Notes Distributing Collaborative Multi-Robot Planning with Gaussian Belief Propagation

Rom Parnichkun

April 11, 2022

1 Factor Graphs

A joint probability distribution $p(V)$ is factorised with a factor graph as:

$$p(V) = \prod_{j}^{N_f} f_j(V_j), \tag{1}$$

where given a set of variables $V = v_{i\,i=1:N_v}$, a factor node f_j is connected to a subset of the variables, $V_j \subseteq V$, and N_v is the number of variables.

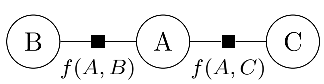

Figure 1: An example of a factor graph.

Solvers such as GTSAM exploit the sparsity revealed by the factor graph for efficient maximum a-posteriori (MAP) inference using sparse linear algebra, and such algorithms are efficient for centralised solvers which have full access to the graph.

2 Gaussian Belief Propagation

GBP is an alternative method to perform inference on a factor graph. It is a subclass of belief propagation which operate using node-wise local computation and message passing, enabling distributed solutions.

The canonical form of a Gaussian distribution is used.

$$\mathcal{N}(x; \mu, \Sigma) = \mathcal{N}(x; \eta, \Lambda), \tag{2}$$

in which $\Lambda = \Sigma^{-1}$ and $\eta = \Sigma^{-1}\mu$. In GBP, variables V are assumed to be Gaussian; thus each variable has a belief $b(v_i) = \mathcal{N}^{-1}(v_i; \eta_i, \Lambda_i)$. Factors $F = f_{i\,i=1:N_f}$ are a probabilistic Gaussian constraint between variables. $f_i(V_j)$ is an arbitrary function that connects variables V_j, and it may be non-linear.

2.1 Variable Belief Update

$$b(v_i) = \prod_{f \in n(v_i)} m_{f \to i}(v_i), \tag{3}$$

where $n(v_i) \subseteq F$ is the set of factors that the variable v_i is connected to, and $m_{f\to i}(v_i) = \mathcal{N}^{-1}(v_i; \eta_{f\to i}, \Lambda_{f\to i})$ is the message from a factor to the variable. As we are using the canonical form, the product between distributions can be rewritten as a summation:

$$\eta_i = \sum_{f \in n(v_i)} \eta_{f \to i}, \tag{4}$$

$$\Lambda = \sum_{f \in n(v_i)} \Lambda_{f \to i}. \tag{5}$$

2.2 Variable to Factor Message

$$m_{v_i \to j}(f_j) = \prod_{f \in n(v_i) \setminus f_j} m_{f \to i}(v_i). \tag{6}$$

2.3 Factor Likelihood Update

The likelihood of factor $f(V_j)$ with measurement function $h(V_j)$, observation z_s, and precision of the observation Λ_s, can be expressed as a Gaussian distribution $\mathcal{N}^{-1}(V_j; \eta_f, \Lambda_f)$, where $\eta_f = \Lambda_s(z_s - h(V_j))$

and $\Lambda_f = \Lambda_s$. This however only holds if $h(V_j)$ is linear. In the non-linear case, we linearise using first-order Taylor expansion: $h(V_j) = h(V_j^0) + J(V_j - V_j^0)$.

$$\eta_f = J^T \Lambda_m (J V_j^0 + z_m - h(V_j^0)), \tag{7}$$

$$\Lambda f = J^T \Lambda_m J, \tag{8}$$

where V_j^0 is the linearisation point, the current state of the variables. In this work $z_m = 0$ for all factors, meaning that the factor energy is purely a function of the states.

2.4 Factor to Variable Message

$$m_{f \to i}(v_I) = \sum_{v_j \in V_j v_i} f(V_j) \prod_{v_j \in V_j v_i} m_{v_j \to f}(v_j). \tag{9}$$

3 Method

3.1 Modelling a robot moving through a 2D plane

$$\dot{x}_i = A x_i + B u_i + F w_i, \tag{10}$$

$$A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, B = 0, F = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \tag{11}$$

where $w \sim \mathcal{N}(0, Q_d)$ is a white noise with covariance matrix $Q_d = \sigma_d^2 I$. X_i represents the robot's position and velocity at that particular moment in time.

$$X_i = \begin{bmatrix} x_i & y_i & \dot{x}_i & \dot{y}_i \end{bmatrix}^T. \tag{12}$$

The trajector of the robot is represented by N such states, from X_0 to X_{N-1}. The optimal trajectory solution can be found by solving for the maximum a posteriori (MAP) solution X^* for the trajectory states.

