Practical Course: Vision-based Navigation (IN2106)

Exercise 1

Introduction to SLAM and Lie Groups

By

Md Jamiur Rahman

03697572

04.11.2018

Exercise 1: What is SLAM

1. Why would a SLAM system need a map?

   Answer:

   There are mainly two reasons:

   a. The map is often required to support other tasks; for instance, a map can inform path planning or provide an intuitive visualization for a human operator.

   b. The map allows limiting the error committed in estimating the state of the robot. In the absence of a map, dead-reckoning would quickly drift over time; on the other hand, using a map, e.g., a set of distinguishable landmarks, the robot can “reset” its localization error by re-visiting known areas (so-called loop closure).

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

   Answer:

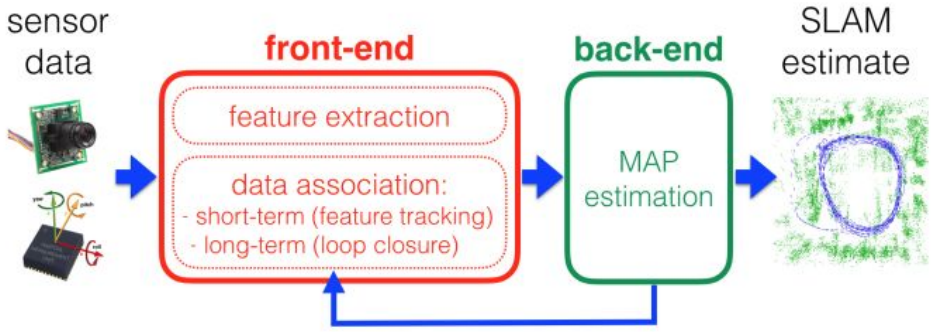

Fig: SLAM technology into real-world applications.

3. Describe the history of SLAM.

The genesis of the probabilistic SLAM problem occurred at the 1986 IEEE Robotics and Automation Conference held in San Francisco. This was a time when probabilistic methods were only just beginning to be introduced into both robotics and AI. A number of researchers had been looking at applying estimation-theoretic methods to mapping and localisation problems; these included Peter Cheeseman, Jim Crowley, and Hugh Durrant-Whyte. Over the course of the conference many paper table cloths and napkins were filled with long discussions about consistent mapping. Along the way, Raja Chatila, Oliver Faugeras, Randal Smith and others also made useful contributions to the conversation.
The result of this conversation was a recognition that consistent probabilistic mapping was a fundamental problem in robotics with major conceptual and computational issues that needed to be addressed. Over the next few years a number of key papers were produced. Work by Smith

and Cheesman and Durrant-Whyte established a statistical basis for describing relationships between landmarks and manipulating geometric uncertainty. A key element of this work was to show that there must be a high degree of correlation between estimates of the location of different landmarks in a map and that indeed these correlations would grow with successive observations.

At the same time Ayache and Faugeras were undertaking early work in visual navigation, Crowley and Chatila and Laumond in sonar-based navigation of mobile robots using Kalman filter-type algorithms. These two strands of research had much in common and resulted soon after in the landmark paper by Smith, Self and Cheeseman. That paper showed that as a mobile robot moves through an unknown environment taking relative observations of landmarks, the estimates of these landmarks are all necessarily correlated with each other because of the common error in estimated vehicle location. The implication of this was profound: A consistent full solution to the combined localisation and mapping problem would require a joint state composed of the vehicle pose and every landmark position, to be updated following each landmark observation. In turn, this would require the estimator to employ a huge state vector (of order the number of landmarks maintained in the map) with computation scaling as the square of the number of landmarks.

Crucially, this work did not look at the convergence properties of the map or its steady-state behavior. Indeed, it was widely assumed at the time that the estimated map errors would not converge and would instead exhibit a random walk behavior with unbounded error growth. Thus, given the computational complexity of the mapping problem and without knowledge of the convergence behavior of the map, researchers instead focused on a series of approximations to the consistent mapping problem solution which assumed or even forced the correlations between landmarks to be minimized or eliminated so reducing the full filter to a series of decoupled landmark to vehicle filters. Also for these reasons, theoretical work on the combined localisation and mapping problem came to a temporary halt, with work often focused on either mapping or localisation as separate problems.

