Mapping Building Outlines to 3D Point Clouds

Anurag Sai Vempati
Autonomus Systems Lab
ETH Zurich
firstauthor@i1.org

Wolf Vollprecht
wolfv@student.ethz.ch

Abstract

While point clouds are easy to obtain through processes such as Structure-From-Motion they only exist in a space for themselves. GPS coordinates, recorded at capture time of the images, can give a sparse and noisy reference to the true position of the point cloud. In this paper we present a method to register a point cloud on reference map data.

1. Introduction

With current state of the art technology, obtaining point clouds through structure from motion has become an easy task. Several attempts deal with a similar problem, such as ...

2. Obtaining the ground truth

images/OSM_Mashup.png

Figure 1. OSM overlaid with editor interface

In order to obtain the ground truth building outlines, we used OpenStreetMaps[1], a collaborative mapping effort that takes place globally. Volunteers are mapping their surroundings and upload it to a central database, where all changes to the map are stored. Of concern for us is only one mapped entity, the building. All points in OSM are stored as nodes. The entity consists of one or more node (in the case of a building it's multiple nodes). By parsing the XML response and mapping all node values to the corresponding building entity, we can obtain all corner points of the closed polygon that describes a building outline in the real world.

In order to adjust the point cloud to the obtained ground truth, we discretized the polygons to point clouds themselves by creating points along the polygon edges in a certain distance. During the Iterative Closest Point matching these points will be matched against the point cloud and provide an error distance.

The OSM data is in a coordinate format consisting of a latitude/longitude pair. This representation maps coordinates to the Globe, which has a spherical shape. However, we would like to work on a 2D surface. The size of the registration is reasonably small to obtain a 2D representation that falls into the boundaries of acceptable error.

There are a number of methods on how to convert coordinates to a 2D representation, which is a topic that has been explored for a long time since large-scale mapping has taken place globally. An exhaustive overview can be found in Snyder (1987) [2]. The methods can be divided into two different categories:

- **Conformal** This category of mappings preserves angles as observed in the real world (ie. the local angles are preserved).
- **Equiareal** Retains equal area
- **Equidistant** Preserving distance
- **Azimuthal** Preserving Direction
- **Shortest Route** Shortest route is observable

[1] http://www.openstreetmap.org/

Since the sphere cannot be flattened out without distortion (ie. the sphere is a non-developable surface) not all of the above properties can hold for a projection at the same time.

For our application of registering a 2D point cloud on the map, we chose to use the Universal Transverse Mercator (UTM) projection, which preserves the angles and shapes, a property we deemed useful for our application.

Contrary to other map projections, UTM is not a single projection but rather divides the globe into 60 different zones.

3. Registering the Point Cloud

After obtaining the ground truth and the 2D projection of the 3D cloud we are now able to register the point cloud on the OSM reference.

Therefore we use "Iterative Closest Point" (ICP) matching algorithm. As the name suggest, ICP is an iterative algorithm to minimize the distance between two (or more) point cloud measurements. ICP is a widely used algorithm for example in robotics (Simultaneous Localization and Mapping [SLAM]).

The input to the ICP algorithm in general is two point-clouds, one called the reference and the other source. The source should be aligned to the reference cloud. The output of the algorithm is a transformation matrix in 2 or 3 dimensions. Furthermore, the algorithm needs to be supplied with a stop condition, for example a maximum number of iterations or a minimal size of change between iterations, which indicates that the algorithm has converged to a minima.

Several important drawbacks of ICP are that it usually only provides rigid transformations (i.e. scale and shear factors are not affected). Allowing infinite scale the ICP solution would scale down to only one point with a distance of zero, as all points would be incident with that one. Another problem is that ICP, without good initialisation, tends to convertge to a local minima. Therefore it is paramount to have an initial rotation and translation that matches the reference somewhat.

For our implementation, the existing `libpointmatcher`[1] library was utilized to provide a framework for customizable ICP algorithms.

Both the map data as well as the point cloud is normalized and scaled to an arbitray scale, starting, so that the left-most point is coincident with the (0, 0) coordinate.

3.1. ICP process

As implemented, the ICP implementation uses a KD-Tree with a variable number of neighbors it considers as potential neighbors. The nearest neigbor is accepted as neighbor and a transformation is searched that decreases the distance (error) between all neighbors.

Generally, there are two different popular ICP variants: Point-to-Point and Point-to-Plane, with the latter usually performing better. However, since we are operating in the 2D space, there are no planes except for the ground plane to be found. Thus we decided to use the more approachable point-to-point algorithm.

One ICP iteration consists of 4 steps:

- We search a KD-Tree structure for the nearest neighbors and weight or reject them depending on the distance
- The error is calculated
- Using the error, a singular value decomposition is calculated
- The rotation matrix is $\mathbf{R} = \mathbf{U} * \mathbf{V}^\intercal$
  The translation vector is $\mathbf{T} = \mathbf{A} - \mathbf{R} * \mathbf{B}$
- Repeat process until either only a very small change between iterations is observed or the error is sufficiently small

3.2. Filtering data

To further reduce the amount of noise, we implemented a filtering chain, consisting of

- Any given point needs at least 50 neighbors in a radius of 2 meters to be considered a valid measurement
- Statistical outliers are removed with a mean of 8 and a standard deviation of 1. This further removes outliers
- Only 1/8 of all remaining points are randomly chosen to reduce the computational load

3.3. Initial alignment

As stated before, the initial alignment of the two point-clouds is crucial for successful ICP matching. Therefore we compute the bounding boxes of the different structures found in the point cloud. We then proceed to align the bounding boxes as good as possible by scaling, shearing, rotating and translating.

To obtain the bounding boxes from the denoised measurements we

- Traverse the KD tree and group structures by neighbors in a certain threshold
- Calculate bounding box and shrink inward slightly to match measurements better
-

References

[1] F. Pomerleau, F. Colas, R. Siegwart, and S. Magnenat. Comparing ICP Variants on Real-World Data Sets. *Autonomous Robots*, 34(3):133–148, Feb. 2013. 2

[2] J. P. Snyder. Map projections: A working manual. Technical report, 1987. Report. 1