Imperial College London

MEng Individual Project

Imperial College London

Department of Computing

TODO

Author:
Akshat Tripathi

Supervisor:
Prof. Andrew Davison

Second Marker:
Prof. Antoine Cully

May 23, 2023

Abstract

TODO

Acknowledgements

TODO

Contents

1 Introduction **5**
- 1.1 Objectives . . . 5
- 1.2 Challenges . . . 5
- 1.3 Contributions . . . 5

2 State of the Art **6**
- 2.1 Multi-Robot Systems . . . 6
  - 2.1.1 Competitive vs Collaborative Behavior . . . 6
  - 2.1.2 Static vs Dynamic Coordination . . . 6
  - 2.1.3 Explicit vs Implicit Communication . . . 6
  - 2.1.4 Homogeneous vs Heterogeneous Robots . . . 6
  - 2.1.5 Centralised vs Decentralised Decision Making . . . 7
- 2.2 Robot Web . . . 7
  - 2.2.1 Factor Graphs . . . 7
  - 2.2.2 Belief Propagation . . . 9
  - 2.2.3 Gaussian Belief Propagation . . . 9
  - 2.2.4 Lie Theory . . . 9
  - 2.2.5 Putting it all together . . . 9
  - 2.2.6 Evaluation . . . 10
- 2.3 Security Issues . . . 11
  - 2.3.1 Denial of Service . . . 11
  - 2.3.2 Identity-Based Attacks . . . 11
  - 2.3.3 Physical Attacks . . . 12
- 2.4 Wifi Fingerprinting . . . 12
  - 2.4.1 Guaranteeing spoof-resilient multi-robot networks . . . 13
  - 2.4.2 Lightweight Sybil-Resilient Multi-Robot Networks by Multipath Manipulation 14
  - 2.4.3 Conclusions . . . 14
- 2.5 Proof of Work . . . 15

3 An Investigation into the Security of RobotWeb **16**
- 3.1 The behaviour of an ideal attacker . . . 16
- 3.2 A simplified system . . . 17
- 3.3 Theoretical properties of the RobotWeb under attack . . . 18
  - 3.3.1 Measuring the strength of an attack . . . 18
  - 3.3.2 Crafting the perfect message . . . 19
  - 3.3.3 Scaling back up to n-dimensional space . . . 20
- 3.4 Hypotheses . . . 20
  - 3.4.1 Bounds on μ can be slowly escaped . . . 20
  - 3.4.2 Bounds on Λ can be quickly escaped . . . 21
  - 3.4.3 Long histories are detrimental . . . 21
  - 3.4.4 Attacks can spread on their own . . . 21
  - 3.4.5 Robots are most vulnerable on startup . . . 22
- 3.5 Experimental evidence . . . 22
  - 3.5.1 Experimental setup . . . 22
  - 3.5.2 Testing hypothesis 1 . . . 22
  - 3.5.3 Testing hypothesis 2 . . . 22

3.5.4 Testing hypothesis 3 23
3.5.5 Testing hypothesis 4 23
3.5.6 Testing hypothesis 5 23

4 Evaluation Plan **25**

5 Conclusion **26**
5.1 Ethical Considerations 26

A First Appendix **27**

Bibliography **27**

List of Figures

2.1 Example factor graph 7
2.2 Robot Web factor graph 10
2.3 RobotWeb's robustness to garbage measurements 10

3.1 Single Variable in a Factor Graph 17
3.2 Representative factor graph around a single variable 17

Chapter 1

Introduction

1.1 Objectives

The field of distributed robotics is a growing one, partly due to the growth of the number of applications involving it (autonomous vehicles and drone delivery systems) and partly because of increased interest in areas which would greatly benefit from it, such as the Lunar Gateway Project, or the Mars 2020 Perseverance Mission.

An open problem in this distributed robotics is that of effective inter-robot communication. Inter-robot communication provides several benefits to distributed robotics; it allows robots to 1. acquire a more accurate view of their environment, 2. gain access to a larger map of the world, 3. improve their path planning by incorporating others' plans and 4. carry fewer resources, such as sensors and instead rely on others in the swarm.

A subset of distributed robotics research is dedicated to finding ways to allow such communication in situations where some of the robots may act with hostility. The source of this hostility could be unnatural, where a bad actor solely seeks to disrupt the system, or it could be a natural consequence of competition in the environment, such as a self-driving car trying to prevent others from overtaking it.

This hostility must be protected against if the benefits of communication and collaboration are to be seized. This is the principal focus of this Master's Thesis.

1.2 Challenges

1.3 Contributions

Currently, the main contributions of this paper can be found in the background research section and some of the preliminary results on the effectiveness of attacks on robot networks. This is subject to change with time.

Chapter 2

State of the Art

The first half of 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 [?], 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 [?]. Multi-robot systems have several advantages over single-robot systems; they are more effective, efficient, flexible, and resilient [?]. 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 [?].

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 [?]. Examples of collaboration range from teams of robots constructing a lunar habitat [?] to exploring unknown environments [?].

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. A possible set of rules would be that each robot must maintain a certain distance between it and all others. Dynamically coordinated multi-robot systems would instead make decisions whilst performing the task and may communicate to do so [?].

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 [?]. 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 [?], where the authors use it to allow a team of robots to play a game of football for the RoboCup Simulation League [?].

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 [**?**]. 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" [**?**], 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 [**?**], where factors are drawn as squares and variables are drawn as circles.

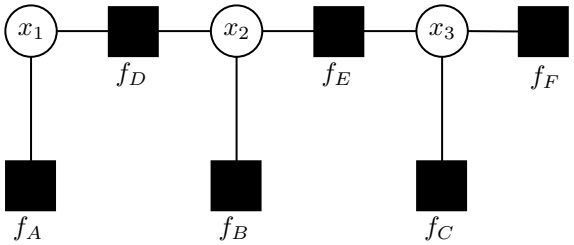

Figure 2.1: An example of a factor graph

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 [**?**]. 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 [**?**]. 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 [**?**].

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

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 [**?**]. 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 [**?**]. However, empirical evidence has shown that "Loopy-BP" can still converge to provide useful results in a variety of problem domains [**?**].

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 [**?**].

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 [**?**] 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.

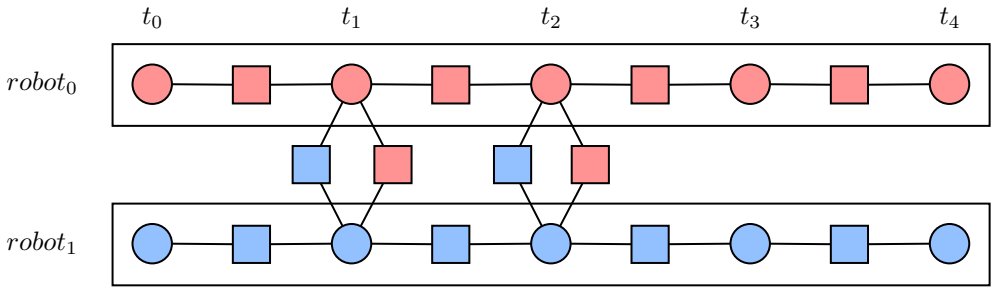

Figure 2.2: An example of a factor graph in the Robot Web. Each robot's variables are connected by odometry factors. At times t_1 and t_2, both robots sense each other, and so exchange measurements by creating inter-robot-measurement factors on the graph.

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

The Robot Web has been shown to improve the accuracy of robot localisation, and most importantly the inter-robot measurements have been shown to provide further improvements over schemes where robots would only measure landmarks.

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.

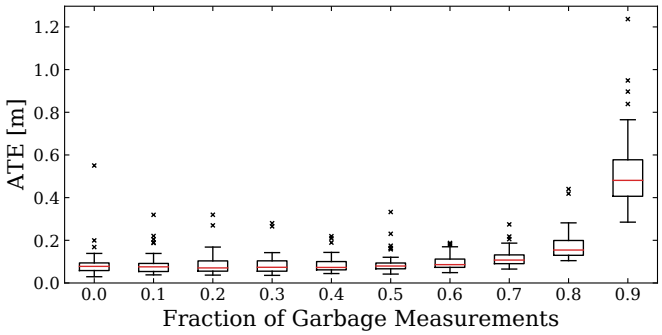

Figure 2.3: A graph showing that the RobotWeb is robust to up to 70-80% of "garbage" measurements, where faulty sensors report random measurements. ATE refers to the average Absolute Trajectory Error measured over 50 runs in an environment with 50 robots and 10 beacons running for 100 timesteps. Taken from [**?**, Figure 5]

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. Sending messages with extremely low standard deviations, to lull others into a false sense of security.
2. Sending these messages whilst assuming the identity of another robot.
3. Sending these messages from many nonexistent robots, also known as a Sybil attack.

2.3 Security Issues

In this section, we will discuss several general security issues that can arise in robot networks. We will focus on issues that affect the accuracy of a robot's internal model of the world. This excludes attacks which may result in an attacker gaining control over a robot, yet includes attacks performed by hijacked robots.

2.3.1 Denial of Service

A denial of service attack seeks to deny service. In a robot network such as the Robot Web, this would prevent one or more robots from being able to access messages sent by their peers, and essentially cut them off from the network.

The simplest way for an attacker to perform a DoS attack is to use a signal jammer, which can be constructed using off-the-shelf equipment [**?**]. This would continuously transmit signals within the range of frequencies allowed by the communication medium, both interfering with and irrecoverably corrupting any messages sent. More sophisticated attackers may craft harder-to-detect jammer attacks by mimicking legitimate messages or only transmitting when it senses communication [**?**]. In addition to these, there exist a whole host of jamming attacks (and defences) targetting specific communication protocols.

Another form of a DoS attack would be to disconnect specific robots from the network, using the network protocol's existing defences. For example, by convincing others that the target is a bad actor, triggering their defences to remove the target from the network.

2.3.2 Identity-Based Attacks

Identity-based attacks can wreak havoc on robot networks. Using the model defined by Douceur [**?**], we can describe a robot network as one consisting of E entities (robots) each claiming at least one identity i from the set of all identities I. When the network is under an identity-based attack, at least one of the following two properties will hold:

1. Two entities, $e1, e2$ will present the same identity i
2. The number of identities in I will exceed the number of entities in E.

If only the former holds, then the network is under a spoofing attack, whereas if only the latter holds, then the network is under a Sybil attack. Note: there is no guarantee that only one of these will hold at a time, and so we must be prepared to defend against both simultaneously.

Spoofing Attacks

Devices exchange information by sending packets or frames of data. For our purposes, we can ignore the differences between packets and frames, and use the terms interchangeably. Packets are used to encapsulate the data sent with relevant metadata, such as the source and destination IDs of the packet. This metadata is the main target of spoofing attacks.

In a spoofing attack, the attacker first finds the identity of a legitimate device, for example by first intercepting packets and then extracting the source ID from them. After this point, the attacker sends misleading packets impersonating the target.

This can have several benefits for the attacker. Firstly, they could covertly inject misinformation into the network by impersonating an already trusted robot. Secondly, they could trigger defences in the network to flag target for misinformation and remove it.

Sybil Attacks

In a Sybil attack, the attacker will create many fake identities to gain undue influence on the network. In a robot network, this would allow them to indirectly influence the actions of victim robots. For example, one could lead two self-driving cars to conclude that their best course of action is to crash, by claiming many false identities would be hit if they were not to.

Douceur [**?**] proves Sybil attacks are always possible in distributed systems where there is no central arbiter of truth. They present and prove four lemmas about identities in large-scale distributed systems. The first two lemmas are concerned with entities which directly validate the identities presented to them, whilst the second two lemmas involve entities which rely upon other, potentially untrustworthy, identities for validation. The lemmas are as follows:

1. If an attacker has ρ times as many resources as the weakest entity, then they can successfully present up to $\lfloor \rho \rfloor$ distinct identities.
2. If an entity doesn't validate all identities simultaneously, then an attacker can present an unbounded number of distinct identities.
3. If an entity trusts q other identities to validate an identity, then at least f attackers are required to perform the attack, where $f >= q$ or the resources commanded by the attackers exceed $q + f$ times the weakest entity's resources.
4. If all c non-attackers don't coordinate when they validate identities, and an entity again trusts q other identities for validation, then even a weak attacker can present $\lfloor \frac{c}{q} \rfloor$ identities.

We plan to solve this problem by treating the physical world as a central arbiter of truth, albeit with some degree of error, due to imperfections in sensors.

