CONTENTS

1 INTRODUCTION

Traditional attempts at occupancy grid mapping employ a *inverse approach* [Thrun, 2003]. These attempts use a model which employs probabilistic inference. Sensor readings are fed into the model and environmental obstacles are derived using the model [Konolige, 1997, Elfes, 1989, Matthies and Elfes, 1988, Thrun, 1993].

Standard models for robot mapping assume that each cell has conditional independence. Cells are separate entities from each other with no connection or influence to the rest of the grid. There are a variety of issues which arise from this assumption because of the unreliability of robot sensors and environment types where a holistic map cannot be generated and we must defer to inferencing based on what can be observed.

In this paper we introduce Conditional Random Fields (CRFs) to model the occupancy grid such that we can represent dependencies and influence of each cell. CRFs allow algorithms such as Loopy Belief Propagation to provide a method for inferencing the occupancy of cells.

1.0.1 *Motivation*

Humans intuitively know that a wall is a continuous stretch of space even if a portion of the wall is obscured by a foreground object. The traditional mapping paradigm does not encode this intuition and therefore does not fully take advantage of contextual information for each cell.

A laser scan is typically a 'sweep' across a range, these sweeps return the angle and distance at which an object is detected. Due to the nature of a sweeping scan, objects further away but still within scan distance may be in between scans of the same set (a single scan in a sweep may be visualised as a thin line). Angles are discrete in the sweep while the range of objects, which may be detected, is continuous.

One of the problems of the independancy assumption is it assumes sensors and readings are perfect. This is rarely the case in a real world scenario, and there are often conflicts in sensor readings. A cell at a coordinate space may be occupied in one set of scans and unoccupied

in the next. Even scans in a single sweep may resolve an area to be both occupied and empty if a discretised method of representation were used (such as an occupancy grid). We use conditional probability to encode observations, which may be erroneous, onto a persistent grid which accounts for conflicting sensor readings.

Noisy environments which results from a variety of factors such as specular reflection or simply dust obscuring laser scans may be resolved using the inferencing method proposed in this paper.

put under either here or the problem statement: high angle increment in laser scans creating gaps, blocks of occupied space like a square cannot be fully mapped, decrease certainty as robot moves, low sensor certainty i.e. specular readings, can lead to developments in a creating a semantic map - references.

1.0.2 *Problem Statement*

Mapping certain environments may have excess noise making distinct objects hard to visually distinguish and produce grids with low certainty making robot localisation difficult. We introduce inferencing algorithms to provide a form of de-noising.

Traditional occupancy grids cannot encode contextual information due to the independancy assumption between cells. Scans being dispersed too widely misses out scanning potential objects and foreground occlusion can cause the mapping process to be incomplete. Parameters for inferencing algorithms may be learned from a similar environment beforehand to infer the state of these unknown areas of the map.

1.0.3 *Summary of Contributions*

This paper demonstrate the method in which an occupancy grid may be translated into a Conditional Random Field. This opens up all the possible methods of inference such as Loopy Belief Propagation which may be performed on a CRF. The reverse is also implemented, going from a CRF post processing to an occupancy grid.

A learning algorithm was implemented specific to maximising the accuracy of the resultant occupancy grid generated after performing inference on the CRF.

The model proposed in this paper allows cells to be encoded such that they are dependent on every other cell in the grid. We approach the potentially large computation overhead by utilising several modifications to existing algorithms sacrificing accuracy for speed while

still producing an overall improved accuracy from basic occupancy grid mapping.

A robot simulator was created to provide absolute ground truths which could be compared against after scans and inferencing for an accurate measure of the models performance. The system was setup to be integrated with ROS, a framework allowing for direct hardware interaction with the robot.

Experimental results are provided in the form of a visual map after inferencing and metrics such as precision and recall.

1.0.4 *Thesis Outline*

2

BACKGROUND

2.1 PROBABILITY THEORY

A probability distribution is defined by the probability of each event in a set of events. An event is a possible configuration in the space all possible outcomes, Ω. There is a set of measurable events S to which we may assign probabilities.

$$a \in S, S \subset \Omega$$

For example, in an die roll, $\Omega = \{1,2,3,4,5,6\}$. $a = \{1\}$ refers to the case where the die is rolled to be 1. An event $a = \{1,2\}$ is the case of the roll be either 1 or 2.

The event space satisfies three basic properties

1. It contains an empty event $\varnothing$ and trivial event Ω.
2. It is closed under union. If $\alpha, \beta \in S$ then $\alpha \cup \beta \in S$.
3. It is closed under complementation. If $\alpha \in S$ then $\Omega - \alpha \in S$.

These properties imply closure under other boolean operations.

A probability distribution P over (Ω, S) refers to the mapping of a to real values with conditions:

1. $P(\alpha) \geqslant 0, \forall \alpha \in S$
2. $P(\Omega) = 1$
3. The probability that one of two mutually disjoint events occurring is the sum of their probabilities
   If $\alpha, \beta \in S$ and $\alpha \cap \beta = \varnothing$ then $P(\alpha \cup \beta) = P(\alpha) + P(\beta)$

In this paper there are several sections (2.5.1, 2.6, 3.4) which utilise conditional probability. We can define a correlation between two events α and β and update the beliefs of one based on the observations of the other. The conditional probability of β given α is:

$$P(\beta|\alpha) = \frac{P(\alpha \cap \beta)}{P(\alpha)}$$

Bayes's Rule shows:

$$P(\beta|\alpha) = \frac{P(\alpha|\beta)P(\beta)}{P(\alpha)}$$

2.1.1 *Random Variables*

Random variables are defined by a function that maps each event in Ω a value [Bishop et al., 2006]. Random variables can only take values defined in a domain or set of discrete values. In this paper, uppercase letters like X, Y are used to denote random variables and lowercase x, y denotes the value of the random variables.

Binary random variables will have, for example, x^0 and x^1 representing their possible states. Moreover, we generally use $P(x)$ as shorthand for $P(X = x)$.

2.1.2 *Binary Variables*

In this paper we primarily deal with binary variables. The outcome of an event can either belong to one of two states. Specifically we deal with the probability of a cell on a grid being occupied and empty (non-occupied).

The probability of a cell, c, being occupied ($val(occupied) = 1$) is given by:

$$p(c = 1|\mu) = \mu$$

where $0 \leqslant \mu \leqslant 1$ and the implications of it being a binary variable give the probability of emptiness as $p(c = 0|\mu) = 1 - \mu$. The probability distribution over c can be written as:

$$Bern(c|\mu) = \mu^c(1 - \mu)^{1-c}$$

This is the *Bernoulli* distribution [Clewer, 1999, Kullback, 1935]. Suppose we observe the value of c so that we have a data set $\mathcal{D} = \{c_1, ..., c_n\}$, we can use this distribution to construct a likelihood function which will allow us to visualise the probabilities of values that c can take.

$$p(\mathcal{D}|\mu) = \prod_{n=1}^{N} p(c_n|\mu) = \prod_{n=1}^{N} \mu^{c_n}(1 - \mu)^{1-c_n}$$

The maximum of the likelihood function is the most likely value of c. This maximum will be used in section 3.5 to learn the optimum parameters for our inference algorithms.

2.1.3 *Marginal and Joint Distributions*

The distribution of events that can be described over a random variable X is called the *marginal distribution*, $P(X)$. In this paper we will often find the marginals of a cell to determine whether it is occupied

P(cell = occupied) and empty P(cell = empty). This is a normalised distribution if the probability of both sum to 1.

A joint distribution allows us to consider the cell state given another random variable, which could be the cell state of a neighboring cell. A joint distribution allows us to obtain the probability a cell C is occupied given its neighbour N is occupied.

The joint distribution over a set of random variables $\mathcal{X} = \{X_1, ...X_n\}$ of random variables is written as $P(X_1, ..., X_n)$. The distribution specifies the probabilities to events that these random variables refer to. The distribution is over all events involving all variables in $\mathcal{X}$.

	N^1	N^0
C_1	0.9	0.1
C_0	0.1	0.9

Table 1: Sample Joint distribution

Figure 1 is a possible assignment of probabilities for a joint distribution between two cells. These cells have random variables for the events that a cell is occupied and empty. The joint distribution specifies the probability that C is occupied given N is occupied, C is empty given N is empty, C is empty given N is empty and C is occupied given N is empty. The joint distribution specifies a valid probability distribution if the probabilities add to 1.

We define the *marginal distribution* over a random variable X_n as the distribution over all events involving X_n in $\mathcal{X}$. We obtain $P(X_n = x_n)$ for all possible values of x_n.

$$Pr(X = x) = \sum_y Pr(X = x|Y = y)Pr(Y = y) \tag{1}$$

We can find the marginal probability of any of these events by summing over it (equation 1). For example, to find the conditional distribution that C is occupied based on observations of N we add the row of probabilities it belongs to. This also works column wise, to find the probability N is occupied we sum the column it belongs to.

Suppose we observe the probability of occupancy of N as 0.8:

$$Pr(C = occupied|N) = (0.9 * 0.8) + (0.1 * 0.2) = 0.74$$

2.2 GRAPHICAL MODELING

The theory which precedes all data structures used for inference in this paper is contained in graphical modeling. Complex systems may

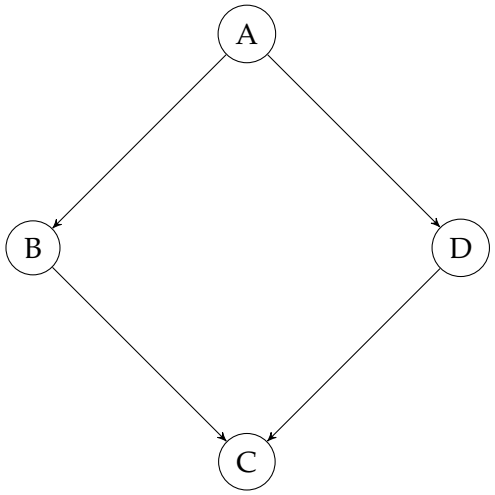
be represented in a graph as a structured probabilistic model. These systems have multiple interrelated aspects, the possible states of these aspects are described as a set of random variables.

A structured probabilistic model allows inferences on the value of variables known to the model [Koller and Friedman, 2009]. A *joint distribution* (section 2.1.3) is constructed. This distribution describes the space of all possible assignments to some set of random variable X.

In a directed graph, G, such as Bayesian Networks (section 2.4), with edges in E(G) for some vertices $X_1, ..., X_n \in V(G) : uX_1, X_1X_2, ..., X_nv$. There exists a *cycle* in G if there is a non-empty path from u to u. If G were a general graph, it is *connected* if for any vertices $u, v \in V$ there exists a path from u to v.

We define neighbours of a vertex $u \in V(G)$ of a graph as:

$$N(u) = \{v \in V | uv \in G\}$$



Independencies $(C \perp A|B, D)$

$(D \perp B|A)$

Figure 1: A sample Bayesian Network demonstrating independences.

In figure 1 there is a list of independences of the example Markov network (detailed in section 2.5).

For a subset of variables X, Y, Z, $(X \perp Y|Z)$ is equivalent to the statement, X is independent of Y given Z. In an example we can simplify calculation of a target distribution, P, by reasoning:

$$P(A|D, B, C) = P(A|D, B)$$

This is logical because information about C is obtained through D and B. The model can be used to find the probability of A being a specified value, given observations in the other variables. It allows the distribution to be written down tractably and is transparent such that it is understandable semantically.

Representation of the graph in this manner allows the distribution to be 'broken up' into smaller factors. Since we can ignore C in the calculation at A provided we processed B and D, the space of possible configurations is decreased and we do not need to define a redundant factor between A and C.

We can represent a generic global function as the product of these factors. We formally define a set of factors $F = \{f_1, ..., f_m\}$ and variables $\mathcal{X} = \{X_1, ...X_n\}$. A factor is defined as $f_j : X_j \rightarrow [0, k]$ where $X_j \subseteq \mathcal{X}$ for some $k \in \mathbb{R}, k > 0$. X_j denotes all possible values for variables in $\mathcal{X}$. $x = \{x_1, x_2, ..., x_n\} \in X$ are the values of all variables in $\mathcal{X}$. $x = \{x_{j1}, x_{j2}..., x_{jn}\} \in X_j$ are the values of the arguments for X_j of factor f_j [Kao, 2005].

2.2.1 *Multivariate Time Series*

Work has been done by [Eichler, 2012] on multivariate time series, this work is out of the scope of this paper but may have interesting applications to robot mapping. Potential applications include sampling environments which are not constant as assumed by this paper. Environments where there are moving objects may be modeled.

The technique described uses non-linear and non-parametric models for analysis on the variables. This technique models itself after work by [Fan and Yao, 2003, Rothman, 1999].

The representation is done through mixed graphs, where each variable is a complete time series represented by a single node and directed edges indicate possible Granger-casual relationships. Undirected edges representative of dependence between nodes at a specific state in time.

Graphical modeling has also been applied to univariate cases [Reale, 1998] which treated both cases separately. The work involves using multivariate AR models which simplifies conditional dependencies.

This is done though the use of *inverse variance lemma* and *demoralisation* to reduce the space of models to consider.

Occupancy grids model the belief a system has on whether a space is occupied or non-occupied. Occupancy grids allow application of fuzzy logic belief systems in the representation of an environment. A grid can be built up without any prior map thought this is not always the case.

The seminal paper on occupancy grids and the methods used in construction and utilisation for robot mapping is by [Elfes, 1989].

An occupancy grid $O(x)$ is a discrete state stochastic process. $O(x)$ defines a map of continuous spatial coordinates:

$$\begin{aligned} x &= (x_1, x_2, ..., x_n) \\ x_n &\in (Z) \end{aligned}$$

The grid is arranged in a discrete spatial lattice.

0.2	0.3	0.2	0.3	0.9	0.9	0.5	0.1	0.1
0.3	0.3	0.4	0.9	0.9	0.8	0.7	0.1	0.4
0.1	0.2	0.6	0.8	0.9	0.7	0.7	0.2	0.3

Figure 2: The structure of a 9x3 occupancy grid.

In this paper we deal with occupancy grids which are two dimensional, $x = \{x_1, x_2\}$.

A *cell* (circles in Figure 2) in the grid is the one to one mapping of $O(x)$ for a given set of x_n. A cell is represented as individual circles in Figure 2.

A cell has an associated value, $s(C)$ which is a discrete random variable corresponding to two states. These states are occupied and non-occupied (empty). The cell states are exclusive and exhaustive such that:

$$P[s(C) = occupied] + P[s(C) = empty] = 1$$

We may store the probability of occupancy (shown as the label for cells in Figure 2) and derive the probability of emptiness (this comes in handy in algorithms discussed in section 2.6).

A state value of $s(C) = 0.5$ is an unknown cell. This paper displays the occupied probability in cells unless specified otherwise. A value

of 1 indicates absolute certainty a cell is occupied.

The occupancy grid is used in the modeling of the environment for algorithms which aim to localise (see section 2.8) the position of the robot through various algorithms which are described in detail in the literature [Temeltas and Kayak, 2008, Dissanayake et al., 2001, Weingarten et al., 2004, Bailey and Durrant-Whyte, 2006]. The grid may also be converted into another data structure which can provide a basis for inference and analysis.

The occupancy grid can only be formulated as an estimation of an environment. The formulation of a grid must consider factors such as the accuracy of sensors and the odometry readings. In the case of localisation, it is an estimation theory problem. The robot attempts to match features in the environment which it has seen before and using that data attempts to localise its position, see section 2.8 on SLAM for more detail. This process can become unreliable in environments where features are similar.

Bayesian networks encode a joint distribution in a directed acyclic graph. The nodes of the graph $\mathcal{G}$ are the random variables and the edges correspond to the influence of the joined nodes on each other.

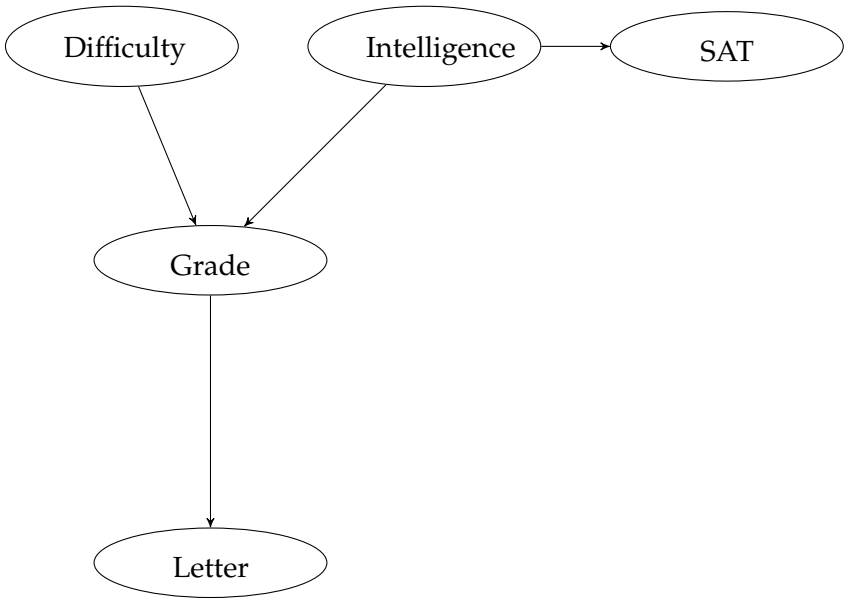


Figure 3: Student Bayesian network.

d^0	d^1
0.6	0.4

i^0	i^1
0.7	0.3

	s^0	s^1
i^0	0.95	0.05
i^1	0.2	0.8

	g^1	g^2	g^3
i^0, d^0	0.3	0.4	0.3
i^0, d^1	0.05	0.25	0.7
i^1, d^0	0.9	0.08	0.02
i^1, d^1	0.5	0.3	0.2

	l^0	l^1
g^1	0.1	0.9
g^2	0.4	0.6
g^3	0.99	0.01

Table 2: Conditional probability distributions for the Student Bayesian network.

