Distributing Collaborative Multi-Robot Planning with Gaussian Belief Propagation

Aalok Patwardhan[1], Riku Murai[2] and Andrew J. Davison[1]

***Abstract*— Precise coordinated planning over a forward time window enables safe and highly efficient motion when many robots must work together in tight spaces, but this would normally require centralised control of all devices which is difficult to scale. We demonstrate GBP Planning, a new purely distributed technique based on Gaussian Belief Propagation for multi-robot planning problems, formulated by a generic factor graph defining dynamics and collision constraints. In simulations, we show that our method allows extremely high performance collaborative planning where robots are able to cross each other in busy, intricate scenarios. They maintain shorter, quicker and smoother trajectories than alternative distributed planning techniques even in cases of communication failure. We encourage the reader to view the accompanying video demonstration.**

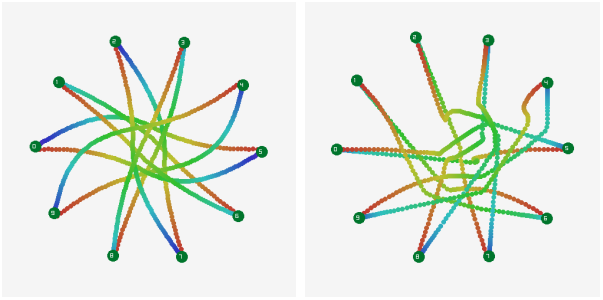
I. INTRODUCTION

As automation increases, multiple robotic devices will need to operate together in limited spaces, whether working robots in a home, farm or industrial setting, or autonomous vehicles on a road network. To achieve both safety and efficiency, these robots must unavoidably *coordinate* when planning their motions and other actions. At the most basic level, robots must take minimal account of each other's plans in order to avoid collisions. When coordination is stronger, extremely high efficiency is possible, and large numbers of devices can move smoothly and intricately around each other as they carry out their tasks.

It might be assumed that highly coordinated planning requires centralised control, where a single computer receives state information from all robots and solves a unified multiple trajectory optimisation problem before sending commands back to them all. In this paper we show that precise performance is in fact possible from an alternative distributed technique which uses efficient local per-robot computation and purely peer-to-peer communication.

Our solution formulates multi-robot planning as a dynamic optimisation problem characterised by a factor graph with variables representing robot positions and velocities in Euclidean space over a bounded forward time window. These are connected by constraint factors which represent each robot's dynamics and the need to avoid collisions with other robots as well as static obstacles. Planning consists of performing inference on the factor graph to determine marginal distributions for the future poses which best satisfy all constraints.

To achieve distributed computation, the full joint factor graph is divided up into fragments which individual robots have responsibility for storing and updating. The graph is optimised using Gaussian Belief Propagation (GBP), an iterative and distributed message passing algorithm which can achieve the same global solution as a centralised solver. Messages which need to be sent between the graph fragments held by different different robots can be transmitted by regular peer-to-peer communication of small messages. The particularly appealing property of GBP for multi-robot systems is that global convergence can be achieved with many types of communication schedule, and we show that it can work well even when peer-to-peer communication is unreliable and a large fraction of messages may be lost.



(a) GBP planner: smooth and efficient trajectories (b) ORCA: longer and jerky trajectories

Fig. 1: Experiment where 10 robots must cross to opposite sides of a circle without colliding. Our GBP planner (a) achieves shorter and smoother trajectories than ORCA (b).

Our GBP Planning algorithm is generic, but we demonstrate it here in simulations of two scenarios of multi-robot planning in tight spaces. In the first, robots must simultaneously exchange places across a circle formation to move from one fixed configuration to another. In the second, two continuous streams of robots meet at a busy junction and aim to cross each other while maintaining high flow rates. In both, the robots jointly plan and update their trajectories online to achieve smooth (dynamically realistic) and efficient trajectories, and we quantitatively measure performance significantly beyond what has been demonstrated with alternative distributed planning techniques.

II. RELATED WORK

Multi-robot trajectory planning problem in the presence of safety and dynamic constraints is NP-hard in general [1], but iterative methods can find practically useful solutions for planning over a forward time window. Techniques can be categorised into centralised and distributed algorithms. Centralised approaches can achieve high accuracy, but require all robots to be coordinated by a hub with full knowledge

of robot states and constraints; this represents clear challenges to scalability. We therefore focus here on distributed approaches, like ours, which can operate with per-robot parallelism and require only peer-to-peer communication.

In single-robot planning, a robot solves for a forward path which takes it towards its goal while balancing the desire to move smoothly and in accordance with dynamics constraints against the need to avoid obstacles. Multi-robot planning in tight spaces essentially consists of conflict resolution, because each robot's future planned states become obstacles for every other robot. One approach is priority-based planning such as in [2], where high priority robots largely use single-robot planning methods and lower-priority robots treat them as as dynamic obstacles with unchanging trajectories which must be avoided. This means that lower priority robots will often have to wait and take a lot of time to reach their goals.

It is more generally interesting to aim at situations where all robots have the same or similar priority. Here conflict resolution becomes more subtle because every change to one robot's plan could affect every other one. A widely used distributed multi-robot planner is the Optimal Reciprocal Collision Avoidance (ORCA) algorithm [3], which assumes that agents share equal responsibility for pairwise collision avoidance. In ORCA, every robot has perfect instantaneous knowledge of the position and velocities of its neighbours. Robots react accordingly by changing their own velocities, but cannot make decisions to resolve conflicts which may arise further into the future. As we will show in comparative results later, ORCA produces trajectories which are often jerky when robots crowd into small spaces.

