A Robot Web for Distributed Many-Device Localisation

Riku Murai[1], Joseph Ortiz[1], Sajad Saeedi[2], Paul H.J. Kelly[1], and Andrew J. Davison[1]

[1]Department of Computing, Imperial College London, UK
{riku.murai15,j.ortiz, p.kelly, a.davison}@imperial.ac.uk
[2]Department of Mechanical and Industrial Engineering, Ryerson University, Toronto, Canada
s.saeedi@ryerson.ca

Abstract

We show that a distributed network of robots or other devices which make measurements of each other can collaborate to globally localise via efficient ad-hoc peer to peer communication. Our Robot Web solution is based on Gaussian Belief Propagation on the fundamental non-linear factor graph describing the probabilistic structure of all of the observations robots make internally or of each other, and is flexible for any type of robot, motion or sensor. We define a simple and efficient communication protocol which can be implemented by the publishing and reading of web pages or other asynchronous communication technologies. We show in simulations with up to 1000 robots interacting in arbitrary patterns that our solution convergently achieves global accuracy as accurate as a centralised non-linear factor graph solver while operating with high distributed efficiency of computation and communication. Via the use of robust factors in GBP, our method is tolerant to a high percentage of faults in sensor measurements or dropped communication packets.

1. Introduction

As we head towards a future where embodied artificial intelligence is ubiquitous, we expect that multiple robots, vehicles and other smart devices which share the same environment will need to communicate and coordinate their actions, whether their goal is explicit cooperation or just safe independent action. One clear possibility is that all devices could use a unified cloud-based 'maps' system, presumably owned by one company or government, which tracks and coordinates all devices. An alternative, which we investigate here, is a distributed system based on per-device local computation and storage, and peer to peer communication between heterogenous devices from different makers using standardised open protocols. Inspired by the original design of the World Wide Web, we call this concept the **Robot Web**.

A key outstanding problem in multi-robot systems has been true distributed localisation: how can a set of moving devices which move and observe each other within a space estimate their locations, using noisy actuators, sensors and realistic peer-to-peer communication? So much work in multi-robot systems has avoided this problem by running in labs with motion capture-style tracking.

In this paper we present a breakthrough Robot Web solution to general, fully distributed and asynchronous many-robot localisation. Our solution is based on the fundamental probabilistic factor graph representation of perception and state estimation. We show that Gaussian Belief Propagation (GBP) [35] is the key inference algorithm with the appropriate properties of distributed processing/storage and message passing which permits a convergent, exact solution to the full, dynamically changing estimation problem via ad-hoc communication between robot peers.

In our solution, each robot stores and maintains its own part of the full factor graph, and updates and publishes a Robot Web Page of outgoing messages for other robots to download and read whenever possible. Remarkably, using GBP the whole factor graph can efficiently converge to localisation estimates as accurate as full batch optimisation, but without any device ever needing to store or process more than its own local graph fragment. Robots communicate via ad-hoc, asynchronous messages containing only small vectors and matrices. Significantly, GBP can deal with graphs which have any type of parameterisation (e.g. 2D or 3D robot movement, or any type of non-linear sensor measurements) and which can change dynamically in arbitrary ways — for instance robots can join or leave the web whenever needed, or reconfigure their sensors online. We will show that it can also cope with and reject a high fraction of 'garbage' measurements, for instance caused by faulty sensors, and deal with highly unreliable communication channels.

Our approach is designed for scalability. All communication is via a simple interface, and robots do not need any privileged information about each other, such as even how many other robots are involved. The whole Robot Web therefore can be fully dynamic, with robots joining or leaving at will. We believe that this formulation of many-robot localisation could be the foundation for a new era of distributed Spatial AI. We proceed here to an immediate exposition of the method, and review the related work that it builds on later in the paper.

2. Robot Web: Core Design and Structure

We will use the term 'robot' for any device involved in the Robot Web, but some of these could be beacons, sensor nodes, or any other type of participating entity, which could be moving or stationary.

The fundamental structure of the Robot Web is the full probabilistic factor graph which represents the states of robots as variables and the measurements they make, or any other information which is available such as pose or smoothness priors, as factors. Determining estimates of the robot states is a matter of performing inference on this factor graph to produce marginal distributions over the variables. We will assume that all factors take the form of Gaussian functions of the involved state variables, and use Gaussian Belief Propagation as the mechanism for inference. Note that GBP supports robust (heavy-tailed) factors, and non-linear measurement functions via the methods proposed in [13], and therefore this model is very broadly practically applicable. These are the same assumptions behind most centralised factor graph inference libraries, such as GTSAM, Ceres or g2o.

The key concept of the Robot Web is to distribute responsibility for storing and updating the full factor graph, by dividing it up between the robots taking part. Figure 1 illustrates this for an elementary case of three moving robots, each with internal odometry sensing and an outward-looking sensor able to make observations of the other robots. We use different colours to highlight the parts of the factor graph for which each robot is responsible. Robot i stores:

- The set of variables $\mathbf{x}_i^t$ representing its state at discrete times t. It could store a whole history of states, or a finite window. Most commonly these states will be multi-dimensional variables which directly represent robot pose, though any other aspects of internal state could be included. We will discuss pose parametrisation in detail later.
- The set of factors f_i representing internal, proprioceptive measurements or priors. Each of these factors connects to one or more of the robot's own state variables. Common examples would be a unary factor representing a GPS pose measurement, or a binary factor connecting two temporally consecutive states representing an odometry or inertial measurement.
- A set of factors g_{ij}^t representing exteroceptive measurements made by this robot of other robots. Specifically, factor g_{ij} represents a measurement made by this robot of another robot j at time t. This factor connects one state variable $\mathbf{x}_i^t$ from robot i with state variable $\mathbf{x}_j^t$ of robot j at the corresponding time.

There is an important design choice here: factors representing inter-robot measurements are stored by the robot making the measurement. This is because the details of measurement factors depend on the type and calibration of the sensor involved, and in this way those details only need to be known to the robot carrying the sensor. Note also that we assume for now that all robots have globally synchronised clocks for timestamping of measurements (though we will see that all computation and communication can be asynchronous).

The factor graph evolves and grows dynamically. At initialisation, a robot will have just one variable node. As it moves, and measures its own incremental motion with odometry or similar, it adds the appropriate variables and factors to its internal factor graph. GBP runs continuously on the robot's internal factor graph, producing always-updating marginal distributions for each variable. The message passing pattern of GBP within a robot's internal factor graph is not important, but should be rapid and global enough the keep the graph fragment mostly close to convergence.

When the robot uses an outward-looking sensor to make an observation of the relative location of another robot, it creates a factor for this measurement, connects it to its current live pose variable, and the factor takes part in local GBP. The other end of this factor will initially be unconnected, because the appropriate variable to attach it to is stored by another robot: the factor-to-variable edge crosses the 'dotted line' boundary separating factor graph fragments. When local GBP generates an outgoing message from this factor which crosses the dotted line, that message is made available to the other robot that needs it by posting it to a 'Robot Web Page'.

We can see from Figure 1 that there are two types of factor that cross a particular robot's 'dotted line'. As well as those for measurements it has made itself, there are factors for measurements where other robots have observed it with their sensors. Since the robot does not know in advance at which times other robots have observed it, it also publishes outgoing messages from every variable in its history. (Later on we will consider possibilities for improving on this by

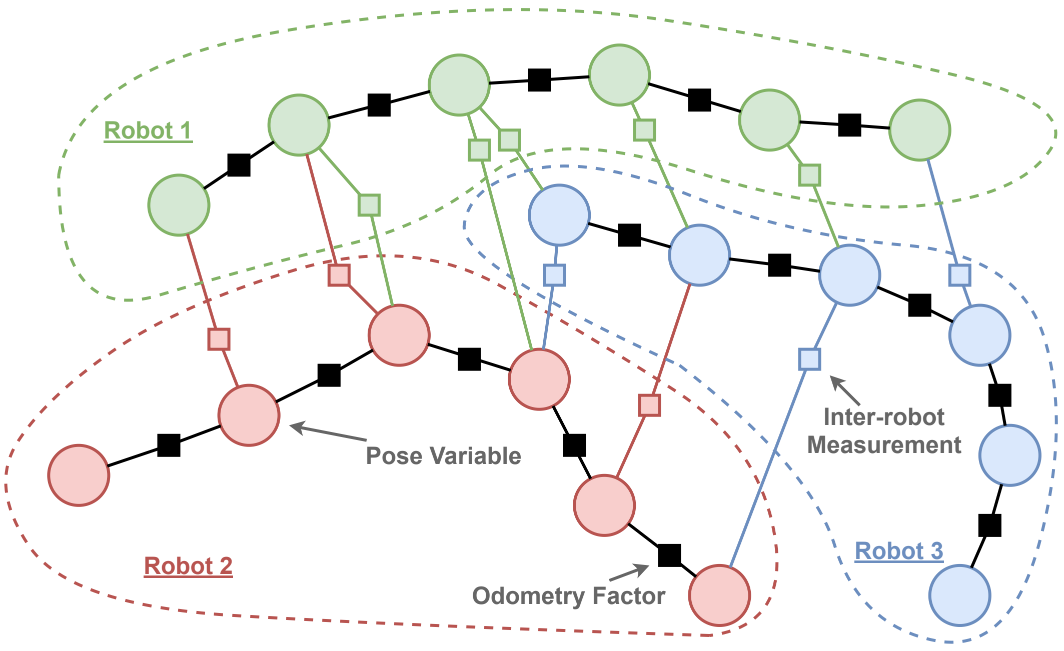


Figure 1: In the Robot Web, we assume that a set of robots move through a space while using their sensors to observe each other. The circles represents the variables, and the squares are the factors. The full factor graph for multi-robot localisation is used. Resposibility for storing and updating it is divided up between the multiple robots participating, as shown by the coloured regions separated by dotted lines. Each robot maintains its own pose variable nodes, odometry factors, and factors for the inter-robot measurements made by its sensors, and carries out continuous GBP on this graph fragment. Message passing across dotted line boundaries is via Robot Web Pages published and updated by each robot, and happens on an asynchronous and ad-hoc basis.

not needing to publish the robot's whole state history.)