An example is shown in Figure 3. Each variable X in the model has an associated *conditional probability distribution* (CPD). This specifies a distribution over the values of X. All possible joint assignments of values for each node is shown in Table 2 along with an example CPD configuration. This network can be broken down into components we wish to analyses. For example, we see that a student's "Grade" is dependent on the variables difficulty and Intelligence. The dependency is shown in the joint probability distribution table. Whilst we see that

since there is no flow of edges into Difficulty or Intelligence they are determined outside of this model.

We can find the probability of a given state by using:

$$P(I, D, G, S, L) = P(I)P(D)P(G|I, D)P(S|I)P(L|G) \tag{2}$$

For example, to find a specific state where the student is intelligent, the course is easy, the student got a 'B' grade (g^2), with a high score on the SAT and getting a weak letter can be calculated by substituting the probabilities in the CPDs.:

$$\begin{aligned} P(i^1, d^0, g^2, s^1, l^0) &= P(i^1)P(d^0)P(G^2|i^1, d^0)P(s^1|i^1)P(l^0|g^2) \\ &= 0.3 \times 0.6 \times 0.08 \times 0.8 \times 0.4 = 0.004608 \end{aligned}$$

We can say that letter is independent of difficulty and Intelligence given grade. (explained briefly in section 2.2). We can calculate the marginal probability of letter assuming we know grade. We do so by summing out the variable of letter so we get a distribution over letter. An example of this process is shown in section 2.5.

Markov Random Fields (MRFs) or Markov networks are undirected cyclic graphs allowing representations of general graphs with foundations developed by [Preston, 1973, Spitzer, 1971]. MRFs have been used in the field of computer vision, with the first such work by [Geman and Geman, 1984].The works are based off a specific model called the *Ising Model* [Ising, 1925].

In a generalised model [Koller and Friedman, 2009], for MRFs we define a factor:

$$\phi(A, B) : Val(A, B) \mapsto \mathbb{R}^+$$

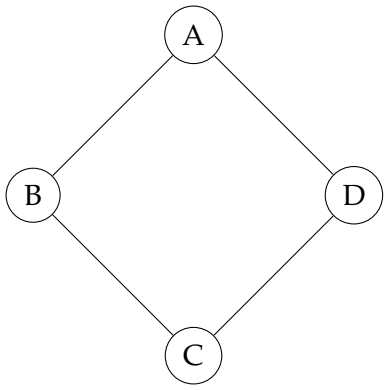


Figure 4: A Markov network.

$\phi_1(a, b)$ correlates positively with compatibility between A and B and similarly for $\phi_2(b, c)$ for nodes B and C and so on for all edges in Figure 4. These factors define local interactions between directly related variables. For nodes I and J and a defined $\phi(I, J)$, we draw an undirected edge between nodes I and J. Unlike Bayesian networks we cannot represent the relation between nodes purely as a conditional probability distribution. A joint distribution over D is a factor over D while a conditional distribution $P(X|U)$ is a factor over $\{X\} \cup U$ which is created using a method known as *parameterisation*. Parameterisation involves creating these factors based on the structure of the underlying graph. Factors can be over arbitrary subsets of variables.

The product of the factors, $\phi_1(X, Y) \times \phi_2(Y, Z)$, of three disjoint sets of variables, X, Y and Z is:

$$\psi(X, Y, Z) = \phi_1(X, Y) \times \phi_2(Y, Z) \tag{3}$$

where ψ is a factor and Val(X,Y,Z) $\mapsto \mathbb{R}$.

Following that we can define a *Gibbs distribution* to define an undirected parameterisation of a distribution. A distribution P_ψ parame-

terised by a set of factors $\psi = \{\phi_1(D_1), ..., \psi_K(D_K)\}$ is a Gibbs distribution if it follows:

$$P_\psi(X_1..., X_n) = \frac{1}{Z}\hat{P_\psi}(X1, ...X_n) \quad (4)$$

$$\hat{P_\psi}(X_1..., X_n) = \phi_1(D_1) \times \phi_2(D_2) \times \phi_m(D_m) \quad (5)$$

$$Z = \sum_{X_1,...,X_n} \hat{P_\psi}(X_1, ...X_n) \quad (6)$$

$P_\psi(X_1..., X_n)$ is referred to as the global model and Z is called the partition function which is used to normalise so that the probability distribution is valid.

Given $I(H) = \{X \perp Y|Z) : sep_H(X, Y|Z)\}$ (graph independencies discussed in section 2.2). If P ($P(\bar{x}) > 0, \forall\bar{x}$) satisfies I(H) then H is an I-map (independency map) of P. The *Hammersley Clifford theorem* states if P factorises over H then H is an I-map of P [Kleinberg and Tardos, 1999]. In other words, it is the equivalence of an MRF and Gibbs distribution given a positive distribution [Kao, 2005].

Ising attempted to explain empirically observed facts about ferromagnetic materials. Specifically, we look at a model which creates links between neighbouring dipoles [Kindermann et al., 1980]. Ising formulated a MRF as follows:

Consider the sequence, $0, 1, 2, ..., n$. This sequence corresponds to a point on a line where each point has a binary state, up or down, which refers to its spin.

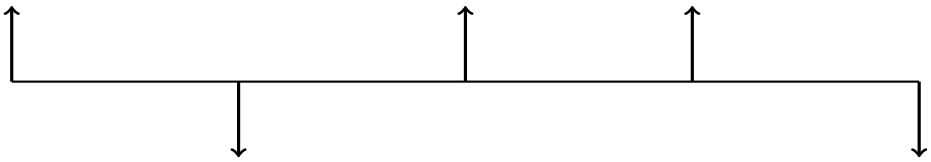


Figure 5: Dipole spin.

Figure 5 shows a possible state configuration of each point. The probability measure, on the set of all possible configurations, Ω, is referred to as a *random field*.

A sample configuration $w = \{w_0, w_1, ..., w_n\}$ is chosen where w_j belongs to a state $\varnothing_j$. The state we refer to is either up or down, where $\varnothing_j = 1$ if w_j is spinning up and $\varnothing_j = -1$ if w_j is spinning down.

A probability measure on Ω for a configuration w has energy $U(w)$.

$$U(w) = -J\sum_{i,j} \varnothing_i(w)\varnothing_j(w) - mH\sum_i \varnothing_i(w)$$

The first term refers to the energy caused by interaction of the spins for neighbouring dipoles. This term only links neighbouring dipoles as a simplifying assumption made by Ising.

J is the link potential on which $U(w)$ may either be positive (attracting) or negative (repulsive).

m and H are constants specific to the property of the material and the external magnetic field respectively.

Ising then goes on to define formulas for the probabilities of each configuration. He also derives the partition function given by equation (6).

The sequential configuration shown in Figure 5 may be generalised to n dimensions. A typical two dimensional configuration is shown in Figure 6, where '+' refers to a up state and '-' is a down state.

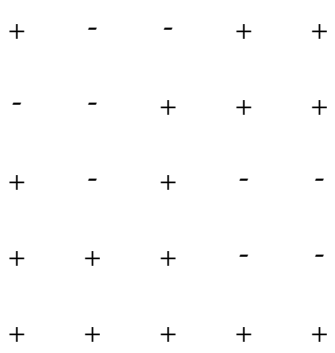


Figure 6: 2 dimensional dipole spin.

A subclass of Markov networks called a *Pairwise Markov Network* exists which is used in the approach chapter and described in section **??**. In this case the distribution of all the factors are over single or pairs of variables.

A pairwise Markov network over a graph $\mathcal{H}$ has a set of node potentials, $\{\phi(X_i) : i = 1, ..., n\}$, and a set of edge potentials, $\{\phi(X_i, X_j) : (X_i, X_j) \in \mathcal{H}\}$. Equation (6) still applies when normalising and the product of factors include both edge and node potentials.

2.5.1 *Conditional Random Fields*

The Ising model described in section 2.5 attempts to assign a probability measure to a sample space Ω without taking into consideration observable variables. *Conditional Random Fields* (CRFs) are a variation of Markov Random Fields which allow observations to be embedded.

A MRF can be used to encode a conditional distribution $P(Y|X)$ where Y is a set of target variables given X, the set of observed variables.

Referring back to Figure 4, we can add observations, X, represented by the black squares.

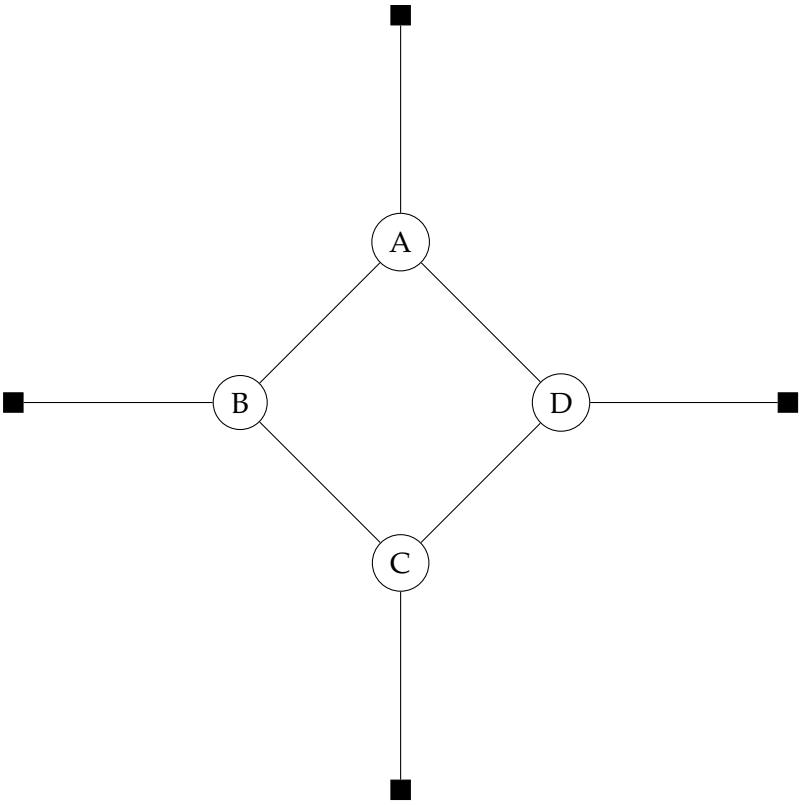


Figure 7: A conditional random field.

A CRF, $\mathcal{H}$, may be formally defined as the undirected graph which nodes are contained in $Y \cup X$. The network is annotated with the set of factors:

$$\phi(D_1), ..., \phi_m(D_m), D_i \not\subseteq X$$

The CRF encodes the conditional distribution as follows:

$$\begin{aligned} P(Y|X) &= \frac{1}{Z(X)} \widetilde{P}(Y, X) \\ \widetilde{P}(Y, X) &= \prod_{i=1}^{m} \phi_i(D_i) \\ Z(X) &= \sum_{Y} \widetilde{P}(Y, X) \end{aligned}$$

Each factor defines the edges in $\mathcal{H}$ between variables. Variables appearing together in the scope of a factor will be connected by an edge in the graph representation, the same as MRFs.

The general configuration for a CRF used in this paper is shown in Figure 7.

Belief propagation defines the process of message passing between nodes on directed non-cyclic graphs. Belief propagation may be used to create dependencies between nodes through the use of messages passed along edges in a graph.

The messages are passed along in a ordered sequence such that during each message passing round a node will receive messages from its immediate neighbours. Furthermore, a node cannot send its message along until it has received all the messages required [Koller and Friedman, 2009].

Each node takes the multiple of all incoming message and multiplies them with the corresponding potential. Then it may sum out one or more variables and sends the processed number out as a message. Belief propagation may also be referred to as the sum-product algorithm.

2.6.1 *Belief Propagation in a Cluster Graph*

Given a cluster graph.
$C = \{1, 2, 3, 4\}$

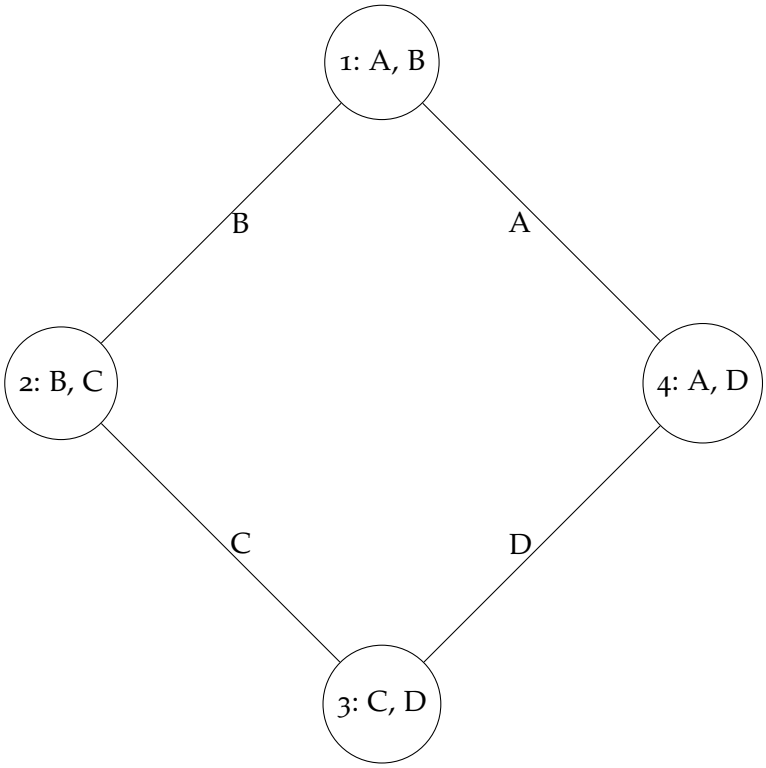


Figure 8: An undirected cluster graph.

Each cluster C_i stores a set of variables in which it holds beliefs, ψ_i. For example, cluster 1 (C_1) holds beliefs in A and B.

In an iteration, a message, $\delta_{1\to4}$, from $C_1 \to C_4$ is the message of cluster C_2 to cluster C_1, multiplied by the initial belief of cluster C_1, summed over B (since cluster 4 does not need to know about B).

$$\delta_{1\to4} = \sum_{B} \delta_{2\to1}(B)\psi_1(A,B)$$

Incoming messages to a cluster is taken from all neighbouring clusters. In undirected graphs, the incoming message cannot be affected by itself. A setpset $S_{i,j}$ is the set of variables which C_i and C_j have in common. A generalised message is given by:

$$\delta_{i\to j}(S_{i,j}) = \sum_{C_i - S_{i,j}} \psi_i \times \prod_{k\in(N_i-\{j\})} \delta_{k\to i} \tag{7}$$

In the first iteration all messages are set to 1 in the calculation of incoming messages to i ($\delta_{k\to i}$) which is a component of equation (7), given by:

$$\prod_{k\in(N_i-\{j\})} \delta_{k\to i}$$

In cases where there are no incoming messages then it is a null set and defaults to 1.

After a iteration, when every cluster has received messages from its neighbours, the cluster updates its beliefs β_i based on the messages received and its current beliefs.

$$\beta_i(C_i) = \psi_i \times \prod_{k\in N_i} \delta_{k\to i} \tag{8}$$

In the next iteration the new beliefs are passed in the messages.

Iteration is done until the graph reaches a point of convergence. Convergence and completeness is guaranteed for trees [Koller and Friedman, 2009]. In this case the algorithm begins at the leaf nodes and messages propagate to the root. Once the root receives messages it will pass messages back towards the leafs.

2.6.2 *Loopy Belief Propagation*

The previous definition for belief propagation works in directed non-cyclic graphs. A definition is required for general graphs, which is

called Loopy Belief Propagation (LBP).

LBP is simply normal belief propagation applied to cyclic graphs. In a general graph, convergence is not guaranteed and the algorithm is run until satisfaction in the context of the application. Loopy belief propagation generally provides a good approximation of the marginals, one such example is shown graphically by [Murphy et al., 1999]. In this case LBP converges and provides a good approximation.

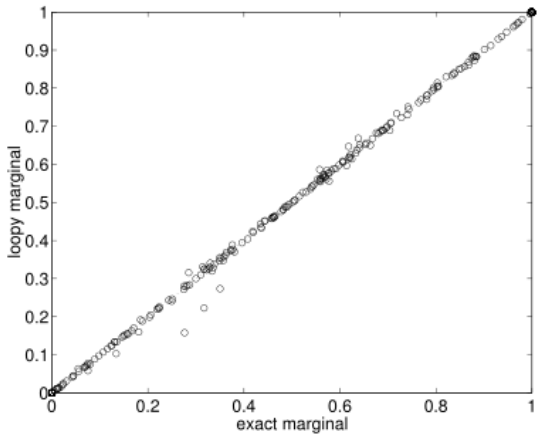


Figure 9: Graphical comparison of marginals generated by LBP against the exact marginals.

There is need to pick a good initial edge on which to pass messages. It is generally best to pick a node with the least amount of neighbours (such as a corner cell) for the first iteration. insert citation

After each iteration, the order in which messages are passed begin at the last node processed in the previous iteration.

There are several strategies for ordering. Breadth first may be considered or a scan across the graph. Scanning across can be done in image processing when LBP is used to de-noise an image. A scan passes sweeping messages across the graph from left to right, right to left, up to down, down to up in four separate iterations.

Since convergence is not guaranteed a stopping condition needs to be defined, this could be when the algorithm detects a loop where

the values of the LBP cycles or a specified number of iterations in a trade off between accuracy and processing time.

2.6.3 *Pairwise Potentials*

We can introduce conditional probabilities to each cluster. In a given graph a factor $\psi_{i\to j}$ is associated with a edge between $C_i \to C_j$. $\psi_{i\to j}$ represents the amount of influence C_i has on C_j.

Factoring in these potentials to the messages are trivial. They are multiplied along with the message passed. Modifying equation (7):

$$\delta_{i\to j}(S_{i,j}) = \psi_{i\to j} \sum_{C_i - S_{i,j}} \psi_i \times \prod_{k\in(N_i-\{j\})} \delta_{k\to i} \tag{9}$$

2.7 LEARNING PARAMETERS

The parameters for graphical models require learning to provide accurate inferencing for a set of variables. This can be done through an approach called *maximum likelihood estimation*.

The goal is to approximate a probability distribution function by creating a likelihood function which is controlled by our unknown parameters we wish to learn. A simple example is a coin flipping scenario. Assuming we do not know if the coin is rigged to land on a single side a hypothesis is formulated based on a series of coin tosses.

Coin tosses are governed by a single unknown parameter θ which is associated with the frequency of heads. Each toss is assumed to be independent from each other and without any external variables such that they are identically distributed.

We define a hypothesis space Θ, which is the set of possibilities, $\theta \in [0, 1]$. An objective function is a metric for the accuracy of each hypothesis in a dataset $\mathcal{D}$. Assuming we get $\mathcal{D} =< T, T, H, T, H, T, T >$. The probability of heads is θ and logically the probability of tails is $1-\theta$. Since we assumed independacy in coin tosses we can write the probability of a sequence as a function of θ:

$$P(< T, T, H, T, H, T, T >: \theta) = (1-\theta)(1-\theta)\theta(1-\theta)\theta(1-\theta)(1-\theta)$$

The likelihood function is equivalent to the equation for that specific data set.

$$\begin{aligned} L(\theta :< T, T, H, T, H, T, T >) &= (1-\theta)(1-\theta)\theta(1-\theta)\theta(1-\theta)(1-\theta) \\ L(\theta :< T, T, H, T, H, T, T >) &= (1-\theta)^5\theta^2 \end{aligned}$$

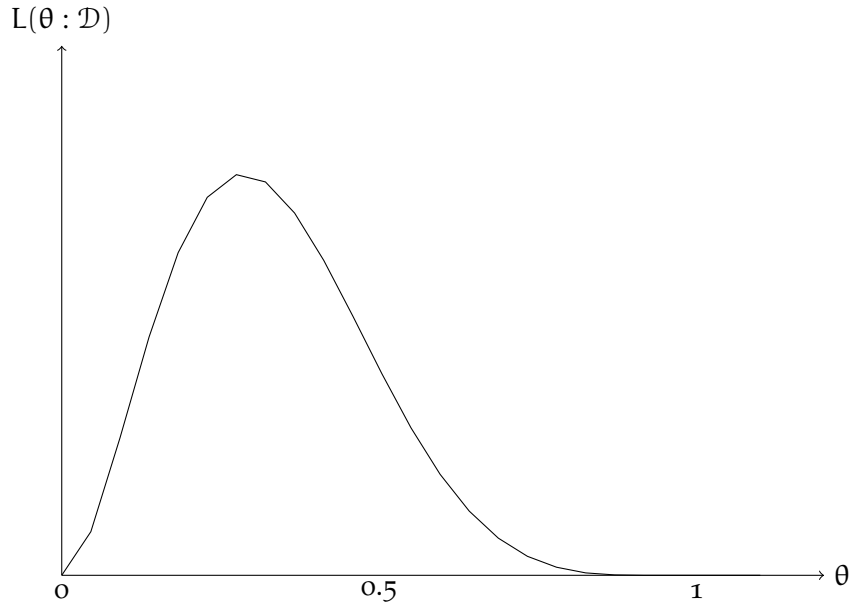


Figure 10: Likelihood function for the sequence of tosses:
$< T, T, H, T, H, T, T >$.

This likelihood function can be used as the objective function, we select θ to maximise the likelihood function to obtain the optimal parameters, this value is called the *maximum likelihood estimator*. Figure 10 shows this graphically.

A general case may be formulated, where M[1] is the number of heads and M[0] is the number of tails:

$$L(\theta : \mathcal{D}) = \theta^{M[1]}(1-\theta)^{M[0]}$$

A log-likelihood is easier to work with, which simply takes the logarithm of the likelihood function. Since the log-likelihood is monotonically related to the likelihood the value of θ at the maximum is equivalent.

$$l(\theta : \mathcal{D}) = M[1] \log \theta + M[0] \log 1 - \theta$$

To find the maximum we derive and equate to 0.

$$\hat{\theta} = \frac{M[1]}{M[1] + M[0]}$$

A parametric model is defined by a function $P(\varepsilon : \theta)$.
talk about how the likelihood function is convex

2.7.1 *Gradient descent*

In the case where we cannot easily approximate the function as a general curve, we may use a method called gradient ascent/descent by

using the assumption that the function is convex. In this paper we use gradient descent because we aim the minimise the mean squared error (section 2.10.1). Gradient descent allows us to find the local minimum of a function [Avriel, 2003].

In an conditional random field, we initialise all the pairwise potentials to their default state. From this state we calculate the gradient, $\mathcal{G}$, with respect to the current pairwise potential λ_i. $\mathcal{G}$ represents the direction in which we want to move λ_i. Thus we derive:

$$\lambda_i = \lambda_i + \alpha(\mathcal{G}) \tag{10}$$

We perform this operation each iteration. α ($\alpha \in \mathbb{R}$) is the rate of learning which we define. A lower rate of learning allows the algorithm to become more precise as the 'jump' each iteration is smaller, however the algorithm will take longer to run. We continue to adjust the value of λ_i until we reach a stopping condition. Generally this is when the current state of the model cannot be improved, for example, the mean squared error cannot be minimised further. This requires each iteration to recalculate the mean squared error after the potential has been changed.

2.8 SLAM

Robot mapping of an environment has many variants such as multi-robot mapping, incorporation of moving objects and off-line analysis among others. Robot mapping has been thoroughly explored in the literature, [Hähnel et al., 2003, Thrun et al., 2000, Wolf and Sukhatme, 2008, Rosencrantz et al., 2002].

SLAM works off recognising familiar features in a map where there is perceptual ambiguity with different features looking similar to the robot it may be difficult to obtain a high certainty during localisation. This uncertainty is factored into occupancy grids so future readings may correct an error of a pose at a given point in time instead of the error escalating as more laser scans are processed. Odometry readings are not exact and are taken into consideration when calculating values of the grid. Certainties of viewed cells in the grid are lower than if odometry was assumed to be perfect. Each cell of the grid may be computed as independent random variables. Cell values update based on external readings, in this case every laser scan cycle. If a laser scan does not provide enough certainty of the position of a robot, the odometer reading is used instead. A general cycle will take both the laser scan and the odometry into consideration.

2.8.1 *Robot Simulation*

2.8.1.1 *Ray Casting*

Ray casting is used to determine the first occupied cell encountered by a laser scan. Ray casting has been used in various applications which require an approximation of a straight line drawn in a discretised space [Schroeder, 2001, Laine and Karras, 2010, Roth, 1982]. Ray casting can be used to simulate a single laser scan at a point in time. A line is drawn from the position of the robot to its max view distance and objects encountered along the way are detected as 'collisions'.

Ray casting may be performed in n dimensions but in the case of this paper we only require two-dimensional ray casting. Standard ray casting uses a set of Cartesian coordinates to represent the source and destination as well as cells traced along the way.

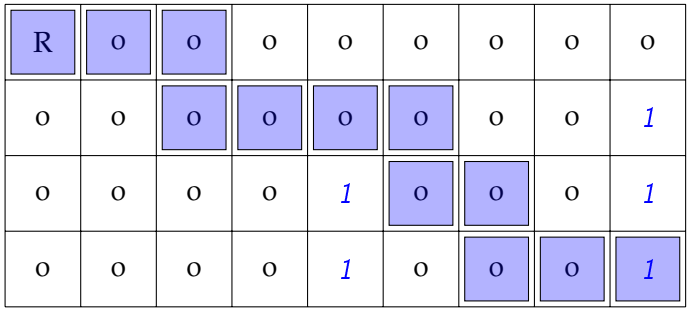


Figure 11: Sample ray cast path.

The algorithm is implemented as an iterative process whereby we jump from cell to cell from the origin, (o_x, o_y), to destination point, (d_x, d_y). The algorithm may be summarised in the following steps.

1. Check if current point (x, y) is either occupied, or our destination point. If so, then stop.

2. If $abs(d_x - o_x) > abs(d_y - o_y)$ then move x towards d_x else move y towards d_y.

We can use an error variable to 'smoothen' the line drawn as shown in Listing 1.

Listing 1: Ray casting pseudocode

```
function raycast(o_x,o_y,d_x,d_y) {
        current_x = origin_x
        current_y = origin_y
        dx = abs(x - currentx)
        dy = abs(y - currenty)
        if d_x > currentx then sx = 1 else sx = -1
        if d_y > currenty then sx = 1 else sy = -1

        error = dx - dy

        while current point is not at destination {
                temperror = error * 2
                if temperror > -dy {
                        error -= dy
                        currentx += sx
                }
                if temperror < dx {
                        error += dx
                        currenty += sy
                }
                Do something with currentx and currenty
        }
}
```

ROS provides a hardware abstraction and several libraries such as localisation algorithms that we use to perform basic SLAM and robot interaction [Quigley et al., 2009].

ROS operates on the concepts of nodes which is essentially an executable in a ROS package. These nodes communicate with each other via topics which broadcast messages to other nodes. The initial design for the simulator and CRF implementation had to be refactored to work as ROS nodes. Message passing involves continuously broadcasting a defined message type on a topic. A node can subscribe to this topic and retrieve the raw message.

ROS node architecture allowed the main functions to be separated into two distinct nodes: The robot simulator and the inference algorithms processing messages from the robot. These two nodes needed to pass a variety of messages between each other, for example the twist, pose and LaserScan messages. Each of these messages are broadcasted and received in their own topics. This required both nodes to be running multiple threads accessing shared memory.

The rate at which messages were passed between each node had to be calibrated based on the speed and laser scan rate. A rate which is too fast causes bottlenecks in the processing of incoming messages. A rate too slow would produce environment readings de-synced from real time scans. This would have an adverse effect on the accuracy of the localisation algorithm if the robot is in constant motion. A delay would mean the system would be processing readings a few seconds after the robot has already passed a given point and thus will not know if the robot was running into an obstacle.

The optimal rate also needs to take into account the max range of the laser scans. A larger range means the processing can afford to be slower. This frees up cycles to perform more complicated operations on the conditional random field. The rate had to be synced between all the various topics between the nodes. Certain processes were dependent on the execution of a previous block of instructions. Each process had a varying execution speed dependent on the complexity of the environment and uncertainty of the system on the robot's current position.

Integration with a real robot in a real environment is trivial in this setup. The robot simulator node would be replaced by the real robot without any major changes in existing code on the ROS side.

There are four terms which we use to refer to results returned by a classification algorithm, true positives, false positives, false negatives and true negatives. True positives are a correct result, false positives are an unexpected result, false negatives are results which are missing and true negatives are the correct absence of a result.

2.10.1 *Mean Squared Error*

The mean squared error (MSE) is a metric allowing comparison between inferencing algorithms. The MSE quantifies the different between the values obtained by an estimation produced by the inferencing algorithm to the expect value [Everitt and Skrondal, 2002].

We take the square of the difference between the estimated value to the ground truth in the case of a occupancy grid. We define the formula to obtain the MSE as follows, where $\hat{Y}$ is a vector of estimated values and Y is the vector of true values:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\hat{Y}_i - Y_i)^2 \tag{11}$$

The MSE is especially useful in the case of this paper as it allows an easy way to compare the accuracy of one occupancy grid generated to another.

2.10.2 *Precision and Recall*

We can use precision and recall as a metric for an algorithm performing classification on a data set. [Manning et al., 2008]. We define a set of objects $\{x_1, ..., x_n\}$ in our data set $\mathcal{D}$. Each object belongs to one more sets of predefined classes $c_n, c_n \in \mathcal{C}$. The *recall* is the number of objects the classification algorithm detects for a given class c_n where x_n actually belongs in c_n, the number of true positives, divided by the total number of objects belonging to c_n.

$$\text{Recall} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}} \tag{12}$$

Precision is the true positives divided by the total number of objects returned in the classification algorithm.

$$\text{Precision} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}} \tag{13}$$

We define the metric of *accuracy* to be the sum of all correct results divided by the total of all results.

$$\text{Accuracy} = \frac{\text{true positives} + \text{true negatives}}{\text{tp} + \text{tn} + \text{fp} + \text{fn}} \tag{14}$$

2.10.3 *ROC curves*

Receiver Operating Characteristic (ROC curves) allow us a way to visualise the performance of a classifier graphically [Tan et al., 2007]. We graph the true positive rate against the false positive rate.

The false positive rate is indicated on the x axis and the true positive (*sensitivity*) on the y axis. The graph depicts the trade-offs between sensitivity, which we vary. The sensitivity refers to the *recall* shown in equation (12). The manner in which we vary the sensitivity is dependent on the implementation of the classifier. The false positive rate may be calculated by:

$$\text{False Positive Rate} = \frac{\text{false positives}}{\text{false positives} + \text{true negatives}} \tag{15}$$

A perfect classification algorithm would be a single point with the maximum rate of true positives (1) and 0 false positives. A typical ROS curve is shown in figure 12.

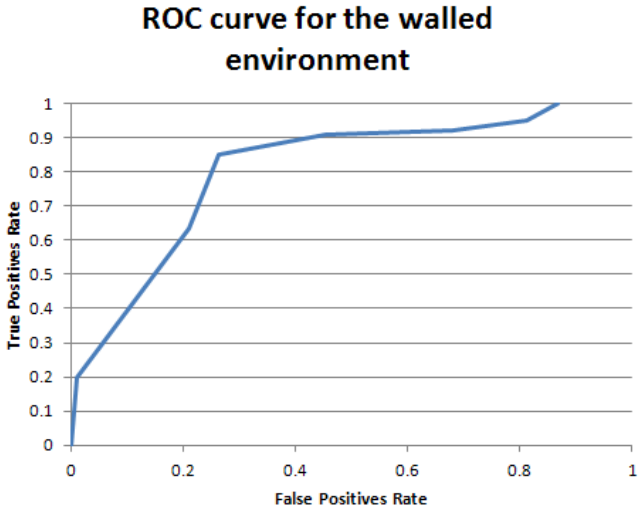


Figure 12: Example of a ROC curve taken from the results section of this paper.

2.10.4 *Cross Validation*

Cross validation is used to measure the performance of a classifier on an independent data set. The goal is to provide an estimate on the accuracy of the classification method after being trained on a similar data set.

In general, we partition a data set in subsets. One of which we refer to as the *training set*, the set which we use to train the classifier. The training set is used in validating the analysis of the other sets, referred to as the *validation set*.This allows us to train a classifier using all the data we have and test it on the same data. Furthermore a value in the validation set is not trained on itself (type-III errors [Mosteller, 2006]), which would be pointless and possibly lead to over fitting.

We can then calculate the mean square error, for example, to get a metric on its performance. This can be an iterative process, where the model adjusts itself after each validation changing its parameters such that it matches the training set more closely (gradient descent). The model may fit too well, which is referred to as *over fitting* which can be a problem when trying to classify unknown data sets which may only partially match the characteristics of the training set.

In the case of binary variables we simply take the more likely state, for example, a cell may have a 0.6 chance of being occupied. We label the cell as occupied and compare it to the ground truth to classify the result as either a true positive or false negative.