Several existing methods use iterative communication to enable longer look-ahead. The work in [4] presents an online solver using Alternating Direction Method of Multipliers (ADMM) for a small number of robots moving through an environment with moving obstacles. Here the robots move together in formation and their paths do not intersect so few conflicts need to be resolved. The distributed model predictive control algorithm in [5] has been demonstrated for impressive criss-crossing behaviour in teams of real quadrotor robots, though again the gaps between robots are generally larger than we are able to achieve with our approach. The graph neural network method in [6] tackles learned multi-robot planning with local communication, though so far only in a very abstract grid-world setting.

While the methods above use elements of standard energy minimisation theory, to achieve distribution they combine this with more non-standard methods, for instance for the determination of trajectory collisions in [5]. In our work we show for the first time a way to use standard optimisation machinery for distributed planning. We are inspired in particular by methods like GPMP2 [7]. This is planner for a single robot in a known static environment, but importantly showed that planning with dynamics could be formulated as inference on a Gaussian factor graph, and therefore that general factor graph solvers such as GTSAM [8] which are normally used for estimation could be easily adapted to planning applications. The definition of all cost terms in

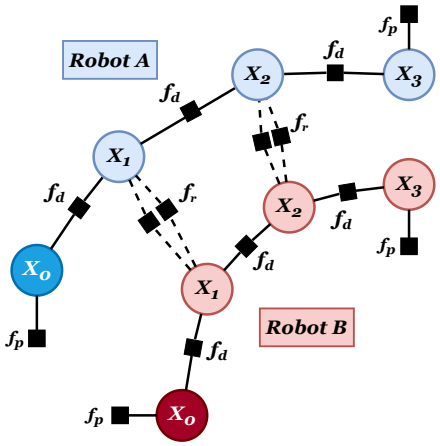


Fig. 2: Multi-robot planning factor graph with two robots (A and B) moving from bottom-left to top-right highlighting pose factors f_p, dynamics factors f_d and inter-robot factors f_r. Note that unary static obstacle factors f_o are connected to each robot's states but for clarity are not shown here.

a planning cost function as quadratic in a set of variables means that planning is solved by the same sparse least-squares inference computation as arises in robot estimation problems such as SLAM.

Very recently, Murai *et al.* showed in their Robot Web work [9] that a highly distributed solution to the distributed estimation problem of multi-robot localisation is possible using efficient distributed computation via Gausian Belief Propagation and peer-to-peer message passing. In this paper, we adapt this method to the domain of multi-robot collaborative planning, and show that it is possible to take advantage of the same distributed computation structure. We will show that this enables highly coordinated multi-robot motion planning beyond previous distributed planning techniques while remaining very flexible and general.

III. THEORETICAL BACKGROUND

A. Factor Graphs

A factor graph represents the factorisation of a joint function $p(\mathbf{X})$ into components f_s which depend on subsets $\mathbf{X}_s$ of all variables $\mathbf{X}$:

$$p(\mathbf{X}) = \prod_s f_s(\mathbf{X}_s) \ . \tag{1}$$

Graphically, it is an undirected bipartite graph, where each factor is represented by a square node attached by edges to circular nodes representing the variables it involves.

In a Gaussian factor graph all factors have the form of Gaussian distributions:

$$f_s(\mathbf{X}_s) = K e^{-\frac{1}{2}\left[(\boldsymbol{z}_s - \boldsymbol{h}_s(\mathbf{X}_s))^\top \boldsymbol{\Lambda}_s (\boldsymbol{z}_s - \boldsymbol{h}_s(\mathbf{X}_s))\right]} \ , \tag{2}$$

where $\boldsymbol{h}_s(\mathbf{X}_s)$ is the (possibly non-linear) functional form of the measurement or constraint that the factor represents,

$\boldsymbol{z}_s$ is its observed or expected value, and $\boldsymbol{\Lambda}_s$ is the precision (inverse covariance) of the constraint.

If we take the negative log of $p(\mathbf{X})$ to form the 'energy' $E(\mathbf{X}) = -\log p(\mathbf{X})$, we obtain:

$$\begin{aligned} E(\mathbf{X}) &= \sum_s -\log f_s(\mathbf{X}_s) \qquad (3) \\ &= L + \frac{1}{2}\sum_s (\boldsymbol{z}_s - \boldsymbol{h}_s(\mathbf{X}_s))^\top \boldsymbol{\Lambda}_s (\boldsymbol{z}_s - \boldsymbol{h}_s(\mathbf{X}_s)) \end{aligned}$$

(Here K and L are unimportant constants.)

Factor graph inference, for instance in probabilistic estimation problems such as SLAM, involves finding the values of variables $\mathbf{X}$ which maximise the product $p(\mathbf{X})$. Clearly this is equivalent to finding $\mathbf{X}$ to *minimise* $E(\mathbf{X}) = -\log p(\mathbf{X})$. Equation 3 shows that a for a Gaussian factor graph, this comes down to minimising a least squares sum over constraints.

So whenever a planning problem can be formulated as a least squares minimisation, it is equivalent to performing inference on a Gaussian Factor Graph. This opens up the use for planning of the many factor graph inference tools which were originally developed for probabilistic inference. This was highlighted in [7], [8], but previously only demonstrated for single robot planning using centralised solvers such as GTSAM [8]. In this work we use a factor graph to model multi-robot planning, as illustrated in Figure 2. We will explain the details of the factors shown in Section IV.

B. Gaussian Belief Propagation (GBP)

GBP allows inference on Gaussian factor graphs via distributed computation. Convergence is achieved via node-wise computation and message passing around the graph, so storage of the graph can be shared between multiple devices communicating by radio or some other channel. Gaussian Belief Propagation (GBP) is the special case of more general loopy belief propagation. GBP has empirically excellent performance, obtaining exact solutions for the marginal means of all variables with rapid convergence [10]. GBP has recently be used to solve challenging computer vision tasks such as bundle adjustment [11], [12] and scene flow estimation [13]. In multi-robot systems, RobotWeb used GBP for decentralised and scalable localisation of over 1000 robots [9].

