# Notes on FutureMapping 2: Gaussian Belief Propagation for Spatial AI

Rom Parnichkun

August 11, 2022

## 1 Overview

This note lays out the formulation of belief propagation, and subsequently Gaussian belief propagation (GBP). Both these algorithms take advantage of a graphical structure called **Factor Graphs**. Within it, there are **factor nodes** and **variable nodes**. Variable nodes contain variables that we would like to compute the marginal probability of, and factor nodes relates the various variable nodes together with a joint distribution.

### 1.1 Example Problem

Understanding this algorithm is difficult when only looking at the mathematical formulations and abstractions therefore this section provides concrete examples where the GBP algorithm may be used.

- **Background:** There are two robots, robot A and robot B. Robot A measures the distance from itself to robot B with a sensor (which has Gaussian noise). Robot A also has a belief on its own position (characterized by a Gaussian distribution).
- **Problem:** Find the position of robot B.
- **Solution:** In order to find the position of robot B, we can simply add robot A's position with the relative distance between robot A and robot B from the sensor. However, there are distributions associated with both the position and the measurement, therefore we should view robot A and robot B's positions to be in a joint distribution (**factor node**), then marginalize robot B's position to get its distribution (belief).

The previous example illustrates a single message passing scheme, which does not fully take advantage of Gaussian belief propagation.

- **Background:** Now instead of two robots imagine hundreds of robots. Each robot has the ability to measure another robot that is in its visibility range. They all also have a belief on its own position.
- **Problem:** Find the position of every robot so that they consistent with each other.
- **Solution:** Each measurement of another robot creates a new factor node. A factor node represents the flow of belief from one variable node to another. Therefore in this case, if there are multiple robots that have measurements of a robot, all the beliefs from the multiple robots are combined to update the belief of the robot being measured. This process iteratively updates the position of every robot in the graph and slowly converges to states that are consistent to one another in a distributed manner.

It is important to point out that GBP provides a way to create a globally consistent graph with distributed computation. Additionally, GBP supports ad-hoc message passing making it suitable for robotics applications.

## 2 Tutorial on Belief Propagation

Marginal distribution is found by taking the joint distribution, and summing over all the other variables:

$$p(x) = \sum_{\mathbf{x} \setminus x} p(\mathbf{x}). \tag{1}$$

The total joint probability is defined as the product of all factors in a graph: [1]

$$p(\mathbf{x}) = \prod_{s} f_s(\mathbf{x}_s), \tag{2}$$

Setting $F_s$ as the product of all factors associated with $f_s$ (a factor that is directly connnected to $x$) we get:

$$p(\mathbf{x}) = \prod_{s \in n(x)} F_s(x, \mathbf{X}_s), \tag{3}$$

where $n(x)$ is the set of factor nodes that are neighbors of $x$ and $\mathbf{X}_s$ is the vector of all variables in the subtree conntected to $x$ via $f_s$. Note that with equation 3 we reduce the problem of finding the joint probability to a local one $s \in n(x)$.

Next, combining equations 1 and 3 we get:

$$p(x) = \sum_{\mathbf{x} \setminus x} \Big[ \prod_{s \in n(x)} F_s(x, \mathbf{X}_s) \Big]. \tag{4}$$

We can reorder the sum and product to obtain:

$$p(x) = \prod_{s \in n(x)} \Big[ \sum_{\mathbf{X}_s} F_s(x, \mathbf{X}_s) \Big]. \tag{5}$$

Intuitively, this reordering is simply changing the marginalisation to be over a subtree, then multiplying over all the marginals to get the final marginal. It effectively decouples the marginalisation process to a subtree (from the summation over $\mathbf{x} \setminus x$ to $\mathbf{X}_s$) allowing us to define a "message" from $f_s$ as shown below. [2]

$$\mu_{f_s \to x} = \sum_{\mathbf{X}_s} F_s(x, \mathbf{X}_s). \tag{6}$$

[1] Note that $p(\mathbf{x})$ here may note be normalized.

