23 October 2019

General Information:

You have to submit your solution via Moodle. We allow **groups of two students**. Please list all names in the submission comments, upload one solution per group. 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 visit us during office hours (every Monday 14:00 – 15:00) 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 libraries.

If you work under Linux you have to install the freeimage lib using:

```
$ sudo apt-get install libfreeimage3 libfreeimage-dev
```

Expected submission files: main.cpp, screenshot.png

Exercise 1 – Camera Intrinsics, Back-projection, Meshes

In this exercise we want to generate 3D meshes from depth and color maps which can be captured by camera like a Microsoft Kinect. We provide a basic virtual sensor class that reads the camera data from image files (see VirtualSensor.h). Your task is to project the depth from a single frame to 3D world space and to save it as a 3D mesh with the corresponding colors of the color map. You only have to modify “main.cpp”. Follow the comments in this source file.

You need to download the recorded camera data from the **TUM RGB-D SLAM Dataset** (https://vision.in.tum.de/data/datasets/rgbd-dataset). Use the Freiburg 1 dataset “fr1/xyz”, extract the tgz archive to the data folder (make sure the variable filenameIn in main() is set respectively).

23 October 2019

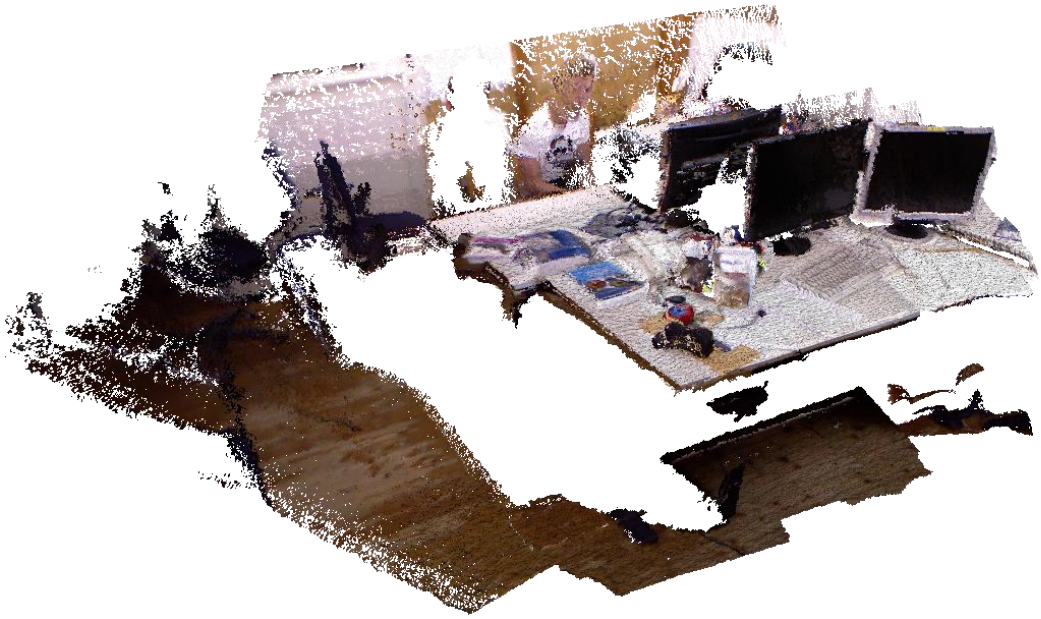

Figure 1: Multiple aligned reconstructed 3D meshes of the TUM RGB-D SLAM Dataset.

The Kinect sensor consists of two cameras (see figure to the right); one camera captures color and the other one is used to reconstruct the depth (IR camera). The VirtualSensor class gives you access to both intrinsics and both extrinsics. Note that the dataset already contains aligned color and depth data, thus, the intrinsics/extrinsics of the color and depth camera can be assumed to be the same.

The intrinsics of the cameras have the following form:

$$Intrinsics = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}$$

All parameters are in pixel units!

In addition the VirtualSensor class gives you access to the trajectory of the Kinect sensor (GetTrajectory()). Have a look at the dataset website to know where this trajectory comes from.

23 October 2019

Tasks:

1. Back-Projection

a. Use the intrinsics of the depth camera to back-project the pixels of the depth map to the camera space of the depth camera.
b. Transform the back-projected 3D points to world space
   - use the inverse of the extrinsic transformation matrix of the depth camera to transform the points to the sensor camera space (note that this task is optional since the extrinsics of the color and the depth camera are the identity in this dataset).
   - use the provided trajectory to transform the points to world space.
c. Assign the color information to the 3D world space vertices.

2. Write a 3D mesh

a. Write a mesh that uses the topology of the pixel grid and the 3D points in world space including per vertex colors (use the OFF file format; there is a sample file "off_sample.off").
   - Only use triangles with valid depth values, and an edge length smaller than the given threshold (edgeThreshold).

3. Submit your solution

a. Make a nice snapshot of a reconstructed mesh similar to Fig.1 using MeshLab (file → save snapshot).
b. Submit your **main.cpp** and the **snapshot** via Moodle.