Exercise Sheet 1

Anselm Coogan

<2019-10-22 Tue>

1 Exercise 1: What is SLAM

1.1 Why would a SLAM system need a map?

- we need SLAM in applications where a prior map is not available but the robot needs to be aware of its surroundings and relative position
- can inform path planning
- provides intuitive visualization for humans

1.2 How can we apply SLAM technology into real-world applications?

- SLAM is useful for military applications, agriculture, autonmous driving, cleaning/service robots
- to apply SLAM you need a set of sensors (LIDAR, depth, RGB, DVS, etc.) which are then processed by the system's front-end. This front-end interprets the sensor information (fusion, segmentation, filtering, feature tracking, etc.). It is also responsible for loop closure. The processed sensory data is then sent to the back-end part of the system that usually performs MAP estimation finally resulting in a SLAM estimate.

1.3 Describe the history of SLAM

1.3.1 The classical age (1984-2004)

- introduction of probabilistic SLAM formulations: MAP, Extended Kalman Filter, Rao-Blackwellised Particle Filters
- decoupling of main challenges: efficiency and robust data association

1.3.2 Algorithmic analysis age (2004-2015)

- study of SLAM fundamentals: observability, convergence, and consistency
- development of main open source libraries for SLAM (Ceres, GTSAM, g2o, iSAM, SLAM++)
- using sparsity to create efficient SLAM solvers

1.3.3 Robust perception age (2015-present)

- SLAM "solved" for specific conditions (e.g. using LIDAR to map an indoor 2D non-dynamic environment with a slow-moving robot)
- actively researched/unsolved problems:
  - robust performance (in the face of outliers, changing maps, non-rigid world, degredation of sensors)
  - high-level understanding (semantic maps, dynamic vs static parts)
  - resource awareness (dealing with influx in CPU stress, taking advantage of idle processor)
  - task-driven perception (filter non-relevant perceptual information, produce adaptive map representations)

2 Exercise 2: git, cmake, gcc, merge-requests

- Prepends the path where cmake will look for modules with the **include** and **find_package** commands with the current dir (.) and the subdirectory **cmake_modules**
- CMAKE_CXX_STANDARD
  - sets the optional property for the build to use the appropriate compiler and linker flags for the C++11 standard to use the appropriate flags for C++ 11. If a newer compiler/linker is available it will be used (i.e. 14 or 17). If only older versions are available, it will fallback to those.
  - Disables the decay to older C++ standards. If C++11 or newer is not available, compilation will fail.

  - Forces the build to use standard C++11 dialect and not an extended variant of this (such as gnu++11). Having the property set to ON (default) may result in cross-platform build bugs

- CMAKE_CXX_FLAGS Sets global flags and specific flags for specific build targets (specified through e.g. `cmake -DCMAKE_TYPE=DEBUG ..`)
  - O0: compiles without optimization
  - O3: compiles with full optimization
  - g: compiles with GDB debugging enabled
  - EIGEN_INITIALIZE_MATRICES_BY_NAN: initializes matrices with NaN values, makes it easier to find uninitialized matrices when debugging
  - NDEBUG: disables assert statements in code
  - ftemplate-backtrace-limit=0: disables the limit of instantiation notes for a single error (default is 10)
  - Wall: enables all warnings during build time
  - EXTRA_WARNING_FLAGS: set of flags defined previously to account for different compilers (gcc and clang) e.g.: Wsign-compare (warning generated when signed and unsigned values are compared), Wno-exceptions (disables clang enabled exception warnings)
  - march: is set to native so that compiler generates instructions for the build system's CPU

3 Exercise 3: SO(3) and SE(3) Lie groups

Theorem:

Show that

$$exp(\xi^{\wedge}) = \begin{pmatrix} \sum_{n=0}^{\infty} \frac{1}{n!}(\phi^{\wedge})^n & \sum_{n=0}^{\infty} \frac{1}{(n+1)!}(\phi^{\wedge})^n \rho \\ 0^T & 1 \end{pmatrix}$$

where

$$\xi^{\wedge} = \begin{pmatrix} \phi^{\wedge} & \rho \\ 0^T & 0 \end{pmatrix}$$

is a 4x4 matrix and show that

$$\sum_{n=0}^{\infty}\frac{1}{(n+1)!}(\phi^\wedge)^n = \frac{sin(\theta)}{\theta}I_{(3,3)} + (1-\frac{sin(\theta)}{\theta})aa^T + \frac{1-cos(\theta)}{\theta}a^\wedge = J$$