[2] A message is like a local marginalisation.

Substituting Equation 6 to 5 results in

$$\boxed{p(x) = \prod_{s \in n(x)} \left[\mu_{f_s \to x}\right].} \tag{7}$$

Next, we break down $F_s(x, \mathbf{X}_s)$ as

$$\begin{aligned} F_s(x, \mathbf{X}_s) &= f_s(x, x_1, \dots, x_M) \\ &\quad \times \left[G_1(x_1, \mathbf{X}_{S_1}), \dots, G_M(x_M, \mathbf{X}_{S_M})\right] \\ &= f_s(x, x_1, \dots, x_M) \prod_{m \in n(f_s)} G_m(x_m, \mathbf{X}_{S_m}), \end{aligned} \tag{8}$$

in which the factor $f_s$ is a function of all nodes conntected to it, and $G_m(x_m, X_{S_m})$ is defined as follows.

$$G_m(x_m, \mathbf{X}_{S_m}) = \prod_{l \in n(x_m) \setminus f_s} F_l(x_m, \mathbf{X}_{m_l}). \tag{9}$$

Equation 9 shows the recursive nature of computing $F_s$ ($F \to G \to F \dots$), therefore we can reorganize and substitute equation 8 to equation 6 to reflect this nature.

$$\begin{aligned} \mu_{f_s \to x}(x) &= \sum_{\mathbf{X}_s} \left[f_s(x, x_1, \dots, x_M) \prod_{m \in n(f_s)} G_m(x_m, \mathbf{X}_{S_m})\right] \\ &= \sum_{x_1, \dots, x_M} \sum_{\mathbf{X}_{S_1}, \dots, \mathbf{X}_{S_M}} \left[f_s(x, x_1, \dots, x_M) \prod_{m \in n(f_s)} G_m(x_m, \mathbf{X}_{S_m})\right] \\ &= \sum_{x_1, \dots, x_M} f_s(x, x_1, \dots, x_M) \prod_{m \in n(f_s)} \sum_{\mathbf{X}_{S_1}, \dots, \mathbf{X}_{S_M}} G_m(x_m, \mathbf{X}_{S_m}) \end{aligned} \tag{10}$$

Similar to Equation 6, $G_m$ can be thought of as a message coming from the variable, hence we define the message $\mu_{x_m \to f_s}(x_m)$ as follows:

$$\mu_{x_m \to f_s}(x_m) = \sum_{\mathbf{X}_{S_m}} G_m(x_m, \mathbf{X}_{S_m}) \tag{11}$$

Substituting $\mu_{x_m \to f_s}$ to equation 10 we get

$$\boxed{\begin{aligned} \mu_{f_s \to x}(x) = &\sum_{x_1, \dots, x_M} f_s(x, x_1, \dots, x_M) \\ &\times \prod_{m \in n(f_s)} \mu_{x_m \to f_s}(x_m). \end{aligned}} \tag{12}$$

Next, we substitute Equation 9 to 11 to get

$$\mu_{x_m \to f_s}(x_m) = \sum_{\mathbf{X}_{S_m}} \prod_{l \in n(x_m) \setminus f_s} F_l(x_m, \mathbf{X}_{m_l}), \tag{13}$$

and swap the order of the sum and product to obtain:

$$\mu_{x_m \to f_s}(x_m) = \prod_{l \in n(x_m) \setminus f_s} \sum_{\mathbf{X}_{m_l}} F_l(x_m, \mathbf{X}_{m_l}). \tag{14}$$

Finally we derive the message from variables that completes the recursion

$$\boxed{\mu_{x_m \to f_s}(x_m) = \prod_{l \in n(x_m) \setminus f_s} \mu_{f_l \to x_m}(x_m).} \tag{15}$$

In order to find the marginal distribution for $x$, we start from all of the leaf nodes of the factor graph relative to $x$, and pass messages inwards torwards $x$. To initialise the leaf nodes, a variable leaf node sends a message $\mu_{x \to f}(x) = 1$ to its only connected factor and a leaf factor sends $\mu_{f \to x}(x) = f(x)$.