Let us be precise about the concept of a Robot Web Page. At some intermediate time, robot i will have built up a factor graph with a set of variable nodes $\mathbf{x}_i^t$, internal factors f_i, and outward-facing factors g_{ij}^t, and these will be taking part in continuous internal GBP. On a regular basis, the robot publishes a page with this information; for example for Robot 2 in Figure 1:

Node	Latest Outgoing Message	
	Info. Vector $\boldsymbol{\eta}$	Precision Matrix Λ
$\mathbf{x}_2^1$	2.342 0.212	0.234 0.011 0.011 0.234
$\mathbf{x}_2^2$	3.453 0.323	0.264 0.041 0.041 0.264
$\mathbf{x}_2^3$	4.333 0.586	0.444 0.056 0.056 0.444
$\mathbf{x}_2^4$	5.764 0.637	0.308 0.122 0.122 0.308
$\mathbf{x}_2^5$	6.345 0.767	0.555 0.099 0.099 0.555
$\mathbf{x}_2^5$	6.345 0.767	0.555 0.099 0.099 0.555
$\mathbf{x}_2^6$	6.345 0.767	0.555 0.099 0.099 0.555
g_{21}^2	9.332 3.343	1.225 0.886 0.886 1.225
g_{21}^3	8.267 2.667	1.665 0.555 0.555 1.655
g_{23}^5	3.543 7.374	0.672 0.336 0.336 0.672

This means that Robot 2 has pose variables for 6 timesteps, and publishes the latest appropriate outgoing messages for factors connected to them. Also, it observed Robot 1 at timesteps 2 and 3, and observed Robot 3 at timestep 5, and publishes outgoing messages from its outward-looking factors representing these measurements. All messages have the form of an information vector and precision matrix. In this example we assume two-dimensional pose representation, so the information vectors are 2×1 and precision matrices are 2×2.

Now, whenever it has the opportunity, a robot will log onto the Robot Web Page of any nearby robot, read it, and check whether there are any messages it can use. For instance, if robot 2 reads the page above from robot 1, the g_{12}^2 and g_{12}^4 messages are useful. If robot 2 has used its sensor to make a measurements of robot 1 at time 3, it will also read the $\mathbf{x}_1^3$ message. It then incorporates these messages as incoming messages into its own local factor graph, and continues local GBP on that fragment.

The properties of GBP [35] mean that the communication between robots can be completely asynchronous and ad-hoc, but the overall graph made up of many fragments will converge to global estimates.

The key motivation for the design choices we have made is a desire for **distributed scalability**. We force all inter-robot communication to be via our simple web protocol, and our design means that different robots do not need to know anything else about each other. We see this as similar to one of the arguments explaining the success of Amazon as a large organisation with many different segments: Jeff Bezos's famous 'API mandate', demanding that all capabilities within the company would have to be designed and exposed as openly defined APIs. Because there shouldn't be any hidden dependencies between business sections, each can operate in a more independent and innovative way. In the Robot Web, a huge advantage of this approach is for instance that robots can be added to or removed from the web in a fully dynamic way, or change their internal methods or software freely as long as they retain the same Web interface. The internal complexity of each robot's processing may be slightly increased because of this, but this is a small price to pay for global scalability.

In terms of how inter-robot communication will actually happen, our use of web terminology leads to the obvious idea that each robot would explicitly form a WWW node, run a web server, publish web pages there, and then other robots would use peer-to-peer Wi-Fi connections to read the pages. The fact that inter-robot communication can happen in arbitrary patterns and in a read-only style allows many other alternatives for communication, which we will discuss further in Section 6.3.

In the next section we recall the details of GBP, and present a new and general formulation for the steps of GBP in the case of Lie Group variables which represent general robot transformations. The reader that is already familiar with implementing GBP, or who does not need to understand the algorithm in full detail, can jump directly to Section 4 where we present demonstrations of the method in action in simulation and experiments on its performance.

3. Gaussian Belief Propagation Inference with Lie Groups

A factor graph has a bipartite structure, which means that variable nodes are only connected to factor nodes, and vice versa. Inference with GBP proceeds iteratively via message passing between variable and factor nodes, which can in principle happen in many different types of pattern but still with convergent overall behaviour. There are four elemental operations in GBP. The first two are message passing at variable and factor nodes, the main steps which take inference towards convergence. When each node performs message passing, it uses the latest incoming messages from *all but one* of the nodes to which it is connected to calculate an outgoing message which it then passes out to the remaining node. The other useful operations involve gathering and combining the incoming messages into a node from *all* connected nodes. At a variable node, this is used to calculate an updated belief at that node, for display or decision making purposes. At a factor node, we use this to calculate the current likelihood of the factor, for instance to relinearise or to apply a robust weighting.

These four operations (two for variable nodes, and two for factor nodes) are always the same, independent of the specific type of variable or factor or whether the messages it is exchanging are with other nodes within its local robot graph fragment or with nodes held by other robots communicated via Robot Web Pages. To implement a Robot Web, therefore, one simply has to follow the rules of GBP as laid out in FutureMapping 2 [13] or 'A visual introduction to Gaussian Belief Propagation' [35]. If the state space of variables is simple, such as linear position in 2D, then the mathematical details of implementation are exactly as given in FutureMapping 2, except that now in Robot Web we pass some messages internally and some across the inter-robot Web interface.

In most realistic robot localisation problems however, there are additional details to consider due to the state space being a robot pose with rotation and translation, where careful thought about parameterisation is needed. Here we go through the four operations of GBP with that in mind, going beyond FutureMapping 2 by using Lie Theory for the general implementation of GBP in these spaces.

We use concepts from Lie Theory and the notation — such as $\oplus, \ominus$ operators — defined in Solà *et al.*'s excellent tutorial, 'A micro Lie theory for state estimation in robotics' [38]. Let $\bar{\mathcal{X}}$ represent the point estimate of a member of a Lie Group. A Gaussian distribution around this point estimate is represented using the tangent space as follows:

$$\mathcal{X} \sim \mathcal{N}(\bar{\mathcal{X}}, \Lambda^{-1}) \, , \tag{1}$$

where:

$$\mathcal{X} = \bar{\mathcal{X}} \oplus \xi \, , \tag{2}$$

and

$$\xi \sim \mathcal{N}(0, \Lambda^{-1}) \, . \tag{3}$$

The core of how we use Lie Theory within GBP is the choice that *all messages to or from Lie Group variables take the form of a point estimate, represented by the full over-parameterised Group element, together with a minimal precision matrix defined in the tangent space around that element.* So, when variable $\mathcal{X}$ represents a transformation which is a member of a Lie Group, all messages to it will take the form $\bar{\mathcal{X}}, \Lambda$, where Λ is a precision matrix in the tangent space at point estimate $\bar{\mathcal{X}}$.

This approach gives maximum flexibility and minimises the need for independent devices to have knowledge or memory of each other (which might be required with alternative ideas, such as that messages would represent perturbations around some remembered stored element).

3.1. Message Passing at a Variable Node

Consider a Lie Group variable connected to $N \geq 1$ factor nodes $i = 1 \dots N$. In a message passing step it must output a message to one of these factors $i = a$. Each of the $N-1$ incoming messages has the form $\bar{\mathcal{X}}_i, \Lambda_i$; for each message the precision matrix is in the local tangent space around the Lie Group element. To combine them, we must first transform all of the precision matrices into the same tangent space. There are different possible choices for which tangent space to use, but a good choice is to use a previous estimate of the variable's state $\bar{\mathcal{X}}_0$ (calculated the last time we did a belief update at that node; see Section 3.3) because this requires minimal transformation of messages. To transform the mean of an incoming message into this tangent space we perform:

$$\boldsymbol{\tau}_i = \bar{\mathcal{X}}_i \ominus \bar{\mathcal{X}}_0 \ . \tag{4}$$

And to transform its precision matrix:

$$\Lambda_{\boldsymbol{\tau}_i} = \mathbf{J}_r^\top(\boldsymbol{\tau}_i)\Lambda_i \mathbf{J}_r(\boldsymbol{\tau}_i) \ , \tag{5}$$

where $\mathbf{J}_r(\boldsymbol{\tau})$, right Jacobian of $\mathcal{X} = \text{Exp}(\boldsymbol{\tau})$, is:

$$\mathbf{J}_r(\boldsymbol{\tau}) = \lim_{\epsilon \to 0} \frac{\text{Exp}(\boldsymbol{\tau} + \epsilon) \ominus \text{Exp}(\boldsymbol{\tau})}{\epsilon} \ . \tag{6}$$

Now that all $\boldsymbol{\tau}_i$ and $\Lambda_{\boldsymbol{\tau}_i}$ are defined in the tangent space of $\bar{\mathcal{X}}_0$, we can add all precision matrices to determine the total precision:

$$\Lambda_a = \sum_{i \neq a}^{N} \Lambda_{\boldsymbol{\tau}_i} \ . \tag{7}$$

Next we can combine the messages to obtain the tangent vector of the outgoing message:

$$\boldsymbol{\tau}_a = \Lambda_a^{-1} \sum_{i \neq a}^{N} \Lambda_{\boldsymbol{\tau}_i} \boldsymbol{\tau}_i \ . \tag{8}$$

Finally we apply this tangent vector to $\bar{\mathcal{X}}_0$ to obtain the group element which is the mean of the outgoing message:

$$\bar{\mathcal{X}}_a = \bar{\mathcal{X}}_0 \oplus \boldsymbol{\tau}_a \ , \tag{9}$$

and warp the total precision to the tangent space of $\bar{\mathcal{X}}_a$.

$$\Lambda_{\bar{\mathcal{X}}_a} = \mathbf{J}_r^{-\top}(\boldsymbol{\tau}_a)\Lambda_a \mathbf{J}_r^{-1}(\boldsymbol{\tau}_a) \ . \tag{10}$$

The outgoing message to factor a is: $\bar{\mathcal{X}}_a, \Lambda_{\bar{\mathcal{X}}_a}$, together with it's most recent point estimate $\bar{\mathcal{X}}_0$.

3.2. Message Passing at a Factor Node

The general definition of a Gaussian factor is:

$$f(\mathbf{x}) = K e^{-\frac{1}{2}\left[(\mathbf{z} - \mathbf{h}(\mathbf{x}))^\top \Lambda (\mathbf{z} - \mathbf{h}(\mathbf{x}))\right]} \ , \tag{11}$$

representing the probability of obtaining vector measurement $\mathbf{z}$ from a sensor. Here $\mathbf{h}$ is the functional form of the dependence of the measurement on state variables. Matrix Λ is the precision (inverse covariance) of the measurement, usually (but not necessarily) a constant matrix representing the accuracy of a sensor or the strength of a prior. $\mathbf{x}$ represents the state space of all of the N variables connected to the factor. To send messages from a factor node, f must be a Gaussian distribution of the state variables. If h is non-linear and all variables are in a standard state space like $\mathbb{R}^2$ or $\mathbb{R}^3$, $\mathbf{x}$ is a simple stacked vector, and we can follow the derivation in [13] to relinearise f to a Gaussian.

