25 November 2020

General Information:

You have to submit your solution via Moodle. We allow **groups of two students**. Please list all names and matriculation numbers in the **team_members.txt** file and upload the solution **individually**. You have **one week of working time** (note the deadline date in the footer). To get the 0.3 bonus, you have to pass at least four exercises, with the fifth one being either completely correct or borderline accepted. The solutions of the exercises are presented the day after the submission deadline respectively. Feel free to use the Moodle forum or schedule a Zoom meeting to ask questions!

To inspect your result, you need a 3D file viewer like **MeshLab** (http://www.meshlab.net/). The exercise zip-archive contains the source templates, a CMake configuration file and the required data. To get the Eigen library, refer to exercise 1.

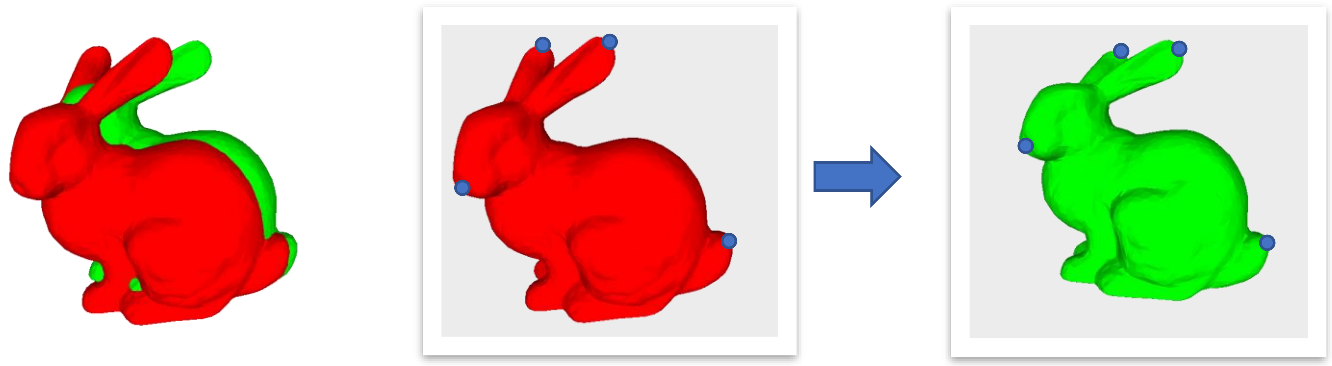
Expected submission files: ProcrustesAligner.h, bunny_procrustes.off

Exercise 3 – Coarse Alignment – Procrustes

In this exercise we want to align 3D shapes. We will explore the Procrustes algorithms and test it on a set of two shapes with sparse correspondences provided.

Tasks:

1. Coarse Registration – Procrustes Algorithm



In this task you have to implement the Procrustes Algorithm to align two bunny meshes (red & green). Exploiting the sparse point correspondences shown as blue dots, we are able to estimate the transformation T that transforms the red bunny to match the green bunny.

The sparse point correspondences were extracted by picking similar points using MeshLab, and are provided in the code. Your task is to fill in the missing parts of the algorithm in the ProcrustesAligner.h. The resulting mesh bunny_procrustes.off should almost perfectly align the green and red bunny. To check your result, you can open the meshes in the same MeshLab window.

25 November 2020

Remarks

We are given a source shape X, a target shape Y, and point matches (x_i, y_i), such that $x_i \in X$ and $y_i \in Y$. Our goal is to estimate a rigid pose (R, t) that transforms source points x_i into target points y_i as $y_i = Rx_i + t$. We show two methods of computing the pose (R, t).

First Approach

With the Procrustes algorithm we compute the rotation between mean-centered points, i.e. we estimate a rotation matrix R that should satisfy the following equation for every point pair (x_i, y_i):

$$R(x_i - \underline{x}) = y_i - \underline{y}\#(1)$$

On the other hand, we compute the translation between the means of both sets of points:

$$\underline{t} = \underline{y} - \underline{x}\#(2)$$

From equations 1 and 2 we get:

$$y_i = R(x_i - \underline{x}) + \underline{y} = R(x_i - \underline{x}) + (\underline{t} + \underline{x}) = Rx_i + (-R\underline{x} + \underline{t} + \underline{x})\#(3)$$

It follows that $\tilde{R} = R$ and $t = -R\underline{x} + \underline{t} + \underline{x} = -R\underline{x} + \underline{y}$.

Second Approach

The second method to achieve the same result is to apply the Procrustes algorithm in two steps. To transform a source point x_i to a target point y_i, we first apply a translation vector t, resulting in a translated point $x' = x_i + \underline{t}$. In the next step we apply the rotation R, but since the Procrustes algorithm computes rotation between mean-centered points, we can only apply the rotation on the mean-centered point $x' - \underline{y}$, and after rotation we add the mean $\underline{y}$ back. That results in the final transformed point:

$$y_i = R\left(x' - \underline{y}\right) + \underline{y} = R\left(x_i + \underline{t} - \underline{y}\right) + \underline{y} = Rx_i + \left(R\underline{t} - R\underline{y} + \underline{y}\right)\#(4)$$

It follows that $\tilde{R} = R$ and $t = R\underline{t} - R\underline{y} + \underline{y}$.

Result Equivalence

It is easy to prove that the results are equivalent. We can transform from the translation in the first approach to the translation in the second approach by using equation 2:

$$-R\underline{x} + \underline{t} + \underline{x} = -R\left(\underline{y} - \underline{t}\right) + \left(\underline{y} - \underline{x}\right) + \underline{x} = R\underline{t} - R\underline{y} + \underline{y} \# (5)$$

2. Submit your solution

a. Upload the resulting mesh: bunny_procrustes.off
b. Submit the following file: ProcrustesAligner.h