GBP proceeds by iterating message passing between variables and factors around the loopy factor graph, and at any time the current marginal belief estimates for variables can be obtained. Here we lay out the main steps; see [9], [14], [10] for full details.

We represent a Gaussian distribution in canonical/information form as:

$$\mathcal{N}(\mathbf{X}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \mathcal{N}(\mathbf{X}; \boldsymbol{\eta}, \boldsymbol{\Lambda}) \ , \qquad (4)$$

where $\boldsymbol{\Lambda} = \boldsymbol{\Sigma}^{-1}$ and $\boldsymbol{\eta} = \boldsymbol{\Lambda}\boldsymbol{\mu}$ are the precision matrix and information vector respectively. In GBP, variables $\mathbf{X}_s$ are assumed to be Gaussian; thus, each variable $\mathbf{x}_k$ has a belief $b(\mathbf{x}_k) = \mathcal{N}^{-1}(\mathbf{x}_k; \boldsymbol{\eta}_k, \boldsymbol{\Lambda}_k)$. Factors $F = \{f_s\}_{s=1:N_f}$ are Gaussian constraints between variables; the factor $f_s(\mathbf{X}_s)$ is an arbitrary function that connects variables $\mathbf{X}_s$, and it may be non-linear.

1) Variable Belief Update: A variable $\mathbf{x}_k$ updates its belief by taking the product of all incoming messages from its connected factors:

$$b(\mathbf{x}_k) = \prod_{f \in n(\mathbf{x}_k)} \mathbf{m}_{f \to k}(\mathbf{x}_k) \ , \qquad (5)$$

where $n(\mathbf{x}_k) \subseteq F$ is the set of factors that the variable $\mathbf{x}_k$ is connected to, and $\mathbf{m}_{f\to k}(\mathbf{x}_k) = \mathcal{N}^{-1}(\mathbf{x}_k; \boldsymbol{\eta}_{f\to k}, \boldsymbol{\Lambda}_{f\to k})$ is the message from a factor to the variable. As we are using information form, product can be rewritten as a summation:

$$\begin{aligned} \boldsymbol{\eta}_k &= \sum_{f \in n(\mathbf{x}_k)} \boldsymbol{\eta}_{f\to k} \ , \qquad (6) \\ \boldsymbol{\Lambda}_k &= \sum_{f \in n(\mathbf{x}_k)} \boldsymbol{\Lambda}_{f\to k} \ . \qquad (7) \end{aligned}$$

However, for conciseness, we will keep the use of product notation.

2) Variable to Factor Message: A message from a variable $\mathbf{x}_k$ to a factor $f_j \in n(\mathbf{x}_k)$ is the product of all incoming factor to variable messages apart from the message from f_j:

$$\mathbf{m}_{\mathbf{x}_k \to j}(f_j) = \prod_{f \in n(\mathbf{x}_k) \setminus f_j} \mathbf{m}_{f\to k}(\mathbf{x}_k) \ . \qquad (8)$$

3) Factor Likelihood Update: The likelihood of a factor $f_s(\mathbf{X}_s)$ connected to variables $\mathbf{X}_s$ with measurement function $\boldsymbol{h}_s(\mathbf{X}_s)$, observation $\boldsymbol{z}_s$, and precision of the observation $\boldsymbol{\Lambda}_s$, can be expressed as a Gaussian distribution $\mathcal{N}^{-1}(\mathbf{X}_s; \boldsymbol{\eta}_f, \boldsymbol{\Lambda}_f)$, where $\boldsymbol{\eta}_f = \boldsymbol{\Lambda}_s(\mathbf{z}_s - \boldsymbol{h}_s(\mathbf{X}_s))$ and $\boldsymbol{\Lambda}_f = \boldsymbol{\Lambda}_s$. This however only holds if $\boldsymbol{h}_s(\mathbf{X}_s)$ is linear. In the non-linear case, we linearise using first-order Taylor expansion: $\boldsymbol{h}_s(\mathbf{X}_s) \approx \boldsymbol{h}_s(\mathbf{X}_s^0) + \boldsymbol{J}_s(\mathbf{X}_s - \mathbf{X}_s^0)$. The likelihood of the linearised factor takes the form [14]:

$$\begin{aligned} \boldsymbol{\eta}_f &= \boldsymbol{J}_s^\top \boldsymbol{\Lambda}_s \left(\boldsymbol{J}_s \mathbf{X}_s^0 + \mathbf{z}_s - \boldsymbol{h}(\mathbf{X}_s^0) \right) \ , \qquad (9) \\ \boldsymbol{\Lambda}_f &= \boldsymbol{J}_s^\top \boldsymbol{\Lambda}_s \boldsymbol{J}_s \ , \qquad (10) \end{aligned}$$

where $\boldsymbol{J}_s$ is the Jacobian and $\mathbf{X}_s^0$ is the linearisation point: the current state of the variables. In our work $\mathbf{z}_s = 0$ for all the factors unless stated otherwise, meaning that the factor energy is purely a function of the states.

4) Factor to Variable Message: A message from a factor to variable $\mathbf{x}_k$ is:

$$\mathbf{m}_{f\to k}(\mathbf{x}_k) = \sum_{\mathbf{x} \in \mathbf{X}_s \setminus \mathbf{x}_k} f(\mathbf{X}_s) \prod_{\mathbf{x} \in \mathbf{X}_s \setminus \mathbf{x}_k} \mathbf{m}_{\mathbf{x}\to f}(\mathbf{x}) \ . \qquad (11)$$

We take the product between the factor likelihood and messages from $\mathbf{X}_s \setminus \mathbf{x}_k$ before marginalising out all variables but $\mathbf{x}_k$.

