Imperial College London

[1cm]

MEng Individual Project

Imperial College London

Department of Computing

There is an impostor Among Us

Author:
Akshat Tripathi

Supervisor:
Prof. Andrew Davison

Second Marker:
Unclear

January 19, 2023

Abstract

TODO

Acknowledgements

TODO

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Objectives

1.2 Challenges

1.3 Contributions

Chapter 2

Background

This chapter will provide the theoretical background for this thesis. First, we will discuss the field of multi-robot systems, providing the reader with an understanding of how the field has evolved and how it may further evolve. Next, we examine RobotWeb [1], the research that this thesis seeks to build upon. Finally, we discuss various security issues present in the field to arrive at the research question for this thesis.

2.1 Multi-Robot Systems

The study of multi-robot systems concerns itself with studying how to allow multiple robots to operate in the same environment [2]. Multi-robot systems have several advantages over single-robot systems; they are more effective, efficient, flexible, and resilient [3]. These robots can behave competitively or collaboratively, coordinate statically or dynamically, communicate explicitly or implicitly, consist of homogeneous or heterogeneous robots, and make decisions centrally or decentrally [4].

2.1.1 Competitive vs Collaborative Behavior

Multiple robots which share a common goal are considered to be behaving collaboratively, whereas if each robot aimed to complete its own goal at the expense of all others, it would be said to be behaving competitively [4]. Examples of collaboration range from teams of robots constructing a lunar habitat [5] to exploring unknown environments [6].

2.1.2 Static vs Dynamic Coordination

If a multi-robot system operates using a set of predetermined rules, then it can be said to be coordinating itself statically [4]. A possible set of rules would be that each robot must maintain a certain distance between it and all others [4]. Dynamically coordinated multi-robot systems would instead make decisions whilst performing the task and may communicate to do so [4].

2.1.3 Explicit vs Implicit Communication

Most multi-robot systems communicate explicitly by sending messages to each other via a hardware communication interface, for example, a wifi module [4]. However, there is still a sizeable minority of approaches that send messages through their environment (implicit communication) and rely on others to sense these messages to receive them. An example of implicit communication is found in [7], where the authors use it to allow a team of robots to play a game of football for the RoboCup Simulation League [8].

2.1.4 Homogeneous vs Heterogeneous Robots

Multi-robot systems can either contain robots with identical hardware, which are known as homogeneous systems, or individual robots may have different hardware, making them heterogeneous

systems. Heterogeneous systems allow a greater degree of specialisation within a multi-robot system but also add additional decision-making complexity.

2.1.5 Centralised vs Decentralised Decision Making

A multi-robot system is said to have centralised decision-making if all robots communicate with a central agent, which may or may not be a robot itself, to receive instructions. Centralised schemes perform better with smaller groups of robots and in static environments, they also introduce a single point of failure in the central agent [4]. Decentralised schemes, however, avoid vesting authority into a central agent and instead treat each agent as an equal part of the system, which allows them to avoid the problems associated with centralised schemes. However, decentralised schemes lose the guarantee that they will converge to an optimal solution, as decisions are made with incomplete information. In addition to centralised and decentralised schemes, multi-robot systems may also be organised in a hierarchical manner, where some robots would be chosen as local leaders, but no global leader would exist.

2.2 Robot Web

This thesis seeks to build upon the work done in "A Robot Web for Distributed Many-Device Localisation" [1], which describes a method for *heterogeneous* robots in a *decentralised* multi-robot system to *collaborate* via *explicit communication* to localise *dynamically*.

Robots in the Robot Web move along predefined paths, estimating their location via internal odometry. When a robot senses another, it communicates its measurement to the other robot, and then both robots use the measurement to update their location estimates. Since we live in an imperfect world, each sensor measurement carries with it a small amount of noise, which is reflected in the Robot Web by a degree of uncertainty attached to each robot's location estimate and represented by a Gaussian distribution.

This section will introduce the reader to the core concepts used in the Robot Web and assemble them to provide the reader with an understanding of how the Robot Web functions and some of its limitations.