There exists variations of basic cross validation:

k-fold cross validation
: The data set is partition into k equal size subsets. A single subset is used as the validation set while the other subsets are the training set. A different validation is then picked and similarly for the training sets until all sets are iterated over. The results from each fold may then be averaged for a single estimation.

leave-one-out cross validation
: This is a specific instance of k-fold where k is equal to the number of values in the data set.

3

APPROACH

3.1 ROBOT SIMULATION

A robot simulation was constructed from scratch to facilitate the maximum amount of flexibility in both the setup and subsequent inferencing stages. Robot simulators were considered such as ROS's Gazebo package, however accessing ROS package data structures directly (in memory) would be troublesome and modification even more so.

The robot simulation initially builds a grid which represents the ground truth state. This is the state which we can use as a comparison to obtain an accuracy metric for inference algorithms. A completely accurate ground truth cannot be guaranteed for a real life environment. A real environment would require the ground truth to be built before experimentation, which can either be done manually (unviable) or by sending a robot to map the environment beforehand, neither of which can be guaranteed to be completely accurate.

The ground truth map refers to an occupancy grid, $O(x)$ with state variables for each cell $s(C)$ only belong in a state of absolute occupancy or absolute emptiness. $s(C) \in \{0, 1\}$.

The simulator initialises a ground truth robot position on the grid, again this implies absolute certainty of the robot position. This is different in the localisation process in ROS where a robot position is represented in a particle field [Zaman et al., 2011].

Thus, two separate variables are tracked by the simulation. The position of the robot and the occupancy of each cell in the grid. Both these variables will have uncertainty factored in during simulation.

ROS has a set of predefined messages to be passed between its packages. The robot simulation utilises a *LaserScan* message to encode all the information about laser scans. Odometry information and instructions to the robot is encoded in *Odometry* and *Twist* messages respectively.

0	0	1	0	1	0	1	0	1
0	1	1	0	1	0	1	1	0
1	0	1	1	0	1	1	0	1
1	1	1	1	1	1	0	1	1
0	0	1	1	1	1	1	0	0
1	1	0	1	1	1	0	1	1
1	0	1	1	0	1	1	0	1
0	1	1	0	1	0	1	1	0
1	0	1	0	1	0	1	0	1

Figure 13: Example of a ground truth occupancy grid.

3.1.1 *Laser Scans*

The laser scan message has a structure as follows:

Listing 2: ROS's LaserScan message structure

```
1
Header header            # timestamp in the header is the
    acquisition time of

float32 angle_min        # start angle of the scan [rad]
float32 angle_max        # end angle of the scan [rad]
6 float32 angle_increment  # angular distance between measurements
    [rad]

float32 time_increment   # time between measurements [seconds]
float32 scan_time        # time between scans [seconds]

11 float32 range_min        # minimum range value [m]
float32 range_max        # maximum range value [m]

float32[] ranges         # range data [m] (Note: values < range_
    min or > range_max should be discarded)
float32[] intensities    # intensity data
```

The simulator bases its scanning algorithm off what is needed in the laser scan message. The sensor simulator scans through a set of angles A in the given range. Each scan runs a ray casting (section 2.8.1.1) algorithm to the points on the edge of circle as shown in Fig-

ure 14. The robot's position is marked as R on the grid. The robot has a limited view angle and view distance, it can change its orientation to view around it without moving to a new set of coordinates.

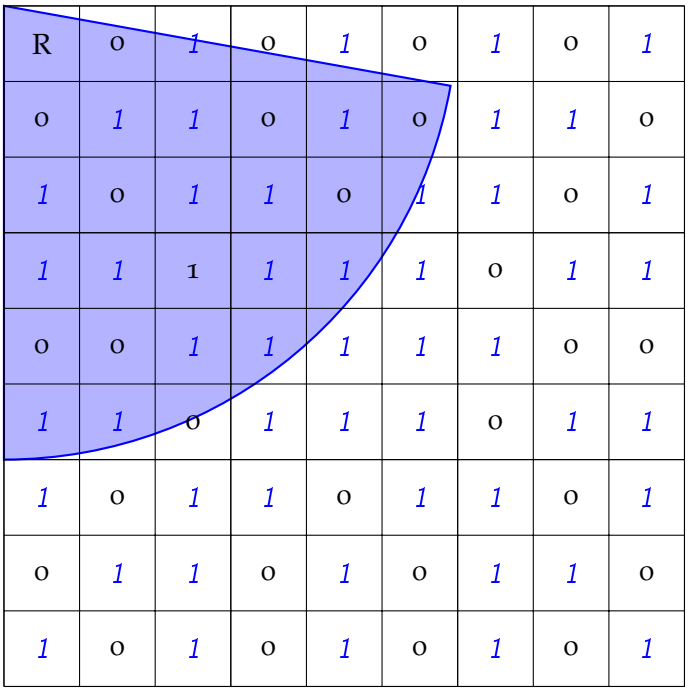


Figure 14: Laser scan circle.

The ray casting algorithm for an angle a_x will determine the distance at which an object is 'hit' if any. The distance, d, at which an object is detected is recorded for a_x, a distance of -1 is set in cases where the scan range is empty.

$$a_x \mapsto d$$

Ray casting does cause several problems since it's impossible to draw a perfectly straight line in a grid. For example Figure 15 shows a case where ray casting "fails" by producing a false positive return for a blocked path. In these cases the ray casting algorithm can go either way (x or y axes) when moving towards the target point and blockage is dependent on which axes it chooses. In the figure, ray casting stops in the first grid when it detects a blocked point because the robot cannot see beyond an occupied cell.

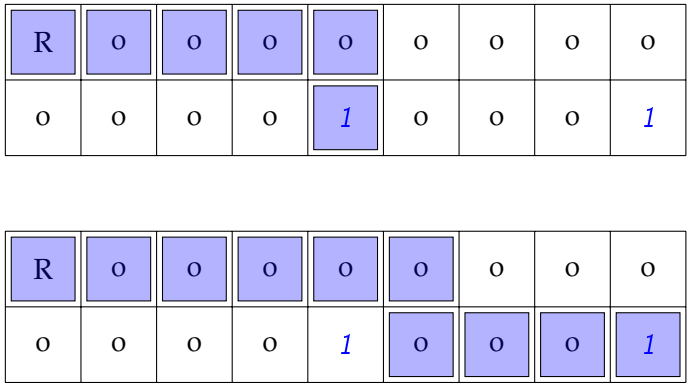


Figure 15: Ray cast comparison.

This problem occurs infrequently and since we factor noise into sensor readings regardless it can largely be ignored. Furthermore, a 'bad' scan at one angle may be correct in the next angular increment in the same scan set. Erroneous readings for a coordinate space is handled appropriately in the discretisation algorithm described in section 3.2. Though for a perfect real world simulation this method is inadvisable.

Noise for scans is factored into the distances using $f(d) = d \pm C$, where C is a randomly generated value between specified bounds $[0, \infty)$ representing the accuracy of the sensor itself.

An unavoidable source of noise (though not necessarily bad for real world simulation) occurs in the rounding of occupied cells. This occurs because real space is not discretised and when tracing out a circle such as the one in Figure 14 it is impossible to draw a continuous curve. This causes problems in the discretisation process in section 3.2 because information is essentially lost during construction of a *LaserScan* message due to having to round to the nearest cell to approximate the general area.

3.1.2 *Odometry*

The odometry and twist messages are defined as:

Listing 3: ROS's Odometry and Twist message structure.

```
Odometry:
Header header
string child_frame_id
geometry_msgs/PoseWithCovariance pose
geometry_msgs/TwistWithCovariance twist

Twist:
geometry_msgs/Vector3 linear
geometry_msgs/Vector3 angular
```

ROS sends the robot twist messages to instruct it to move. The odometry messages are the readings from the robot's onboard odometry sensors. Noise is factored in at each interval of movement, for each twist message the robot follows the same formula as the one for scan distances ($f(d) = d \pm C$). Figure 16 shows the comparison of a robot moving with different ranges for the value of C.

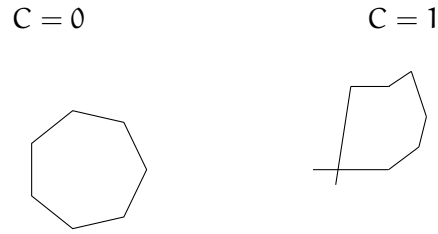


Figure 16: Comparison of C = 0 and C = 1 as parameters for robot movement simulation.

3.1.3 *Robot Simulator Pseudo code*

Listing 4: Robot Simulator Psuedocode

```
main {
circle = drawCircle(robot.x, robot.y)
foreach (point in circle) {
        scanSet += rayCast(point)
}

convertToLaserScan(scanSet)
send laserScan
}
```

3.1.4 *Related Works*

[Tarnoff et al., 1992] presents a way to do Off-Line programming centered around complex industrial robot systems. The paper describes the method of establishing a geometric and kinematic world model calibrated to high accuracy. This paper does not handle kinematic motion though it should be considered as a potential feature to utilise in certain robots.

Calibration of sensors is done by compact ranging sensors and *laser triangulation sensors* [Dietrich et al., 1990]. The system uses a ultrasound ranging sensor which records the time it takes for the wave to hit hard surface and return. The robot simulation described in this pa-

per is simplistic and does not do any calibration that could improve accuracy of the simulation.

Occupancy grid construction is based off the *LaserScan* message described by Listing 2 in section 3.1. This method is used in combination with ROS's built in localisation packages. The *gmapping* package is used purely to generate robot poses which is then fed into occupancy grid constructed. Extracting the occupancy grid used in working memory of ROS packages such as gmapping is difficult and thus it is easier to construct an occupancy grid on which to do inference.

The discretisation of laser scans is the reverse process of the construction of the laser scans into the *LaserScan* message in section 3.1.1. Every angle described in the message is used to obtain a set of (x, y) coordinates. The coordinates are only approximate and cannot be guaranteed to be the exact value of the ground truth, even without any noise. This issue was mention in the construction of the laser scans. Since information about the exact position of occupancy is lost when scans are converted in *LaserScan* messages the algorithm can only approximate the exact coordinate space which the scan refers to.

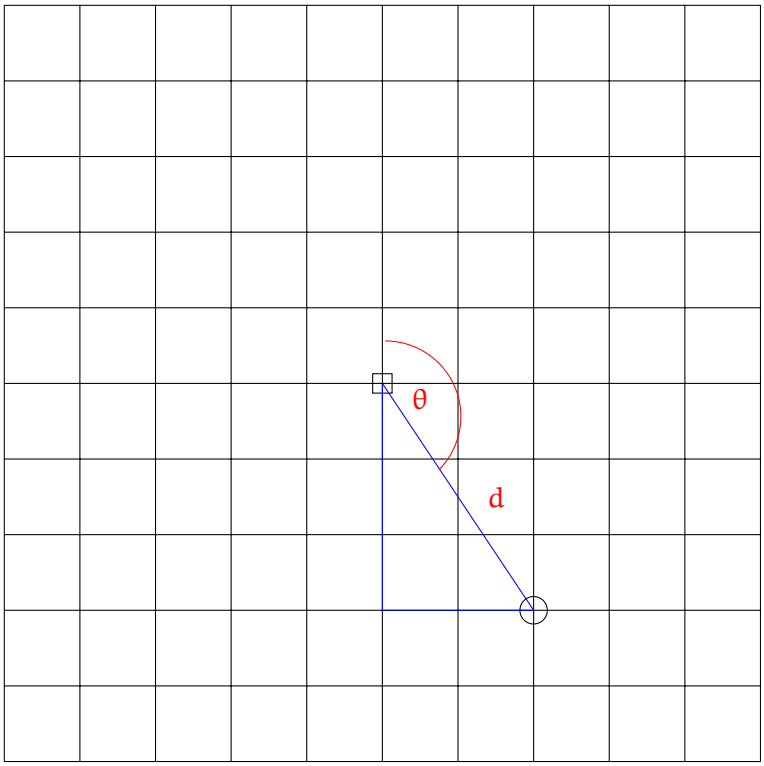


Figure 17: Discretisation example, the square represents the robot and the circle represents an object detected through laser scans.

In Figure 17 we see that basic trigonometry may be used to retrieve x and y coordinates when given an angle and hypotenuse equivalent to the distance, d, passed in the scan message.

In a general case the (x, y) coordinates are not integers and must be rounded. The way we choose to do the rounding will effect the cell which will be updated on the occupancy grid. Regardless of the method chosen, information is lost and the discretisation process cannot be guaranteed to reconstruct the exact ground truth even without noise.

The rounding problem becomes apparent when pairing it with the robot simulation. The simulation works by drawing a circle around the robot to calculate its maximum view distance. Since its only possible to approximate a circle in a grid there is a loss of information due to rounding. This loss of information carries over in the emphLaserScan message.

Once the (x, y) coordinates are processed a ray casting algorithm is run to that point. All cells which are in the line will be recorded as empty, while the last point is recorded as occupied (unless $d = -1$ indicating everything in the range is empty).

There may be erroneous readings for a cell. That is, a cell may be recorded as both occupied and empty by different scans in the same set. Furthermore, some scans may overlap and confirm the readings for a cell. A mechanism is needed on which the certainty of the value of the cell is dependent on the number of scans 'hitting' the space. The following function was introduced to obtain a general probability for the occupied state, where x is the number of occupied hits minus the number of empty hits.

$$Pr[Cell = Occupied] = f(x) = 0.5 + \frac{1}{\pi}\arctan(x) \tag{16}$$

It is notable that the probability will never reach 1 or 0, indicating absolute certainty, however it will approach them.
The probability obtained from the distribution is the ultimate value of the associated cell on the occupancy grid.

3.2.1 *Related Works*

There have been various attempts in the robotics field to improve this uncertainty in various stages of both the formulation of the occupancy grid and the localisation process.

Dusty or angled surfaces can skew sensory information. A improved mapping method is devised by [Nagla et al., 2012] to reduce the effect of specular reflection. Specular reflection refers to a wave hitting a surface at an acute angle without the wave bouncing back

back to the transducer [Murphy, 2000].

The method involves using a fuzzy logic algorithm to filter specular information. Knowledge of the type of environment and the surfaces prevalent in the environment can be used to estimate the probability of a given reading being specular.

Specular reflection can be handled by using two sets of parameters learned before hand, a range confidence factor and an orientation probability [Lim and Cho, 1992]. A MURIEL algorithm was introduced which attempts to calculate the probability of a cell being specular at different robot poses. Data is gathered incrementally and applied for each cell space. The approach showed a 6.6% improvement and a 16.5% reduction in computation time.

A different approach to what is detailed in this paper to mapping using occupancy grids and inference on the grids involves gathering all sensory data prior to map formation, referred to as the forward approach [Thrun, 2003]. [Collins, 2011] introduces a *ConForm* approach which expands on this. ConForm analyses sensory readings in the context of the ground truth map and each cell in the context of the reading set. These readings are filtered and passed to a mapping layer to go through a forward model which generates the occupancy grid. Every individual scan in a sensor reading gets evaluated against two sets of variables, the set of scans in the sensor reading (Acceptability / Agree-ability) and trait verification which processes the cell in the current perceived state of the environment.

The Agreeability algorithm constructs dependencies between each scan. A 'profile' (processed scan set) is evaluated from multiple inputs obtained from a subset of the scanned space. Scans which infer the same hypothosis are accepted whilst scans which disagree are flagged for erroneous readings.

The ConForm approach achieves a similar goal to the one in this paper (though as an offline operation), it seeks to address various issues of occupancy grids one of which is the assumption of independence of cells. Larger grids require a sliding window approach to building the CRF [Collins, 2011].

[Liu and von Wichert, 2013] provides more methods in analysis of an occupancy grid after construction. Semantic mappings are established through Bayesian reasoning. The goal is to assign labels to objects in the environment such as rooms or corridors. Relations such as rectangularity of rooms can be established and used in the reasoning to establish a probabilistic model. Techniques from both

'place labeling' [Wolf and Sukhatme, 2008, Pronobis et al., 2010, Martinez Mozos et al., 2007] and 'object labeling' [Limketkai et al., 2005, Douillard et al., 2008] literature is taken to build a probabilistic generative model of the environment.

The Conditional Random Field is the data structure on which we base inferencing algorithms off, which in itself is based off the structure of the occupancy grid constructed in section 3.2.

Imprinting the probabilities of real world readings onto the lattice is achieved by a link potential. The potential may be learnt through a variety of methods and will be optimised depending on the environment. The potential adjusts the accuracy of the sensor readings, where some environments may be dusty or reflective which will affect the accuracy of sensor readings.

3.3.1 *Related Works*

[Quattoni et al., 2007] uses hidden CRFs to capture intermediate structures using hidden-state variables. This produces a model where inference and parameter estimation is done by basic graphical modeling algorithms. A hidden CRF uses intermediate hidden variables conditioned on observations and with a set of factors describing their relation to each other. These hidden variables model the latent structure of the input domain.

CRFs have been shown to be superior to Markov Random Fields in detecting man-made structures in images [Kumar and Hebert, 2003]. A CRF allows relaxation of the assumption of conditional independence between observations in the field of computer vision [Quattoni et al., 2004].

Related problems may be correlated, for example, in this paper we model cell occupancy which may be correlated with a heat map in an office environment. We can model two related problems by using a triangular-chain CRF [Jeong and Geunbae Lee, 2008]. Triangular-chain CRFs encode both the sequence and meta-sequence labels in a single model.

Observations over time, in this paper, are assumed to be independent from each other to maintain tractability. [Stewart et al., 2008] provides a way for discriminative labeling of structured data so that they may be used to classify higher order structures in the labels. A random field of parameterised features is used to model the labels which are functions based off groupings of the observations.

A method for learning the potential is the search and score approach [De Santana et al., 2007].
A graph is built node by node with a score recorded at each stage. An example of the implementation of the method is described based on

genetic algorithms and fuzzy logic to induce Bayesian networks. The method uses fuzzy systems with genetic algorithms. A scoring system is calculated through the fuzzy system and the genetic algorithm allows exploration through possible structures.

3.4.1 *Application to the Conditional Random Field*

The cluster graph discussed in section 2.6 can be translated into the CRF that is required for inference on our occupancy grid. As defined previously in the section 2.3 on occupancy grids, each cell on the occupancy grid has an associated state variable
$s(C) \in \{occupied, empty\}$ [Elfes, 1989].

All clusters hold beliefs on whether the spatial coordinate they represent are occupied and empty. Each cluster has an associated joint distribution it uses to update its beliefs conditioned on its neighbor's beliefs and its current beliefs.

Clusters are the individual cells on the occupancy grid.

Each cluster holds a belief:

$$\psi_i = \{P[s(C) = Occupied], P[s(C) = empty]\} \tag{17}$$

ψ_i holds the probability of occupancy and emptiness for a given cell.

Similar to occupancy grids:

$$P[s(C) = Occupied] + P[s(C) = empty] = 1$$

Clusters are linked via undirected edges and are arranged in the same lattice structure as an occupancy grid. These clusters will be referred to as the hidden layer.

We further add clusters $O_{(x,y)}$ into the graph. $O_{(x,y)}$ is the *observed* probability for discretised space associated with coordinates (x, y) which we obtain post-localisation of the robot. These clusters are layered on top of our existing lattice. These clusters have their own pairwise link potentials. They can be viewed as having fixed beliefs, that is they do not change their beliefs based on incoming messages, however they will still send out messages.

The hidden layer encodes the relationship between each cell such that neighbouring cells will broadcast its beliefs of occupancy at each message passing iteration. All clusters in the set of O encode the observations onto the grid, this relation allows adjustments based on the accuracy of the localisation process in a given environment.

3.4.2 *An Example*

Modifying the graph in Figure 8 to match the structure of the conditional random field produces the following graph.

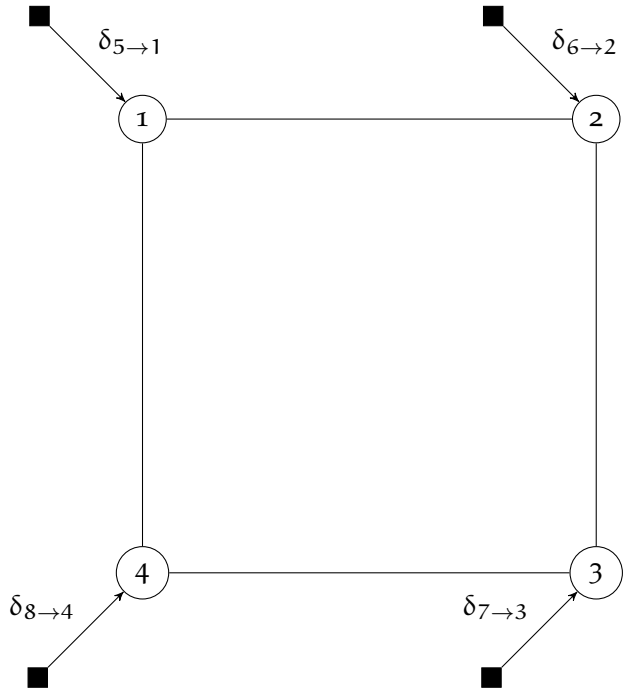


Figure 18: The structure of a 2x2 occupancy grid mapped to a conditional random field.

Node	Observed Occupancy	Observed Emptiness
5	0.6	0.4
6	0.8	0.2
7	0.3	0.7
8	0.4	0.6

Table 3: Observations as their own nodes embedded onto the hidden layer.

In figure 18, the *circles* $C(n)$ are the hidden nodes and as a group is referred to as the hidden layer.

Rectangles $R(n)$ represent the observations projected onto the graph, the observations are for the occupied belief. The labels indicate their real world equivalent mapping. Due to the fact the cell states are exclusive and exhaustive we can infer the probability it is empty $(1 - P[S(n) = \mathtt{occupied}])$.

Rectangular nodes contain fixed beliefs and are therefore one directional since the only node in which it sends messages to is the associated coordinate cell on the hidden layer. Messages to a node cannot be influenced by the receiver of that message and since no other nodes are connected to rectangular nodes the logic of the edge in the graph can be simplified to flow in one direction.

Each node will have an associated belief $\psi_i(n)$ on occupancy, $P[s(n) = Occupied]$, and emptiness, $P[s(n) = empty]$, of the node's associated coordinate space after each iteration of LBP. The initial belief of nodes in $C(n)$ are constant and equal for both occupied and empty states. This means that we do not need to consider their current beliefs in the calculation of messages.

The set of $\{1, 0\}$ infers absolute belief that the cell is occupied. A set of $\{0.5, 0.5\}$ is an unknown cell and is the default belief of every node in the hidden layer.

In a simplified case for pairwise potentials we can define two sets which can be kept constant during processing.

	N^1	N^0
C_1	0.9	0.1
C_0	0.1	0.9

Table 4: Joint distribution from cell to neighbouring cell, hidden pairwise potential.

	N^1	N^0
C_1	1	0
C_0	0	1

Table 5: Joint distribution from observation to cell, link pairwise potential.

In a generalised case, each edge in the graph will have its own pairwise potential which may or may not be the same as other potentials.

The first iteration is initialised at C_1 representing (0, 0).

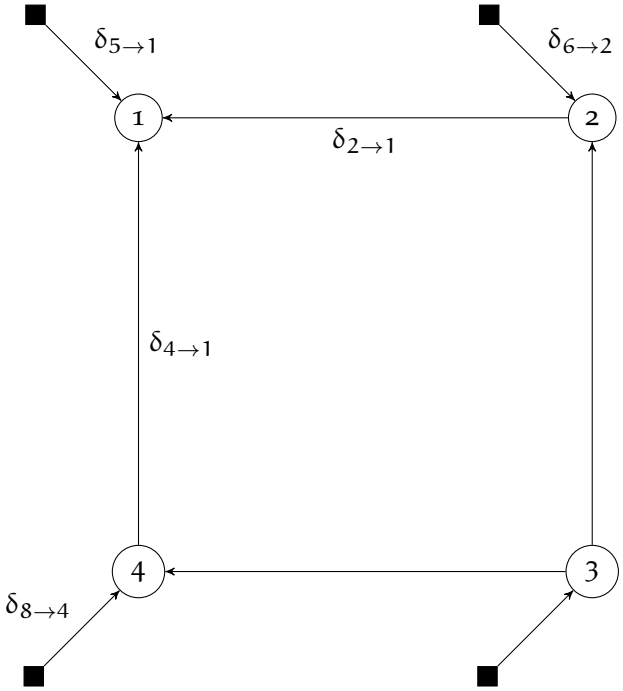


Figure 19: Message flow in the first iteration of LBP.

The initial messages incoming to C_1 are:

$$\begin{aligned}
\delta_{2\to1} &= (\psi_{C\to C}) \times (\delta_{3\to2}) \times (\delta_{6\to2}) \\
\delta_{4\to1} &= (\psi_{C\to C}) \times (\delta_{3\to4}) \times (\delta_{8\to4}) \\
\delta_{5\to1} &= \psi_{R\to C} \times \psi_5 \\
\delta_{6\to2} &= \psi_{R\to C} \times \psi_6 \\
\delta_{8\to4} &= \psi_{R\to C} \times \psi_8
\end{aligned}$$

Since we do not know all the incoming messages of other hidden nodes the value of those messages are set to 1. The messages from rectangular nodes can be calculated immediately. We sum across the two states using the joint distribution defined in table 4 to obtain the unnormalised probabilities for occupied and empty. For example, to calculate $\delta_{2\to1}$:

$$\begin{aligned}
\delta_{2\to1}(O) &= (\psi_{C\to C}) \times (1) \times (1 \times 0.8) \\
\delta_{2\to1}(E) &= (\psi_{C\to C}) \times (1) \times (1 \times 0.2)
\end{aligned}$$

We sum across the rows to obtain:

$$\begin{aligned}
\delta_{2\to1}(O) &= (0.9 \times 0.8) + (0.1 \times 0.2) = 0.74 \\
\delta_{2\to1}(E) &= (0.1 \times 0.8) + (0.9 \times 0.2) = 0.26
\end{aligned}$$

	N^1	N^0
C_1	0.9×0.8	0.1×0.2
C_0	0.1×0.8	0.9×0.2

Table 6: Using the joint distribution to calculate probabilities of states.

The same process is repeated $\delta_{4\to1}$ (calculating link potentials is trivial in this case because we simply multiply by 1), evaluating for the variable representing the occupied state gives:

$$\begin{aligned}\delta_{2\to1}(O) &= 0.74\\ \delta_{4\to1}(O) &= 0.42\\ \delta_{5\to1}(O) &= (1 \times 0.6) = 0.6\end{aligned}$$

Similarly, we can obtain the messages inferring whether a cell is empty.

$$\begin{aligned}\delta_{2\to1}(E) &= 0.24\\ \delta_{4\to1}(E) &= 0.58\\ \delta_{5\to1}(E) &= (1 \times 0.4) = 0.4\end{aligned}$$

C_1 updates its beliefs based on equation (8).

$$\begin{aligned}B_1(P(S(n) = \text{occupied}) &= 0.74 \times 0.42 \times 0.6 = 0.18648\\ B_1(P(S(n) = \text{empty}) &= 0.24 \times 0.58 \times 0.4 = 0.05568\end{aligned}$$

Normalising the beliefs so they add to 1 gives:

$$\begin{aligned}B_1(P(S(n) = \text{occupied}) &\approx 0.77\\ B_1(P(S(n) = \text{empty}) &\approx 0.23\end{aligned}$$

The messages incoming to C_4 are:

$$\begin{aligned}\delta_{1\to4} &= (\psi_{C\to C}) \times (\delta_{2\to1}) \times (\delta_{5\to1})\\ \delta_{3\to4} &= (\psi_{C\to C}) \times (\delta_{2\to3}) \times (\delta_{7\to3})\\ \delta_{8\to4} &= (\psi_{R\to C} \times \psi_8)\end{aligned}$$

We follow the same process as before, except now we reuse the evaluated value for the message $\delta_{2\to1}$. Evaluating for the variable representing the occupied state gives:

$$\begin{aligned}\delta_{1\to4}(O) &= (\psi_{C\to C}) \times (0.74) \times (1 \times 0.6)\\ \delta_{3\to4}(O) &= (\psi_{C\to C}) \times (1) \times (1 \times 0.3)\\ \delta_{8\to4}(O) &= (1 \times 0.4) = 0.4\end{aligned}$$

Similarly to C_1, the normalised beliefs for C_4 are:

$$B_4(P(S(n) = \mathrm{occupied}) \approx 0.505$$
$$B_4(P(S(n) = \mathrm{empty}) \approx 0.495$$

Message passing continues in this fashion until all edges in the hidden layer are processed once. Messages are persistent and are reused in the next iteration until updated. A stopping condition is defined as either a set upper bound count for iterations or until convergence is detected depending on which condition is reached first.

3.4.3 *Loopy Belief Propagation Pseudocode*

Listing 5: Loopy Belief Propagation

```
main {
        stack nodes_to_process
        list processed_nodes

        while (iteration) {
                nodes_to_process.push(corner node)
                //breadth first traversal
                while (nodes_to_process is not empty) {
                        currentnode = stack.pop()
                        processed_nodes += currentnode

                        nodes_to_process += (Neighbours(
                            currentnode) - processed_nodes) -
                            fixed_value_nodes

                        update_belief(currentnode)
                }
                processed_nodes.clear()
                nodes_to_process.push(currentnode)
        }
}

function update_belief(currentnode) {
        messagesum = currentnode.belief

        for each n in Neighbour(currentnode) {
                for each n_n in Neighbour(n)
                        n_messages *=
                                lookup_message(n_n, neighbour)

                current_message = n_messages
                        * n.belief
                        *  pairwise_potential(currentnode, n)

                messagesum += current_message
                global_message_list += current_message

        }

        currentnode.belief = normalise(messagesum)

}
```

3.4.4 *Optimisation*

In larger graphs, running LBP may be very costly and not viable for real time applications. In the case of this paper, if an occupancy grid were to have finer resolution then LBP requires some modifications for run time.

One method implemented in this paper is to restrict message propagation to within dynamic bounds. Message passing is limited to within a certain distance of the robot.

Nodes beyond the defined distance are not processed and message passing does not continue. When the robot moves the bounds move with it, this can be visualised as a circle around the robot.

This technique sacrifices accuracy in the case of very large objects which cover a large portion of the grid, however in other cases performance improves noticeably. It also provides scalability for the algorithm when mapping physically large environments requiring large grids.

3.4.5 *Related Works*

[Kao, 2005] discusses the various techniques on which marginalisation of can be achieved and links belief propagation to free energy approximation and parity-check codes. Once belief propagation has been performed on a graph [Braunstein et al., 2005] provides a method in which detailed probabilistic information obtained from observations are used to fix variables which simplifies the marginalisation process.

This paper uses discrete variables in the representation of cell states, continuous variables may also be used to represent the cell state. In this setup a variable is defined to be in a range of occupancy rather than belonging to two discrete states. [Sudderth et al., 2010] presents methods in which it is possible use a modified belief propagation algorithm on sets of continuous variables, demonstrating its applications in distributed localisation in sensor networks.

Reparameterisation can be done on a subgraph in the form of a hypertree of the initial graph on which belief propagation is performed. The goal is to provide an alternative factorisation using functions that represent the marginal distributions. This allows a comparison between the exact marginals and the marginals approximated by loopy belief propagation on a cyclic graph [Wainwright et al., 2003].

There are two sets of pairwise potentials which require to be optimised before inferencing on an environment. The link potentials and the hidden potentials. We use gradient descent (section 2.7.1) to approximate the optimum configuration.

The hidden potential is the joint distribution on which we define relationships between neighbouring cells and its usage is shown in the section on loopy belief propagation (section 2.6).

	N^1	N^0
C_1	oo	oe
C_0	eo	ee

Table 7: Joint distribution cell state combinations

Table 7 shows us the possible combinations we require a default weighting for. As described in the background section on gradient descent, we initialise the joint distribution to default weights.

	N^1	N^0
C_1	0.9	0.1
C_0	0.1	0.9

Table 8: Default weights

The values are then adjusted in the direction of the gradient in increments proportional to a defined variable α. For example, we seek to optimise the occupied-occupied potential. We define $\alpha = 0.01$ and add α to 0.9. If we determine the gradient to be in the opposite direction we would add $-\alpha$ instead.

	N^1	N^0
C_1	0.91	0.1
C_0	0.1	0.9

Table 9: Adjusted Weighting

Now we perform loopy belief propagation using this new joint distribution. After which the mean squared error (MSE) is calculated to get a metric on which we can base the performance of this configuration for the pairwise potentials. This process is repeated until a *stopping condition*.

Suppose we get a MSE for the configuration in table 9 higher then the previously calculated MSE which uses the configuration in table

8, in this case we know that with our current value of α this is the closest approximation to the optimal weight for occupied-occupied we can obtain. The current options are to either settle for the current value or to decrease the value of α so that we may obtain a more accurate result.

Each iteration also adjusts the weightings for the other combinations of states for cells. Furthermore, the same process is performed for the link potentials in the same iteration.

3.5.1 *Local minimum assumption*

A particular issue involving the nature of using loopy belief propagation seems to affect the accuracy of the optimal parameters found. Loopy belief propagation only provides an approximation and as such gradient ascent cannot be guaranteed to find the optimal parameters.

This paper makes the assumption that the likelihood function is convex and hence any local minimums are a global minimum, this is not true in all cases though gradient descent does find acceptable approximations (we see in the results section that the accuracy of the occupancy grid constructed from scans is guaranteed to improve using gradient descent).

Several methods were used to provide an understandable representation for the mapped environment intuitive for humans. A basic command line printout of the exact representation of numbers was used to fine-tuning during parameter learning (section 3.5).

fig:textprintout

The ROS packaged *rviz* was used in conjunction with ROS's built in localisation packages to provide a detailed real time representation of the robot moving and sensor sweeps. This required integration with the existing modules (robot simulation and inference algorithms) written separate to ROS.