IV. METHOD

We now explain the details of how we formulate multi-robot planning as a factor graph. Although GBP can optimise factor graphs for any type of Gaussian variables and factors, for the rest of the paper we will consider a 2D scenario

where robots are modelled as objects with two degrees of movement freedom on a plane.

The state $\mathbf{x}_k$ of a robot at time t_k represents the robot's position and velocity at that particular moment in time:

$$\mathbf{x}_k = [\boldsymbol{x}_k^\top, \dot{\boldsymbol{x}}_k^\top]^\top = [x_k, y_k, \dot{x}_k, \dot{y}_k]^\top \ . \tag{12}$$

The short-term planned trajectory $\mathbf{X}$ of a robot with a lookahead horizon of t_{K-1} seconds is parameterised by K such states at discrete times t_k into the future, from the current state $\mathbf{x}_0$ of the robot at the current time t_0 to the horizon state $\mathbf{x}_{K-1}$ at time t_{K-1}. Each state is a variable in a factor graph and is connected to the previous state in time with a factor representing dynamics. The states are therefore all Markovian in time, resulting in a sparse linear system. The optimal solution for the planned trajectory (the poses from the current state until the lookahead horizon) can be found by solving for the maximum a posteriori (MAP) solution $\mathbf{X}^*$ over the trajectory states.

In the presence of moving obstacles and other robots the safety of the robot (collision avoidance) is paramount. It is the states in the immediate future that are of more importance when to optimising a trajectory. The time gaps between consecutive states therefore increase for states that are further along the planned trajectory.

There are four types of factors that we consider. A unary pose factor f_p is connected to the current and horizon states of the robot and represents the prior information on those states. A dynamics factor f_d encourages a smooth trajectory and represents a noise-on-acceleration dynamics model. An obstacle factor f_o penalises a state from being close to a stationary obstacle in the scene. Finally, an inter-robot factor f_r penalises trajectories that bring two robots within collision range at a particular timestep, and takes into account each robot's future positions. An example of the resulting factor graph may be seen in figure 2.

A. Pose Factor

We define a unary pose factor f_p connected to the current and horizon states of the robot with measurement function and precision matrix:

$$\boldsymbol{h}_p(\mathbf{x}_k) = \mathbf{x}_k \ , \tag{13}$$

$$\boldsymbol{\Lambda}_p = \sigma_p^{-2}\mathbf{I} \tag{14}$$

$$\boldsymbol{z}_k = \mathbf{x}_k \ . \tag{15}$$

The robot current and horizon states ($\mathbf{x}_0$ and $\mathbf{x}_{K-1}$) are connected to strong pose factors with small values of σ_p, ensuring that the trajectory is anchored at these states during optimisation at a timestep. At the next timestep, these pose factors are modified to reflect the fact that the robot has moved (changed its current state), after which trajectory optimisation takes place.

B. Dynamics Factor

For a trajectory to be smooth and dynamically feasible, each state of the robot is connected to the next in time via a factor f_d defined in [7] as

$$\boldsymbol{h}_d(\mathbf{x}_k, \mathbf{x}_{k+1}) = \boldsymbol{\Phi}(t_{k+1}, t_k)\mathbf{x}_k - \mathbf{x}_{k+1} \ , \tag{16}$$

$$\boldsymbol{\Lambda}_d = \begin{bmatrix} \frac{1}{3}\Delta t_k^3 \mathbf{Q}_d & \frac{1}{2}\Delta t_k^2 \mathbf{Q}_d \\ \frac{1}{2}\Delta t_k^2 \mathbf{Q}_d & \Delta t_k \mathbf{Q}_d \end{bmatrix}^{-1} , \tag{17}$$

with $\Delta t_k = t_{k+1} - t_k$, $\mathbf{Q}_d = \sigma_d^2 \mathbf{I}$ and

$$\boldsymbol{\Phi}(t_b, t_a) = \begin{bmatrix} \mathbf{I} & (t_b - t_a)\mathbf{I} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} , \tag{18}$$

is the state transition matrix from time t_a to time t_b. This factor is derived from a noise-on-acceleration model of dynamics, encouraging a zero acceleration (and therefore a feasible and smooth) trajectory.

C. Obstacle Factor

A unary factor f_o is connected to each state of a robot, representing the distance of that state's position from static obstacles in the environment [7]. A pre-computed 2D signed distance field (SDF) of the environment is used. The obstacle factor is defined as:

$$\boldsymbol{h}_o(\mathbf{x}_k) = \begin{cases} 1 - \frac{d_o(\boldsymbol{x}_k)}{r_R} & d_o(\boldsymbol{x}_k) \leq r_R \\ 0 & \text{otherwise} \end{cases} , \tag{19}$$

$$\boldsymbol{\Lambda}_o = \sigma_o^{-2}\mathbf{I}. \tag{20}$$

where $d_o(\boldsymbol{x}_k)$ is the signed distance of point $\boldsymbol{x}_k$ from the nearest static obstacle and r_R is the robot radius. It is assumed that a robot can obtain the signed distance instantaneously from the SDF.

D. Inter-robot Factor

If a robot A encounters another robot B within its communication range r_C, an inter-robot factor f_r is created for each timestep of both robot trajectories. The states $\mathbf{x}_k$ of both robots A and B are connected for all future timesteps $k \geq 1$. This ensures that the two robots will not occupy the same position at the same time (collide).