2.3.3 Physical Attacks

Finally, since this thesis focuses on the security of robot networks, we will discuss the possibility of physical attacks on the system, and our limited ability to protect against them. We define a physical attack as any attempt to compromise the ability of a robot to perform its task. This includes colliding with the robot but also includes more subtle tactics, such as obscuring the robot's sensors. Physical attacks could also be used similarly to Sybil attacks, where several attackers would surround a robot and feed it misinformation.

In this thesis, we will not attempt to protect against attacks where several robots would physically collide, since this would require heavy hardware modifications to existing robots. Instead, we will focus on occlusion attacks and physical Sybil attacks, at the very least allowing a robot to detect them.

Several different techniques have been explored in preventing the types of identity-based attacks discussed in this chapter. In the 2^{nd} half of this chapter, we will examine and evaluate these. We broadly group these approaches into 2 main groups; the first uses the physical characteristics of signal propagation to bind an identity to an entity, whilst the second exploits the fact that no entity can perform an unlimited amount of computation.

2.4 Wifi Fingerprinting

There are many ways for devices to communicate wirelessly, many of which use radiowaves. Wifi is the name of a family of networking protocols that allow for this, it derives from the IEEE 802.11 standard. In order to use Wifi, a device must have at least one wireless antenna which can transmit and receive within the bands specified in the IEEE 802.11 standard; usually 2.4GHz and 5GHz.

When a device sends a packet using Wifi, it transmits a radio signal for a given number of nanoseconds from its antenna. This signal will attenuate as it travels further and further through space. After a certain distance, also known as the communicating range of the antenna, the signal will fade into background noise. The signal leaves the antenna in all directions simultaneously, as a

radio wave. Eventually, a small part of this wave will reach the receiver's antenna and the packet will be decoded out of it.

Other parts of the wave will either diffract around corners, reflect off some large objects, or scatter off many small objects [**?**]. Consequently, this means that a receiver may receive a packet from several different directions simultaneously, that is that it could encounter different parts of the same wave from different directions at the same time. This phenomenon is known as multipath scattering.

Multipath scattering has two interesting properties; it is practically impossible to predict, ahead of time, the distribution of signals around a receiver and it is unlikely that two receivers will observe the same signal propagation with sufficient multipath scattering. These properties allow for devices to "fingerprint" every packet they receive, such that two different transmitters cannot have the same fingerprint unless they are simultaneously located at the same place.

However, implementing multipath scattering-based algorithms have some technical constraints; namely that the receiving antenna must be able to measure the signal in each direction, and that a sufficient amount of multipath scattering must occur. Usually, these algorithms struggle in outdoor environments, where the environment may not provide objects for multipath scattering to occur.

The following papers discuss methods to circumvent these constraints, focussing on securing robot networks from spoofing and Sybil attacks.

2.4.1 Guaranteeing spoof-resilient multi-robot networks

Gil et al, [**?**] provides an interesting approach to some of the aforementioned problems, most notably the problem of requiring expensive hardware to allow the receiving antenna to measure the signal in each direction. They do this by inventing an algorithm that allows them to build a "virtual spoofer sensor" only using commercially available wifi hardware, which creates a "spatial fingerprint" from each transmission in the network. They use the output from this sensor to calculate a confidence metric α indicating their algorithm's confidence that a robot's identity is its entity. Finally, they characterise the theoretical performance of the "virtual spoofer sensor" and provide empirical evidence to support their claims by undertaking several experiments.

The authors build the "virtual spoofer sensor" by building upon *Synthetic Aperture Radar* (SAR) techniques, which allow a single antenna to be used to simulate an antenna array. SAR involves moving the antenna to different locations and taking snapshots of the signals received. These snapshots are then combined using signal-processing techniques to emulate a multi-antenna array [**?**]. The "spatial fingerprint" calculated, is then compared to the fingerprints of other clients, and clients with identical fingerprints are assumed to be Sybil attackers.

The authors evaluate their algorithm in the context of the following problem statement. Given an environment with several "clients", each expecting service from mobile "servers", dynamically find the optimal layout for the servers such that each "client" is served. A subset of clients are assumed to be malicious and are carrying out Sybil attacks in order to influence the "servers" to move closer to them.

The authors perform four experiments to validate their hypotheses:

1. They compare the performance of the "virtual spoofer sensor" in both an indoor and simulated outdoor environment, as expected, finding that multipath scattering is more effective in indoor settings, but also that adding a single reflector to the environment vastly improves performance.
2. They compare the effect of a stationary, moving, and power-scaling Sybil attacker on the ability of the "virtual spoofer sensor" to correctly classify agents, resulting in no false negatives, but many false positives.
3. They evaluate their system on the multi-agent coverage problem [**?**], finding that it can provide near-optimal results even when there are $3\times$ more spoofed agents.
4. They apply their system to a drone delivery problem, where the "server" needs to visit each real "client" to deliver a package and again find that their system provides near-optimal

results when there are 3× more spoofed agents.

2.4.2 Lightweight Sybil-Resilient Multi-Robot Networks by Multipath Manipulation

Huang et al. [?] take a different approach to Gil et al. [?]; instead of relying upon the environment to provide multipath scattering, they actively cause it by using backscatter tags. This offers two main advantages over Gil et al.: 1. the environment has a lesser effect on the performance of the Sybil attack detector and 2. the robots' antennae no longer need to move when capturing a fingerprint. Using the captured signal information, the robots again compute a fingerprint per transmission, normalise it to mitigate the effects of any power scaling attacks, and finally compare the normalised fingerprint to those of all others, treating any identities with sufficiently similar fingerprints as Sybil attackers.

Backscatter tags scatter signals that they encounter by rapidly absorbing and reflecting them. Backscatter tags also simplify the fingerprinting process, since robots are no longer required to perform small movements for SAR, nor must the software engage in expensive linear algebra to construct a multi-antenna array.

A key property of backscatter tags is that they operate passively and don't require a power supply. This reduces the cost of implementing this scheme as tags can be simply and inexpensively attached to robots. Another useful property is that the backscattering of the final signal is highly correlated with the positions of the tags, transmitting and receiving antennae, meaning that if two identities have very similar backscattering patterns, they are likely to originate from the same entity.

When a robot receives a signal, it receives a raw signal, which is first smoothed out with a moving average filter, to create the backscattered signal. Then it decodes a message out of the backscattered signal, and uses it, with signal processing techniques to deduce how much each tag contributed to the backscattering, and when each tag was "activated". Then the robot does more filtering to remove any backscattering from the environment, the resulting signal will be used to construct a signature for the transmission.