Here we consider the case where at least one of the connected variables represents a Lie Group. Now parts of $\mathbf{x}$ will be group elements rather than simple vectors, and messages to and from those variables take the form of group elements together with precision matrices in the tangent space of those elements.

We first use the most recent variable states from the incoming messages from all variables connected to the factor to form stacked state representation $\mathbf{x}_0$. For example for a factor connected to one $\mathbb{R}^2$ and two $\mathbf{SE}(2)$ variables:

$$\mathbf{x}_0 \quad = \quad \begin{bmatrix} \bar{\mathcal{X}}_0^1 \\ \bar{\mathcal{X}}_0^2 \\ \bar{\mathcal{X}}_0^3 \end{bmatrix} \in \langle \mathbb{R}^2, \mathbf{SE}(2), \mathbf{SE}(2) \rangle \ . \tag{12}$$

where $\bar{\mathcal{X}}_0^i$ is the variable i's state. We will use $\mathbf{x}_0$ as the linearisation point for the factor, and perform the calculations needed for message passing in the tangent space around this point. For our example, a vector $\boldsymbol{\tau}$ in this compound tangent space is:

$$\boldsymbol{\tau} = \begin{bmatrix} \boldsymbol{\tau}_1 \\ \boldsymbol{\tau}_2 \\ \boldsymbol{\tau}_3 \end{bmatrix} \in \langle \mathbb{R}^2, \mathfrak{se}(2), \mathfrak{se}(2) \rangle \ . \tag{13}$$

We define the boxplus operator $\boxplus$ which composes $\mathbf{x}_0$ with a compound tangent vector by choosing the appropriate operation for each variable:

$$\mathbf{x} = \mathbf{x}_0 \boxplus \boldsymbol{\tau} = \begin{bmatrix} \bar{\mathcal{X}}_0^1 + \boldsymbol{\tau}_1 \\ \bar{\mathcal{X}}_0^2 \oplus \boldsymbol{\tau}_2 \\ \bar{\mathcal{X}}_0^3 \oplus \boldsymbol{\tau}_3 \end{bmatrix} \ . \tag{14}$$

Note that for the first variable here, which is a standard vector variable, this composition is a simple addition. We also define $\boxminus$ as the inverse of this operation:

$$\boldsymbol{\tau} = \mathbf{x} \boxminus \mathbf{x}_0 \ . \tag{15}$$

Let us return to the factor definition of Equation (11) and rewrite it as:

$$f(\mathbf{x}) = K e^{-\frac{1}{2}E} \ , \tag{16}$$

where:

$$E = \mathbf{r}^\top \Lambda \mathbf{r} \ , \tag{17}$$

and

$$\mathbf{r} = \mathbf{z} - \mathbf{h}(\mathbf{x}) \,. \tag{18}$$

We linearise the non-linear measurement function $\mathbf{h}$ via a first-order Taylor expansion around $\mathbf{x}_0$:

$$\begin{aligned} \mathbf{h}(\mathbf{x}) &\approx \mathbf{h}(\mathbf{x}_0) + \mathtt{J}(\mathbf{x} \boxminus \mathbf{x}_0) & (19)\\ &= \mathbf{h}(\mathbf{x}_0) + \mathtt{J}\boldsymbol{\tau} \,. & (20) \end{aligned}$$

Here $\mathtt{J}$ is the Jacobian of the measurement function with respect to the compound tangent space:

$$\begin{aligned} \mathtt{J} &= \frac{\partial \mathbf{h}}{\partial \boldsymbol{\tau}}\Big|_{\mathbf{x}=\mathbf{x}_0} & (21)\\ &= \begin{pmatrix} \frac{\partial \mathbf{h}}{\partial \boldsymbol{\tau}_1} & \frac{\partial \mathbf{h}}{\partial \boldsymbol{\tau}_2} & \frac{\partial \mathbf{h}}{\partial \boldsymbol{\tau}_3} \end{pmatrix} \,. & (22) \end{aligned}$$

Substituting Equation (20) into Equation (18) and rearranging:

$$\begin{aligned} \mathbf{r} &= \mathbf{z} - (\mathbf{h}(\mathbf{x}_0) + \mathtt{J}\boldsymbol{\tau}) & (23)\\ &= -\mathtt{J}\boldsymbol{\tau} + (\mathbf{z} - \mathbf{h}(\mathbf{x}_0)) & (24)\\ &= -\mathtt{J}\boldsymbol{\tau} \\ &\quad + \mathtt{J}\mathtt{J}^\top(\mathtt{J}\mathtt{J}^\top)^{-1}(\mathbf{z} - \mathbf{h}(\mathbf{x}_0)) & (25)\\ &= -\mathtt{J}(\boldsymbol{\tau} \\ &\quad - \mathtt{J}^\top(\mathtt{J}\mathtt{J}^\top)^{-1}(\mathbf{z} - \mathbf{h}(\mathbf{x}_0))) \,. & (26) \end{aligned}$$

Then substituting Equation (26) into Equation (17), we obtain:

$$\begin{aligned} E &= (\boldsymbol{\tau} - \mathtt{J}^\top(\mathtt{J}\mathtt{J}^\top)^{-1}(\mathbf{z} - \mathbf{h}(\mathbf{x}_0)))^\top \\ &\quad \mathtt{J}^\top\Lambda\mathtt{J} \\ &\quad (\boldsymbol{\tau} - \mathtt{J}^\top(\mathtt{J}\mathtt{J}^\top)^{-1}(\mathbf{z} - \mathbf{h}(\mathbf{x}_0))) \,. & (27) \end{aligned}$$

We are interested in a Gaussian distribution in the tangent space of $\mathbf{x}_0$, which will have this general mean/precision form:

$$f(\mathbf{x}) = K e^{-\frac{1}{2}\left[(\boldsymbol{\tau}-\boldsymbol{\mu})^\top \Lambda'_\tau (\boldsymbol{\tau}-\boldsymbol{\mu})\right]} \,, \tag{28}$$

Matching Equation (28) with Equation (27), we obtain the mean and precision matrix of the linearised factor in tangent space:

$$\begin{aligned} \boldsymbol{\mu} &= \mathtt{J}^\top(\mathtt{J}\mathtt{J}^\top)^{-1}(\mathbf{z} - \mathbf{h}(\mathbf{x}_0)) & (29)\\ \Lambda' &= \mathtt{J}^\top\Lambda\mathtt{J} \,. & (30) \end{aligned}$$

We can then finally find the information vector of the linearised factor in tangent space $\boldsymbol{\eta}$:

$$\boldsymbol{\eta}_\mu = \Lambda'\boldsymbol{\mu} \,. \tag{31}$$

Having linearised the factor, we can now perform message passing. Consider our example factor connected to one $\mathbb{R}^2$ and two $\mathbf{SE}(2)$ variables. The factor's information vector and precision matrix are partitioned as follows:

$$\begin{aligned} \boldsymbol{\eta}_\mu &= \begin{pmatrix} \boldsymbol{\eta}_{\mu_1} \\ \boldsymbol{\eta}_{\mu_2} \\ \boldsymbol{\eta}_{\mu_3} \end{pmatrix} & (32)\\ \Lambda' &= \begin{bmatrix} \Lambda'_{11} & \Lambda'_{12} & \Lambda'_{13} \\ \Lambda'_{21} & \Lambda'_{22} & \Lambda'_{23} \\ \Lambda'_{31} & \Lambda'_{32} & \Lambda'_{33} \end{bmatrix} \,. & (33) \end{aligned}$$

If we choose the output variable to be the third variable, we need to first condition the factor on the incoming messages from variables 1 and 2 and then marginalise to achieve an output distribution over variable 3. Conditioning is achieved by adding as follows:

$$\begin{aligned} \boldsymbol{\eta}_C &= \begin{pmatrix} \boldsymbol{\eta}_{\mu_1} + \boldsymbol{\eta}_1 \\ \boldsymbol{\eta}_{\mu_2} + \boldsymbol{\eta}_2 \\ \boldsymbol{\eta}_{\mu_3} \end{pmatrix} & (34)\\ \Lambda'_{Cs} &= \begin{bmatrix} \Lambda'_{11} + \Lambda_1 & \Lambda'_{12} & \Lambda'_{13} \\ \Lambda'_{21} & \Lambda'_{22} + \Lambda_2 & \Lambda'_{23} \\ \Lambda'_{31} & \Lambda'_{32} & \Lambda'_{33} \end{bmatrix} \,, & (35) \end{aligned}$$

where $(\boldsymbol{\eta}_i,\ \Lambda_i)$ describe the incoming message $(\bar{\mathcal{X}}_i, \Lambda'_{\bar{\mathcal{X}}_i})$ from variable i in the tangent space of $\mathbf{x}_0$.

$$\begin{aligned} \boldsymbol{\tau}_i &= \bar{\mathcal{X}}_i \boxminus \mathbf{x}_0^i & (36)\\ \Lambda'_i &= \mathbf{J}_r^\top(\boldsymbol{\tau}_i)\Lambda'_{\bar{\mathcal{X}}_i}\mathbf{J}_r(\boldsymbol{\tau}_i) & (37)\\ \eta_i &= \Lambda'_i\boldsymbol{\tau}_i \,. & (38) \end{aligned}$$

Here $\mathbf{x}_0^i$ represents the ith block in the composite.

To complete message passing, from this joint distribution we must marginalise to obtain a distribution $\boldsymbol{\eta}_3$, Λ_3 over the output variable, and for this we follow Equations 44–51 from FutureMapping 2 [13].

This distribution is still in the tangent space of the linearised factor, so finally we transform back to a Lie Group element with information matrix in its own tangent space as follows to form the outgoing message $\bar{\mathcal{X}}_{o3}, \Lambda_{\bar{\mathcal{X}}_{o3}}$:

$$\begin{aligned} \boldsymbol{\tau}_3 &= \Lambda_3^{-1}\boldsymbol{\eta}_3 & (39)\\ \bar{\mathcal{X}}_{o3} &= \bar{\mathcal{X}}_0^3 \oplus \boldsymbol{\tau}_3 & (40)\\ \Lambda_{\bar{\mathcal{X}}_{o3}} &= \mathbf{J}_r^{-\top}(\boldsymbol{\tau}_3)\Lambda_3\mathbf{J}_r^{-1}(\boldsymbol{\tau}_3) \,. & (41) \end{aligned}$$

3.3. Belief Update at a Variable Node

At any stage, we can calculate a new marginal distribution at a variable node given all of the message passing that has happened to date; this is the current estimate of the position of a robot at a particular timestep. We combine *all* of the latest incoming messages $\bar{\mathcal{X}}_i, \Lambda_i$ at the variable node. First, we use the previously calculated belief mean $\bar{\mathcal{X}}_0$ at the variable (or some simple initialisation if this is the first time we are doing a belief update at this node), and transform all incoming messages into that tangent space using Equation (4) and Equation (5), to obtain $\boldsymbol{\tau}_i$ and $\Lambda_{\boldsymbol{\tau}_i}$.