2.2.1 Factor Graphs

A factor graph is an undirected bipartite graph used to represent the factorisation of a probability distribution $p(X)$. A probability distribution can be said to be factorised if it is written in the form:

$$p(X) = \prod_i f_i(X_i) \tag{2.1}$$

The nodes of a factor graph can either represent variables (X_i) or factors (f_i). There are several different ways to draw factor graphs, but we will use the one defined in [9], where factors are drawn as squares and variables are drawn as circles.

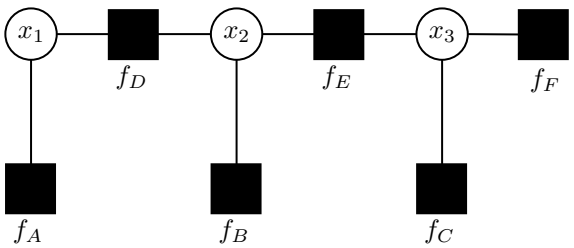

Figure 2.1: An example of a factor graph

The above factor graph represents the following factorisation:

$$p(X_1, X_2, X_3) = f_A(X_1)f_B(X_2)f_C(X_3)f_D(X_1, X_2)f_E(X_2, X_3)f_F(X_3) \tag{2.2}$$

Assuming that each variable takes discrete values, suppose we wanted to find the probability that $X_1 = z$ for some value of z using the above factor graph. Then we would need to find:

$$p(X_1 = z, X_2, X_3) = \sum_{i=X_2} \sum_{j=X_3} p(X_1 = z, X_2 = i, X_3 = j) \tag{2.3}$$

And by 2.2 we get:

$$p(X_1 = z, X_2, X_3) = \sum_{i=X_2} \sum_{j=X_3} f_A(z)f_B(i)f_C(j)f_D(z, i)f_E(z, j)f_F(j) \tag{2.4}$$

which can be rearranged to form:

$$p(X_1 = z, X_2, X_3) = f_A(z) \sum_{i=X_2} \left(f_D(z, i)f_B(i) \left(\sum_{j=X_3} f_E(z, j)f_C(j)f_F(j) \right) \right) \tag{2.5}$$

Similarly, if we wanted to find the probability that $X_2 = z$ for some z we would need to find:

$$p(X_1, X_2 = z, X_3) = f_b(z) \left(\sum_{i=X_1} f_D(i, z)f_A(i) \right) \left(\sum_{j=X_3} f_E(z, j)f_C(j)f_F(j) \right) \tag{2.6}$$

Noticing how the sum over X_3 in both 2.5 and 2.6 is the same, we may want to "cache" the result when dealing with large factor graphs, to improve performance. To do this we can associate calculations with nodes in the factor graph. We call these associations "messages".

The general form of a message from variable i to factor j is the product of the messages from all other neighbouring factors [10]. Put formally:

$$m_{x_i \to f_j} = \prod_{s \in N(i) \setminus j} m_{f_s \to x_i} \tag{2.7}$$

The general form of a message from factor j to variable i is the product of the messages from all other neighbouring variables and the factor applied to all other variables except i [10]. Put formally:

$$m_{f_j \to x_i} = \left(\sum_{X_j \setminus x_i} f_j(X_j) \right) \left(\prod_{k \in N(j) \setminus i} m_{x_k \to f_j} \right) \tag{2.8}$$

Finally, the marginal value of a variable is simply the product of all incoming messages to it [10].

$$p(x_i) = \prod_{s \in N(i)} m_{f_s \to x_i} \tag{2.9}$$

2.2.2 Belief Propagation

The above equations are used by the Belief Propagation algorithm, an iterative message-passing algorithm used to calculate the marginal value for each variable in a factor graph [10]. Each iteration of Belief Propagation has 3 phases:

1. Variables send messages to each of their neighbouring factors 2.7.
2. Factors send messages to each of their neighbouring variables 2.8.
3. Each variable updates its "belief" (its estimated marginal value) 2.9.

The original Belief Propagation algorithm was designed to be used in tree-like graphs, i.e. graphs without loops [10]. However, empirical evidence has shown that "Loopy-BP" can still converge to provide useful results in a variety of problem domains [10].