Nodes of the factor graph represents the robot states through time (see Figure 2).

There are four types of factors that this study considers.

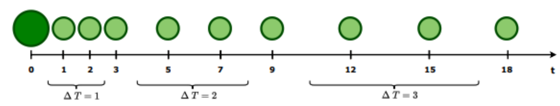

Figure 2: Nodes represents the robot's state.

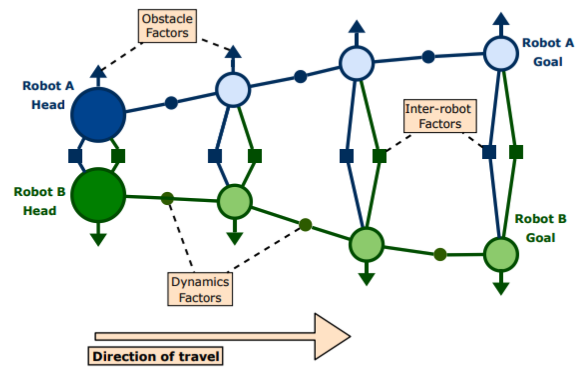

Figure 3: An example of the factor graph.

3.2 Pose Factor

$$h_p(X_i) = X_i, \tag{13}$$

$$\Sigma_p = \sigma_p^2 I, \tag{14}$$

3.3 Dynamics Factor

$$h_d(X_i, X_{i+1}) = \Phi(t_{i+1}, t_i) X_i - X_{i+1}, \tag{15}$$

$$\Sigma_d = \begin{bmatrix} \frac{1}{3}\Delta t_i^3 Q_d & \frac{1}{2}\Delta t_i^2 Q_d \\ \frac{1}{2}\Delta t_i^2 Q_d & t_i Q_d \end{bmatrix}, \tag{16}$$

where

$$\Phi(t_b, t_a) = \begin{bmatrix} 1 & (t_b - t_a)1 \\ 0 & 1 \end{bmatrix} \tag{17}$$

is the state-transition matrix from time t_a to time t_b. This factor encourages a zero acceleration and therefore a feasible and smooth tarjectory.

3.4 Obstacle Factor

$h_O(X_i)$ is equal to 1 at or within the obstacle boundary and decreases exponentially to 0 at a distance of one robot radius away from the obstacle (kinda similar to a reward function). The factor covariance is $\Sigma_O = \sigma_O I$.

3.5 Inter-robot Factor

The factor has a non-zero energy cost if the distance between the robots at a particular timestep is less than the critical distance distance $r^* = 2r_{robot} + \epsilon$ where ϵ is a small safety distance.

$$g(p) = \begin{cases} 1 - \frac{P}{r^*}, & p \leq r^* \\ 0, & \text{otherwise} \end{cases} .^1 \tag{18}$$

There is the possibility that robot tarjectories could interesect in between two consecutive timesteps. To avoid this issue we allow for K-opint linear interpolation between timesteps in the inter-robot factor.

$$h_r(X_{A,i}, X_{B,i}) = [g(r_{i+\frac{k}{K}}]|_{0 \leq k \leq K-1}. \tag{19}$$

$$\Sigma_r = \sigma_i I \tag{20}$$

where $\sigma_i = \sigma_r t_i$.

3.6 Goal State Update

Goal state X_{N-1} is updated as:

$$X_{N-1} \leftarrow \begin{bmatrix} x_{N-1} + \tau v^*_{N-1} \\ v^* \end{bmatrix}, \tag{21}$$

where

$$\tau = \begin{cases} 1 & ||x_{N-1} - x_0|| \leq r_{max} \\ (\dot{x_0} \cdot v^*) & \text{otherwise} \end{cases} \tag{22}$$

and $r_{max} = t_{N-1}||v^*||$.

[1] This is known as a truncated hinge loss.