Next we sum all precision matrices to determine the total precision:

$$\Lambda_{\boldsymbol{\tau}} = \sum^{N} \Lambda_{\boldsymbol{\tau}_i} \ , \tag{42}$$

and combine the messages to obtain the total tangent vector:

$$\boldsymbol{\tau} = \Lambda_{\boldsymbol{\tau}}^{-1} \sum^{N} \Lambda_{\boldsymbol{\tau}_i} \boldsymbol{\tau}_i \ . \tag{43}$$

Finally we apply this tangent vector to $\bar{\mathcal{X}}_0$ to obtain the group element which is the mean of the belief of the variable:

$$\bar{\mathcal{X}} = \bar{\mathcal{X}}_0 \oplus \boldsymbol{\tau} \ , \tag{44}$$

and warp the total precision into the tangent space of the new $\bar{\mathcal{X}}$:

$$\Lambda_{\bar{\mathcal{X}}} = \mathbf{J}_r^{-\top}(\boldsymbol{\tau}) \Lambda_{\boldsymbol{\tau}} \mathbf{J}_r^{-1}(\boldsymbol{\tau}) \ . \tag{45}$$

3.4. State Update at a Factor Node

By using all of the latest incoming messages at a factor, we can compute the current likelihood at that factor, and interpret this as a Mahalonobis distance which indicates the 'energy' of the factor in terms of how many standard deviations away from its most probable value the current states of the variables are. This can be used for visualisation (to see which factors in a network are currently the most 'stretched' during optimisation), or for applying the effect of a robust kernel to downweight its effect .

At a factor, we combine the latest states of all the connected variables to form a stacked state representation $\mathbf{x}_0$. This acts as a linearisation point, and is used to compute the current Mahalonobis distance of the factor:

$$M = \sqrt{(\mathbf{z} - \mathbf{h}(\mathbf{x}_0))^\top \Lambda (\mathbf{z} - \mathbf{h}(\mathbf{x}_0)))} \tag{46}$$

As shown in FutureMapping 2 [13], robust measurement functions in GBP can be handled by applying a robust scaling factor k_R to the precision matrix and the information vector of the factor potential:

$$\tilde{\Lambda}' = k_R \Lambda' \tag{47}$$

$$\tilde{\boldsymbol{\eta}}_\mu = k_R \boldsymbol{\eta}_\mu \ . \tag{48}$$

This scaling factor depends on the Mahalonobis distance, and can implement any robust kernel such as Huber, Tukey or DCS [1]. Through this mechanism a factor will weaken itself by the appropriate probabilistic amount when the combination of variables it connects to mean that the measurement the factor represents is likely to be an outlier. Overall, when many robust factors are connected in a large factor graph, GBP is able to use this mechanism to achieve lazy, reversible data association. We will see the impact of this in our results later, where we show that we are able to deal with large proportion of incorrect measurements in multi-robot localisation.

4. Demonstrations and Experiments

We present extensive simulation demonstrations of Robot Web localisation for the case of many robots with planar 2D motion and noisy odometry and inter-robot range bearing measurements. Our simulation uses metric units, and models an application like a warehouse setting where tens or hundreds of robots roam through an environment 100m across with randomly generated paths. Usually we add a handful of known beacon landmarks to the environment, whose positions are known in advance to all robots, but are widely spread so that robot-landmark measurements are much less frequent than robot-robot measurments. The main role of the the landmarks is to anchor the whole web to an absolute coordinate frame over long periods of operation.

Our simulation uses a fully distributed program structure equivalent to what could be achieved on a true multi-robot system. All storage and computation occurs within robot-specific modules, and all communication is via the Robot Web protocol.

4.1. Implementation Details

All variable nodes in the current simulation are represented using $\mathbf{SE}(2)$, and three different factors are implemented:

Anchor Factor: If needed, we can use unary anchor factors which are priors on the poses of robots before they start moving. In most experiments, we use these factors to represent fairly well known initial robot positions at the start of motion, though note that in Section 4.6 we show that new robots can be added to an existing web without any pose priors.

An anchor factor is defined as $\mathbf{h}(\mathbf{x}) = \mathbf{x}$ with measurement $\mathbf{z} \in \mathbf{SE}(2)$, which is the prior pose estimate. The uncertainty assigned to the anchor factors in our main experiments is: $\sigma_x = 0.1$m, $\sigma_y = 0.1$m, and $\sigma_\theta = 0.01$ rad.

Odometry Factor: For robot odometry, a binary factor $\mathbf{h}(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{x}_1 \ominus \mathbf{x}_2$ with measurement $\mathbf{z} \in \mathfrak{se}(2)$ represents a relative pose measurement. The uncertainty assigned to odometry factors per metre step is: $\sigma_x = 0.1$m, $\sigma_y = 0.01$m, and $\sigma_\theta = 0.01$ rad. As each robot moves along its x-axis, the uncertainty on x is higher than y.

Range Bearing Factor: We use a range bearing sensor for the measurements between robots, or between robots and landmarks. The measurement function is $\mathbf{h}(\mathbf{x}_1, \mathbf{x}_2) = (r, b)$, where r is the Euclidean distance between $\mathbf{x}_1, \mathbf{x}_2$ and b is the angle between the $\mathbf{x}_1, \mathbf{x}_2$ in the coordinate frame of $\mathbf{x}_1$. The range bearing measurement is defined as

$z \in \langle \mathbb{R}, \mathbf{SO}(2) \rangle$ The uncertainty assigned to range/bearing factors is: $\sigma_r = 0.01$m, $\sigma_b = 0.05$ rad, with the sensor range limited to 30m.

We use a communication pattern which simulates a limited peer-to-peer communication budget, where each robot connects to and reads the Robot Web page from other robots in a sequential, random pattern with closer robots more likely to be selected. The idea is that this is similar to a robot sequentially switching its Wi-Fi connection between peers with strong signals.

We generated a noisy distance sample between robot i and robot j as $d_{ij} \sim \mathcal{N}(\bar{d}_{ij}, 0.1)$, where d_{ij} is the random sample and $\bar{d}_{ij}$ is the ground truth distance between robot i and j. We define the neighbourhood of i, $N(i)$, as the set of robots which robot i can communicate with. The probability that robot i communicates with robot $j \in N(i)$ is:

$$p(j) = \frac{1/d_{ij}^2}{\sum_{n \in N(i)} 1/d_{in}^2} \, . \tag{49}$$

In this work, we assume that $N(i)$ includes all robots; however, this approach is general for any definition of neighbourhood.

4.2. Basic Operation with Some Known Beacons

We first show the essential operation of our method for a set of 20 robots which run in a square arena of width 100m with 4 known beacons: see Figure 2. They observe each other when within a range of 30m. Robots keep a sliding window of 5 poses, and all of the factors associated with at least one of those poses. We carry out 3 full GBP iterations per movement step of each robot. Each robot communicates with one other robot randomly per GBP iteration, reading any relevant outgoing messages from that robot's Robot Web page.

Our robots are all following randomly generated movement patterns, and have a simple avoidance mechanism which stops them when two robots would have collided.

Using a sliding window allows the average size of the factor graph, and the amount of computation needed, to remain fixed. This allows the system to operate over an arbitrarily long period while maintaining constant computational cost. What we lose by doing this is the possiblity to improve estimates of older variables in the graph using new observations. In our experiments, where the robots run in an area with a few known beacons, this is not a big problem, because the value of keeping old variables becomes vanishingly small over time, and in fact we have found that Robot Web works remarkably well with small time windows in realistically simulated application settings. We will discuss this further in Section 6.2.

4.3. Convergence and Computational Properties

A key property of our method is that the marginal estimates generated by message passing with a fixed computation and communication budget on our ever-changing factor graph may not necessarily be at complete convergence during live operation, though that is often not a problem if useful robot pose estimates are still achieved. Nevertheless, here we show that when enough computation and communication is regularly applied, the localisation results are convergent and estimates as accurate as a batch solution on a centralised processor can be achieved. Importantly, this can be achieved with highly efficient, realistic settings for distributed GBP computation and communication.

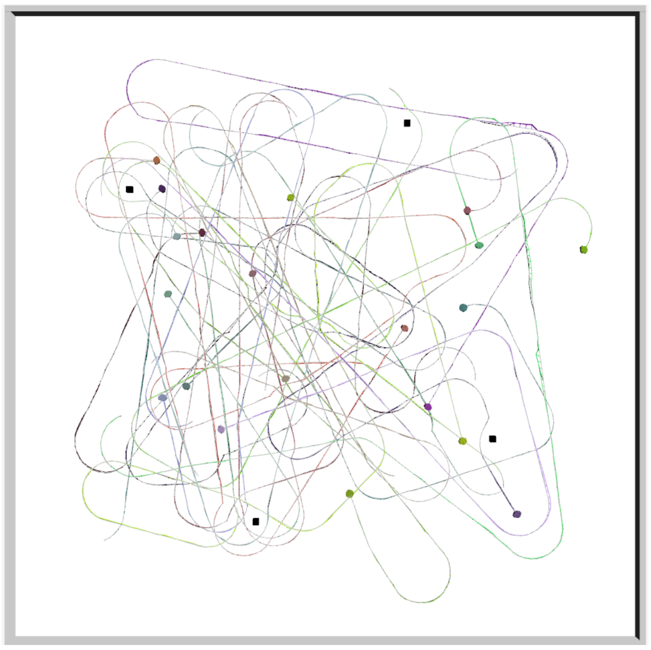
Here we present an experiment to compare the accuracy of distributed Robot Web GBP against a centralised solution of the same factor graph using GTSAM [15]. We run robots in a square arena of width 100m with 10 known beacons where all robots move through 100 pose steps. The robots observe each other when within a range of 30m. GBP linearises at every iteration, and is allowed to optimise for $[100, 50, 20, 20]$ iterations per step for $N = [16, 32, 64, 128]$ respectively.