2.2.3 Gaussian Belief Propagation

A special case of the Belief Propagation algorithm is Gaussian Belief Propagation, which applies to problems where all variables follow a Gaussian distribution, and all factors are Gaussian functions of their inputs.

Under Gaussian Belief Propagation, each message can be interpreted as a Gaussian and so must contain sufficient information to produce one. A naive way of achieving this is to include a mean vector and a covariance matrix in each message. However, this approach is computationally expensive as it requires a full matrix multiplication whenever messages are multiplied which is an order $O(n^3)$ operation. An alternative approach is to use the *canonical form* of the multivariate Gaussian distribution.

The canonical form uses an *information vector* (η) and a *precision matrix* (Λ) defined as follows:

$$\eta = \Sigma^{-1}\mu \qquad\qquad \Lambda = \Sigma^{-1}$$

where Σ is the covariance matrix and μ is the mean vector. Now multiplying messages is made more efficient as it only requires the addition of both messages' η and Λ values, making it an order $O(n^2)$ operation in the worst case. A further performance improvement can be made by recognising that the precision matrix is a sparse matrix [10].

2.2.4 Lie Theory

Lie theory is a subset of group theory focussed on studying *Lie groups*. Lie theory is a vast and abstract field, from which we only need to borrow a few concepts. The first is that positions and rotations can be represented as Lie groups, for example, the group $SO2$ represents a rotation in 2D space and the $SE3$ group represents a rigid motion in 3D space. The second core concept is the *tangent space* which allows small deviations to be applied to the Lie group uniformly regardless of the value it operates on. This concludes our whirlwind tour of Lie theory, we invite the reader to read [11] for a more detailed tutorial.

2.2.5 Putting it all together

Now that we have covered all of the prerequisites to understanding how the Robot Web operates, we shall now demonstrate how they can be assembled into the Robot Web.

Every robot in the Robot Web needs to estimate its current location at all times, this is called localisation. One simple localisation method is to use odometry, which uses internal sensors to measure its displacement from its previous location. Since no sensor is perfect, this introduces a small amount of noise, which can be accurately modelled using a Gaussian distribution. The Robot Web simulates odometry using a factor graph, each known position of the robot maps to a pose variable, and the variables of each pair of successive positions are connected by an odometry factor.

On every timestep, the robot performs an iteration of Gaussian Belief Propagation to estimate its current position.

The Robot Web further improves the accuracy of robots' locations by allowing robots to measure each other using external sensors. When a robot senses another, it creates a factor in its factor graph between its and the other's latest pose variables. When each robot wants to send a message to another, it publishes the message to its **Robot Web Page**, which the other robot will eventually read and use to update its location estimate.

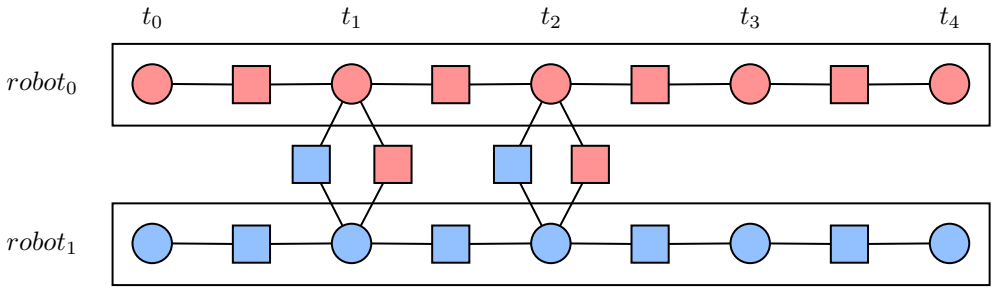

Figure 2.2: An example of a factor graph in the Robot Web. Each robot's variables are connected by odometry factors. At times t_1 and t_2, both robots sense each other, and so exchange measurements by creating inter-robot-measurement factors on the graph.

The Robot Web represents the locations and sensor measurements of all robots using general Lie groups, rather than any specific group. This has the consequence that any type of sensor or robot can be a part of the Robot Web. For example, a drone moving in 3D space can interact with a car moving on a plane.