The authors then evaluate their implementation in both an indoor office environment and an outdoor rooftop environment. They find that their method is virtually indifferent to the surrounding environment, as they measure an average true positive rate of 97.6% and an average false positive rate of 5.1%.

2.4.3 Conclusions

Although both sets of authors extensively test their system, they make some problematic assumptions, which may be exploited by attackers.

1. They do not account for Sybil attacks using multiple antennae, which could transmit the same message at the same time, but with variable power levels. Each antenna would create a different multipath scatter, and if the relative powers between them were varied, then they could theoretically construct many false fingerprints.

2. They also do not account for collaboration between different attackers, which would function similarly to the previous vulnerability, where geographically distributed attackers could simultaneously send the same message, with different power scales, creating another set of false fingerprints. This method could produce a larger range of false identities but may encounter synchronisation problems between attackers.

3. Attackers could leverage methods similar to Huang et al. and physically augment their antennae to manipulate their multipath scattering, for example, one could create moveable barriers to prevent some scattering from occurring.

4. Both sets of authors assume that any noise from the environment will not be malicious; Gil et al. assume that it will follow a Gaussian distribution, whilst Huang et al. assume that standard signal processing techniques would be sufficient to filter it out. Both of these assumptions fail to account for the possibility that coordinated attackers emit noise above background levels, but not so high that it would seriously interfere with transmissions. This

would disrupt every transmission's signature and would prevent any pair of transmissions from looking alike.

2.5 Proof of Work

Proof of Work is underpinned by the insight that no entity in a network can perform unlimited computation. This leads to defences reliant on the idea that generating identities should be computationally expensive, to prevent any entity from being able to create an unbounded number of them. A PoW identity generation scheme would express an identity as the solution to some puzzle, and thus the identity can be validated by checking if it is a valid solution to the puzzle. This imposes two constraints: 1. the puzzle must be hard to solve and 2. the solution to a puzzle must be easy to verify, which means that, formally, the puzzle belongs to the *NP-Complete* class of problems.

A problem with this strategy is that it requires wasting computational resources, which may be in short supply for the embedded systems used in robotics. Furthermore, it requires that normal robots constantly pay a steep price for their security even without the presence of attackers.

Gupta et al. [**?**] design an iterative algorithm (**GMCom**) that solves the 2 problem, where if attackers spend T resources, whilst J new non-attacking identities are presented, then each non-attacker only needs to spend $O(\sqrt[2]{TJ} + J)$ resources. They refer to this property as the *assymetry* of the algorithm, as attackers must spend many more resources than non-attackers.

GMCom organises identities into a group, where a subset of them form a committee. When a new identity tries to join the group, it must first solve a puzzle set by the committee. Occasionally, the committee will seek to purge all attackers from the group, by issuing a *purge puzzle*, which must be solved by all identities before a new committee is formed, otherwise the identity will be purged from the group. This limits the wasted computation that good entities must perform since they only need to solve a single puzzle when entering a group, and occasionally thereafter, whilst an attacker would need to solve puzzles for each identity it claims, and would need to solve them repeatedly to avoid being purged.

However, this approach is not without its limitations. The authors assume that attackers will always only be able to command a fraction, α, of the total resources of the system, however, the heterogeneous nature of robots means that this may not always be guaranteed. For example, in a drone delivery system, each drone would likely only possess a small amount of computational power, but an attacker may attack the system using their desktop computer.

Chapter 3

An Investigation into the Security of RobotWeb

This chapter will investigate the behaviour of the RobotWeb when it is under attack. We will start by definining the expected behaviour of an attacker, including its incentives to attack. From then, we will construct and examine a simplified version of the RobotWeb to determine how victims of an attack will behave. Finally, we present and test several hypotheses about the behaviour of the RobotWeb under attack.

3.1 The behaviour of an ideal attacker

In order to predict the nature of an attacker's behaviour, we must first understand how normal robots behave in the RobotWeb. At all times, a robot will have a noisy estimate of its current position, as it moves the associated noise will grow. When robot r_1 encounters another r_2, it measures it and creates a factor between its own and r_2's current pose variable. Both robots will then exchange messages to each other over the factor, both using the messages to improve their own location estimates and decrease their uncertainty. In addition to other robots, a robot may measure a fixed beacon, which once again creates a factor between them, however, here only the robot will update its position estimate, as the beacon does not move. The quality of all sent message is dependent on the sender's own position accuracy and the accuracy of its sensors.

An attacker can take the form of either another robot or a beacon in the RobotWeb. Regardless of the form taken by the attacker, its goal would be to control the other robots' position estimates, by sending specially crafted messages.

All robots require a method to localise themselves as part of their normal operations, even attackers. Normal robots use the RobotWeb to do this, while we have no guarantees on the methods that potential attackers may use. An attacker may use an external system for localisation, or even a private RobotWeb, which renders it impervious to any consequences for its actions. However, some attackers may instead choose to participate in the RobotWeb in order to avoid the complexity of using an alternative system. Participating attackers would then have incentive to partially preserve the RobotWeb as they themselves are dependent on it. This dependency is likely cause attackers to only attack a small subset of robots, or only attack robots for a small amount of time.

It is self-evident that all attackers would seek to be effective i.e. thier attacks should have a high chance of working especially when no defences are present. In seeking effectiveness in the face of defences, attackers should not want their attacks to be easily identified, as identified attacks can be easily defended against. Thus it stands to reason that attacks are likely to be subtly executed, such that their victims would be able to believe them. For example, an attack that suggests that a small robot has moved 10km in 5s is guaranteed to fail.

Finally, in the event that multiple attackers are present in the environment, they would do best to collaborate rather than compete, as it would raise the effectiveness of them all. Since competing attackers are less effective than collaborating attackers, we shall spend less time considering

them.

3.2 A simplified system

Any theoretical investigation of the RobotWeb would require the investigator to analyse its backbone, the factor graph. However, the investigator would soon find themselves ensnared by complexity; caught in an intricate web of variable and factor nodes; messages constantly scurrying between them. Worse yet, the web would constantly be spun and unspun as robots moved closer and further from each other. In the face of this complexity, it becomes clear that a simplification is needed.

