3D Camera Localization - Probabilistic Robotics

Francesco Chichi

September 19, 2017

1 Introduction

This project aims to reconstruct the poses of a vehicle, with a Least-Squares approach, using robot's odometry and anonymized observations. Via G2O simulator, we obtain a landmarks based map, the robot's perceptions and the transition of the robot between two poses (the odometry).

2 Implementation

2.1 Data Structure

All the project's data structures are parsed from the G2O file by the *g2o_parser* class.

- **Landmark:** This object store the position of a landmark and his ID.
- **Pose:** Used to represent a camera pose and his ID.
- **Transition:** This structure represent the transition of the robot from pose A to B. In it are stored the two poses and their relative ID.
- **Observation:** This kind of data structure is used to represent the robot's observations. It contains the pose's ID, a vector of 2d landmarks and two vector with the relative ID and depth of each one.
- **DictPoints:** This dictionary is used to store match of 2D and 3D projection of a landmark.

2.2 Camera

This class is used to represent a camera, initialized with the resolution (rows and columns), the camera matrix and the initial position in the space. It contains the two principal methods *projectPoints* and *unprojectPoint*, used to transform a point from 2D to 3D and vice versa.

2.3 Distance Map

A distance map approach is used in order to compute the data associations. After each transition, a distance map is computed using the G2O's observations as reference and are computed the corrispondences with the projections of the landmarks observed.

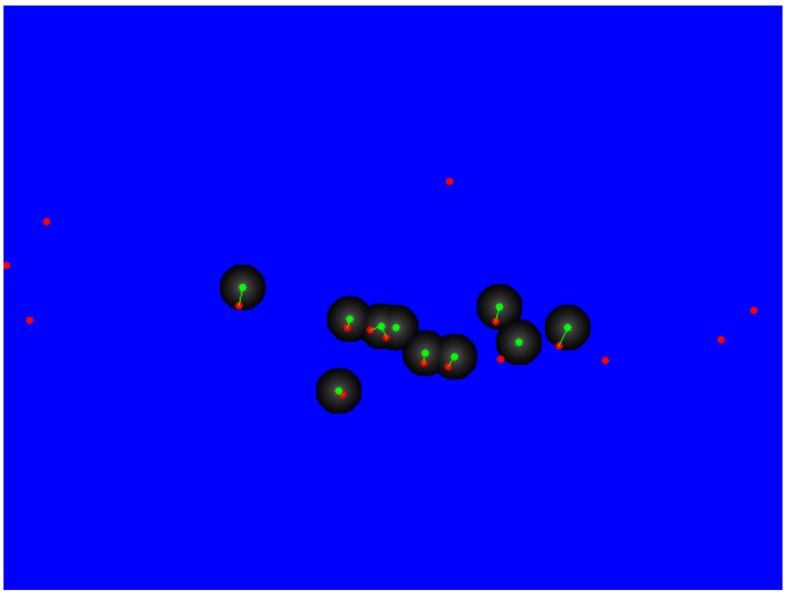

Figure 1: The green points are observed points while the red are the projected landmarks.

2.4 Main

The main class is used to parse the data coming out from the G2O simulation into the data structures of the project. Once the data are completely parsed, for each transition of the robot, the landmarks are filtered and only those within a range are stored and used to compute the data association, in order to correct, using the least squares algorithm, the predicted position of the camera.

3 Least-Squares Approach

3.1 State

In the state there are the position of the camera in the space, represented as a rotation matrix and a transition vector:

$$\mathrm{X} \in SE(3) : \mathrm{X} = (\mathrm{R}|\mathrm{t})$$

The increments $\Delta\mathrm{x}$ is represented as a $\mathbb{R}^6$ vector: the position x,y,z and the rotations among the three axes α_x,α_y,α_z.

3.2 Measurement

The measurements are the projections of the landmarks in the camera.

$$z^{[m]} \in \mathbb{R}^2 : \mathbf{z}^{[m]} = (u^{[m]} v^{[m]})^T$$

3.3 Prediction

The prediction of the camera pose at time t is obtained applying the odometry transition at time t to the pose *t-1 (current_camera_pose* $\times$ *cameraToRobot* $\times$ *motion* $\times$ *robotToCamera).*

$$h^{[n]} = proj(KX^{-1}p^{[n]}).$$

3.4 Error

The error is computed as the difference between the projected landmark, using the predicted pose, and the observed one:

$$e^{[n,m]} = h^{[n]} - z^{[m]}$$

3.5 Jacobian

For each landmark, the column i of the 2×6 jacobian is computed using the formula of the numerical differentiation:

$$\frac{f(i+\epsilon)-f(i-\epsilon)}{2\epsilon}.$$

For each landmark the procedure is the same: at the i-th element of the state **X** are summed an ϵ, then is projected the point using a camera in the modified state, then an error **ep** is computed as the difference between this projected point and the reference one ($z^{[m]}$);
The same procedure is applied subtracting the ϵ, computing in the same way **em**.
Finally the element of the state is computed as $\frac{ep-em}{2\epsilon}$