We present results for general GBP, where each robot keeps a full history of pose variables, and Windowed GBP, where each robot maintains a sliding window of its most recent 5 poses and only processes messages relating to these. In these experiments, for both versions of GBP all robots are allowed to communicate with each other on every iteration.

We report the root mean square Absolute Trajectory Error (ATE), and root mean square Relative Pose Error (RPE) [39], averaged over 10 runs with randomised robot motions in Figure 3, for varying numbers of robots in the area. We see that GBP and GTSAM have similar ATE across all evaluations, with only a small loss of accuracy for GBP when the number of robots is low. Here the absolute metric relatively disconnected nature of the measurement graph favours a batch solver, with more GBP iterations probably needed for full convergence.

As the number of robots increases, the difference in ATE across the different approaches becomes negligible. Windowed GBP reaches comparable accuracy to GTSAM and GBP, even with a significantly smaller computational cost when compared to full GBP optimisation. We see that in the relative metric RPE, this similar performance is achieved even with a low number of robots, showing in particular how effective GBP is at optimising the relative properties of networks, as was observed in [13].

In terms of computational performance, it would not be meaningful to quantitatively report the speed of our C++ CPU simulation of the Robot Web algorithm, which is designed to be fully distributed across a large number of devices. However, in fact our simulation is pleasingly fast and can run comfortably in real-time on a laptop for problems


(a) With inter-robot and landmark observations.

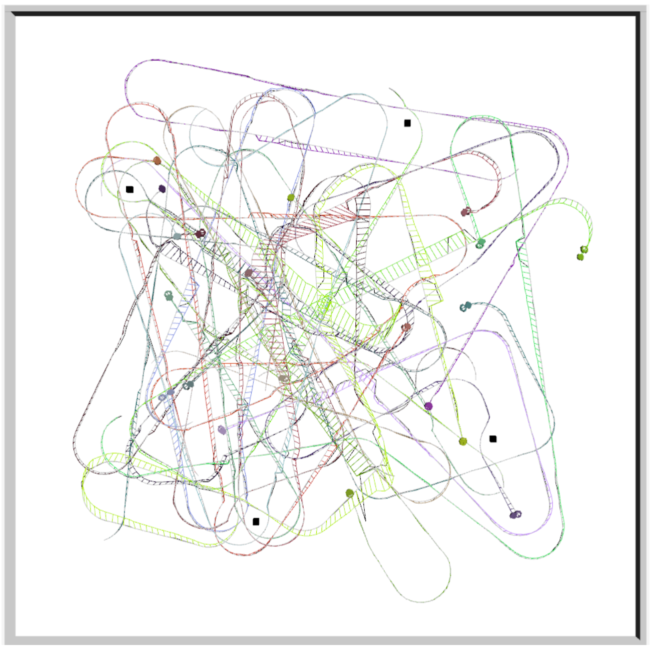
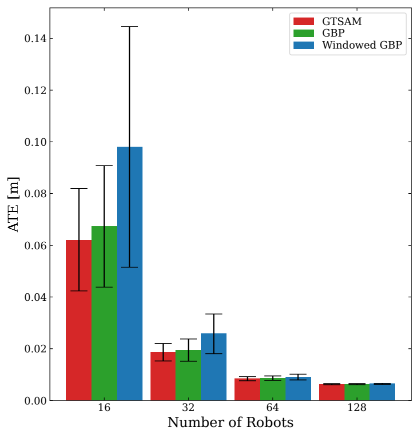
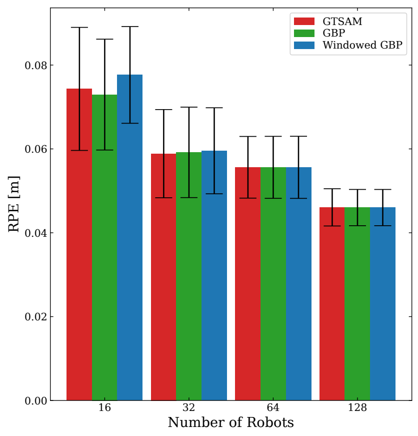

(b) With landmark observations only.

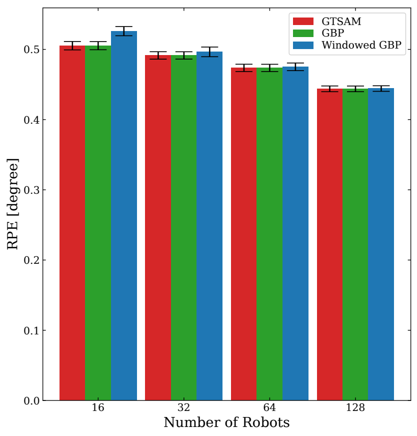
Figure 2: Demonstration of Robot Web with 20 robots and 4 known beacon landmarks — robots are coloured circles, and black squares are the landmarks. The whole simulated arena is 100m across, and robots make range-bearing measurements of nearby other robots and the landmarks when within a range of 30m. For a few seconds of operation (300 steps) here, we show the deviation between the planned groundn truth trajectory (grey) and actual trajectory (colour) for each robot, with the differences highlighted by connecting lines. We contrast (a), the full Robot Web, where estimated trajectories are consistently accurate, even for robots which rarely or never observe landmarks; and (b) where the robots estimate their poses individually only using landmark observations (no inter-robot sensor measurements). Here robots drift significantly when far from landmarks.



(a) Mean RMSE ATE



(b) Mean RMSE Translational RPE



(c) Mean RMSE Rotational RPE

Figure 3: N robots are moving around in an environment with 10 known landmarks for 100 poses each. GTSAM optimises the factor graph after every pose insertion rather than solving after all poses are inserted to keep the comparison fair. GBP uses the full factor graph to optimise, while Windowed GBP only uses only the last 5 poses. The results are the average of 10 runs with different random initialisation, and the error bar represents one standard deviation of uncertainty.

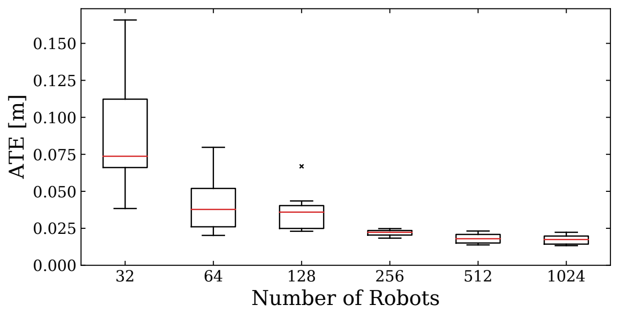


Figure 4: Extreme scaling: we increase the number of robots in the arena to over 1000, with each robot communicating with only one other per iteration of Windowed GBP, and therefore having a per-robot bounded computation and communication workload. The average ATE in all robots' poses continues to decrease as we increase the number of robots.

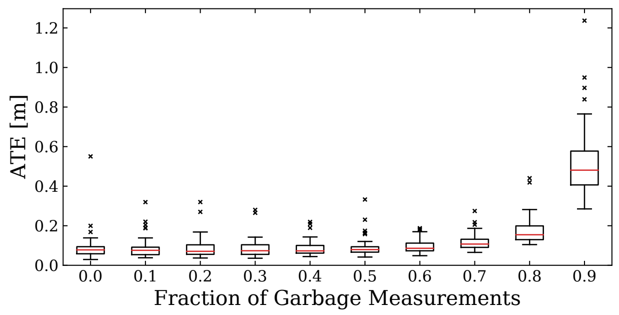


Figure 5: Robust factors enable remarkable resilience to a large fraction of 'garbage' inter-robot sensor measurements, with Average Trajectory Error remaining low up to 70–80% of corrupt measurements to which a large amount of uniform noise is added. The red line shows the median, the box extends from the first quartile to the third quartile, and the whisker extends from the box by 1.5 inter-quartile range, and the outliers are marked with a cross. 50 robots are moving in an environment with 10 known landmarks for 100 poses each. Each result is a summary of 50 runs with different random initialisation.

involving 100 robots or beyond using Windowed GBP, in particular because it is designed to take advantage of CPU parallelism using OpenMP.

Instead, we present an experiment which demonstrates the scaling properties of Windowed GBP in a mode where the computation and communication work *per robot* is bounded. Figure 4 shows the average ATE of all robot pose estimates as the number of interacting robots in our arena is raised from 32 to 1024. Each robot measures nearby robots, but is allowed to communicate with *one other robot* per GBP iteration. Robot Web handles this extreme packing and scaling straightforwardly, and the ATE for all robots continues to decrease as robots are added due to the favourable high inter-connectedness of the whole graph, despite the minimal communication allowed. These results indicate the true potential of Robot Web methods towards very high numbers of simple interacting devices.

4.4. Operation with Garbage Measurements and Robust Factors

A well known property of loopy Gaussian Belief Propagation is that while it obtains correct marginal mean estimates of variables, these are accompanied by overconfident precision/covariance measures. This is certainly true, and the properties of how and why this happens depend on the graph topology. However, a common misunderstanding among roboticists is that this means that GBP is not a useful algorithm because uncertainty gating for data association is not possible. A little consideration of any particular variable in the graph at convergence explains why we can in fact do properly-formulated data association using GBP. For a variable, we calculate its marginal belief by multiplying the probability distributions from the latest messages from all the factors it connects to. The fact that the marginal mean is correct for all variables at convergence shows that the *relative* precisions of local messages into a variable are convergently correct. *This* is what is important for data association — to be able to judge multiple hypotheses represented by different factors in the graph against each other.

We use robust factors on inter-robot measurements, and show that our approach can straightforwardly deal with a high fraction of 'garbage' sensor measurements. These garbage measurments do not need to be identified or dealt with in any explicit manner, but are simply automatically downweighted by the robust factors and end up having negligible influence on the whole graph. Note that this is a kind of 'lazy data association', because GBP's final commitment to whether to accept a measurement can often happen some time after the measurement being reported, or could even be reversed later on if other supporting measurements arrive.