The conceptual breakthrough came with the realisation that the combined mapping and localisation problem, once formulated as a single estimation problem, was actually convergent. Most importantly, it was recognised that the correlations between landmarks, that most researchers had tried to minimize, were actually the critical part of the problem and that, on the contrary, the more these correlations grew, the better the solution. The structure of the SLAM problem, the convergence result and the coining of the acronym ‘SLAM’ was first presented in a mobile robotics survey paper presented at the 1995 International Symposium on Robotics Research. The essential theory on convergence and many of the initial results were developed by Csorba. Several groups already working on mapping and localisation, notably at MIT, Zaragoza, the ACFR at Sydney and others, began working in earnest on SLAM 1 applications in indoor, outdoor and sub-sea environments.

At this time, work focused on improving computational efficiency and addressing issues in data association or ‘loop closure’. The 1999 International Symposium on Robotics Research (ISRR’99) was an important meeting point where the first SLAM session was held and where a degree of convergence between the Kalman-filter based SLAM methods and the probabilistic localisation and mapping methods introduced by Thrun was achieved. The 2000 IEEE ICRA

Workshop on SLAM attracted fifteen researchers and focused on issues such as algorithmic complexity, data association and implementation challenges. The following SLAM workshop at the 2002 ICRA attracted 150 researchers with a broad range of interests and applications. The 2002 SLAM summer school hosted by Henrik Christiansen at KTH in Stockholm attracted all the key researchers together with some 50 PhD students from around the world and was a tremendous success in building the field. Interest in SLAM has grown exponentially in recent years, and workshops continue to be held at both ICRA and IROS. The SLAM summer school ran in 2004 in Tolouse and will run at Oxford in 2006.

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

What does the following lines do?

1. set(CMAKE_MODULE_PATH
   ${CMAKE_CURRENT_SOURCE_DIR}/cmake_modules/" ${CMAKE_MODULE_PATH})

   Answer:
   Adding additional modules such as Eigen, Intel TBB into Cmake list.
   Ref: *https://cmake.org/*

2. a. set(CMAKE_CXX_STANDARD 11)
   b. set(CMAKE_CXX_STANDARD_REQUIRED ON)
   c. set(CMAKE_CXX_EXTENSIONS OFF)

   Answer:
   Enabling a particular C++ standard 11 in Cmake.
   a. The behaviour of the CXX_STANDARD target property is to add the relevant compiler and linker flags to the target to make it build with the specified C++ standard.
   b. There is a minor wrinkle in that if the compiler doesn't support the specified standard, rather than causing a warning or error, by default CMake falls back to the latest standard the compiler does support instead. To prevent this fallback behaviour, the CMAKE_CXX_STANDARD_REQUIRED target property should be set to ON. On some platforms, this results in linking to a different library (e.g. -std=c++11 rather than -std=gnu++11).
   c. The default behavior is for C++ extensions to be enabled. It is important to be consistent in whether or not extensions are enabled when linking targets, as mixing different settings can result in trying to mix standard library implementations.
   Ref: *https://crascit.com/*

3. SET(CMAKE_CXX_FLAGS_DEBUG "-O0 -g")
   SET(CMAKE_CXX_FLAGS_RELEASE "-O3 -DNDEBUG -ftree-vectorize -march=${CXX_MARCH}")
   SET(CMAKE_CXX_FLAGS_RELWITHDEBINFO "-O3 -g -DNDEBUG -ftree-vectorize -march=${CXX_MARCH}")
   SET(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -ftemplate-backtrace-limit=0")

   Answer:
   To choose the CMake build type we set the variable CMAKE_BUILD_TYPE to one of Release|Debug|RelWithDebInfo|MinSizeRel. The build type can be fixed in the CMakeLists.txt, or, more commonly, passed as an argument to cmake on the command line. The particular C++ flags used for a given build type are compiler dependent and are stored in the CMAKE_CXX_FLAGS_ variable.
   Ref: *https://ecrafter.wordpress.com*

4.
   a. add_executable(calibration src/calibration.cpp)
      Answer:
      Add an executable to the project using the specified source file (calibration.cpp).

   b. target_link_libraries(calibration ceres ${Pangolin_LIBRARIES})
      Answer:
      Link a target to Pangolin library.

      Ref: *https://cmake.org/*

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

Proof: