A Low-Dimensional Representation for Robust Partial Isometric Correspondences Computation
Source Code Documentation

Alan Brunton

September 4, 2015

1 General Information

1.1 GeoXL 3.5 License

The source code is part of the GeoXL system. The base system is licensed under a permissive BSD license, and each individual module (visual studio project with files; compiled into a loadable DLL) is subject to a specific license (mostly BSD and GPL). Each individual module comes with its own license - see the <module-name>.tags.h files (or alternatively, use the module manager provided within the executable) for details. Please be also aware of license restrictions of the various third party modules included in this distribution.

1.2 Three-Parameter Matching Module License

The code for the paper "A Low-Dimensional Representation for Robust Partial Isometric Correspondences Computation" [3] is licensed under GNU GPL2 (third-party modules might use a different license, as explained in the source archive):

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation. `http://www.gnu.org/licenses/old-licenses/gpl-2.0.html`

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

2 Setup

The executable is self-contained - just copy the whole folder hierarchy to a directory and start "GeoXL35.exe" in the folder "application/bin/«platform»/«configuration»". There is also a batch file as a short cut.

The source code can *almost* be directly compiled using Visual Studio 2010. You need to have QT 4.x.x installed (or use the stripped down, matching version provided along with the sources in the third-party folder). If you use later versions of visual studio, you need to have Visual Studio 2010 installed in parallel and choose the compatibility mode that uses the VS 2010 toolchain. Almost? Two preparations are requirered: If you compile in 64-bit mode (recommended) you have to set an environment variable `%QTDIR64%` that points to the QT 4 directory such that "`include`, `lib`, `bin`" are subdirectories. For 32-bit builds, the environment variable `%QTDIR%` needs to be set the same way, but pointing to the 32-bit versions of the library (the QT installer mostly does this automatically).

3 Usage

Correspondence computation between two shapes as described in [3] can be done in the following two modes:

1. User provides only the two input shapes,

   In this mode, the user imports two meshes separately or import an .object file (containing the two meshes), computes internally features and then run the algorithm to compute correspondences.

   (a) Import input shapes.
   The user can find two test input shapes. These are taken from Vlasic et al. [2](Press "meshes" under "samba" and extract the files "mesh_0118.obj", "mesh_0123.obj"), or can use his own two input shapes (supported file formats: obj).

   For the two shapes that the user chose, they can be imported into *GeoXL* in the following way:

   - Click "File → Import",
   - Under "Import Settings", "FileName" insert the file location path of the shape.
   - For the source shape set "PointCloud Name" to "root/pc0" and set "Target PointCloud" to "root/pc0".
   - For the target shape set "PointCloud Name" to "root/pc1" and set "Target PointCloud" to "root/pc1".

   (b) Compute internal features

   - Select the tool "GMOD14Features" in the Tool pane of the scene editor.
   - Select one of the shapes by clicking on it with the "Select" tool.
   - Click the button "PrecomputeSeedFindingDescriptors".
   - Follow the above procedure for the second shape.
   - Switch the tool to "GMOD14ShapeMatching".
   - Open the settings dialog (see 3.1 below) and set the parameter "featureMatchSource" to "internal".

(c) Run the algorithm to compute correspondences

- Select the tool “GMOD14ShapeMatching”.
- Click the button “computeCorrespondences”.

2. User provides the two input shapes and sparse correspondence information,

   In this mode, the user imports two meshes separately or import an .object file (containing the two meshes), provides sparse correspondence and then runs the algorithm to compute dense correspondences.

   (a) Same as step 1a

   (b) User provides sparse correspondence.

   - Select the tool “GMOD14ShapeMatching".
   - Open the settings dialog and set the parameter “featureMatchSource" to “external".
   - Insert in the field “strExtSparseCorrFile" the name of the file with the sparse correspondence. This file should be located in the location indicated by by the field “m_strDirSwapPath".
     File format: first line is of the format *numLines numCols*, where *numLines* is the number of subsequent lines in the file, which should be equal to the number of points (vertices) in the shape specified by the “target" property of the module (“root/pc1" by default), and *numCols* is the number of columns, which should be 1 for pairwise correspondence. Every other line is of the format *idx icorr* where $idx = Line\# - 1$, and either $icorr = -1$ to indicate that this point has no correspondence, or *icorr* equals the index of the corresponding point on the shape specified by the “source" property of the module (“root/pc0" by default).

   (c) Same as step 1c

- To open an .object file in *GeoXL*, click “Open". In the pop-up window, select through the file explorer the .object file to be opened. To actually see it, double click on the file created.

- To save an .object file in *GeoXL*, click “File → Save as...". In the pop-up window, select through the file explorer the location where the .object will be stored.

3.1 Parameter settings

To view or edit the parameter settings go to *Tools* and select the module “animRecGMOD14Match". Press the button in the right of “Edit" in the “settings" field.

1. surface settings

(a) numNN: number of nearest neighbors in a k-nn graph (used for point clouds). Default value set to 8 (see paper page 8).

(b) maxSampleSpacing: maximum spacing allowed between samples. Default value set to 0.1

2. feature settings

   (a) featureMatchSource
   - internal: features are computed internally (see 1b)
   - external: sparse correspondence is provided (see 2b)

   (b) strExtSparseCorrFile: name of the file with the sparse correspondence. This file should be located in the location indicated by "m_strDirSwapPath".

3. growing settings

   (a) stretchTreshold: allowed stretching parameter (see paper pages 5 and 10).

   (b) growingKillSteps: number of steps to run each instance of isometric region growing. Simply needs to be a large value–default value will suffice in almost all cases.

   (c) numFeatureTries: maximum number of oriented point matches to be tried (see paper page 9).

4. clustering settings

   (a) numClustersMin: minimum number of clusters allowed

   (b) clusterMergeThreshold: parameter ρ as described in Section 4.4 of the paper

   (c) clusteringNumNN: sets the number of "neighbors" each partial match is connected to for graph-based cluster (eg. GDL clustering). Default value of 10 usually works.

   (d) mergeMode: set it to "cluster - GDL"

5. visualization settings These parameters are not necessary for the algorithm, but can be used for visualizing partial or full results. They can be safely ignored.

6. comparison settings These settings were used to compare to other methods for Section 5 of the paper.

   - extResType: type of external result (i.e. format according to which method)
   - strCorrFile: file name containing external result for comparison
   - bEvalExternalCorrs: flag indicating whether to evaluate external result
   - strFirstSubFolder:
   - numSubFolders:
   - strCumulativeFilename: filename to output cumulative error curve data
   - strCoverageFilename: filename to output coverage data

3.2 Output

1. partials_matches_clusters_merged.object: It contains computed correspondence information. It is stored in the path location indicated by "m_strDirSwapPath".
   (a) under "ar4result/partialMatches": the user can browse the computed partial correspondence
   (b) under "ar4result/clusteredMatches": the user can browse the computed correspondence after the clustering step as described in the paper

References

[1] `http://www.staff.uni-mainz.de/wandm/software.html`.

[2] Vlasic et al., "Articulated Mesh Animation from Multi-view Silhouettes", ACM Trans. Graph. 2008. (`http://people.csail.mit.edu/drdaniel/mesh_animation/index.html`).

[3] Brunton et al., "A Low-Dimensional Representation for Robust Partial Isometric Correspondences Computation", Graphical Models, 2014.