Here, we demonstrate the robustness of GBP using the method for handling robust factors from FutureMapping 2 [13] and the robust kernel from DCS [1], which has robust characteristics in between Huber and Tukey. 50 robots run in an environment with 10 known landmarks for 100 poses each. Each robot performs 10 iterations of GBP per movement step, with a sliding window of 5, and at each GBP step robots communicate with one neighbour. In Figure 5, we show what happens to the ATE when we add to a random fraction of the range bearing sensor measurements a huge amount of uniform noise $r_n \sim \mathcal{U}(0, \pi)$ $b_n \sim \mathcal{U}(0, 30)$ with a fixed probability to simulate non-Gaussian noise which

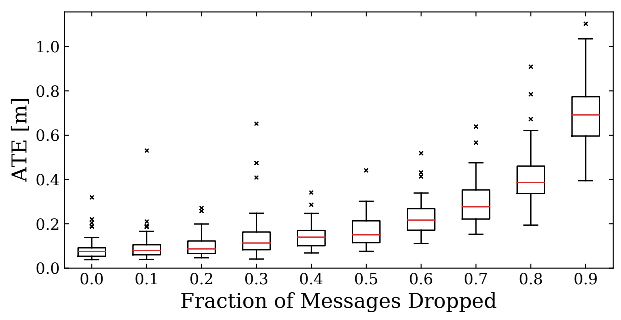


Figure 6: Robot Web is highly robust to a high fraction of randomly dropped messages. The red line shows the median, the box extends from the first quartile to the third quartile, the whisker extends from the box by 1.5 inter-quartile range, and outliers are marked with a cross. 50 robots are moving in an environment with 10 known landmarks for 100 poses each. The result is a summary of 50 runs with different initialisation.

makes these measurements essentially useless. We see that a huge fraction of up to 80% of measurements can be completely corrupted but still handled by the robust measurement kernel with very little effect on the overall accuracy of the network. This again shows the advantage of the heavily inter-connected network which GBP allows us to efficiently and incrementally optimise in a distributed manner. In this network, each pose estimate is highly overconstrained, and this is what allows the robust kernel to weed out garbage measurements.

4.5. Operation with Unreliable Communication

In multi-robot systems, another potential problem is the reliability of the communications. Often robots will communicate with best-effort, meaning messages can get lost in the network. Robustness against such data loss can be challenging; however, GBP is not significantly affected, as the message scheduling can be random. Here, we imagine that data transmission is quantised at the level of individual messages, as it might be with certain types of communication technology, and experiment to see the effect of the loss of a random fraction of messages between robots.

In Figure 6, we force the network to drop the messages randomly with a fixed probability which we gradually increase, and investigate how that affects ATE. For example, if Robot i sends 3 rows of message $\{M_1, M_2, M_3\}$ to Robot j, the network may drop M_2, and Robot j will only receive $\{M_1, M_3\}$. Other experimental settings are the same as in the previous experiment on corrupted measurements, except that we now use a fixed 10% setting garbage range bearing sensor measurements are garbage. In this experiment we also see very advantageous properties for GBP, which retains a low ATE up to at least 50% message loss in this setting.

4.6. Joining and Leaving the Robot Web

The Robot Web is fully dynamic, because each robot does not need any information about the group as a whole, so robots can join or leave freely. To demonstrate this, as shown in Figure 7, we run our simulation in a mode where new robots are added to the web one by one. There are 4 widely-spread known beacons in the scene, and initially just one robot moving, which has a poor pose estimate when it wanders away from the beacons. We randomly add high noise to 10% of 'garbage' range bearing sensor measurements are garbage. New robots are then dropped into the scene, each with a ground truth pose randomly chosen within the arena. It has no prior information about its pose, so is initialised at the centre of the arena and starts to participate in the Robot Web. It does not start to move until it believes that it has a good pose estimate. This decision is based on each robot monitoring the robust scaling of its factors, which is based on the Mahalonobis distance. Our implementation checks whether (a) the average scaling for all outgoing factors are > 0.95, and (b) that there are at least 8 different observations. Until these criteria are met, the newly-added robots send empty messages on the inter-robot factors and therefore do not affect the already-initialised robots until they are confident enough to start moving and properly taking part in the web.

We can see that once all 30 robots have been initialised, there is usually at least one robot in all areas of the scene and a strong localisation web has been produced where robots are rarely out of range of any others. Overall, the robot web has highly fault-tolerant behaviour, and accuracy which only gets better as more robots are involved and the whole graph has more redundancy.

5. Related Work

Robot Web uses the standard full Gaussian factor graph representing the multi-robot localisation problem, and is most closely related to the wealth of factor graph formulations and solvers in robotics, as well explained in the work of Dellaert and Kaess [16, 15]. Widely-used and high performance solver libraries for Gaussian factor graphs include GTSAM [14], Ceres [2], and g2o [27].

Most methods for inference on factor graphs assume a centralised computer with access to the whole graph and focus on either efficient batch solution or incremental inference on graphs that are continually changing. Centralised pose-graph optimisation algorithms suitable for multiple robots are well-explored in the literature [23, 5, 30, 26, 3]. MR-iSAM2 [45] extends iSAM2 [24] to build an incremental, centralised graph optimisation method for multiple robots. However, centralised methods require a base sta-

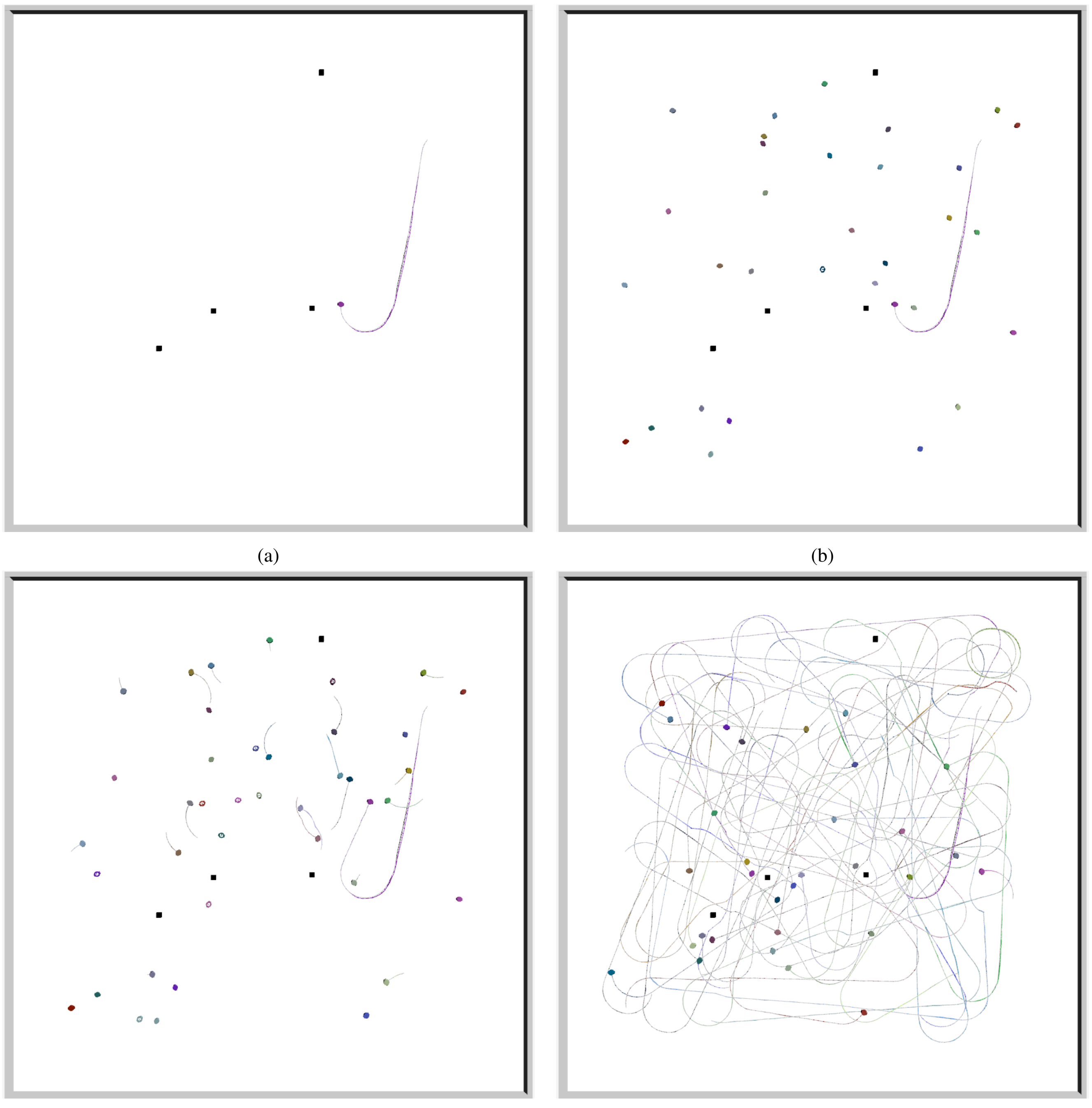

(a) (b) (c) (d)

Figure 7: Demonstration of Robot Web with dynamic robot insertion. (a) Initially, 1 robot moves around in a room with 4 beacons. (b) 30 new robots are added into the room. All new robots have no prior information about their position and stay stationary until localised. (c) As the initial robot moves around, it and other added robots make observations of each other. Once the added robots are localised, they start moving around. (d) All 30 robots are well localised and are moving around the scene. Path deviation is highlighted by the lines connecting the planned trajectory (grey) and actual trajectory.

tion, and are vulnerable to failure or this station, can require high communication bandwidth, create privacy concerns, and generally are not scalable [29].

There have been many previous attempts at distributed multi-robot localisation and SLAM which use local computation and peer-to-peer communication, but these are generally far more limited or specific than Robot Web; there is a useful recent survey by Halsted *et al*. [21]. Leung *et al.* demonstrated that in a distributed network of robots, where the network is never fully connected, it is possible to achieve exact and centralised-equivalent estimates for localisation [32] and SLAM [31]. Many of the existing results in multi-robot SLAM perform a version of gradient descent, albeit in a distributed fashion [19]. A recent example is the work by Tian *et al.* [40] utilizing block coordinate descent on the Riemannian manifold. The work by Tian *et al.* has also been used in Kimera-Multi [8] for pose-graph optimisation.

Other papers formulate the graph optimisation problem as a linear minimum mean-squared-error problem and utilize stationary iterative methods [20] such as the Jacobi method or the Gauss-Seidel method. For example, the Jacobi method is used by Barooah and Hespanha [6] and Aragues *et al.* [4] to solve the linearised system equations iteratively. The Jacobi method is implemented in a distributed manner as described in [17]. Choudhary *et al.* [9] demonstrated a distributed algorithm that is scalable to up to 50 robots. The algorithm utilizes a distributed Gauss-Seidel method for solving linearised equations. This method has been used as a back-end optimisation module in other works such as DOOR-SLAM [28] and the recent work by Cieslewski *et al.* [10]. It is also possible to use direct (non-iterative) methods to solve the linearised equations. Unlike iterative methods, the direct methods are exact, but become slow as the number of the variables grows. For instance, in the decentralized data fusion smoothing and mapping (DDF-SAM), Cunningham *et al.* [11] presented an algorithm that relies on Gaussian elimination.