The *gmapping* package which used particle filters to generate robot poses needed to have it's active memory extracted to get the grid containing probabilities of occupancy which were then processed by the inference algorithms.

fig:rivz

Rviz did not provide easy methods on which to display specific probabilities, for example, varying the transparency of a colored cell to indicate certainty. The third form of representation is a basic png image which showed a snapshot of the occupancy grid. The snapshot is captured through a filed dumped from ROS and processed in a python script to generate the png file.

3.6.1 *Related Works*

There exists many extensions to the basic environment visualisation, represented in this paper, in the literature [Adams, 2005, Almirao et al., 2007, Goodrich and Schultz, 2007]. A cognitive task analysis assists with human understanding of domain and robot appropriate tasks [Humphrey, 2009].

4 CONCLUSION

4.1 RESULTS

The goal of this paper was to provide a method of inference to improve the accuracy of occupancy grids. As discussed previously, there are specific limitations to occupancy grids stemming from its assumption of independency of cells. Two fundamental problems may be solved through inferencing on the occupancy grid.

- Noisy environments.
- Occluded objects in the background by a small foreground object and areas the robot sensors cannot reach.

A set of environments were constructed to demonstrate the affect of inferencing algorithms specific to the two mentioned cases. We also provide a basis for comparison of the loopy belief propagation algorithm by performing the same tasks as an established Matlab toolbox package implementation. Testing on a real life environment was also done, an office environment was mapped out and used to evaluate the effectiveness of LBP [mat]. fix

We use precision and recall, ROC curves and the MSE (section 2.10) as the basis on which we compare different states of the occupancy grid.

One notable attribute of LBP on any given CRF generated in this paper is that they converge. There is no mathematically proof in this paper but experimental data has suggested any occupancy grid imprinted onto a CRF will allow LBP to converge.need more proof? The number of iterations for LBP is set to 2 for all applications of LBP, from empirical evidence it was determined the majority of grids converged at this number (and with larger grids more then 2 iterations takes a very long processing time).

4.1.1 *Matlab comparison and de-noising*

The Matlab toolbox implements various algorithms for general graphs. We have a specific case where we take an image and add noise and then use LBP to denoise the image after learning optimal parameters.

Figure 20: Ground truth for LBP.

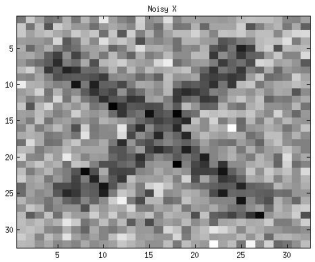


Figure 21: Matlab version of the noisy image.

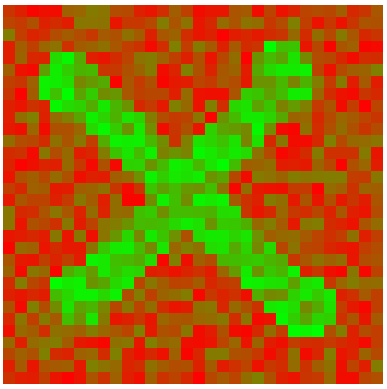


Figure 22: Simulated noisy scan for LBP.

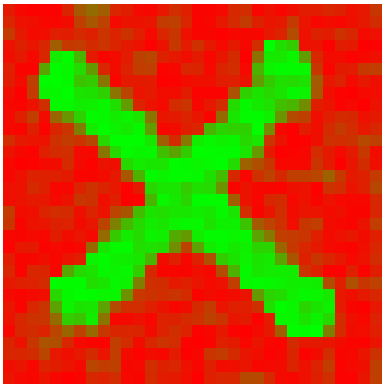


Figure 23: After LBP, without learning.

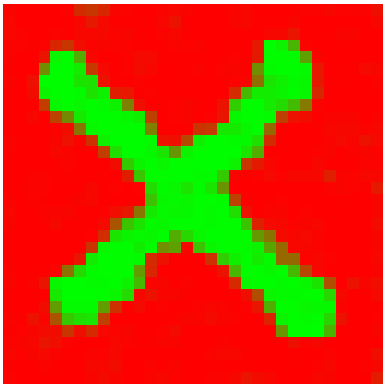


Figure 24: After LBP, with learning.

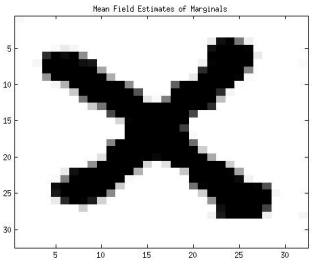


Figure 25: Matlab's version, after LBP.

	N^1	N^0
C_1	10	1
C_0	1	10

Table 10: Default Hidden Pairwise Potential.

	N^1	N^0
C_1	10.1	1.2
C_0	1.1	10.1

Table 11: Learned Hidden Pairwise Potential.

	N^1	N^0
C_1	1	0
C_0	0	1

Table 12: Default Link Pairwise Potential.

	N^1	N^0
C_1	1.2	0
C_0	0	1

Table 13: Learned Link Pairwise Potential.

The matlab toolbox does not easily provide a way to modify its psuedolikelihood parameter learning, the order in which it performs LBP and the method it uses to add random noise to the original image all of which has an effect on the ultimate state of the occupancy grid. This means that the implementation in this paper might not use the same parameters as the Matlab toolbox. We can, however, use visual inspection to see that the de-noised images closely mirrors each other.

For specific numerical results we can compare the de-noised image to the ground truth and noisy scan to gauge LBP's performance in this specific image. The mean squared error (MSE), precision, recall and accuracy is used to distinguish between each occupancy grid. In calculating the precision, recall and accuracy we do not include inferred cells which have a 0.5 occupancy probability (unknown). The parameter learning was done off the same image, with a different seed for the noise generated in the noisy scan.

ROC curves for the occupied were generated by adjusting the threshold at which we consider a cell as occupied. This classifies each cell into the two states of either occupied or empty and compared to the ground truth to generate the True Positive Rate and False Positive Rate at each adjustment.

	MSE	Precision	Recall	Accuracy
Before LBP	0.0825	n/a	n/a	n/a
After LBP, without learning	0.0219	0.9727	0.9519	0.9442
After LBP, with learning	0.0197	0.9526	0.9653	0.9383

Table 14: Rounded metrics for the outcome of each application of LBP.

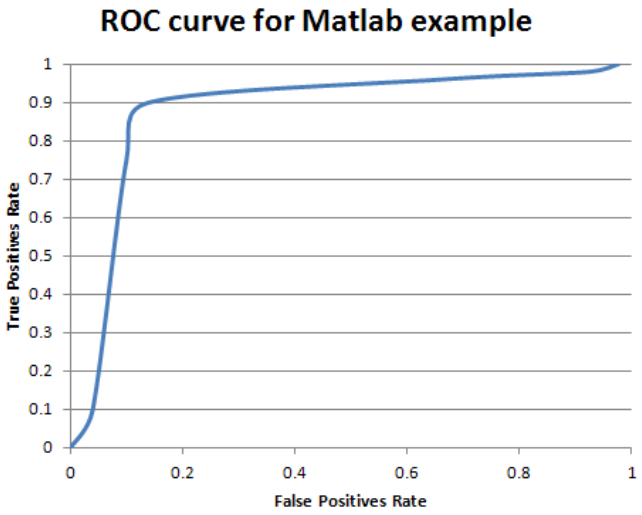


Figure 26: Simulated noisy scan for LBP.

4.1.1.1 *De-noising*

As expected, learning has improved the results both in MSE and precision and recall. Intuitively, we can expect a higher accuracy in noisy environments with a sensible definition for the hidden and link pairwise potentials and we see this is the case.

4.1.2 *Occluded Cells*

Occluded cells refers to any cells within the robot's scan range that it cannot physically scan at its current position. Generally it is hard to predict the cells behind a large object obscuring many cells, however we may infer cells behind smaller objects. Figure 27 shows a simplified setup of a environment where a small wall obscures a continuous background object with the robot in the upper left hand corner. Another feature which is noticeable in the raw occupancy grid that has been constructed is that cells further away from the scan origin may

be in between scan angles. If the scan angle increment is large enough there are gaps in the field of vision of the robot, this is another feature that may be improved upon by inferencing algorithms.

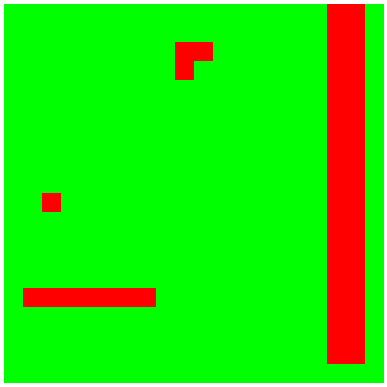


Figure 27: Ground truth of a occluded environment with wall type objects.

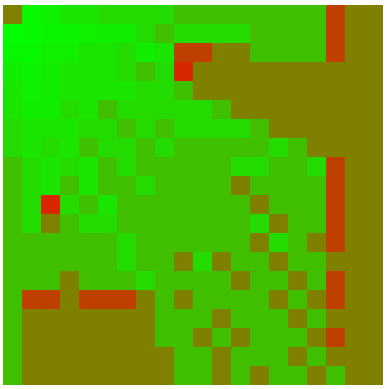


Figure 28: Scan of a occluded environment with wall type objects (robot in upper left corner).

We then perform loopy belief propagation to infer that the background object is continuous despite the initial occupancy grid generated showing otherwise. The same default parameters are used from the previous section and described by Table 23. We generate the test environments such that the robot has moved to every cell in the grid in which it can physically access (excludes cells inside enclosed spaces). We also assume that the robot has perfect odometry and scan sensors to reduce the randomness so it is easier to demonstrate this exact application of the inferencing algorithm.

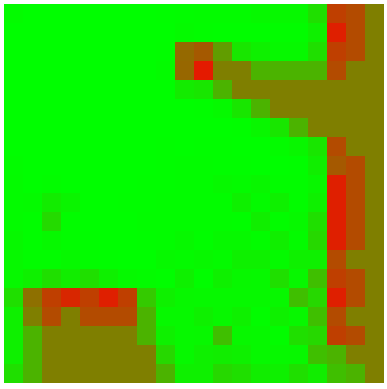


Figure 29: Occupancy grid after LBP without learning

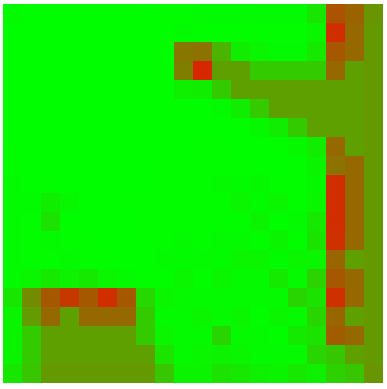


Figure 30: Occupancy grid after LBP with learning.

	MSE	Precision	Recall	Accuracy
Before LBP	0.112	n/a	n/a	n/a
After LBP, without learning	0.0623	1	0.5	0.973
After LBP, with learning	0.055	1	0.35	0.952

Table 15: Rounded metrics for the outcome of each application of LBP.

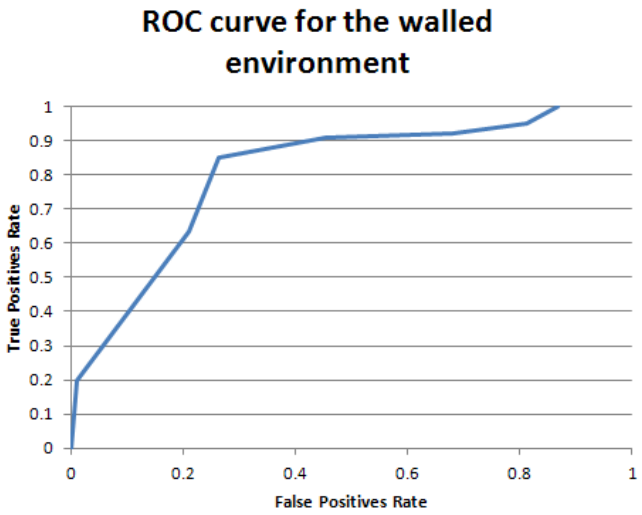


Figure 31: ROC curve for the occluded environment.

We may now learn new parameters based off this environment which we can generalise and apply to a similar environment. The learning algorithm generates the optimal parameters for Figure **??**.

	N^1	N^0
C_1	10	1
C_0	1	10

Table 16: Default Hidden Pairwise Potential.

	N^1	N^0
C_1	10.3	1.3
C_0	0.9	10.2

Table 17: Learned Hidden Pairwise Potential.

	N^1	N^0
C_1	1	0
C_0	0	1

Table 18: Default Link Pairwise Potential.

	N^1	N^0
C_1	1.1	0
C_0	0	1.3

Table 19: Learned Link Pairwise Potential.

Using the pairwise potentials in the table above we attempt to apply it to a similar environment where a background object is occluded by a foreground object. The process is run twice, once with default parameters and once for the learned parameters.

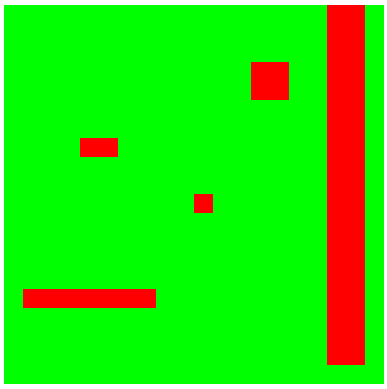


Figure 32: Testing occluded environment's ground truth.

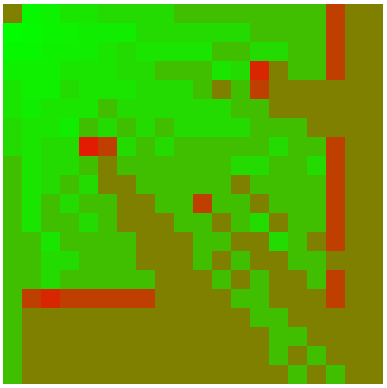


Figure 33: Testing occluded environment's scan (robot in upper left corner).

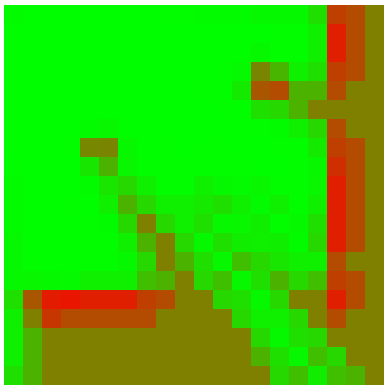


Figure 34: Testing occluded environment after LBP without learning

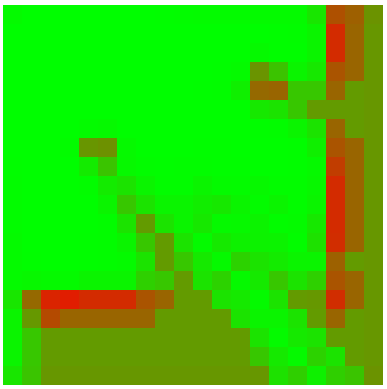


Figure 35: Testing occluded environment after LBP with learning.

	MSE	Precision	Recall	Accuracy
Before LBP	0.0825	n/a	n/a	n/a
After LBP, without learning	0.0219	0.9727	0.9519	0.9442
After LBP, with learning	0.0197	0.9526	0.9653	0.9383

Table 20: Rounded metrics for the outcome of each application of LBP.

There is a small but notable increase from using the learned parameters as opposed to running LBP with the default parameters. The difference in the occupancy grid after an application of LBP can be seen visually, the background walls are inferred to be continuous objects.

The other notable feature is the the original scan of the environments shows that after a certain distance the scan 'spreads' such that there are streaks of unknown cells within the robot's scan view. LBP infers the value of these unknown cells providing a further increase in accuracy.

4.1.2.1 *Enclosed Environments*

The other form of occlusion that can occur in occupancy grids are inaccessible environments, for example, we can visualise this as an office environment. In an office environment there may be cubicles which the robot cannot scan inside. We apply the same steps previously to this new environment.

The learning environment is a template which we use to learn the general structure of objects typical in the testing environment. For simplicity, both environments are fully scanned by the robot, excluding the cells which are not physically accessible by the robot.

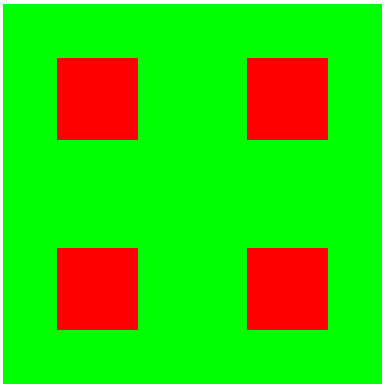


Figure 36: Learning enclosed environment's ground truth.

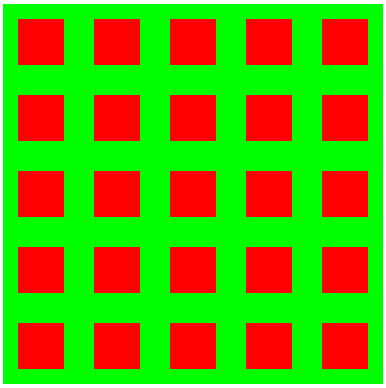


Figure 37: Testing enclosed environment's ground truth.