To start, we choose to limit our investigation to a single variable, chosen arbitrarily, in the factor graph. Fortunately, the properties of Belief Propagation allow us to reason about this variable without loss of generality. Equation 2.9 shows that the belief of a given variable is solely dependent on its connected factors.

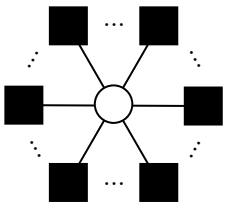

Figure 3.1: A single variable connected to a number of factors

One problem still remains with the above setup - a single variable can be connected to any number of factors. As a further simplification, we can group the factors together based off their shared characteristics, which include their origin (if they are internal to the robot or not), their type (what kind of sensor they represent) and their intentions (whether they will help or hinder the robot). For our purposes, we will split the set of factors F, into sets G and B based off whether the factors are "good" or "bad" for the robot. Good factors aim to steer the variable towards the a ground truth value, while bad factors aim to steer it away. Notice that the sets G and B form a cover of F, that is $F = G \cup B$.

We then take this a step further and replace each set of factors with a single "representative factor".

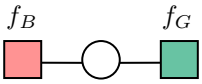

Figure 3.2: The above factor graph using "representative factors". f_G and f_B respectively represent the sets of good factors (G) and bad factors (B).

We calculate the messages from each of these representative factors as such:

$$m_{f_G \to x_i} = \prod_{g \in G} m_{f_g \to x_i} \quad m_{f_B \to x_i} = \prod_{b \in B} m_{f_b \to x_i} \tag{3.1}$$

Thus the belief of the variable becomes $p(x) = m_{f_G \to x} m_{f_B \to x}$, and since $G \cup B$ covers every factor connected to the variable, we can show that this replacement can be made without altering the variable's final result by equation 2.9.

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

For completeness we now present the equations for contents of the messages $m_{f_G \to x_i}$ and $m_{f_B \to x_i}$.

Each message is has an information vector η and a precision matrix Λ.

$$\eta_G = \sum_{g \in G} \eta_g \quad \Lambda_G = \sum_{g \in G} \Lambda_g \tag{3.2}$$

$$\eta_B = \sum_{b \in B} \eta_b \quad \Lambda_B = \sum_{b \in B} \Lambda_b \tag{3.3}$$

3.3 Theoretical properties of the RobotWeb under attack

3.3.1 Measuring the strength of an attack

From the above setup we will now aim to quantify the strength of an attack. For mathematical convenience and ease of understanding, we will derive these equations in 1 dimension, and later present the n-dimensional forms.

To measure the strength of an attack we want to understand the impact of the bad factors on the variable's final belief. To do this we shall use the Kullback-Leibler (KL) divergence between the variable's belief when it is safe and when it is under attack. The KL divergence is a measure of far the distribution Q is from the distribution P. The further the belief distribution under attack (BDA) is from the distribution suggested by the good factors, the stronger we say the attack is. Similarly, the closer the BDA is to the distribution suggested by the bad factors, the stronger the attack.

The KL divergence between 2 distributions of continuous random variables is:

$$D_{KL}(P||Q) = \int_{-\infty}^{\infty} log\left(\frac{p(x)}{q(x)}\right) dx \tag{3.4}$$

However a special case exists for Normal distributions, $N_1(\mu_1, \sigma_1^2)$ and $N_2(\mu_2, \sigma_2^2)$.

$$D_{KL}(N_1||N_2) = log\left(\frac{\sigma_2}{\sigma_1}\right) + \frac{\sigma_1^2 + (\mu_1 - \mu_2)^2}{2\sigma_2^2} - \frac{1}{2} \tag{3.5}$$

In 1 dimension the messages sent from f_G and f_B are:

$$m_{f_G \to x_i} = (\eta_G, \Lambda_G) = (\frac{\mu_G}{\sigma_G^2}, \frac{1}{\sigma_G^2}) \tag{3.6}$$

$$m_{f_B \to x_i} = (\eta_B, \Lambda_B) = (\frac{\mu_B}{\sigma_B^2}, \frac{1}{\sigma_B^2}) \tag{3.7}$$

From 2.9 we see that the BDA is equal to:

$$m_{f_G \to x} m_{f_B \to x} = (\eta_G + \eta_B, \Lambda_G + \Lambda_B) \tag{3.8}$$

$$= \left(\frac{\mu_G}{\sigma_G^2} + \frac{\mu_B}{\sigma_B^2}, \frac{1}{\sigma_G^2} + \frac{1}{\sigma_B^2}\right) \tag{3.9}$$

$$= \left(\frac{\mu_G \sigma_B^2 + \mu_B \sigma_G^2}{\sigma_G^2 \sigma_B^2}, \frac{\sigma_B^2 + \sigma_G^2}{\sigma_G^2 \sigma_B^2}\right) \tag{3.10}$$

The above distributions follow the following Normal distributions:

$$G \sim N\left(\frac{\eta_G}{\Lambda_G}, \frac{1}{\Lambda_G}\right) \tag{3.11}$$

$$B \sim N\left(\frac{\eta_B}{\Lambda_B}, \frac{1}{\Lambda_B}\right) \tag{3.12}$$

$$BDA \sim N\left(\mu_{BDA}, \sigma_{BDA}^2\right) \tag{3.13}$$

$$\sim N\left(\frac{\eta_G + \eta_B}{\Lambda_G + \Lambda_B}, \frac{1}{\Lambda_G + \Lambda_B}\right) \tag{3.14}$$

$$\sim N\left(\frac{\mu_G \sigma_B^2 + \mu_B \sigma_G^2}{\sigma_G^2 + \sigma_B^2}, \frac{\sigma_G^2 \sigma_B^2}{\sigma_G^2 + \sigma_B^2}\right) \tag{3.15}$$

So now we take the KL divergence between the good factors' distribution G and BDA. To make this easier to follow we derive each term in the sum separately.