This factor has a non-zero energy if the distance between the robots at a particular timestep is less than a critical distance $r^* = 2r_R + \epsilon$ where r_R is the radius of a robot, and ϵ is a small 'safety distance'. We define the inter-robot factor as:

$$\boldsymbol{h}_r\left(\mathbf{x}_k^A, \mathbf{x}_k^B\right) = \begin{cases} 1 - \frac{d_r(\boldsymbol{x}_k^A, \boldsymbol{x}_k^B)}{r^*} & d_r(\boldsymbol{x}_k^A, \boldsymbol{x}_k^B) \leq r^* \\ 0 & \text{otherwise} \end{cases} \tag{21}$$

where:

$$d_r(\boldsymbol{x}_k^A, \boldsymbol{x}_k^B) = \left\| \boldsymbol{x}_k^A - \boldsymbol{x}_k^B \right\| \ , \tag{22}$$

is the distance between the two robots at timestep t_k. We augment the notation for robot states with the superscripts A and B to distinguish between the two robots. The factor precision matrix takes the form $\boldsymbol{\Lambda}_r = (t_k \sigma_r)^{-2}\mathbf{I}$, such that the factor is weaker for states that are further in the future.

It can be seen in Figure 2 that there are two inter-robot factors for each timestep of the trajectories. This symmetry in factor creation represents a shared responsibility between the two robots for planning safe trajectories but is also

a redundancy in the design. In future work this could be extended further to allow heterogeneous robots to define their own trajectory parameters, such as safety thresholds or self-importance.

E. Online Algorithm

Each robot performs Algorithm 1. At each timestep, the current state $\mathbf{x}_0$ of each robot is updated using the simulation timestep duration ΔT. In the case of a moving horizon, the horizon state $\mathbf{x}_{K-1}$ is updated similarly.

When new robots are within communication range r_C of each other, inter-robot factors are created between them if they do not already exist, and destroyed when the robots are mutually out of range.

Finally robots perform iterative trajectory optimisation using GBP under two regimes of message passing:

1) Internal GBP: M_I iterations of message passing are conducted between the states of a robot, and their associated pose, dynamics and obstacle factors (any factors that are not connected to states of other robots). We imagine that an individual robot can be continuously performing internal message passing, and so M_I can be arbitrarily large.

2) Inter-robot GBP: M_R iterations of message passing are conducted between the states of one robot that are connected to the states of another robot via the inter-robot factors f_r.

The bottleneck in a distributed multi-robot system is likely to be the inter-robot communication, and so we would want the number of iterations M_R of inter-robot GBP to be low. This number represents the amount of negotiation between robots.

Algorithm 1 For one robot R_i

Update $\mathbf{x}_0$ and $\mathbf{x}_{K-1}$ by ΔT.
Let $C(R_i)$ be a set of robots currently connected to R_i.
Let $N(R_i) = \{R_j \mid \|R_i - R_j\| < r_C\}$ be a set of robots within communication radius of R_i.
while Running **do**
  for $R_j \in N(R_i) \backslash C(R_i)$ **do**
    Create inter-robot factors $f_r(R_i, R_j)$ with a newly observed robot R_j.
  end for
  for $R_j \in C(R_i) \backslash N(R_i)$ **do**
    Delete inter-robot factors $f_r(R_i, R_j)$ with a robot R_j out of range.
  end for
  Perform M_I internal and M_R inter-robot GBP iterations to adapt trajectory to the newly updated states and information.
end while

V. EVALUATION

Our method for multi-robot planning is generalised and can be applied to many scenarios in which robots traverse a 2D environment. We consider scenarios where coordination between robots is of importance.

In the first experiment robots begin in a circle formation and travel to the antipodal sides. In the second experiment we investigate the performance of our algorithm for robots traversing a junction, typical of highly coordinated robots in a warehouse space. Finally we also show that the asynchronous nature of our method for short-term trajectory planning which relies on message passing can be useful in planning safe motion, especially in the realistic case of communications failure due to dropped messages.

A. Baseline

We compare our method with the popular Optimal Reciprocal Collision Avoidance (ORCA) [3] which is a multi-agent distributed collision avoidance system. ORCA can be used in high coordination problems and can be said to use a basic form of 'communication' where each robot can perfectly observe the positions and velocities of its neighbours. At each timestep, ORCA optimises for each robot's instantaneous velocity in response to its neighbours. Our method, while also distributed, is a short term state planner that can react to new information as the robot follows its trajectory. Both methods are adaptive to changes in the environment and can be applied to scenarios where many robots need to coordinate in *Euclidean space*, without the necessity for the fine-tuning of many parameters. Noting the differences between our method and ORCA we use the latter as a baseline for comparison.

B. Metrics

We compare metrics for efficiency and smoothness of trajectories. Robots working together in high-coordination problems must minimise their distance travelled as well as the total time taken to reach their goals. The planned trajectories must also be smooth and collision-free. As a robot moves through the environment (with its future trajectory being constantly replanned) we store the current state position $\mathbf{x}_0$. Using this stored history we calculate the following metrics:

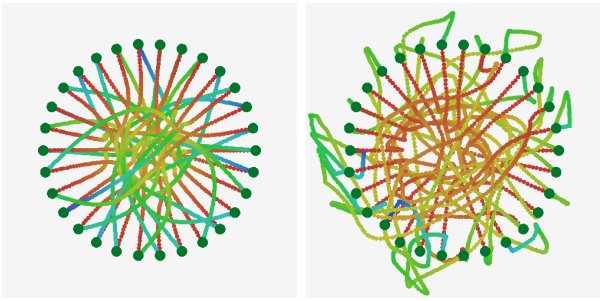
1) Distance travelled: The total distance travelled by the robot until its goal is reached. Efficient trajectories should minimise this metric.

2) Makespan: The total time taken for all robots to reach their goals. A system of many robots working together efficiently should minimise this metric.

