by axis.

src.ate_calc.**compute_ate_horn** (*gt_tst*, *pred_tst*)
Calculate the absolute trajectory error between two poses. Based on Horn align method. Explained by https://vision.in.tum.de/.

gt_tst : *list*
List of ground truth poses.

pred_tst : *list*
List of predict poses.

alignment_error : *np.array (3xn)*
A matrix of errors by axis.

trans_error : *list*
The sum of error by row.

CHAPTER 6

Indices and tables

- *Index*
- *Module Index*
- *Search Page*

S

A

C

G

M

R

S