Proof:

$$exp(\xi^\wedge) = \sum_{n=0}^{\infty}\frac{1}{n!}(\xi^\wedge)^n$$

$$\begin{aligned}
&= \sum_{n=0}^{\infty}\frac{1}{n!}\begin{pmatrix}\phi^\wedge & \rho\\ 0^T & 0\end{pmatrix}^n\\
&= I_{(4,4)} + \begin{pmatrix}\phi^\wedge & \rho\\ 0^T & 0\end{pmatrix}\\
&+ \frac{1}{2}\begin{pmatrix}\phi^\wedge & \rho\\ 0^T & 0\end{pmatrix}\begin{pmatrix}\phi^\wedge & \rho\\ 0^T & 0\end{pmatrix}\\
&+ \frac{1}{6}\begin{pmatrix}\phi^\wedge & \rho\\ 0^T & 0\end{pmatrix}\begin{pmatrix}\phi^\wedge & \rho\\ 0^T & 0\end{pmatrix}\begin{pmatrix}\phi^\wedge & \rho\\ 0^T & 0\end{pmatrix}\\
&+ \dots\\
&= I_{(4,4)} + \begin{pmatrix}\phi^\wedge & \rho\\ 0^T & 0\end{pmatrix}\\
&+ \frac{1}{2}\begin{pmatrix}(\phi^\wedge)^2 & \phi^\wedge\rho\\ 0^T & 0\end{pmatrix}\\
&+ \frac{1}{6}\begin{pmatrix}(\phi^\wedge)^3 & (\phi^\wedge)^2\rho\\ 0^T & 0\end{pmatrix}\\
&+ \dots\\
&= I_{(4,4)} + \sum_{n=1}^{\infty}\frac{1}{n!}\begin{pmatrix}(\phi^\wedge)^n & (\phi^\wedge)^{(n-1)}\rho\\ 0^T & 0\end{pmatrix}\\
&= I_{(4,4)} + \begin{pmatrix}\sum_{n=1}^{\infty}\frac{1}{n!}(\phi^\wedge)^n & \sum_{n=1}^{\infty}\frac{1}{n!}(\phi^\wedge)^{(n-1)}\rho\\ 0^T & 0\end{pmatrix}\\
&= \begin{pmatrix}I_{(3,3)} + \sum_{n=1}^{\infty}\frac{1}{n!}(\phi^\wedge)^n & \sum_{n=1}^{\infty}\frac{1}{n!}(\phi^\wedge)^{(n-1)}\rho\\ 0^T & 1\end{pmatrix}\\
&= \begin{pmatrix}\sum_{n=0}^{\infty}\frac{1}{n!}(\phi^\wedge)^n & \sum_{n=0}^{\infty}\frac{1}{(n+1)!}(\phi^\wedge)^n\rho\\ 0^T & 1\end{pmatrix}
\end{aligned}$$

Now, the second part of the derivation:

$$\sum_{n=0}^{\infty} \frac{1}{(n+1)!}(\phi^\wedge)^n =$$

$$\begin{aligned}
& I_{(3,3)} + \frac{1}{2!}\theta a^\wedge + \frac{1}{3!}\theta^2 a^\wedge a^\wedge + \frac{1}{4!}\theta^3 a^\wedge a^\wedge a^\wedge + \dots \\
&= aa^T - a^\wedge a^\wedge + \frac{1}{2!}\theta a^\wedge + \\
&+ \frac{1}{3!}\theta^2 a^\wedge a^\wedge - \frac{1}{4!}\theta^3 a^\wedge \\
&- \frac{1}{5!}\theta^4 a^\wedge a^\wedge + \dots \\
&= aa^T + \frac{1}{\theta}(\frac{1}{2!}\theta^2 - \frac{1}{4!}\theta^4 + \frac{1}{6!}\theta^6 - \dots)a^\wedge + \\
&+ \frac{1}{\theta}(-\theta + \frac{1}{3!}\theta^3 - \frac{1}{5!}\theta^5 + \frac{1}{7!}\theta^7 - \dots)a^\wedge a^\wedge \\
&= aa^T + \frac{1}{\theta}(1 - 1 + \frac{1}{2!}\theta^2 - \frac{1}{4!}\theta^4 + \frac{1}{6!}\theta^6 - \dots)a^\wedge + \\
&+ \frac{1}{\theta}(-(\theta - \frac{1}{3!}\theta^3 + \frac{1}{5!}\theta^5 - \frac{1}{7!}\theta^7 + \dots)a^\wedge a^\wedge \\
&= aa^T + \frac{1}{\theta}(1 - (1 - \frac{1}{2!}\theta^2 + \frac{1}{4!}\theta^4 - \frac{1}{6!}\theta^6 + \dots))a^\wedge + \\
&+ \frac{1}{\theta}(-(\theta - \frac{1}{3!}\theta^3 + \frac{1}{5!}\theta^5 - \frac{1}{7!}\theta^7 + \dots)a^\wedge a^\wedge \\
&= aa^T + \frac{1}{\theta}(1 - cos(\theta))a^\wedge + \frac{1}{\theta}(-sin(\theta))a^\wedge a^\wedge \\
&= aa^T + \frac{1}{\theta}(1 - cos(\theta))a^\wedge + \frac{1}{\theta}(-sin(\theta))(aa^T - I_{(3,3)}) \\
&= aa^T + \frac{1}{\theta}(1 - cos(\theta))a^\wedge - \frac{sin(\theta)}{\theta}aa^T + \frac{sin(\theta)}{\theta}I_{(3,3)} \\
&= \frac{sin(\theta)}{\theta}I_{(3,3)} + \frac{1}{\theta}(1 - cos(\theta))a^\wedge + (1 - \frac{sin(\theta)}{\theta})aa^T
\end{aligned}$$

q.e.d.