3) Smoothness: Robot trajectories must be smooth, in order to be feasible with respect to real-world constraints such as those due to actuators. Since smoothness is intrinsically a geometric property of the path, it should not depend on time taken or velocity. We use the Log Dimensionless Jerk (LDJ), a smoothness metric first defined in [15]. This metric is essentially a log-normalised value of the total squared jerk along the trajectory and is defined as

$$LDJ \triangleq -\ln\left(\frac{(t_{final}-t_{start})^3}{v_{max}^2}\int_{t_{start}}^{t_{final}} |\ddot{v}(t)|^2\,dt\right) \quad (23)$$

where $t \in [t_{start}, t_{final}]$ is the time interval over which the metric is considered, $v(t)$ is the robot velocity at time t, and v_{max} is the maximum velocity along the trajectory. The



(a) GBP planner: smooth and efficient trajectories

(b) ORCA: longer, jerky trajectories

Fig. 3: Circle experiment for $N_R = 30$ robots for ORCA (a) and GBP planner (b). Colours change along the paths with time, from red (oldest) to blue (newest).

LDJ metric was shown to be a better indicator of smoothness than other dimensionless metrics, and values that are more positive represent 'smoother' paths.

VI. Results and Discussion

For all the following experiments we set the simulation timestep to be $\Delta T = 0.1$s, and the robot radius $r_R = 2$m. Our GBP planner conducts inter-robot GBP at $M_R = 10$ iterations per timestep, and internal GBP at $M_I = 50$ iterations per timestep. We also set $\sigma_p = 1 \times 10^{-15}$. We encourage the reader to view the accompanying video for a visual demonstration.

A. Circle Experiment

Robots are initialised in a circle formation of radius 40m, with an initial speed of $|\dot{\boldsymbol{x}}_0| = 15\text{ms}^{-1}$, towards a stationary horizon state at the opposite side of the circle. We set t_{K-1} as the ideal time taken for a single robot moving from a speed of $|\dot{\boldsymbol{x}}_0|$ to rest across the circle at constant acceleration. This would correspond to a smooth (zero jerk) trajectory. Each robot has a communication range $r_C = 80$m representing all-to-all communication. Factor covariances are set with $\sigma_d = 1$ and $\sigma_r = 0.01$.

This is a complex problem requiring coordination among many robots. Figure 3 shows the paths taken by the GBP planner and ORCA when $N_R = 30$. The colours along the paths (from red, through green to finally blue) represent time. Our GBP planner allows robots to collaborate and reach their goals at similar times.

As the number of robots N_R is varied, Figure 4 shows the distribution of robot distances travelled. The minimum distance would be 80m, the diameter of the circle. With GBP, path lengths are close to this, and are shorter than with ORCA. The spread of robot distances travelled is also smaller with our method, due to collaboration between robots, whereas in ORCA each robot acts for itself in choosing a path at the expense of longer paths for others.

Figure 5 shows the makespan as N_R is varied. This metric increases in an approximately linear fashion with GBP, and performs better than ORCA for higher N_R. ORCA performs

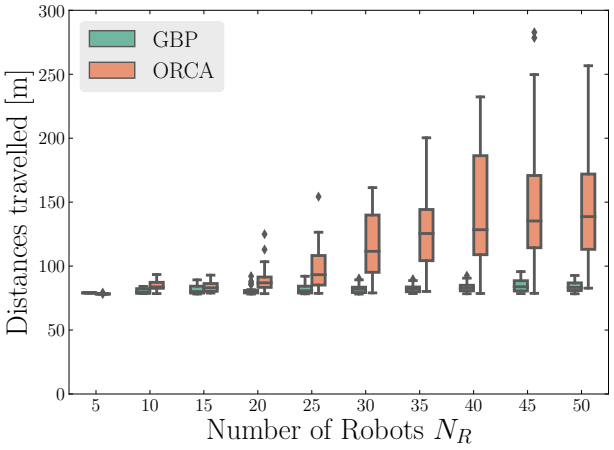


Fig. 4: Circle experiment: distribution of distances travelled as number of robots in the formation N_R is varied. The GBP planner creates shorter trajectories than ORCA, and also has a smaller spread of distances showing that robots can collaborate to achieve their goals.

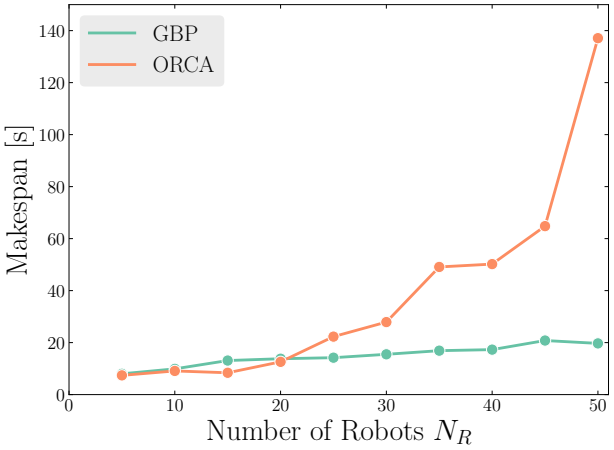


Fig. 5: Circle experiment: makespan (total time for all robots to reach goals) as N_R increases. GBP planner makespans increase linearly. At low N_R ORCA performs similarly or better, at the expense of jerkier trajectories.

better at when N_R is low – robots reach their goals in shorter amounts of time however this is at the expense of non-smooth and therefore unrealistic, jerky trajectories.

Figure 6 shows the distributions of the LDJ metric at each N_R, and it can be seen that the GBP planner creates trajectories that are far smoother than ORCA. As N_R increases, this difference in performance between the GBP planner and ORCA increases. In general the *worst performing* GBP planner robots had trajectories that were smoother than the *best performing* robot using ORCA.