2.2.6 Evaluation

insert percentage here

The Robot Web has been shown to improve the accuracy of robot localisation by in a scenario where there are ....

Furthermore, the Robot Web has proven to be robust to a large number of faulty inter-robot sensors reporting random measurements, with this robustness lasting until 70-80% of inter-robot sensors reported corrupted measurements.

Add the graph from the original paper to back this up

Although the Robot Web is robust to many inter-robot sensors reporting random measurements, it is not robust to a bad actor which may instead report incorrect measurements designed to worsen the localisation of other members of the Robot Web. Possible attacks include but are not limited to:

1. A bad actor creates inter-robot-measurement factors with extremely low standard deviations, to lull others into a false sense of security.
2. A bad actor sends these messages whilst assuming the identity of another robot.
3. A bad actor sends messages from many nonexistent robots, also known as a sybil attack (subsection 2.3.3)

2.3 Security Issues

2.3.1 Byzantine fault tolerance

2.3.2 Masquerade Attacks

2.3.3 Sybil Attacks

Chapter 3

Related Work

3.1 Wifi Scattering

3.1.1 Paper 1 - Gil et al.

3.1.2 Paper 2 - Huang et al.

3.2 Proof of Work

3.2.1 Paper series 1 - Gupta et al.

3.2.2 Paper series 2 - Strobel et al.

3.3 ARP poisoning defenses

3.4 DDoS Attacks

Chapter 4

PROJECT X

Chapter 5

Evaluation

Chapter 6

Conclusion

Appendix A

First Appendix

Bibliography

[1] Murai R, Ortiz J, Saeedi S, Kelly PHJ, Davison AJ. A Robot Web for Distributed Many-Device Localisation. CoRR. 2022;abs/2202.03314. Available from: https://arxiv.org/abs/2202.03314.

[2] Farinelli A, Iocchi L, Nardi D. Multirobot Systems: A Classification Focused on Coordination. IEEE transactions on systems, man, and cybernetics Part B, Cybernetics : a publication of the IEEE Systems, Man, and Cybernetics Society. 2004 11;34:2015-28.

[3] Roldán-Gómez JJ, Barrientos A. Special Issue on Multi-Robot Systems: Challenges, Trends, and Applications. Applied Sciences. 2021;11(24). Available from: https://www.mdpi.com/2076-3417/11/24/11861.

[4] Yan Z, Jouandeau N, Ali A. A Survey and Analysis of Multi-Robot Coordination. International Journal of Advanced Robotic Systems. 2013 12;10:1.

[5] Stroupe A, Huntsberger T, Okon A, Aghazarian H, Robinson M. Behavior-based multi-robot collaboration for autonomous construction tasks. In: 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems; 2005. p. 1495-500.

[6] Burgard W, Moors M, Fox D, Simmons R, Thrun S. Collaborative multi-robot exploration. In: Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065). vol. 1; 2000. p. 476-81 vol.1.

[7] Pagello E, D'Angelo A, Montesello F, Garelli F, Ferrari C. Cooperative behaviors in multi-robot systems through implicit communication. Robotics and Autonomous Systems. 1999;29(1):65-77. Available from: https://www.sciencedirect.com/science/article/pii/S0921889099000391.

[8] Kitano H, Asada M, Kuniyoshi Y, Noda I, Osawa E. RoboCup: The Robot World Cup Initiative. In: Proceedings of the First International Conference on Autonomous Agents. AGENTS '97. New York, NY, USA: Association for Computing Machinery; 1997. p. 340–347. Available from: https://doi.org/10.1145/267658.267738.

[9] Kschischang FR, Frey BJ, Loeliger HA. Factor graphs and the sum-product algorithm. IEEE Transactions on Information Theory. 2001;47(2):498-519.

[10] Ortiz J, Evans T, Davison AJ. A visual introduction to Gaussian Belief Propagation. CoRR. 2021;abs/2107.02308. Available from: https://arxiv.org/abs/2107.02308.

[11] Solà J, Deray J, Atchuthan D. A micro Lie theory for state estimation in robotics. CoRR. 2018;abs/1812.01537. Available from: http://arxiv.org/abs/1812.01537.