A significant limitation of the distributed methods above is that they are synchronous, which means the robots must share their messages at predetermined times to ensure the shared information is up-to-date. In contrast, asynchronous methods offer the flexibility that the robots can operate at their own rate, without waiting for other robots. Examples are the works by Todescato *et al.* [42] and Tian *et al.* [41]. The first reference conducts the optimisation in Euclidean space, while the latter does the optimisation by computing the gradient descent on a Riemannian manifold in a distributed manner. The authors of these papers also investigated the convergence of their distributed algorithms under communication delay.

Robot Web is also asynchronous and distributed, but both much more simple in formulation and more general than these methods. Significantly, Tian *et al.* [41] assume Gaussian noise in their formulation, and are unable to handle outliers, whereas Robot Web supports general robot and sensor models and allows robust factors, making it robust to large fractions of non-Gaussian garbage measurements.

There has been some work on multi-agent distributed localisation using variations of belief propagation in the sensor networks community. For instance, Schiff *et al.* [37] performed multi-robot localisation using non-parametric belief propagation. Wymeersch *et al.* [44] also used belief propagation to perform cooperative positioning in a distributed manner. They used the sum-product algorithm over a factor graph in an ultra-wideband network. Then Caceres *et al.* [7] extended [44] to a network composed of GNSS nodes. These distributed positioning methods were later generalized by adding nonlinear measurement models and utilizing Gaussian message passing [33], [34]; and in 2017, Wan *et al.* [43] proposed a distributed multi-robot SLAM algorithm, using belief propagation. In their method, a mixture of Gaussian and non-parametric models was used to handle nonlinear models. They also assumed measurements are affected by Gaussian noise and use synchronous message passing.

Robot Web goes far beyond these methods to presents a general framework for general robots and sensors. It defines for the first time an open, asynchronous communication framework, and via the focus on GBP with robust factors enables highly robust and scalable performance.

6. Ongoing Research Issues

We have demonstrated the essential operation of the Robot Web in simulation, and provided all of the details which should enable a real-world, truly distributed implementation on multiple robots.

The properties of the method are extremely promising, and we are motivated to explore a wide number of research issues which will enable it to be more generally capable and potentially an important layer in ongoing multi-robot system development, where as explained in the introduction our vision is that many different companies could create robots and devices which can use the Robot Web to cooperate in a fully distributed manner.

Here we discuss some important research directions going forward.

6.1. More General Parameterisation

Our current implementation makes several simplifying assumptions, but we believe that all of these are fairly straightforward to remove within the Robot Web framework with some further work.

- We currently assume that inter-robot measurement factors, stored by the robot with the sensor, always correspond to observations of the position of the centre of the second robot. This would already allow a practical implementation for 2D planar robots which each carry a single observable beacon above their centres. More realistically, each robot might have several or many observable features, and these will be located on any point on its structure. We can deal with this by adding additional internal variables to the second robot, connected to its main pose by ‘perfect’ factors, representing the positions of the observable features relative to its body, with positions that only need to be known to the second robot.

- Our current assumption that all robots have pose variables defined at the same rate and at corresponding times could be relaxed by measurement factors which connect to multiple variable at the receiving robot and interpolate the measurement between poses.
- We assume that robots have fully calibrated sensors which give unbiased measurements, but it would be straightforward to add calibration parameters as extra variables to each robot’s internal graph as needed.

We might take the Robot Web idea even further to also apply *inside* a single robot’s modular body. The different parts, actuators and sensors that make up the robot might use Web interfaces between them to enable distributed joint estimation and very general modularity.

6.2. Efficient Long-Term Operation

If we keep the full history of all pose variables for each robot, and all measurement factors, eventually the computation, storage and communication capacity of each robot would become overloaded. Of these, inter-robot communication is likely to be the main bottleneck. In Section 4.2 we showed one simple approach to dealing with this via time windowing, where poses older than a threshold are discarded, and this gives good performance when robots have bounded drift due to the presence of known beacons.

A more general approach to bounding the growth of the graph could be based on incremental abstraction [36], where past variables and factors are not deleted but grouped into more efficient blocks with minimal loss of accuracy. For instance, a set of well-estimated pose variables from the past could be grouped into an abstract trajectory segment, represented by far fewer variables. Factors could also be grouped. Achieving this incremental abstraction in fully distributed way across multiple robots however will require substantial research.

6.3. Communication

For efficiency and long-term scalability, we should also consider the format of messages and the way that they are tranferred between robots. It seems clear that most messages in GBP, which explicitly represent quantities with known and often large uncertainty, need not be represented by floating point numbers with many significant digits of precision. An important piece of research will be to determine the minimum number of bits of numeric representation needed for each message. We believe that there should be a close theoretical relationship between the information content of a message, defined the amount of information entropy it is expected to reduce in state estimates [12], and the number of bits of data transmission that are needed to send it.

This analysis will be especially important if we consider methods of inter-robot communication which are not the explicit downloading of web-pages over Wi-Fi, but perhaps much more lightweight and ad-hoc channels such as Bluetooth. A further remarkable possibility is that the robots could even communicate visually without the need for radio at all. Imagine that each robot carried a screen displaying its current ‘web-page’, and that robots would simply read each other’s screens with cameras. The one way transmission of messages in our current formulation means that this would work exactly as well as a Wi-Fi method. Visual communication would work much better still if the information on a web-page could be encoded into the most efficient possible binary form and presented as a QR code, or a pattern of flashing LEDs for instance.

6.4. Security

We have shown in Section 4.4 that the Robot Web is remarkably robust to garbage measurements due for instance to faulty sensors, but it seems likely that a ‘bad actor’ robot could choose to deliberately send misleading messages which are subtly altered in a systematic way which would not be detected by simple robust factors.

Again, imagining a long-term future where many different devices from different companies or organisations are to take part in the Web, it may be necessary to introduce a mechanism via which robots can sign and verify messages to guarantee trust in a way similar to that developed for other decentralised systems. But more fundamentally, could we design into the Robot Web mechanisms which encourage every device to share honest and accurate information, because the more they do so the better the quality of information that will be shared back with them?

6.5. Privacy

It would be natural that some robots want to take part in the Robot Web without revealing all of their internal details, and to some extent privacy is already built into its design. Robots can each be running entirely proprietry internal software, and only need to publish Web messages in the correct format to participate. Further, details about how a robot’s sensors and actuators work are held only by that robot, and do not need to be revealed to other devices.

More privacy is possible if desired. If we extend the parameterisation as suggested in Section 6.1 such that the measurable points on a robot are at locations which are not its centre, the relative locations of those features only need to be known to the robot itself, and the explicit pose of the robot need not be published to other robots explicitly. The measurable features (e.g. flashing beacons around or near the main robot) could even be moving continuously or turned on and off for more privacy.

6.6. SLAM and General Federated Learning

The method we have presented so far is for pure multi-robot localisation, where robots estimate their locations by making measurements of each other. It is simple to include fixed landmarks in the scene, either if the positions of these are known in advance to all, or if the landmarks themselves actively communicate and take part in the web, essentially as 'parked' robots.

It will require more research to enable general SLAM, where the robots collaborate to build a shared map of novel static scene elements. We have already given thought to how an initial SLAM implemenation may not be difficult, as long as the number of potential landmarks in the scene is small and they are clearly distinct such that a robot can associate a unique identifier with each. When a robot discovers a landmark, it should add a landmark variable to its local factor graph, and then publish the pose and identifier of that landmark within its Robot Web page. If another robot reads the page and has mapped a landmark with a matching identifier, it will also have its own copy of the landmark variable, which will initially have a different position estimate. The robots should interpret messages they exchange about the same landmark as if these copy variables are connected by a very strong 'identity factor', which will force the copies towards the same value. Over time, as many robots move through the same area, the maps they build of multiple landmarks should all converge towards equal copies of each other.

Other global parameters of a scene could be estimated/learned in this federated style, by using identity factors to drive distributed estimates together; for instance a calibration variable affecting the performance of all robots such as the slipperiness of the floor. Ultimately much more general learning of other characteristics of the scene could be developed, all within the same framework of variable inference using GBP.

More advanced SLAM, where the whole map is too large for any one robot to carry, is clearly more difficult to distribute without a global server, and this will require much further research.

6.7. Robot Web for Planning

In this paper we have shown how GBP allows a distributed Robot Web approach to multi-robot estimation and specifially localisation, but in fact a very similar approach is also feasible for the other key parts of autonomous robot operation: planning and control.

Dellaert and colleages in particular [15] have recently shown that many planning problems in robotics, where the future actions of a robot are to be decided based on satisfying a number of constraints, can be formulated as Gaussian factor graphs and therefore solved using exactly the same computation machinery and software libraries as estimation problems. For example, GPMP2 performs high degree-of-freedom motion planning using GTSAM [18].

This argument clearly extends to the distributed case, and therefore distributed multi-robot collaborative planning can be tackled with the Robot Web and GBP. Example applications are the distributed coordination of many vehicles on a road network, or the self-organisation of many robots into joint configurations and shapes. We are actively working on these problems and will publish new results soon. A particularly interesting issue going forward is the combination of distributed estimation and planning into a single formulation and computational structure and we believe that this is feasible.

7. Robot Web Software

A key tenet of our approach, very much inspired by the original World Wide Web concept, is to use a simple protocol to connect many systems, which could have entirely different internal implementations. The Robot Web is a concept which is different from a software library for factor graph inference or even multi-robot localisation. It just specifies, in the simplest way possible, a communication protocol between devices, each of which carries out its own internal proceessing in whichever way it chooses, whether open source or proprietary.

8. Discussion and Conclusions

We have presented a breakthrough method for distributed multi-robot localisation in the context of a larger 'Robot Web' vision for how heterogeneous groups of intelligent robots and devices of the future could cooperate and coordinate. This approach could be important at a time when many different companies and organisations are building spatially aware devices, and offers a distributed, inter-operable alternative to a single unified cloud maps solution.

As the performance and scale of many-robot systems may greatly improve due to work such as ours, it is important to consider potential ethical concerns. A robust, large-scale robot group or 'swarm' has many possible positive applications, such as the automation of farming or environmental surveillance via many low-cost devices, which could be much more efficient overall than a small number of large devices. However, there are possible negative uses such as swarms of weaponised drones.

A key question is about the possible danger of a robot group which is able to localise and control itself without the need for centralised control — could the group cause harm by escaping the control of its human operators? On the other hand, there is an ethical risk with following a path towards fully centralised control of AI devices using a cloud service run by a single large organisation such as a company or nation state, which possibly places a great deal of power in a single place.