As we are presenting a state planner we assume that robots can perfectly execute their planned trajectories. With real-world robots this can be helped if the planned trajectories

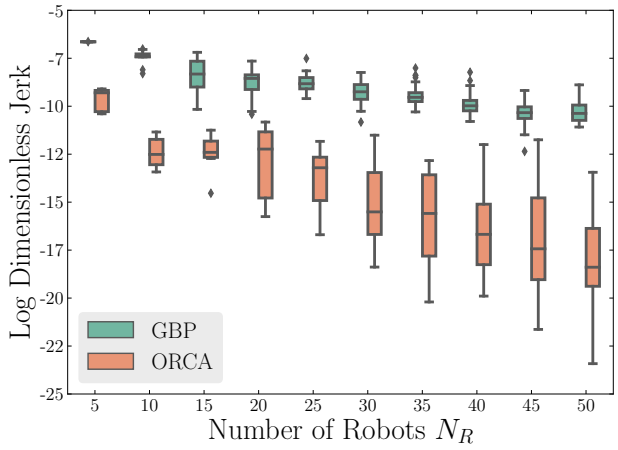


Fig. 6: Circle experiment: distribution of the Log Dimensionless Jerk (LDJ) metric as number of robots N_R increases. Values that are more positive represent trajectory shapes that are smoother. The worst performing GBP planner robots had smoother trajectories than the best robots for ORCA.

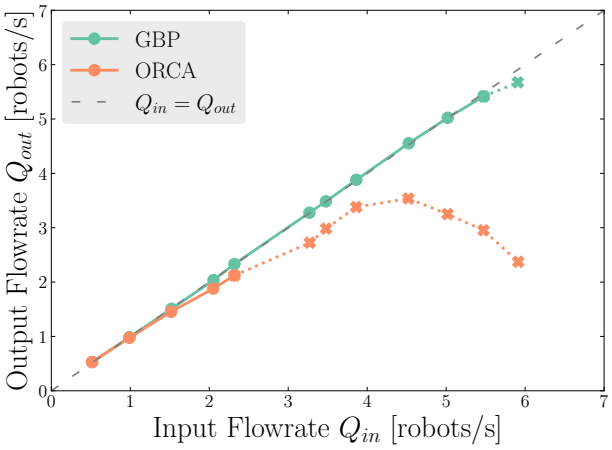


Fig. 7: Juncion experiment: input vs output flowrates. Robots cross the junction at 15ms^{-1}. Points shown as crosses denoted trials where the correctness condition was violated. The GBP planner can sustain flowrates of robots up to $Q_{in} = 5.5$ robots s^{-1}, compared to ORCA which breaks down at $Q_{in} = 2.3$ robots s^{-1}.

are smooth, efficient and induce minimal jerk.

B. Junction Experiment

Robots working in crowded environments may have to operate at high speeds with high levels of coordination. One example would be while traversing junctions between shelves or other obstacles in a warehouse. We simulate one such junction where each channel has width 16m and robots move with velocities of 15ms^{-1} with a horizon of $t_{K-1} = 2$s. We set $\sigma_d = 0.5$ and $\sigma_o = \sigma_r = 0.005$.

Multirobot systems must be able to maintain a high flowrate of robots without blockages occurring in the junction. We vary the rate Q_{in} at which robots enter the central section of the junction and measure the rate Q_{out} at which they exit and compare our method against ORCA. We choose the central section of the junction to measure flow over 500 timesteps to represent steady-state flow behaviour. Note that robots must exit the junction in the same direction that they entered, and that they must not collide; we refer to this as the 'correctness condition'.

Figure 7 shows Q_{out} against Q_{in} for the GBP planner and ORCA. A dashed line representing the ideal case of equal input and output flowrates ($Q_{out} = Q_{in}$) is also shown for comparison. Trials where the correctness condition was violated are shown as crosses. The GBP planner is able to sustain flowrates of up to $Q_{in} = 5.5$ robots s^{-1}, compared to ORCA which begins to break down at $Q_{in} = 2.3$ robots s^{-1}. This can be attributed to the fact that the GBP planner allows robots to communicate, and collaboratively negotiate their planned trajectories with neighbouring robots before they are close to each other.

It is important to note that at high enough flowrates the effectiveness of both the GBP planner and ORCA will break down as the dense packing of incoming robots at those flowrates cannot allow much freedom for robots to adapt their trajectories.

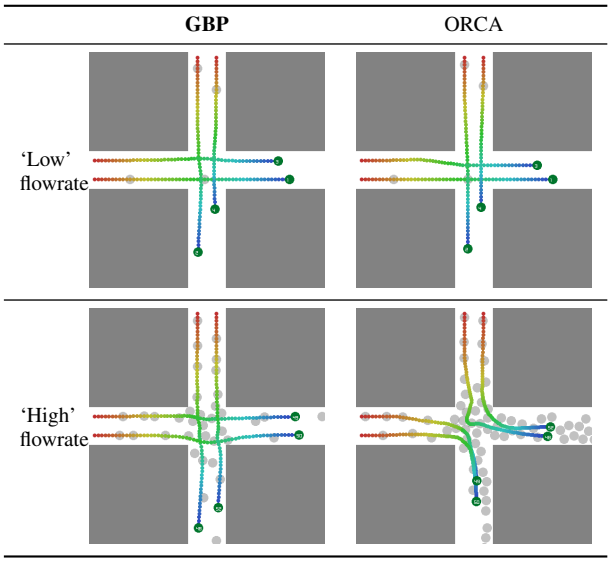


TABLE I: Qualitative comparison of planning performance for our GBP planner and ORCA in the junction experiment at representative low ($Q_{in} = 1$) and high ($Q_{in} = 5.5$) flowrates. Selected robots are shown in colour, with other robots in light gray. Colours along paths represent time. At low flowrates, both methods work well. At the high flowrate shown, GBP is able to allow robots to smoothly swerve past each other, whereas ORCA has broken down — robots push each other and are not able to continue along their desired directions, creating a blockage in the junction.

