Probabilistic Programming for Robotics

Seyed Mahdi Shamsi, Gian Pietro Farina, Marco Gaboardi, and Nils Napp

Autonomous agents use noisy sensors and actuators to interact with a complex external world. As such, inference engines and point estimators are essential to making modern robots work. Often, these are highly specialized and optimized to run in real-time on autonomous agents. These methods are hand-coded and typically make approximations to gain speed, e.g linearization, approximate independence, or static world assumptions. Extended Kalman filters (EKF) are used in a variety of applications from low-level state estimator for feedback control, to high-level mapping applications.

As a high-level language and inference tool, we believe that probabilistic programming has much to offer to the robotics community. As robots become increasingly capable, they also pose greater risks. A wrong state estimate can result in serious damage and injury when robots are equipped with powerful motors. Eliminating the possibility of coding errors in the inference engine by compiling them from high-level specifications would be beneficial both from a safety and design-time perspective. Conversely, robotics can be a rich source of challenging inference problems related to robust long-term autonomy that probabilistic programming might help answer. We present a specific example problem of a common estimation task in autonomous robot and show how PP can address a (vexing) practical calibration issues that practitioners face.

Mapping and Localization

Localization, estimating the robot pose in a world frame, is one of the fundamental tasks of mobile agents. It forms the basis for essentially any high-level robot behavior and the formulation of motion plans. The class of algorithms used for this purpose are known as simultaneous localization and mapping (SLAM). They jointly estimate a representation of the environment (map) and the robot's pose within the map (localization). The robot pose is denoted by X, the sensor observations by Z, motion control inputs to the robot by U, and a map by M. The sensor reading Z_t at time t depends on both the map M and the robot pose X_t. In SLAM we assume that the measurements z_t and the control inputs u_t are observed, and the X_t and M_t need to be inferred.

EKF-SLAM

To illustrate how writing a simple generative model might be useful to practitioners we briefly discuss practical considerations in using EKF-SLAM, since the map representation and calibration issues are

similar to many other SLAM implementations. The representation in EKF (as in all Kalman Filters) is Gaussian. The map is composed of a discrete set of *landmarks*, each of which has a location. Both the robot pose and the map elements are represented as multivariate Guassians. The motion model, $P(X_t|x_{t-1}, u_t)$ and the observation model $P(Z_t|x_t, m)$ are modeled as additive noise terms to the ideal behavior.

In order to work, SLAM systems require calibrated noise models since they define the conditional distributions for motion and observation. While some calibration tasks (i.e. internal camera parameters) are easy to perform, calibrating a full SLAM pipeline is very tedious. Calibrating motion noise requires ground truth observations. Ground truth poses are difficult to collect, and typically require an expensive calibrated external observation system. These systems are both expensive and might represent different operating conditions from the robot's target environment, which could produce drastically different motion models.

Probabilistic programs that define generative models of robot behavior can be used both to perform SLAM and to calibrate noise parameters. The constraints in the data that enable SLAM in the first place are still present in the sensor data of an uncalibrated system. While using a probabilistic program for SLAM would be much slower than using specialized implementation, being able to use regular run time data in order to perform system calibrations for a complicated robotic system would be an invaluable tool in practice, both in the SLAM setting and for other robotics applications.

Writing Probabilistic Programs for Robots

When designing a robot, roboticists typically have a high-level model of what each part of the robot is supposed to do. As a result, we it should be relatively easy to build a model for a robot that relates the different sensor and input variables. Unknown details of the behavior can be expressed as uncertain parameters. We would like to explore the use probabilistic programming languages for this purpose, and to design a domain specific languages that makes it easy for robotics practitioners to quickly write new generative models for their robots. These can then be used to synthesize new estimators, which can be used either for calibrating parameters required by other systems or used directly when specialized estimators do not exist. More generally, since estimation is such an important part of modern robotics, we believe that this approach of writing generative models for robots might have other useful applications throughout a robot's life-cycle, for example: during robot design for quick prototyping, for reliable operation, for for finding faults.