We believe that our paper overall could indicate a positive direction for the operation of distributed multi-robot systems via the specification that the Robot Web allows and demands of an **open communication protocol**. If the majority of moving intelligent devices were to take part in such a system by publishing and reading localisation messages via this open protocol, it would be greatly to the advantage of any newly built devices to also take part, to exchange open messages, and to benefit from the system. This would mean that the whole system might work in a way similar to the World Wide Web, and some degree of global control would be possible via the interpretability of the protocol and perhaps more specific safety measures built into it. We believe that it is better for devices to be exchanging clearly interpretable geometric information than cryptic coded messages (as would emerge for instance in a possible distributed 'graph neural network' system for localisation, where the format of messages is learned rather than designed — and we should add here our view that a learned alternative to our method is also likely to be far less flexible and efficient).

These are ongoing issues to be debated as the technology advances, and we as authors believe that researchers should openly engage with these issues and play a part in designing the correct principles into the technology.

In the longer term future, the distributed coordination of intelligent moving systems is a key part of the concept of 'intelligent matter', where distribution and communication might be at the microscopic level to enable new classes of technology such as micromachines [22] which can self-organise in ways that might approach the capabilities of biological systems [25]. Efficient, robust distributed localisation will be one of the most important enabling layers of such systems.

Acknowledgements

We are grateful to many researchers with whom we have discussed some of the ideas in this paper, especially from the Dyson Robotics Lab and Robot Vision Group at Imperial College, and SLAMcore. We would particularly like to thank Aalok Patwardhan, Stefan Leutenegger, Raluca Scona, Talfan Evans, Eric Dexheimer, Seth Nabarro, Mark van der Wilk, Owen Nicholson, Rob Deaves, Pablo Alcantarilla, Jacek Zienkiewicz, Amanda Prorok, Mac Schwager, Frank Dellaert, Michael Kaess, Tim Barfoot, Richard Newcombe.

References

[1] P. Agarwal, G. D. Tipaldi, L. Spinello, C. Stachniss, and W. Burgard. Robust map optimization using dynamic covariance scaling. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2012. 7, 10

[2] S. Agarwal, M. K., and Others. Ceres solver. `http://ceres-solver.org`. 11

[3] L. A. A. Andersson and J. Nygards. C-SAM : Multi-robot SLAM using square root information smoothing. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2008. 11

[4] R. Aragues, L. Carlone, G. Calafiore, and C. Sagues. Multi-agent localization from noisy relative pose measurements. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2011. 13

[5] T. Bailey, M. Bryson, H. Mu, J. Vial, L. McCalman, and H. Durrant-Whyte. Decentralised cooperative localisation for heterogeneous teams of mobile robots. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2011. 11

[6] P. Barooah and J. Hespanha. Distributed estimation from relative measurements in sensor networks. In *International Conference on Intelligent Sensing and Information Processing*, 2005. 13

[7] M. A. Caceres, F. Penna, H. Wymeersch, and R. Garello. Hybrid cooperative positioning based on distributed belief propagation. *IEEE Journal on Selected Areas in Communications*, 29(10):1948–1958, 2011. 13

[8] Y. Chang, Y. Tian, J. P. How, and L. Carlone. Kimera-Multi: a system for distributed multi-robot metric-semantic simultaneous localization and mapping. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2021. 13

[9] S. Choudhary, L. Carlone, C. Nieto, J. Rogers, H. Christensen, and F. Dellaert. Distributed mapping with privacy and communication constraints: Lightweight algorithms and object-based models. *International Journal of Robotics Research (IJRR)*, 36(12):1286–1311, 2017. 13

[10] T. Cieslewski, S. Choudhary, and D. Scaramuzza. Data-efficient decentralized visual SLAM. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2018. 13

[11] A. Cunningham, M. Paluri, and F. Dellaert. DDF-SAM: Fully distributed SLAM using constrained factor graphs. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, 2010. 13

[12] A. J. Davison. Active Search for Real-Time Vision. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2005. 14

[13] A. J. Davison and J. Ortiz. FutureMapping 2: Gaussian Belief Propagation for Spatial AI. *arXiv preprint arXiv:1910.14139*, 2019. 2, 4, 5, 6, 7, 8, 10

[14] F. Dellaert. Factor graphs and GTSAM. Technical Report GT-RIM-CP&R-2012-002, Georgia Institute of Technology, 2012. 11

[15] F. Dellaert. Factor graphs: Exploiting structure in robotics. *Annual Review of Control, Robotics, and Autonomous Systems*, 4:141–166, 2021. 8, 11, 15

[16] F. Dellaert and M. Kaess. Factor Graphs for Robot Perception. *Foundations and Trends in Robotics*, 6(1–2):1–139, 2017. 11

[17] V. Delouille, R. Neelamani, and R. Baraniuk. Robust distributed estimation in sensor networks using the embedded polygons algorithm. In *International Symposium on Information Processing in Sensor Networks*, 2004. 13

[18] J. Dong, M. Mukadam, F. Dellaert, and B. Boots. Motion Planning as Probabilistic Inference using Gaussian Processes and Factor Graphs. In *Proceedings of Robotics: Science and Systems (RSS)*, 2016. 15

[19] T. Fan and T. Murphey. Majorization minimization methods for distributed pose graph optimization with convergence guarantees. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, 2020. 13

[20] L. A. Hageman and D. M. Young. *Applied Iterative Methods*. Academic Press, 1981. 13

[21] T. Halsted, O. Shorinwa, J. Yu, and M. Schwager. A survey of distributed optimization methods for multi-robot systems. *ArXiv*, abs/2103.12840, 2021. 13

[22] T. Huang, H. Gu, and B. Nelson. Increasingly intelligent micromachines. *Annual Review of Control, Robotics, and Autonomous Systems*, 5:25.1–25.32, 2022. 16

[23] V. Indelman, E. Nelson, N. Michael, and F. Dellaert. Multi-robot pose graph localization and data association from unknown initial relative poses via expectation maximization. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2014. 11

[24] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. Leonard, and F. Dellaert. iSAM2: Incremental Smoothing and Mapping Using the Bayes Tree. *International Journal of Robotics Research (IJRR)*, 31(2):216–235, 2012. 11

[25] C. Kaspar, B. J. Ravoo, W. G. van der Wiel, and W. H. P. Wegner, S. V. annd Pernice. The rise of intelligent matter. *Nature*, 594:345–355, 2021. 16

[26] B. Kim, M. Kaess, L. Fletcher, J. Leonard, A. Bachrach, N. Roy, and S. Teller. Multiple relative pose graphs for robust cooperative mapping. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2010. 11

[27] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. g^2o: A General Framework for Graph Optimization. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2011. 11

[28] P. Lajoie, B. Ramtoula, Y. Chang, L. Carlone, and G. Beltrame. DOOR-SLAM: Distributed, online, and outlier resilient SLAM for robotic teams. *IEEE Robotics and Automation Letters*, 5(2):1656–1663, 2020. 13

[29] P.-Y. Lajoie, B. Ramtoula, F. Wu, and G. Beltrame. Towards collaborative simultaneous localization and mapping: a survey of the current research landscape. *ArXiv*, abs/2108.08325, 2022. 12

[30] M. T. Lazaro, L. M. Paz, P. Pinies, J. A. Castellanos, and G. Grisetti. Multi-robot SLAM using condensed measurements. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, 2013. 11

[31] K. Y. Leung, T. D. Barfoot, and H. H. Liu. Decentralized cooperative SLAM for sparsely-communicating robot networks: A centralized-equivalent approach. *Journal of Intelligent and Robotic Systems (JIRS)*, 66(3):321–342, 2012. 13

[32] K. Y. K. Leung, T. D. Barfoot, and H. H. T. Liu. Decentralized localization of sparsely-communicating robot networks: A centralized-equivalent approach. *IEEE Transactions on Robotics (T-RO)*, 26(1):62–77, 2010. 13

[33] B. Li, N. Wu, H. Wang, J. Kuang, and C. Xing. Gaussian message-based cooperative localization on factor graph in wireless sensor networks. In *International Conference on Wireless Communications and Signal Processing*, 2014. 13

[34] B. Li, N. Wu, H. Wang, P.-H. Tseng, and J. Kuang. Gaussian message passing-based cooperative localization on factor graph in wireless networks. *Signal Processing*, 111:1–12, 2015. 13

[35] J. Ortiz, T. Evans, and A. J. Davison. A visual introduction to Gaussian Belief Propagation. *arXiv preprint arXiv:2107.02308*, 2021. 1, 3, 4

[36] J. Ortiz, T. Evans, E. Sucar, and A. J. Davison. Incremental Abstraction in Distributed Probabilistic SLAM Graphs. *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2022. 14

[37] J. Schiff, E. B. Sudderth, and K. Goldberg. Nonparametric belief propagation for distributed tracking of robot networks with noisy inter-distance measurements. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, 2009. 13

[38] J. Solà, J. Deray, and D. Atchuthan. A micro Lie theory for state estimation in robotics. *arXiv:1812.01537*, 2018. 4

[39] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A Benchmark for the Evaluation of RGB-D SLAM Systems. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, 2012. 8

[40] Y. Tian, K. Khosoussi, D. M. Rosen, and J. P. How. Distributed certifiably correct pose-graph optimization. *IEEE Transactions on Robotics (T-RO)*, 37:2137–2156, 2021. 13

[41] Y. Tian, A. Koppel, A. S. Bedi, and J. P. How. Asynchronous and parallel distributed pose graph optimization. *IEEE Robotics and Automation Letters*, 5(4):5819–5826, 2020. 13

[42] M. Todescato, A. Carron, R. Carli, and L. Schenato. Distributed localization from relative noisy measurements: a robust gradient based approach. In *European Control Conference (ECC)*, 2015. 13

[43] J. Wan, S. Bu, J. Yu, and L. Zhong. Distributed simultaneous localization and mapping for mobile robot networks via hybrid dynamic belief propagation. *International Journal of Distributed Sensor Networks*, 13(8):1–19, 2017. 13

[44] H. Wymeersch, J. Lien, and M. Z. Win. Cooperative localization in wireless networks. *Proceedings of the IEEE*, 97(2):427–450, 2009. 13

[45] Y. Zhang, M. Hsiao, J. Dong, J. Engel, and F. Dellaert. MR-iSAM2: Incremental smoothing and mapping with multi-root Bayes tree for multi-robot SLAM. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, 2021. 11