3D Digital Geometry Processing
Exercise 2 – Registration

Handout date: 02.06.2014

Submission deadline: 02.19.2014, 23:59 h

Note

Copying of code (either from other students or from external sources) is strictly prohibited! Any violation of this rule will lead to expulsion from the class.

What to hand in

A .zip compressed file renamed to "Exercise*n*-YourName.zip" where *n* is the number of the current exercise sheet. It should contain:

- All the source code necessary to compile your application. Please, do not hand in libraries, build folders, binaries. You only need to submit `cc` and `hh` files.
- A "readme.txt" file containing a description on how you solved each exercise (use the same numbers and titles) and the encountered problems.
- Upload the archive at `http://www.dropitto.me/usc-cs599dgp` before the deadline. Password is `ididit`.

Registration

The aim of this second exercise sheet is to implement the ICP registration algorithm to align multiple 3D scans into a common coordinate system. The overall framework for registration is given and you will need to implement several steps of the ICP algorithm, namely subsampling, bad pair rejection, and the optimization.

2.1 Getting Started

- Download the framework from Piazza.
  Note: Additionally to the libraries from the previous exercise, the framework also contains the ANN (Approximate Nearest Neighbor) library for efficient closest point lookup.
- Compile the code.
- In order to load the scans

To load the scans of a bunny, use **command line arguments**: `..\..\data\output-pts ..\..\data\bunny1.obj ..\..\data\bunny2.obj, ...`
To load the scans of a bee, use **command line arguments**: `..\..\data\output-pts ..\..\data\bee1.obj ..\..\data\bee2.obj, ...`

`output-pts` is the final aligned point cloud file name.

- When the program opens you will see the first two scans that you passed on the command line, one in grey the other in green (see Fig. 1). The green scan is the scan that will be aligned to all other grey scans.
  - By pressing 'n' you can load and select the next scan.
  - As in the previous exercise you will be able to navigate the loaded meshes with mouse controls. By pressing 'SHIFT' during mouse controls you can perform manually alignment.
  - 'h' will give you all options.
  - 'r' performs a single registration optimization step.
  - ' ' performs registration with tangential motion enabled.
  - 's' stores all vertices of the aligned scans to the output file `output-pts`. These points will be used in the next exercise for reconstructing one seamless mesh.
- The viewer and all registration steps are controlled from the `RegistrationViewer` class. In there you will find the `keyboard(...)` function which handles all key input. Pressing 'r' and ' ' will call `perform_registration(BOOL)` to do a single registration step aligning the current scans to all other scans. First, correspondences are calculated by calling `calculate_correspondences(...)` with subsampling, closest point search, and bad pair rejection. The correspondences are then used for registration using `register_point2point(...)` and `register_point2surface(...)`.

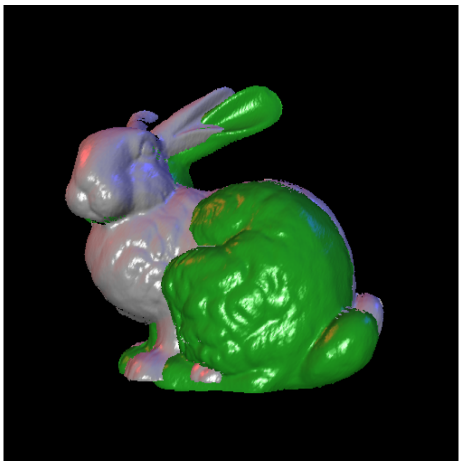

Figure 1: Screenshot of the registration viewer with the first two scans.

2.2 Subsampling

Appropriately subsampling the scan is the first step for ICP. In this case you will implement uniform subsampling with a given sampling radius. Adapt `subsample()` to store the indices of all samples. The input is a list of `Vector3d` which is a simple vector class declared in `Vector.hh`. Note that this method may be speed up by taking advantage of the fact that consecutive points are often close on the mesh. You can see the subsampled points by pressing 'r' or ' ', and should be similar to Fig.2.

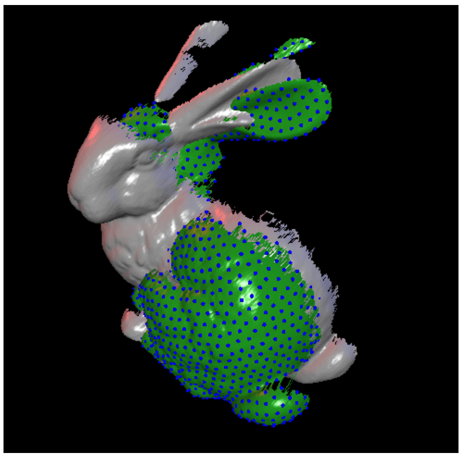

Figure 2: Uniform sampling of the scan.

2.3 Bad Pairs Rejection

Correspondences are calculated in `calculate_correspondences(...)` using a KD-tree lookup based on the ANN library.

The quality of the registration, however, depends on the goodness of the correspondences. Therefore strategies to remove bad pairs need to be employed. Pairs at mesh boundaries will already be removed in the framework. In this task you will further need to remove pairs that have a distance larger than three times the median correspondence distance, and pairs whose normals are not compatible, i.e. the angle between normals is larger than a given threshold.

2.4 Point-2-Point Optimization

Registration based on minimizing the distance of correspondences in the least squares sense is performed in `register_point2point` in class `Registration`.

Instead of using a closed-form solution we solve the registration by linearizing the rotation matrix **R**. A rotation matrix is expressed by the three Euler angles α, β, and γ:

$$\mathbf{R} = \begin{bmatrix} \cos\beta\cos\gamma & -\cos\beta\sin\gamma & \sin\beta \\ \sin\alpha\sin\beta\cos\gamma + \cos\alpha\sin\gamma & -\sin\alpha\sin\beta\sin\gamma + \cos\alpha\cos\gamma & -\sin\alpha\cos\beta \\ -\cos\alpha\sin\beta\cos\gamma + \sin\alpha\sin\gamma & \cos\alpha\sin\beta\sin\gamma + \sin\alpha\cos\gamma & \cos\alpha\cos\beta \end{bmatrix} \quad (1)$$

The matrix may be linearized by assuming that we only have a small rotation and therefore $\sin x = x$ and $\cos x = 1$.

The actual optimization is performed by minimizing the distance of pairwise correspondences $\mathbf{p}_i$ and $\mathbf{q}_i$ in the least square sense:

$$E = \sum_{i=1}^{N} \|\mathbf{R}\mathbf{p}_i + \mathbf{t} - \mathbf{q}_i\|_2^2 \tag{2}$$

where $\mathbf{R}$ is the linearized rotation matrix and $\mathbf{t}$ the translation vector. The optimization can therefore be posed as overdetermined linear system and solved using normal equations:

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{3}$$

In this task you will need to set up matrix $\mathbf{A}$ and vector $\mathbf{b}$. The first three components of $\mathbf{x}$ are the three Euler angles $x[0] = \alpha$, $x[1] = \beta$, and $x[2] = \gamma$, and the other three components are the translation vector $\mathbf{t}$ ($x[3] = \mathbf{t}_x$, $x[4] = \mathbf{t}_y$, $x[5] = \mathbf{t}_z$). Note that solution to $\mathbf{A}\mathbf{x} = \mathbf{b}$ is then calculated automatically using the normal equations and the corresponding rotation matrix and translation vector are extracted.
Press 'r' multiple times to perform registration which should result in an alignment similar to Fig. 3.

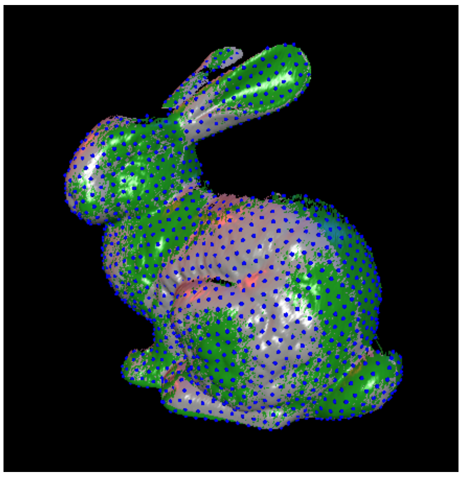
Figure 3: Registered scans.

2.5 Point-2-Surface Optimization

Point-2-point optimization converges rather slowly as it does not allow for motion along the surface. Therefore a second optimization strategy is to optimize the distance of each scan vertex to the tangential plane defined at the target correspondence. The corresponding tangential surface of vertex $\mathbf{p}_i$ is defined by the correspondence $\mathbf{q}_\mathrm{i}$ and its normal $\mathbf{n}_\mathrm{i}$. Minimizing this point-2-surface distance in the least square sense is again a least squares solution:

$$E = \sum_{i=1}^{N} \|\mathbf{n}_i^\top (\mathbf{R}\mathbf{p}_i + \mathbf{t} - \mathbf{q}_i)\|_2^2 \tag{4}$$

Your task is again to set up matrix $\mathbf{A}$ and vector $\mathbf{b}$. You will see that registration will converge very quickly by pressing ' '.
When you have accomplished all tasks you can register all scans by first doing a coarse manual alignment, pressing ' ' multiple times for a fine alignment, and then choosing the next scan by pressing 'n'. Each scan will be registered against all other scans. You

should therefore loop through all scans multiple times. At the end you should have a result similar to Fig. 4. You can store your result by expressing ′s′.

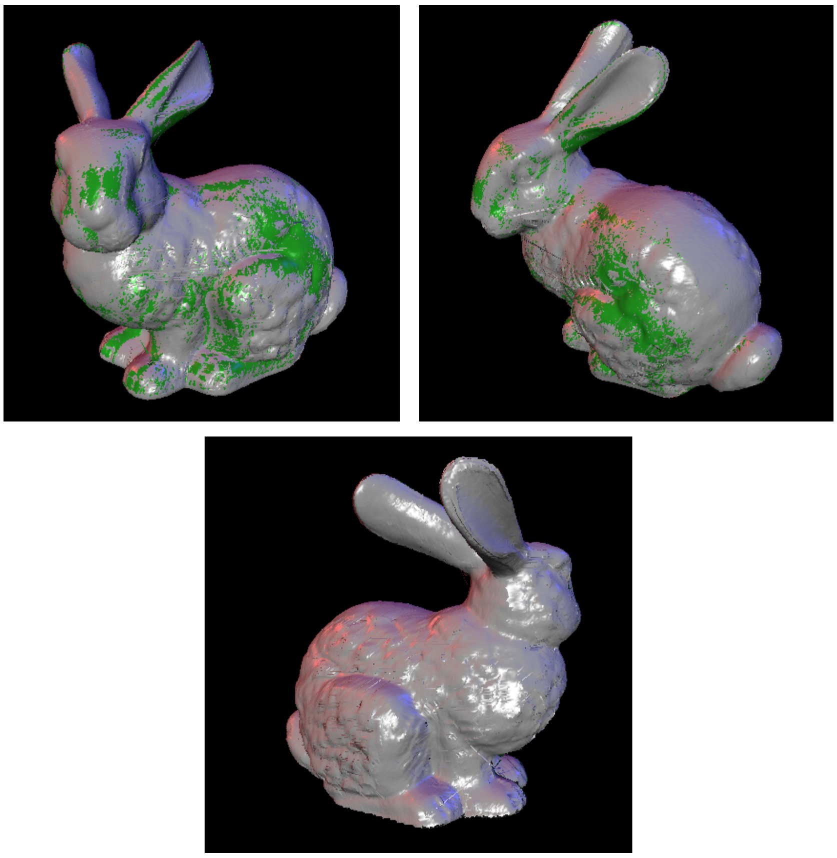

Figure 4: Final alignment of all scans.