C. Communications Failure Experiment

Our GBP planner relies on peer-to-peer communication between robots. At each timestep messages will need to be passed back and forth. It is assumed that each robot follows a protocol similar to [9]; it always broadcasts its state information. We consider a communications failure scenario where a robot is not able to receive messages from robots it is connected to. We would expect more cautious behaviour when planning a trajectory.

We simulate a communication failure fraction γ: at each timestep the robot cannot receive any messages from a randomly sampled proportion γ of its connected neighbours. We repeat the circle experiment with 21 robots at two different initial speeds of $|\dot{\boldsymbol{x}}_0| = 10\text{ms}^{-1}$ and 15ms^{-1}, measuring the makespan. The reported result is an average over 5 different random seeds. To be fair, at any timestep for any robot, the failed communications are exactly the same given a fixed seed for both initial velocities considered.

TABLE II: A comparison of makespans for the circle experiment with 21 robots for two initial velocities under varying rates of communication failure.

Initial speed [ms^{-1}]	10		15	
Comm. failure γ [%]	Makespan [s]	# Collisions	Makespan [s]	# Collisions
0	19.5	0.0	14.9	0.0
10	20.3	0.0	17.1	0.0
20	22.9	0.0	18.9	0.0
30	25.7	0.0	22.5	0.0
40	30.8	0.0	26.5	0.0
50	35.6	0.0	30.6	0.0
60	42.0	0.0	38.8	0.2
70	51.3	0.0	44.6	0.8
80	87.4	0.0	63.4	0.8
90	146.9	1.6	12.6	4.6

Table II shows that as γ increases, it takes longer for all robots to reach their goals. However, trajectories for the 10ms^{-1} case are completely collision free up to $\gamma = 80\%$. As the initial speed increases, collisions happen at lower values of γ as robots have less time to react to faster moving neighbours who they may not be receiving messages from.

Reactive planners such as ORCA require that a robot always has information about all of its neighbours and thus cannot deal with communication failures (partial observation of the neighbourhood). This experiment shows one of the benefits of GBP — safe trajectories can still be planned even with partial information about the environment, which is likely in any realistic settings.

VII. CONCLUSIONS

We have shown that our collaborative method for short-term trajectory planning using Gaussian Belief Propagation (GBP) can ensure smooth, efficient and safe trajectories in problems where a high degree of coordination is required. By performing message passing between robots in a purely peer-to-peer manner, there is no need for a centralised solver; the distributed nature of the problem could be scaled up to large numbers of robots.

Our current experiments are based on multiple iterations of communication between robots per timestep, and assume that robots have perfect localisation. However the results in [9] on multi-robot localisation using GBP offer a lot of promise that with much more efficient inter-robot communication patterns good factor graph optimisation and therefore planning performance could be achieved.

In the future, we will explore other applications of GBP trajectory planning, such as the coordination of larger numbers of robots which can organise themselves into useful formations as well as investigating uncertainty in localisation, and non-holonomic constraints.

ACKNOWLEDGEMENTS

Research presented in this paper has been supported by Dyson Technology Ltd. We are also 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 London. We would particularly like to thank Joe Ortiz, Tristan Laidlow and Ignacio Alzugaray.

REFERENCES

[1] Jingjin Yu. Intractability of optimal multi-robot path planning on planar graphs. *CoRR*, abs/1504.02072, 2015.

[2] J.P. van den Berg and M.H. Overmars. Prioritized motion planning for multiple robots. In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 430–435, 2005.

[3] J. Van Den Berg, S. Guy, M. Lin, and D. Manocha. Reciprocal n-body collision avoidance. *Proceedings of the International Symposium on Robotics Research*, 2009.

[4] Ruben Van Parys and Goele Pipeleers. Online distributed motion planning for multi-vehicle systems. In *2016 European Control Conference (ECC)*, pages 1580–1585, 2016.

[5] Carlos E. Luis, Marijan Vukosavljev, and Angela P. Schoellig. Online trajectory generation with distributed model predictive control for multi-robot motion planning. *CoRR*, abs/1909.05150, 2019.

[6] Qingbiao Li, Fernando Gama, Alejandro Ribeiro, and Amanda Prorok. Graph neural networks for decentralized multi-robot path planning. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 11785–11792, 2020.

[7] Mustafa Mukadam, Jing Dong, Xinyan Yan, Frank Dellaert, and Byron Boots. Continuous-time gaussian process motion planning via probabilistic inference. *CoRR*, abs/1707.07383, 2017.

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

[9] Riku Murai, Joseph Ortiz, Sajad Saeedi, Paul HJ Kelly, and Andrew J Davison. A robot web for distributed many-device localisation. *arXiv preprint arXiv:2202.03314*, 2022.

[10] Joseph Ortiz, Talfan Evans, and Andrew J Davison. A visual introduction to gaussian belief propagation. *arXiv preprint arXiv:2107.02308*, 2021.

[11] Joseph Ortiz, Mark Pupilli, Stefan Leutenegger, and Andrew J Davison. Bundle adjustment on a graph processor. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2416–2425, 2020.

[12] Joseph Ortiz, Talfan Evans, Edgar Sucar, and Andrew J Davison. Incremental abstraction in distributed probabilistic slam graphs. *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[13] Raluca Scona, Hidenobu Matsuki, and Andrew J Davison. From scene flow to visual odometry through local and global regularisation in markov random fields. *IEEE Robotics and Automation Letters*, 2022.

[14] Andrew J Davison and Joseph Ortiz. Futuremapping 2: Gaussian belief propagation for spatial ai. *arXiv preprint arXiv:1910.14139*, 2019.

[15] S. Balasubramanian, A. Melendez-Calderon, Roby-Brami, and A. et al. On the analysis of movement smoothness. *NeuroEngineering and Rehabilitation 12, 112*, 2015.