Lab 1: RGB-D alignment and odometry

Javier Civera
jcivera@unizar.es

Objectives

1. Analyze the main characteristics and limitations of methods for RGB-D alignment using the functions implemented in Open3D
2. Evaluate the accuracy of a simple RGB-D odometry implemented with the functions in Open3D

Results

1 Set up your work environnment

In this practical we will use the library Open3D, `http://www.open3d.org/`. Open3D is an open-source library for 3D data processing that, among others, contains implementations for point cloud alignment and RGB-D odometry.

It also has an excellent documentation of its functionalities. To install it, follow the instructions in the first links of the documentation (`http://www.open3d.org/docs/release/introduction.html` and `http://www.open3d.org/docs/release/getting_started.html`). You will find an index to the documentation here `http://www.open3d.org/docs/release/`.

If you do not want to install it in your local computer, you can use Google Colab. The only point that might be problematic in this case is the 3D visualization of the point clouds, for which you have a working solution in this notebook:

`https://drive.google.com/file/d/1e3hHRS7eIFVSTyfP3bEwDlT_3XI7jUEx`.

You also have in the notebook of the link the functions to read the ground truth camera poses from a text file.

2 Download test sequences

Download one of the datasets of the Redwood collection (`http://redwood-data.org/`), for example the Indoor sequence (`http://redwood-data.org/indoor/dataset.html`). Notice that, apart from the image sequences, you also have the ground truth camera poses.

3 Familiarize with basic functions before starting

For this lab session, we will need to read point clouds and RGB-D frames. Follow the corresponding items in the tutorial, find how to read RGB-D images and transform it to point clouds in the follow-

ing part of the tutorial http://www.open3d.org/docs/release/tutorial/geometry/rgbd_image.html. And find here how to visualize point clouds http://www.open3d.org/docs/release/tutorial/geometry/pointcloud.html (remember that it will not work in Google Colab, you have an alternative visualization here[1] if you use Colab). The most relevant parts for the lab session are the first three, "Visualize point cloud", "Voxel downsampling" and "Vertex normal estimation".

4 Evaluate errors and limitations of ICP

1. Follow this tutorial http://www.open3d.org/docs/release/tutorial/pipelines/icp_registration.html to get an example on how to use the ICP implementation of Open3D.

2. Take two consecutive RGB-D images of the Redwood sequence, transform them into point clouds and run ICP with the identity transformation as the initial seed. Remember that, for using the point-to-plane distance, the normals of the point cloud should be estimated before, as you did in the previous section.

3. Compute the error of the estimated pose. Observe that the ICP function will return the estimated relative transformation $\hat{\mathbf{T}}_{ts}$, and the ground truth files contain absolute poses, e.g., $\mathbf{T}_{ws}$ and $\mathbf{T}_{wt}$. In order to obtain the ground truth relative transformation, you will need to compose the absolute ones $\mathbf{T}_{ts} = \mathbf{T}_{wt}^{-1}\mathbf{T}_{ws}$. And then, in order to get the transformation error, you can do $\mathbf{T}_e = \hat{\mathbf{T}}_{ts}^{-1}\mathbf{T}_{ts}$. As this transformation error might be hard to understand, report the norm of the translation error and the norm of the rotation error in Euler angles. In the shared Colab notebook you have functions to read the ground truth poses from the text files of the dataset.

4. Compare the errors using the point-to-point and the point-to-plane distances. Which one is better? Note that the error depends on the RGB-D pair that you choose, so take many and present aggregated results.

5. Take two more separated RGB-D images and try to align them again using ICP. Increase the separation of the images gradually and try with several pairs. Observe how the error of the estimated transformation grows. Present your results qualitatively (e.g., in a graph) and quantitatively.

5 Feature matching and alignment

1. In this part of the tutorial http://www.open3d.org/docs/release/tutorial/pipelines/global_registration.html there is an example on how to use feature matching + RANSAC alignment.

2. Align two close images and two separated images using feature matching + RANSAC. Is the estimation error bigger or smaller than for ICP?

3. Use feature matching + RANSAC to obtain a pre-alignment of two point clouds that are separated. After that, refine the alignment using ICP. Report cases where feature matching helps to reduce the alignment error of two RGB-D images. Report also if ICP refinement improves the error of feature-matching + RANSAC.

[1] https://drive.google.com/file/d/1Dm8lnZR43Ris7WaHVOL-Hdxtq7KCWO0I

6 RGB-D Odometry

Implement a simple RGB-D odometry with the blocks you used before.

1. First, use the example code in the notebook [2] to read the RGB and depth images from the files.
2. For each image, estimate the relative transformation with respect to the previous one using ICP.
3. Concatenate the relative transformations to obtain the transformation of each frame with respect to the first one. The first frame will be taken as the global reference frame.
4. Report translation and rotation errors. Observe how the drift grows with the number of images. Does this ICP-based odometry fail at some particular frame?

[2] https://drive.google.com/file/d/1Dm8lnZR43Ris7WaHVOL-Hdxtq7KCWO0I