	N^1	N^0
C_1	10	1
C_0	1	10

Table 21: Default Hidden Pairwise Potential.

	N^1	N^0
C_1	10.4	1.3
C_0	0.8	9.9

Table 22: Learned Hidden Pairwise Potential.

	N^1	N^0
C_1	1	0
C_0	0	1

Table 23: Default Link Pairwise Potential.

	N^1	N^0
C_1	1.4	0
C_0	0	0.9

Table 24: Learned Link Pairwise Potential.

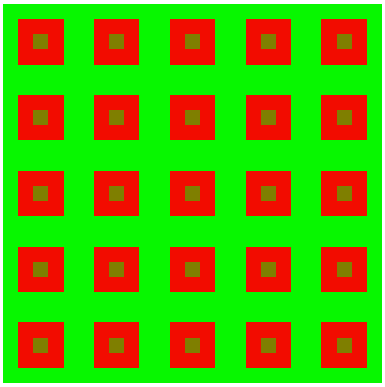


Figure 38: Testing enclosed environment scan.

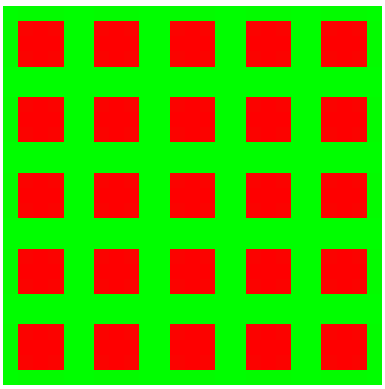


Figure 39: Testing enclosed environment after LBP with learning.

	MSE	Precision	Recall	Accuracy
Before LBP	0.024	n/a	n/a	n/a
After LBP, without learning	0.0007	1	0.5	0.833
After LBP, with learning	0.00005	1	1	1

Table 25: Rounded metrics for the outcome of each application of LBP.

A similar setup can be manipulated to produce poor results for LBP. This time the same environment is constructed but with unfilled boxes (the center is empty).

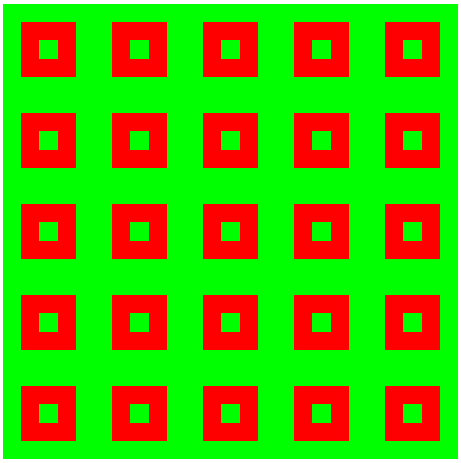


Figure 40: Enclosed environment with filled boxes ground truth.

The same learning is done but this time with an unfilled box, in this specific case the resulting LBP gives a worse MSE and accuracy as opposed to not performing LBP at all. Though this environment was specifically constructed to show cases in which LBP returns poor results and may not come up in real life environments.

4.1.3 *Real Life Office Environment*

An office in a wing of the School of IT building at the University of Sydney was mapped out using basic ROS packages. The resulting occupancy grid was used as the basis for inference. Since there is no easy way of obtaining a ground truth outside of a simulation, leave one out cross validation was used. The LBP used for inferencing on this occupancy grid used restricted bounds to how far the algorithm would propagated messages from the robot (section 3.4.4). The size of the occupancy grid caused message propagation to be very slow and it had to be done for every cell in a large grid. LBP in this case still uses the same default parameters as the previous tests in the simulation.

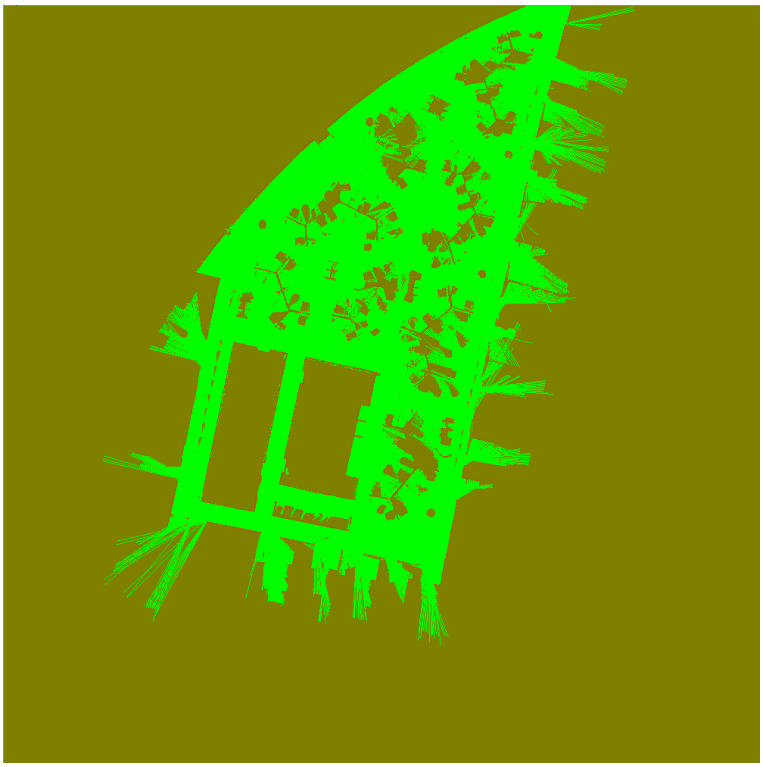


Figure 41: Level 3 School of IT finished scan.

A partial snapshot of the grid during robot mapping was taken to reduce the processing overhead while still giving a good approximation of how the algorithm will extrapolate to the entire map.

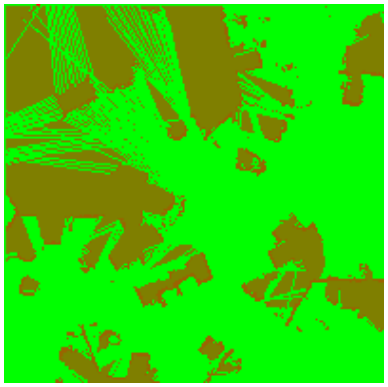


Figure 42: Partial Level 3 School of IT scan.

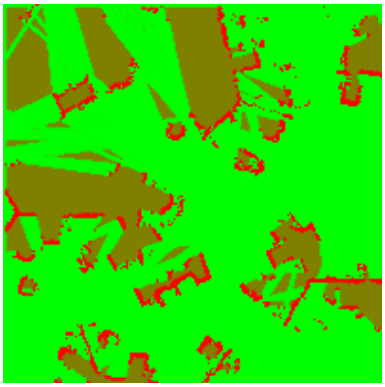


Figure 43: Partial Level 3 School of IT after LBP.

We see in the occupancy grid, after LBP, that diffused scans have been successfully resolved. Furthermore occupied space in the occupancy grid becomes, visually, easier to distinguish.

Numerically we use leave one out cross validation compared against the occupancy grid obtained after mapping. Similar to before, cells which are of unknown occupancy are ignored.

	Precision	Recall	Accuracy
After LBP, without learning	0.8436	0.6229	0.977

Table 26: Rounded metrics for the outcome of each application of LBP.

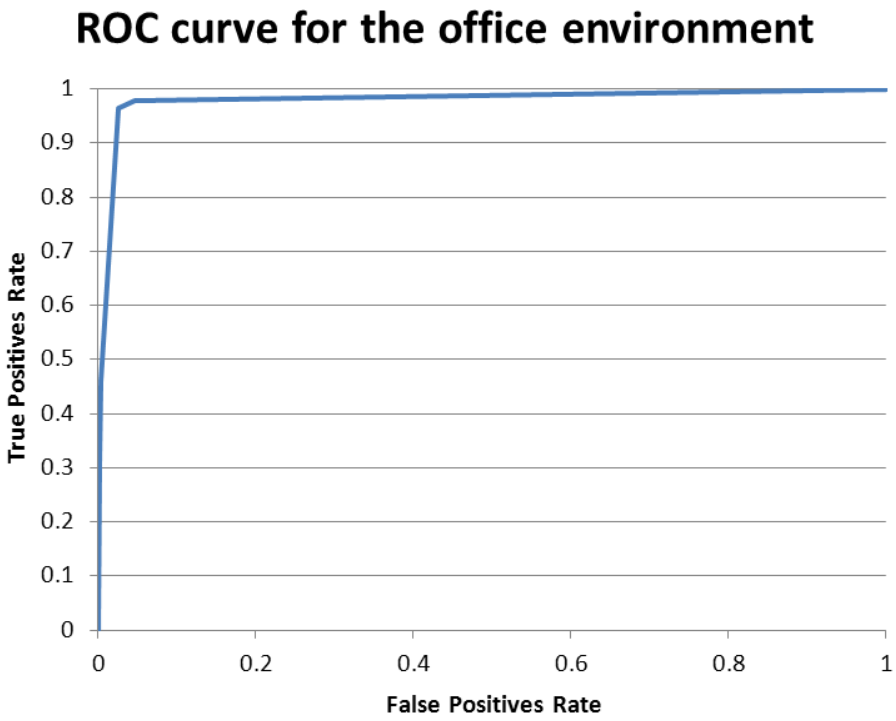


Figure 44: ROC curve for the office environment.

The ROC curve displays a steep jump in true positive rate because of several reasons. The exact occupancy grid could not be extracted from ROS's localisation package. Instead the pose returned by the ROS localisation algorithm is run through as the ground truth for the grid construction algorithm written for this paper. This creates a grid which has higher certainty than it should on whether a given cell is occupied or empty, the occupancy value tends to converge towards either 0 or 1. Furthermore, the default parameters set give a relatively high pairwise potential for occupied cells with occupied neighbors. This contributes to the certainty of occupied cells being high and around the same value for all cells inferred to be occupied.

4.2 CONCLUSION

In simulated cases we see LBP provides improvements in accuracy and MSE in general cases. The specific case constructed which gave a poor result for LBP could be improved upon by either a higher resolution scan of the environment or disabling LBP altogether (by setting hidden pairwise potentials to be zero) which can be done by improv-

ing the parameter learning algorithm to utilise entropy in jumps.

In the results we can visually see improvements in noisy environments and the algorithms ability to inference occluded objects which were discussed previously. The office scan provided a clear example on how inferencing can be done to improve scan dispersion, where the scan does not pick up everything in its range of vision.

In certain cases the parameter learning is done on MSE which produces improved MSE but worst results for precision, recall and accuracy. This is because precision and recall did not process unknown cells while they were considered in the calculation of MSE.
is office environment result good?

4.3 FUTURE WORK

The CRF does not take into account how certainty of a cell factors into its pairwise potential. A method may be used to introduce varying potentials based on the certainty of the cell. For example, a cell with a occupancy belief of 0.5 would not scale, while a belief of 0.9 would increase the hidden pairwise potential for occupied-occupied by a factor. This belief would be passed into a function generated during the parameter learning process.
A way to do this would be to map the generated occupancy value against the optimal occupancy value producing a 2d graph. A function with degree n is created to approximate the graph, n is chosen such that it fits the curve without over fitting. This function generates an one to one mapping which scales the pairwise potential based on certainty.

The order in which we perform LBP may be varied, the effect of that was not tested in this paper. The starting cell and the order of propagation of messages may change the final state of the CRF.

LBP in its current form in this paper was not designed to run in real time, mapping a large environment, this would require further modifications to improving the efficiency and runtime of LBP. The ability to run in real time means that we can measure the performance of an algorithm beyond post occupancy grid construction. For example, we can measure the time it takes for a robot to complete a mapping of a complex environment by introducing inferencing algorithms to modify its on board occupancy grid. Other tasks such as robot navigation around an already mapped environment can be measured, factors such as noise could adversely affect robot navigation, which is improved upon by LBP.

In this paper only one algorithm for inferencing was implemented however the infrastructure in terms of setup with ROS, testing methods and simulation are there to implement more algorithms such as *graph cuts* for inferencing. This would allow comparison between which methods produce the greatest accuracy and which ones are viable for real time inferencing during robot mapping.

BIBLIOGRAPHY

Ugm: Matlab code for undirected graphical models. http://www.di.ens.fr/~mschmidt/Software/UGM.html. Accessed: 2013-05-20.

Julie A Adams. Human-robot interaction design: Understanding user needs and requirements. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 49, pages 447–451. SAGE Publications, 2005.

FM Almirao, FB Da Silva, SD Scott, and ML Cummings. Designing decision and collaboration support technology for operators in multi-uav operations teams. Technical report, Citeseer, 2007.

Mordecai Avriel. *Nonlinear programming: analysis and methods*. Courier Dover Publications, 2003.

Tim Bailey and Hugh Durrant-Whyte. Simultaneous localization and mapping (slam): Part ii. *Robotics & Automation Magazine, IEEE*, 13 (3):108–117, 2006.

Christopher M Bishop et al. *Pattern recognition and machine learning*, volume 1. springer New York, 2006.

Alfredo Braunstein, Marc Mézard, and Riccardo Zecchina. Survey propagation: An algorithm for satisfiability. *Random Structures & Algorithms*, 27(2):201–226, 2005.

Alan Clewer. Cambridge dictionary of statistics. *Journal of Applied Ecology*, 36(5):842–842, 1999.

Thomas Collins. Occupancy grid learning using contextual forward modelling. *Journal of Intelligent & Robotic Systems*, 64(3):505–542, 2011.

Ádamo L De Santana, Carlos Renato L Francês, and João C Weyl Costa. Algorithm for graphical bayesian modeling based on multiple regressions. In *MICAI 2007: Advances in Artificial Intelligence*, pages 496–506. Springer, 2007.

J. Dietrich, G. Hirzinger, J. Heindl, and J. Schott. Multisensory telerobotic techniques. In ThomasC. Henderson, editor, *Traditional and Non-Traditional Robotic Sensors*, volume 63 of *NATO ASI Series*, pages 255–283. Springer Berlin Heidelberg, 1990. ISBN 978-3-642-75986-4. doi: 10.1007/978-3-642-75984-0_18. URL http://dx.doi.org/10.1007/978-3-642-75984-0_18.

MWM Gamini Dissanayake, Paul Newman, Steven Clark, Hugh F Durrant-Whyte, and Michael Csorba. A solution to the simultaneous localization and map building (slam) problem. *Robotics and Automation, IEEE Transactions on*, 17(3):229–241, 2001.

Bertrand Douillard, Dieter Fox, and Fabio Ramos. Laser and vision based outdoor object mapping. In *Proc. of Robotics: Science and Systems (RSS)*, pages 9–16, 2008.

Michael Eichler. Graphical modelling of multivariate time series. *Probability Theory and Related Fields*, pages 1–36, 2012.

Alberto Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6):46–57, 1989.

Brian Everitt and Anders Skrondal. *The Cambridge dictionary of statistics*, volume 4. Cambridge University Press Cambridge, 2002.

Jianqing Fan and Qiwei Yao. *Nonlinear time series: nonparametric and parametric methods*. Springer Verlag, 2003.

Stuart Geman and Donald Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (6):721–741, 1984.

Michael A Goodrich and Alan C Schultz. Human-robot interaction: a survey. *Foundations and Trends in Human-Computer Interaction*, 1(3): 203–275, 2007.

Dirk Hähnel, Dirk Schulz, and Wolfram Burgard. Mobile robot mapping in populated environments. *Advanced Robotics*, 17(7):579–597, 2003.

Curtis Michael Humphrey. *Information abstraction visualization for human-robot interaction*. PhD thesis, 2009.

Ernst Ising. Beitrag zur theorie des ferromagnetismus. *Zeitschrift für Physik A Hadrons and Nuclei*, 31(1):253–258, 1925.

Minwoo Jeong and G Geunbae Lee. Triangular-chain conditional random fields. *Audio, Speech, and Language Processing, IEEE Transactions on*, 16(7):1287–1302, 2008.

Dennis Kao. *Belief Propagation*. PhD thesis, University of Toronto, 2005.

Ross Kindermann, James Laurie Snell, et al. *Markov random fields and their applications*, volume 1. American Mathematical Society Providence, RI, 1980.

Jon Kleinberg and Eva Tardos. Approximation algorithms for classification problems with pairwise relationships: Metric labeling and markov random fields. In *Foundations of Computer Science, 1999. 40th Annual Symposium on*, pages 14–23. IEEE, 1999.

Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

Kurt Konolige. Improved occupancy grids for map building. *Autonomous Robots*, 4(4):351–367, 1997.

Solomon Kullback. On the bernoulli distribution. *Bulletin of the American Mathematical Society*, 41(12):857–864, 1935.

Sanjiv Kumar and Martial Hebert. Discriminative random fields: A discriminative framework for contextual interaction in classification. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 1150–1157. IEEE, 2003.

Samuli Laine and Tero Karras. Two methods for fast ray-cast ambient occlusion. In *Computer Graphics Forum*, volume 29, pages 1325–1333. Wiley Online Library, 2010.