3.6 Correction

The correction follows the least-squares procedure, for each least-squares iteration:

$$\begin{gathered} \mathrm{H} \leftarrow \mathrm{H} + J_i^T\ \Omega_i\ J_i \\ \mathrm{b} \leftarrow \mathrm{b} + J_i^T\ \Omega_i\ e_i \\ \Delta x \leftarrow solve(H\Delta x = -b) \\ X^* \leftarrow X^* \boxplus \Delta x \end{gathered}$$

4 Front-End

In the *Test.h* class is implemented in openCV a GUI that allows the user to visualize and navigate throw Landmarks, camera poses and transitions (figure 2).

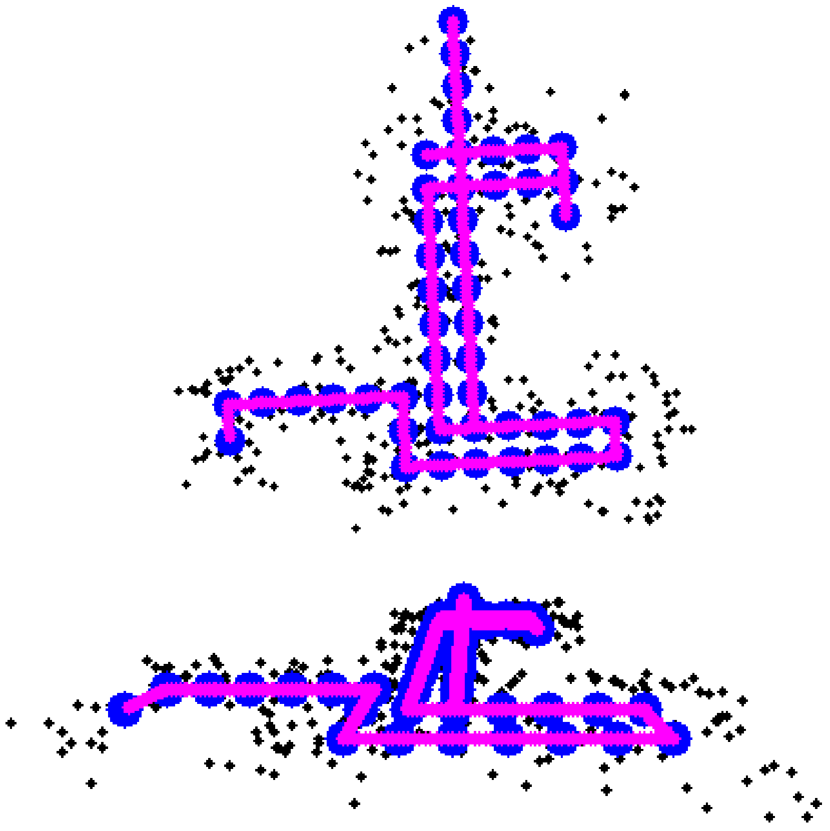

Figure 2: The blue points are camera poses, the black ones are the landmarks and the purple are represented the transition between two poses.

5 Experimental Results

Unfortunately, the purpose of the project has not been achieved.
An initially obstacle fixed was the ideal camera matrix K_i gived by G2O:

$$K_i = \begin{bmatrix} 1 & 0 & 0.5 \\ 0 & 1 & 0.5 \\ 0 & 0 & 1 \end{bmatrix}$$

In order to visualize the point in the GUI, all the projected landmarks in the observations are unprojected using K_i and reprojected using a camera matrix K_r that have a camera focal lengths scaled by a factor $scale$=150 and the optical centers centered respect to the resolution of the image:

$$K_r = \begin{bmatrix} 1 * scale & 0 & 0.5 * cols \\ 0 & 1 * scale & 0.5 * rows \\ 0 & 0 & 1 \end{bmatrix}$$

Another problem fixed concerns the project "*Projective Registration with unknown Associations*" seen during the lessons, where the camera is fixed at the world's center, so trasformation worldToCamera is always an identity. In this way there are two big difference with this project:

- Since that worldToCamera is always an identity, the J_{icp} part of the jacobian becomes: $J_{icp}^{[n]} = (I_{3\times3} | \lfloor -\hat{p}^{[n]} \rfloor_\times)$. In the project the entire jacobian is computed numerically.
- The projection used in the observation function uses the state X, $h^{[n]} = proj(KXp^{[n]})$, because the origin of the world coincide with the camera position. In this project this semplification couldn't be done, so: $h^{[n]} = proj(KX^{-1}.p^{[n]})$.