First deriving the log term we get:

$$log\left(\frac{\sigma_{BDA}}{\sigma_G}\right) = log\left(\sqrt[2]{\frac{\sigma_{BDA}^2}{\sigma_G^2}}\right) \tag{3.16}$$

$$= \frac{1}{2}log\left(\frac{\sigma_{BDA}^2}{\sigma_G^2}\right) \tag{3.17}$$

$$= \frac{1}{2}log\left(\frac{\sigma_G^2\sigma_B^2}{\sigma_B^2+\sigma_G^2} \times \frac{1}{\sigma_G^2}\right) \tag{3.18}$$

$$= \frac{1}{2}log\left(\frac{\sigma_B^2}{\sigma_B^2+\sigma_G^2}\right) \tag{3.19}$$

Next we derive the fractional term:

$$\frac{\sigma_G^2+(\mu_G-\mu_{BDA})^2}{2\sigma_{BDA}^2} = \frac{\sigma_G^2+\left(\mu_G-\frac{\mu_G\sigma_B^2+\mu_B\sigma_G^2}{\sigma_B^2+\sigma_G^2}\right)^2}{2\frac{\sigma_G^2\sigma_B^2}{\sigma_B^2+\sigma_G^2}} \tag{3.20}$$

$$= \left(\sigma_G^2+\left(\frac{\mu_G\sigma_G^2+\mu_G\sigma_B^2-\mu_G\sigma_B^2-\mu_B\sigma_G^2}{\sigma_B^2+\sigma_G^2}\right)^2\right) \times \frac{\sigma_B^2+\sigma_G^2}{2\sigma_G^2\sigma_B^2} \tag{3.21}$$

$$= \left(\sigma_G^2+\left(\frac{\mu_G\sigma_G^2-\mu_B\sigma_G^2}{\sigma_B^2+\sigma_G^2}\right)^2\right) \times \frac{\sigma_B^2+\sigma_G^2}{2\sigma_G^2\sigma_B^2} \tag{3.22}$$

$$= \frac{\sigma_G^2\left(\sigma_G^2+\sigma_B^2\right)^2+\left(\mu_G\sigma_G^2-\mu_B\sigma_G^2\right)^2}{(\sigma_B^2+\sigma_G^2)^2} \times \frac{\sigma_B^2+\sigma_G^2}{2\sigma_G^2\sigma_B^2} \tag{3.23}$$

$$= \frac{\sigma_G^2\left(\sigma_G^2+\sigma_B^2\right)^2+\sigma_G^4\left(\mu_G-\mu_B\right)^2}{(\sigma_B^2+\sigma_G^2)^2} \times \frac{\sigma_B^2+\sigma_G^2}{2\sigma_G^2\sigma_B^2} \tag{3.24}$$

$$= \frac{\left(\sigma_G^2+\sigma_B^2\right)^2+\sigma_G^2\left(\mu_G-\mu_B\right)^2}{2\sigma_B^2(\sigma_B^2+\sigma_G^2)} \tag{3.25}$$

Thus the KL divergence between G and BDA is:

$$\frac{1}{2}log\left(\frac{\sigma_B^2}{\sigma_B^2+\sigma_G^2}\right) + \frac{\left(\sigma_G^2+\sigma_B^2\right)^2+\sigma_G^2\left(\mu_G-\mu_B\right)^2}{2\sigma_B^2(\sigma_B^2+\sigma_G^2)} - \frac{1}{2} \tag{3.26}$$

Similarly the KL divergence between B and BDA is:

$$\frac{1}{2}log\left(\frac{\sigma_G^2}{\sigma_B^2+\sigma_G^2}\right) + \frac{\left(\sigma_G^2+\sigma_B^2\right)^2+\sigma_B^2\left(\mu_G-\mu_B\right)^2}{2\sigma_G^2(\sigma_B^2+\sigma_G^2)} - \frac{1}{2} \tag{3.27}$$

From this we notice a disturbing detail - the KL divergence is quadratically affected by the μ_B. This suggests that the attacker's power is unbounded, so long as it chooses an appropriate σ_B^2. That an attacker can always command a robot to reject the evidence of its own peers and sensors. That the attacker can always craft messages to trick a robot into believing absurdities about its location.

3.3.2 Crafting the perfect message

From the results derived in the previous section, we know that theoretically an attacker will seek to increase its μ_B to ∞ and decrease its σ_B^2 to 0. However, in a real life scenario, it is unlikely that the attacker would fully exploit these powers, for two reasons.

1. Robots are unlikely to believe *incredibly* incorrect values - no sensible robot would believe that it has moved 500km away in the past 3 seconds.

2. Robots don't have infinite numerical precision, so large values of Λ_B ($\frac{1}{\sigma_B^2}$) would cause overflow errors, which would prevent the attacker from controlling the robot's belief.

Given these restrictions, attackers would set their values of μ_B to believable values, whilst choosing the largest value of σ_B^2 possible. We will now suppose that the attacker wishes to move its victim's belief from μ_G to μ_T. As before, the attacker sends a message with the form described in equation 3.7.

Does the standard deviation matter here?

So from equation 3.15, we see that:

$$\mu_T = \frac{\mu_G \sigma_B^2 + \mu_B \sigma_G^2}{\sigma_G^2 + \sigma_B^2} \tag{3.28}$$

Which can be rearranged to the form:

$$\mu_B = \frac{\mu_T \left(\sigma_G^2 + \sigma_B^2\right) - \mu_G \sigma_B^2}{\sigma_G^2} \tag{3.29}$$

Which can be used to calculate the μ_B that an attacker would send given a minimum σ_B^2. Thus the attacker would send the following message:

$$\left(\frac{\mu_T \left(\sigma_G^2 + \sigma_B^2\right) - \mu_G \sigma_B^2}{\sigma_G^2 \sigma_B^2}, \frac{1}{\sigma_B^2} \right) \tag{3.30}$$

TODO: Derive this

3.3.3 Scaling back up to n-dimensional space

3.4 Hypotheses

In this section, we will present 5 hypotheses about the behaviour of the entire RobotWeb under attack, using the above analysis. These hypotheses will then be experimentally tested in the next section.

3.4.1 Bounds on μ can be slowly escaped

An intuitive defence against the attackers is setting an upper bound on how far a message can suggest the robot is from its previous position. However, we believe that this approach is ineffective, and can be easily evaded by attackers. In this subsection, we will lay out how this defence would work and how it can be bypassed.

For the upper bound to be effective, it must occupy a "Goldilocks zone" - it cannot be too large or too small. If the upper bound is too large, it would present attackers with ample opportunities to control the robot's belief, especially since they aim to suggest somewhat plausible positions. If the upper bound is too small, it would effectively prevent the robot from listening to any dissenting messages from other good robots.

