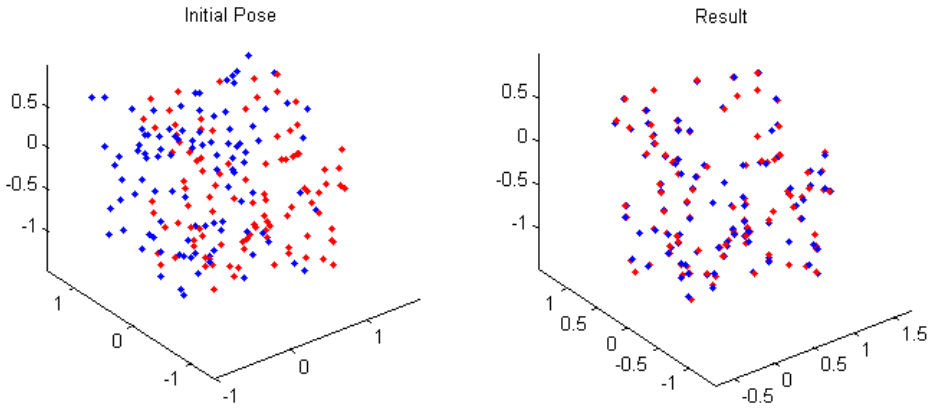
Go-ICP for globally optimal 3D pointset registration

—a C++ implementation (V1.3, 26-Jan-2015)



About

This C++ code implements the Go-ICP algorithm (with trimming strategy for outlier handling). It is free software under the terms of the GNU General Public License (GPL) v3. Please refer to our paper for algorithm details:

Jiaolong Yang, Hongdong Li, Yunde Jia, Go-ICP: Solving 3D Registration Efficiently and Globally Optimally, International Conference on Computer Vision (ICCV), 2013.

Compiling

Use cmake to generate desired projects on different platforms. (See “CMakeLists.txt” in the “src” folder)

Running

Run the compiled binary with following parameters: <MODEL FILENAME> <DATA FILENAME> <NUM DOWNSAMPLED DATA POINTS> <CONFIGURATION FILENAME> <OUTPUT FILENAME>, e.g. “./GoICP model data 1000 config output”, “GoICP.exe model.txt data.txt 500 config.txt output.txt”.

<MODEL FILENAME> and <DATA FILENAME> are the point files of the model and data pointsets respectively. Each point file is in plain text format. It begins with a positive point number *N* in the first line, followed with *N* lines of X, Y, Z values of the *N* points.

<NUM DOWNSAMPLED DATA POINTS> indicates the number of down-sampled data points. Assuming the input data points are randomly ordered, the code will use the first <NUM DOWNSAMPLED DATA POINTS> data points for registration. Please adapt the code should you want to use some other sampling strategies.

<CONFIGURATION FILENAME> is the configuration file containing parameters for the algorithm, e.g. initial rotation and translation cubes, convergence threshold. See “config_example.txt” in the “src” folder for example.

<OUTPUT FILENAME> is the output file containing registration results. By default it contains the obtained 3x3 rotation matrix and 3x1 translation vector only. You can adapt the code to output other results as you wish.

Notes

- Both model and data points should be normalized into $[-1, 1]^3$ prior to running (we recommend first independently centralizing the two point clouds to the origin then simultaneously scaling them). The default initial translation cube is $[-0.5, 0.5]^3$ (see "config_example.txt").
- The convergence threshold is set on the Sum of Squared Error (SSE) as in the code and the paper. For the ease of parameter setting for different numbers of data points, we use Mean of Squared Error (MSE) in the configuration (see "config_example.txt"). We use MSE threshold of 0.001 for the demos. Try smaller ones if your registration results are not satisfactory.
- Building 3D distance transform with (default) 300 discrete nodes in each dimension takes about 20-25s in our experiments. Using smaller values can reduce memory and building time costs, but it will also degrade the distance accuracy.

Acknowledgment

We thank Dylan Campbell for his help in code polishing. This code uses the *nanoflann* library, and a simple matrix library written by Andreas Geiger. The distance transform implementation is adapted from the code of Alexander Vasilevskiy.

If you find the code helpful and use it in your work, please cite our paper. For bugs and advices, please send an email to yangjiaolong@gmail.com.

Change log

V1.3 (26-Jan-2015)
Implemented the intro-selection algorithm
Fixed some minor issues

V1.2 (12-Jun-2014)
Refined the quick-selection algorithm
Added a deconstructor to distance transform class (Thanks to Nima Tajbakhsh)

V1.1 (21-Apr-2014)
Speeded up Trimmed-GoICP (around 2-5 times experimentally) using a quick-selection algorithm

V1.0 (13-Feb-2014)
First complete version for release