To find the marginal for *all* variables, we send the messages outwards from the root back to the leaves.

## 3 Gaussian Belief Propagation

If the relationship between variables is linear, representing the probabilities of the variables with a multivariate Gaussian distribution enables the marginals to also be Gaussian distributions.

### 3.1 Factor Definition

A Gaussian factor can be written as follows.

$$f_s(\mathbf{x}_s) = K e^{-\frac{1}{2}[(\mathbf{z}_s - \mathbf{h}_s(\mathbf{x}_s))^T \Lambda_s (\mathbf{z}_s - \mathbf{h}_s(\mathbf{x}_s))]}. \tag{16}$$

This expression represents the probability of obtaining measurement vector $\mathbf{z}_s$ from the sensor based on the other variables $\mathbf{x}_s$. $\mathbf{h}_s$ is the function which describes the relationship between the variables on the expected measurement $\mathbf{z}_s$. Matrix $\mathbf{\Lambda}_s$ is the precision of the measurement.

With the information form (canonical form), the Gaussian distrubution can be represented as follows.[3]

$$f_s(\mathbf{x}_s) = K e^{[-\frac{1}{2}\mathbf{x}_s^T \mathbf{\Lambda}_s' \mathbf{x}_s + \boldsymbol{\eta}_s^T \mathbf{x}_s]}, \tag{17}$$

where

$$\boldsymbol{\eta}_s = \mathbf{\Lambda}_s \boldsymbol{\mu}_s. \tag{18}$$

### 3.2 State Representation

The probability distribution over state variables also have a Gaussian form as follows.

$$p_m(\mathbf{x}_m) = K e^{[-\frac{1}{2}\mathbf{x}_m^T \mathbf{\Lambda}_m \mathbf{x}_m + \eta_m^T \mathbf{x}_m]}. \tag{19}$$

### 3.3 Linearising Factors

We can linearise any factor to the equation below

$$f_s(\mathbf{x}_s) = K e^{[-\frac{1}{2}\mathbf{x}_s^T \mathbf{\Lambda}_s' \mathbf{x}_s + \boldsymbol{\eta}_s^T \mathbf{x}_s]} \tag{20}$$

$$\boldsymbol{\eta}_s = \mathbf{J}_s^T \mathbf{\Lambda}_s [\mathbf{J}_s \mathbf{x}_0 + \mathbf{z}_s - \mathbf{h}_s(\mathbf{x}_0)] \tag{21}$$

$$\mathbf{\Lambda}_s' = \mathbf{J}_s^T \mathbf{\Lambda}_s \mathbf{J}_s. \tag{22}$$

where $\mathbf{J}_s$ is the Jacobian $\frac{\partial \mathbf{h}_s}{\partial \mathbf{x}_s}|_{\mathbf{x}_s = \mathbf{x}_0}$, $\mathbf{x}_0$ is the point of linearisation.

[3]For more information about the information form read Exactly Sparse Delayed-State Filters.

## 3.4 Message Passing at a Variable Node

For GBP, each incoming message $\mu_{f_l \to \mathbf{x}_m}(\mathbf{x}_m)$ is represented by an information vector $\boldsymbol{\eta}_{ml}$ and a precision matrix $\boldsymbol{\Lambda}_{ml}$. The outgoing message $\mu_{\mathbf{x}_m \to f_s}(\mathbf{x}_m)$ is computed by simply adding

$$\boldsymbol{\eta}_{ms} = \sum_{l \in n(\mathbf{x}_m) \setminus f_s} \boldsymbol{\eta}_{ml} \tag{23}$$

$$\boldsymbol{\Lambda}_{ms} = \sum_{l \in n(\mathbf{x}_m) \setminus f_s} \boldsymbol{\Lambda}_{ml}. \tag{24}$$

This is equivalent to multiplying the distributions.