Now suppose that at time t, the robot is at position μ_t and has a upper bound of ϵ. It would then evaluate each incoming message and reject it if its proposed position μ_t' is a distance ϵ away from μ_t. After filtering out all "bad" messages, the robot would use the remaining messages to determine μ_{t+1}.

Being aware of this scheme, and attacker would seek to incrementally attack the robot. It would start by proposing a position μ_t'' that lies between μ_t and its target location μ_t', such that μ_t'' is accepted by the robot. This would successfully shift the robot's position estimate μ_{t+1} to μ_t''. In the next iteration, the robot would reject any proposals a distance of ϵ away from $\mu_{t+1} = \mu_t''$. Hence over several iterations, the robot's position estimate would slowly shift to μ_t', and thus the attacker would be able to escape the bound.

Would a diagram be useful here?

3.4.2 Bounds on Λ can be quickly escaped

As shown previously, the strength of an attack is highly dependent on its ability to propose arbitrary values of Λ. Which leads to another intuitive defence strategy - robots set an upper bound on the Λ values that they receive. We believe that this strategy is effective when there is only a single identity spreading misinformation, counterbalanced by many others sending reliable information. In this subsection, we will provide a justification for this belief as well as a strategy that an attacker could take to bypass the upper bound on Λ.

The Λ value of a message can be though of as its "pull", or how strongly the message would move a variable towards its proposed location. The greater the value, the stronger the pull. With this in mind, we can think of the robot's location estimate as being "pulled" by good and bad factors, where good factors pull the estimate towards a ground truth, whilst bad factors pull it to an alternate location.

Norms?

When sending a message, each robot decides the strength of the message's pull. Good robots limit their strength to the accuracy of their sensors, while attackers will set their strength to the fullest extent. This once again makes the case for the use of a reasonable upper bound on the strength of a message, similar to the one described above.

We argue that this upper bound is unenforceable in practice, for an attacker can simply "split" their message into several weaker parts. So far in our analysis, we have treated Λ_B as if it were sent in a single message, however from equation 3.3 we can see that it may also be the result of several bad robots sending messages. In fact, if the upper bound on an individual message's Λ is λ, then an attacker could simply send several messages from several different identities to arrive at a strong Λ_B, in a Sybil Attack.

Should I add maths here too?

Aside: Attacks are uncorrelated

Considering the "pull based model" of variables, it stands to reason that in a Sybil Attack, the messages sent by individual identities will not be correlated with each other, as that would greatly simplify the detection and prevention of attacks. Instead, we believe that Sybil Attackers would send messages with wildly different μ values, that would "resolve" to the actual attack that the attacker intends.

3.4.3 Long histories are detrimental

One little discussed feature of the RobotWeb so far has been its time-windowed history. Instead of remembering every past position that that robot has had, it instead chooses to only remember the past h positions. This improves the performance of robots in the RobotWeb, as they store fewer pose variables and thus perform fewer floating-point operations when updating beliefs. In the original paper, Murai et al. show that the impact of time-windowed history on the average trajectory error of each robot is negligible.

Add graph

We believe that keeping the full position history not only has an adverse impact on the computational performance of a robot, but also amplifies the strength of attacks. If an attacker is able to successfully attack a robot at time t, then the pose variable at t, v_t will contain a μ close to the attacker's target μ, and its Λ will be high. If v_t is then used to estimate the robot's position at time $t+1$, then it will effectively also attack the robot, as it would further the attacker's belief. The longer the history kept by a robot, the more power an attacker can gain over it.

3.4.4 Attacks can spread on their own

Similarly to the previous hypothesis, we can assume that any other variable connected to v_t will be attacked by it. Thus a victim of an attack will unintentionally attack all those it contacts, meaning that attacks have a degree of contagion.

Attackers may not be able to prevent or limit contagion, as they would need to strongly pull unintentional victims back to a ground truth value. Since no robot has absolute certainty about the locations of any other, the attacker will actually pull them to values close to their ground truth. However, as there is no mechanism to increase Λ, the attacker will still make the other robots overconfident about their locations, which may bias them in the long term.

3.4.5 Robots are most vulnerable on startup

Our final hypothesis is that robots that have just started up are likely to be the most affected by attacks, as they wouldn't have a strong estimate of their location. This would mean that their Λ_G is much lower than older robots, increasing the strength of the attack.

3.5 Experimental evidence

3.5.1 Experimental setup

We're going to test in a simulator. The simulator will have n robots in total, where 1 is an attacker and the rest are law-abiding citizens. Each robot will carry a single range-bearing sensor, which allows them to sense how far another robot is, and its bearing relative to themselves. Each sensor has a limited range in which it can measure another robot, and a normal robot will only send messages to robots that it can see. In contrast, the attacker will not bind itself to this ideal instead, it will always send messages to its victims. All robots will move along a path, and the attacker's job will be to convince them to move along a different path.

Because the RobotWeb currently isn't deployed anywhere, we have a dearth of real-world situations to test against. This leaves us with the choice of either inventing realistic situations or using simple, artificial situations. Given that right now we want to show that attacks are possible, we will test our hypotheses in lab settings, rather than aim to create realistic situations. The risk of this approach is that some effects may be more pronounced in these settings rather than in the real world, but that's alright because it would help with defenses later on.

We choose to use the following map to test on. The robots will move along the green path, whilst the attackers will try to convince them that they are actually moving along the red path. The robots' belief of their path is the black path. The bottom robot is the attacker.

3.5.2 Testing hypothesis 1

If you remember 3.4.1, it suggests that attackers won't be deterred even if robots use a bound on how far messages can displace them. To test this, we use a single attacker and single victim. The attacker also creates an onramp between the robots' actual positions and where it wants them to go. We vary the number of steps that the onramp has. (A series of checkpoints? and we vary how far each checkpoint is from where the robot thinks it is.). And measure the ATE.

We have 1 attacker and 2 victim, all three of which can always measure each other. The history length is 2, to avoid any issues from 3.4.3 (which we later show is legit). The victims reject all messages that are 5cm away from where it thinks it is. We repeat each experiment 10 times. The attacker presents a confidence level of x.

Looking at the Average Trajectory Error of the victims against the distance between checkpoints, we see that a cliff emerges, where once the distance between checkpoints is small enough, that the victims will always follow the attacker. Thus confirming our hypothesis.

Interestingly enough, we also see that the cliff starts a little short of the displacement bound, which is likely due to the noise in robots' measurements leading them to temporarily believe that they are closer to the robot's path than they actually are.