Jong Hwan Lim and Dong Woo Cho. Physically based sensor modeling for a sonar map in a specular environment. In *Robotics and Automation, 1992. Proceedings., 1992 IEEE International Conference on*, pages 1714–1719. IEEE, 1992.

Benson Limketkai, Lin Liao, and Dieter Fox. Relational object maps for mobile robots. In *International Joint Conference on Artificial Intelligence*, volume 19, page 1471. LAWRENCE ERLBAUM ASSOCIATES LTD, 2005.

Ziyuan Liu and Georg von Wichert. Extracting semantic indoor maps from occupancy grids. *Robotics and Autonomous Systems*, 2013.

Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to information retrieval*, volume 1. Cambridge University Press Cambridge, 2008.

Oscar Martinez Mozos, Rudolph Triebel, Patric Jensfelt, Axel Rottmann, and Wolfram Burgard. Supervised semantic labeling of places using information extracted from sensor data. *Robotics and Autonomous Systems*, 55(5):391–402, 2007.

Larry Matthies and Alberto Elfes. Integration of sonar and stereo range data using a grid-based representation. In *Robotics and Automation, 1988. Proceedings., 1988 IEEE International Conference on*, pages 727–733. IEEE, 1988.

Frederick Mosteller. A k-sample slippage test for an extreme population. In *Selected Papers of Frederick Mosteller*, pages 101–109. Springer, 2006.

Kevin P Murphy, Yair Weiss, and Michael I Jordan. Loopy belief propagation for approximate inference: An empirical study. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pages 467–475. Morgan Kaufmann Publishers Inc., 1999.

Robin R Murphy. *An introduction to AI robotics*. MIT press, 2000.

KS Nagla, Moin Uddin, and Dilbag Singh. Improved occupancy grid mapping in specular environment. *Robotics and Autonomous Systems*, 2012.

Tan Pang-Ning, Michael Steinbach, and Vipin Kumar. Introduction to data mining. In *Library of Congress*, 2006.

Judea Pearl. Fusion, propagation, and structuring in belief networks. *Artificial intelligence*, 29(3):241–288, 1986.

CJ Preston. Generalized gibbs states and markov random fields. *Advances in Applied probability*, pages 242–261, 1973.

Andrzej Pronobis, Patric Jensfelt, Kristoffer Sjöö, Hendrik Zender, Geert-Jan M Kruijff, Oscar Martinez Mozos, and Wolfram Burgard. Semantic modelling of space. *Cognitive Systems*, pages 165–221, 2010.

Ariadna Quattoni, Michael Collins, and Trevor Darrell. Conditional random fields for object recognition. In *In NIPS*. Citeseer, 2004.

Ariadna Quattoni, Sybor Wang, L-P Morency, Michael Collins, and Trevor Darrell. Hidden conditional random fields. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(10):1848–1852, 2007.

Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, 2009.

Marco Reale. *A Graphical Modelling Approach to Time Series*. 1998.

Matt Rosencrantz, Geoffrey Gordon, and Sebastian Thrun. Decentralized sensor fusion with distributed particle filters. In *Proceedings of the Nineteenth conference on Uncertainty in Artificial Intelligence*, pages 493–500. Morgan Kaufmann Publishers Inc., 2002.

Scott D Roth. Ray casting for modeling solids. *Computer graphics and image processing*, 18(2):109–144, 1982.

Philip Rothman. *Nonlinear time series analysis of economic and financial data*, volume 1. Springer, 1999.

Tim Schroeder. Collision detection using ray casting. *Game Developer*, 8(8):50–56, 2001. URL `http://ezproxy.library.usyd.edu.au/login?url=http://search.proquest.com/docview/219085645?accountid=14757`. Copyright - Copyright Miller Freeman Inc. Aug 2001; Last updated - 2010-06-09; DOI - 75356150; 1411659; 34952; GADE; SubjectsTermNotLitGenreText - United States; US.

Frank Spitzer. Markov random fields and gibbs ensembles. *The American Mathematical Monthly*, 78(2):142–154, 1971.

Liam Stewart, Xuming He, and Richard S Zemel. Learning flexible features for conditional random fields. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(8):1415–1426, 2008.

Erik B Sudderth, Alexander T Ihler, Michael Isard, William T Freeman, and Alan S Willsky. Nonparametric belief propagation. *Communications of the ACM*, 53(10):95–103, 2010.

Pang-Ning Tan et al. *Introduction to data mining*. Pearson Education India, 2007.

Nicholas Tarnoff, Adam Jacoff, and Ronald Lumia. Graphical simulation for sensor based robot programming. *Journal of Intelligent and Robotic Systems*, 5(1):49–62, 1992.

Hakan Temeltas and D Kayak. Slam for robot navigation. *Aerospace and Electronic Systems Magazine, IEEE*, 23(12):16–19, 2008.

Sebastian Thrun. Learning occupancy grid maps with forward sensor models. *Autonomous robots*, 15(2):111–127, 2003.

Sebastian Thrun, Wolfram Burgard, and Dieter Fox. A real-time algorithm for mobile robot mapping with applications to multi-robot and 3d mapping. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, volume 1, pages 321–328. IEEE, 2000.

Sebastian B Thrun. Exploration and model building in mobile robot domains. In *Neural Networks, 1993., IEEE International Conference on*, pages 175–180. IEEE, 1993.

Martin J. Wainwright, Tommi S Jaakkola, and Alan S. Willsky. Tree-based reparameterization framework for analysis of sum-product and related algorithms. *Information Theory, IEEE Transactions on*, 49 (5):1120–1146, 2003.

Jan W Weingarten, Gabriel Gruener, and Roland Siegwart. A state-of-the-art 3d sensor for robot navigation. In *Intelligent Robots and*

Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on, volume 3, pages 2155–2160. IEEE, 2004.

Denis Fernando Wolf and Gaurav S Sukhatme. Semantic mapping using mobile robots. *Robotics, IEEE Transactions on*, 24(2):245–258, 2008.

Safdar Zaman, Wolfgang Slany, and Gerald Steinbauer. Ros-based mapping, localization and autonomous navigation using a pioneer 3-dx robot and their relevant issues. In *Electronics, Communications and Photonics Conference (SIECPC), 2011 Saudi International*, pages 1–5. IEEE, 2011.

BIBLIOGRAPHY

Ugm: Matlab code for undirected graphical models. http://www.di.ens.fr/~mschmidt/Software/UGM.html. Accessed: 2013-05-20.

Julie A Adams. Human-robot interaction design: Understanding user needs and requirements. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 49, pages 447–451. SAGE Publications, 2005.

FM Almirao, FB Da Silva, SD Scott, and ML Cummings. Designing decision and collaboration support technology for operators in multi-uav operations teams. Technical report, Citeseer, 2007.

Mordecai Avriel. *Nonlinear programming: analysis and methods*. Courier Dover Publications, 2003.

Tim Bailey and Hugh Durrant-Whyte. Simultaneous localization and mapping (slam): Part ii. *Robotics & Automation Magazine, IEEE*, 13 (3):108–117, 2006.

Christopher M Bishop et al. *Pattern recognition and machine learning*, volume 1. springer New York, 2006.

Alfredo Braunstein, Marc Mézard, and Riccardo Zecchina. Survey propagation: An algorithm for satisfiability. *Random Structures & Algorithms*, 27(2):201–226, 2005.

Alan Clewer. Cambridge dictionary of statistics. *Journal of Applied Ecology*, 36(5):842–842, 1999.

Thomas Collins. Occupancy grid learning using contextual forward modelling. *Journal of Intelligent & Robotic Systems*, 64(3):505–542, 2011.

Ádamo L De Santana, Carlos Renato L Francês, and João C Weyl Costa. Algorithm for graphical bayesian modeling based on multiple regressions. In *MICAI 2007: Advances in Artificial Intelligence*, pages 496–506. Springer, 2007.

J. Dietrich, G. Hirzinger, J. Heindl, and J. Schott. Multisensory telerobotic techniques. In ThomasC. Henderson, editor, *Traditional and Non-Traditional Robotic Sensors*, volume 63 of *NATO ASI Series*, pages 255–283. Springer Berlin Heidelberg, 1990. ISBN 978-3-642-75986-4. doi: 10.1007/978-3-642-75984-0_18. URL http://dx.doi.org/10.1007/978-3-642-75984-0_18.

MWM Gamini Dissanayake, Paul Newman, Steven Clark, Hugh F Durrant-Whyte, and Michael Csorba. A solution to the simultaneous localization and map building (slam) problem. *Robotics and Automation, IEEE Transactions on*, 17(3):229–241, 2001.

Bertrand Douillard, Dieter Fox, and Fabio Ramos. Laser and vision based outdoor object mapping. In *Proc. of Robotics: Science and Systems (RSS)*, pages 9–16, 2008.

Michael Eichler. Graphical modelling of multivariate time series. *Probability Theory and Related Fields*, pages 1–36, 2012.

Alberto Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6):46–57, 1989.

Brian Everitt and Anders Skrondal. *The Cambridge dictionary of statistics*, volume 4. Cambridge University Press Cambridge, 2002.

Jianqing Fan and Qiwei Yao. *Nonlinear time series: nonparametric and parametric methods*. Springer Verlag, 2003.

Stuart Geman and Donald Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (6):721–741, 1984.

Michael A Goodrich and Alan C Schultz. Human-robot interaction: a survey. *Foundations and Trends in Human-Computer Interaction*, 1(3): 203–275, 2007.

Dirk Hähnel, Dirk Schulz, and Wolfram Burgard. Mobile robot mapping in populated environments. *Advanced Robotics*, 17(7):579–597, 2003.

Curtis Michael Humphrey. *Information abstraction visualization for human-robot interaction*. PhD thesis, 2009.

Ernst Ising. Beitrag zur theorie des ferromagnetismus. *Zeitschrift für Physik A Hadrons and Nuclei*, 31(1):253–258, 1925.

Minwoo Jeong and G Geunbae Lee. Triangular-chain conditional random fields. *Audio, Speech, and Language Processing, IEEE Transactions on*, 16(7):1287–1302, 2008.

Dennis Kao. *Belief Propagation*. PhD thesis, University of Toronto, 2005.

Ross Kindermann, James Laurie Snell, et al. *Markov random fields and their applications*, volume 1. American Mathematical Society Providence, RI, 1980.

Jon Kleinberg and Eva Tardos. Approximation algorithms for classification problems with pairwise relationships: Metric labeling and markov random fields. In *Foundations of Computer Science, 1999. 40th Annual Symposium on*, pages 14–23. IEEE, 1999.

Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

Kurt Konolige. Improved occupancy grids for map building. *Autonomous Robots*, 4(4):351–367, 1997.

Solomon Kullback. On the bernoulli distribution. *Bulletin of the American Mathematical Society*, 41(12):857–864, 1935.

Sanjiv Kumar and Martial Hebert. Discriminative random fields: A discriminative framework for contextual interaction in classification. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 1150–1157. IEEE, 2003.

Samuli Laine and Tero Karras. Two methods for fast ray-cast ambient occlusion. In *Computer Graphics Forum*, volume 29, pages 1325–1333. Wiley Online Library, 2010.

Jong Hwan Lim and Dong Woo Cho. Physically based sensor modeling for a sonar map in a specular environment. In *Robotics and Automation, 1992. Proceedings., 1992 IEEE International Conference on*, pages 1714–1719. IEEE, 1992.

Benson Limketkai, Lin Liao, and Dieter Fox. Relational object maps for mobile robots. In *International Joint Conference on Artificial Intelligence*, volume 19, page 1471. LAWRENCE ERLBAUM ASSOCIATES LTD, 2005.

Ziyuan Liu and Georg von Wichert. Extracting semantic indoor maps from occupancy grids. *Robotics and Autonomous Systems*, 2013.

Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to information retrieval*, volume 1. Cambridge University Press Cambridge, 2008.

Oscar Martinez Mozos, Rudolph Triebel, Patric Jensfelt, Axel Rottmann, and Wolfram Burgard. Supervised semantic labeling of places using information extracted from sensor data. *Robotics and Autonomous Systems*, 55(5):391–402, 2007.

Larry Matthies and Alberto Elfes. Integration of sonar and stereo range data using a grid-based representation. In *Robotics and Automation, 1988. Proceedings., 1988 IEEE International Conference on*, pages 727–733. IEEE, 1988.

Frederick Mosteller. A k-sample slippage test for an extreme population. In *Selected Papers of Frederick Mosteller*, pages 101–109. Springer, 2006.

Kevin P Murphy, Yair Weiss, and Michael I Jordan. Loopy belief propagation for approximate inference: An empirical study. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pages 467–475. Morgan Kaufmann Publishers Inc., 1999.

Robin R Murphy. *An introduction to AI robotics*. MIT press, 2000.

KS Nagla, Moin Uddin, and Dilbag Singh. Improved occupancy grid mapping in specular environment. *Robotics and Autonomous Systems*, 2012.

Tan Pang-Ning, Michael Steinbach, and Vipin Kumar. Introduction to data mining. In *Library of Congress*, 2006.

Judea Pearl. Fusion, propagation, and structuring in belief networks. *Artificial intelligence*, 29(3):241–288, 1986.

CJ Preston. Generalized gibbs states and markov random fields. *Advances in Applied probability*, pages 242–261, 1973.

Andrzej Pronobis, Patric Jensfelt, Kristoffer Sjöö, Hendrik Zender, Geert-Jan M Kruijff, Oscar Martinez Mozos, and Wolfram Burgard. Semantic modelling of space. *Cognitive Systems*, pages 165–221, 2010.

Ariadna Quattoni, Michael Collins, and Trevor Darrell. Conditional random fields for object recognition. In *In NIPS*. Citeseer, 2004.

Ariadna Quattoni, Sybor Wang, L-P Morency, Michael Collins, and Trevor Darrell. Hidden conditional random fields. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(10):1848–1852, 2007.

Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, 2009.

Marco Reale. *A Graphical Modelling Approach to Time Series*. 1998.

Matt Rosencrantz, Geoffrey Gordon, and Sebastian Thrun. Decentralized sensor fusion with distributed particle filters. In *Proceedings of the Nineteenth conference on Uncertainty in Artificial Intelligence*, pages 493–500. Morgan Kaufmann Publishers Inc., 2002.

Scott D Roth. Ray casting for modeling solids. *Computer graphics and image processing*, 18(2):109–144, 1982.

Philip Rothman. *Nonlinear time series analysis of economic and financial data*, volume 1. Springer, 1999.

Tim Schroeder. Collision detection using ray casting. *Game Developer*, 8(8):50–56, 2001. URL `http://ezproxy.library.usyd.edu.au/login?url=http://search.proquest.com/docview/219085645?accountid=14757`. Copyright - Copyright Miller Freeman Inc. Aug 2001; Last updated - 2010-06-09; DOI - 75356150; 1411659; 34952; GADE; SubjectsTermNotLitGenreText - United States; US.

Frank Spitzer. Markov random fields and gibbs ensembles. *The American Mathematical Monthly*, 78(2):142–154, 1971.

Liam Stewart, Xuming He, and Richard S Zemel. Learning flexible features for conditional random fields. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(8):1415–1426, 2008.

Erik B Sudderth, Alexander T Ihler, Michael Isard, William T Freeman, and Alan S Willsky. Nonparametric belief propagation. *Communications of the ACM*, 53(10):95–103, 2010.

Pang-Ning Tan et al. *Introduction to data mining*. Pearson Education India, 2007.

Nicholas Tarnoff, Adam Jacoff, and Ronald Lumia. Graphical simulation for sensor based robot programming. *Journal of Intelligent and Robotic Systems*, 5(1):49–62, 1992.

Hakan Temeltas and D Kayak. Slam for robot navigation. *Aerospace and Electronic Systems Magazine, IEEE*, 23(12):16–19, 2008.

Sebastian Thrun. Learning occupancy grid maps with forward sensor models. *Autonomous robots*, 15(2):111–127, 2003.

Sebastian Thrun, Wolfram Burgard, and Dieter Fox. A real-time algorithm for mobile robot mapping with applications to multi-robot and 3d mapping. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, volume 1, pages 321–328. IEEE, 2000.

Sebastian B Thrun. Exploration and model building in mobile robot domains. In *Neural Networks, 1993., IEEE International Conference on*, pages 175–180. IEEE, 1993.

Martin J. Wainwright, Tommi S Jaakkola, and Alan S. Willsky. Tree-based reparameterization framework for analysis of sum-product and related algorithms. *Information Theory, IEEE Transactions on*, 49 (5):1120–1146, 2003.

Jan W Weingarten, Gabriel Gruener, and Roland Siegwart. A state-of-the-art 3d sensor for robot navigation. In *Intelligent Robots and*

Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on, volume 3, pages 2155–2160. IEEE, 2004.

Denis Fernando Wolf and Gaurav S Sukhatme. Semantic mapping using mobile robots. *Robotics, IEEE Transactions on*, 24(2):245–258, 2008.

Safdar Zaman, Wolfgang Slany, and Gerald Steinbauer. Ros-based mapping, localization and autonomous navigation using a pioneer 3-dx robot and their relevant issues. In *Electronics, Communications and Photonics Conference (SIECPC), 2011 Saudi International*, pages 1–5. IEEE, 2011.