Looking at both victims' ATE over time, we see that they do reject the attackers' messages until they reach the onramp, which further lends credence to the hypothesis.

Also note that this means that an attack does take some time to occur, since the attacker must wait for its victim to reach the onramp.

3.5.3 Testing hypothesis 2

Remembering 3.4.2 suggests that a single attacking robot doesn't need to do anything funky to its confidence level, if it creates a bunch of fake identities. To test this we'll vary the number of fake identities that the attacker creates to help it. Again we'll measure ATE.

We have 1 attacker, and 10 victims, all of which can always measure each other. The history size is 2 again, so we avoid any attacks from weaponised histories. Each fake identity that the attacker presents doesn't alter its confidence level. And repeat each experiment 10 times.

Looking at the ATE against number of fake identities, we see that once the attacker presents enough (almost a majority) of fake identities, it can manages to overwhelm the efforts of all good robots and seizes control of localisation in the network. We also note that this doesn't take many ticks to occur. Thus confirming the hypothesis.

3.5.4 Testing hypothesis 3

Remembering 3.4.3 suggests that having a long history vector is bad. To test this we'll vary the length of the history vector and again measure the ATE. Since reducing the amount of history available is also likely to negatively affect performance in the absence of attackers, we will also investigate the role of history in improving the performance of the system.

We have 1 attacker and 10 victims, which are all able to measure each other. The attacker will send messages with a confidence level of x%. We repeat 10 times. We will conduct this experiment both with and without the attacker to measure the effect of history size.

Looking at the ATE against history size when an attacker is present reveals that longer histories do in fact harm robots' localisation, but only until a point after which this effect levels off. This is likely due to the fact that the past poses are fully attacked by then, and so can't attack the robots further.

Looking at the ATE against history size when no attackers are present, tells the opposite story - that a longer history does help in the good times, albeit very slightly. We believe that this effect is only due to the fact that the robots have good odometry, which leads us to rerun the experiment with worsed odometry sensors. We can now see a more pronounced positive effect of logner history vectors, which again levels off.

This suggests that for real-world scenarios there will always be a trade-off to be considered. Longer histories may improve localisation when no attacks are taking place, but at the cost of increased risk and a higher computational load.

3.5.5 Testing hypothesis 4

Remembering 3.4.4 suggests that attacks can spread on their own. We'll test this by having the attacker only attack a single robot, and measuring the loss effect it has on all others around it.

We have 1 attacker, 1 victim and 10 other robots, where the attacker only attacks the middle robot, and all robots can only measure the ones directly adjacent to themselves. We do this to measure the number of hops the attack must take before becoming neutralised. The attacker sends messages with x% confidence. We repeat 10 times.

Looking at the ATE against number of hops, we can see that the contagion effect does exist, as the robots directly adjacent to the victim have a higher ATE than they would in the control group, although it is much less than the main victim. Thus the hypothesis is correct.

This suggests that in real-world attacks, attackers will not be able to rely on any robots that are able to talk to their victims. Meaning attacks aren't as clean as they attackers would want them to be.

3.5.6 Testing hypothesis 5

Remembering 3.4.5 suggests that robots are most vulnerable to attacks when they have just entered the RobotWeb. We will test this by introducing a small delay between a robot starting up and it joining the web and measuring the effect of this delay on the ATE.

We have 1 attacker and 1 victim here, which can measure each other. We'll vary both the history size and the startup delay, as the two are likely to be related. The attacker will have a confidence level of x%. 10 repeats.

Looking at the history size vs startup delay heatmap, we see that the 2 variables are in fact linked...

This suggests that delayed joining might be a strat.

Chapter 4

Evaluation Plan

I plan to evaluate this thesis by simulating many different scenarios, varying the paths the robots will take, the number of attackers in the system, the types of attackers in the system and if I find multiple defences, then varying the different defences too. From these runs, I will compute an error metric measuring the deviation of each robot from its actual path. I then plan to compare the results to scenarios where there are no attackers, no defences and no Robot Web, to see how each effect the localisation of the robots.

I hope to observe that the defences I will implement are effective at preventing attackers from being able to influence the network, this would mean that the case where there are no attackers would have a similar error to the case where there are attackers and defences.

I also hope that the defences are not invasive, and so wouldn't require major changes to the hardware of robots. This includes computational resources, which are likely to be limited.

I would also like to evaluate the defences in a real-world environment.

Chapter 5

Conclusion

5.1 Ethical Considerations

"A new device merely opens a door; it does not compel one to enter"
(Lynn White [**?**])

From the discovery of coffee leading to the "Age of Enlightenment" to the invention of Boolean algebra creating our modern digital age, history has repeatedly shown us that it is impossible to fully understand the implications of new discoveries and nascent technology. With this in mind, we provide a short discussion of potential ethical issues which, we believe, may arise as a consequence of the research conducted in this thesis.

As with all research enabling autonomous robotics, we must consider potential military applications. Many militaries today already use unmanned aerial combat vehicles in their operations, if they were to incorporate this research, they may be able to improve their performance by allowing them to share information securely. However, we do not believe that this is likely to occur as militaries tend to have highly centralised structures, where each robot would have prior knowledge about other trusted robots in the network. Whereas our research focuses on providing security to untrusted robots in decentralised networks.

Another potential misuse of this research would be enhancing the capabilities and security of surveillance robots. In this scenario, an authoritarian regime would use robots to continuously monitor their citizens. The robots would communicate with others in their immediate surroundings to coordinate their search and could be vulnerable to cyber attacks where several are hijacked. The hijacked robots would then send incorrect messages to prevent certain areas from being searched. However, it is unlikely that this research would be an ideal candidate for implementing such a dystopia, as a single party would own the robots and would find it much simpler to implement centralised security measures.

Alongside the unethical misuses of this research, there exist several scenarios where it would confound unethical groups. One intended use case is to implement a common robotic infrastructure for autonomous robots owned by many different parties. Here the decentralised nature of the infrastructure would provide asymmetric robustness against hackers or governments seeking to unilaterally disrupt and destroy the infrastructure as they would not be a single point of failure.

In conclusion, there are many scenarios where this research may be misused to the detriment of humanity, yet we are not convinced that this research would be the most appropriate in those examples. Furthermore, given how this research seeks to defend against bad actors, we believe it is more likely to be a benefit to humanity.

Appendix A

First Appendix