Neural Network Applications in Sensor Fusion For An

Autonomous Mobile Robot

Joris W.M. van Dam, Ben J.A. Kröse and Franciscus C.A. Groen

Faculty of Mathematics, Computer Science Physics and Astronomy

University of Amsterdam

Kruislaan 403

NL - 1098 SJ Amsterdam

dam@fwi.uva.nl

tel: +31 20 525 7463

The investigations were supported by the Foundation for Computer Science in the Netherlands (SION) with financial support from the Netherlands Organisation for Scientific Research (NWO).
Neural Network Applications in Sensor Fusion
For An Autonomous Mobile Robot

Abstract

Key issue in the design of a sensor data fusion system is the conversion of sensor measurements to an internal representation. Thus, in this article, we identify the problems with traditional conversion methods and we introduce a neural network which learn how to convert such measurements.

keywords: learning sensor models, neural networks, sensor fusion, occupancy grids
1 Introduction

In an autonomous mobile robot, it is essential that the robot represents its local environment. This representation can be used for low-level obstacle avoidance ([Millan93]), for the maintenance of a global representation of the environment ([Elles89, Vandorpe94]) and for determining the position of the robot in such a global representation ([Braunegg93]). The robot typically acquires such a representation of its local environment by the use of its sensors. Unfortunately, sensor measurements are uncertain, erroneous and incomplete and the outputs of multiple sensors must be fused to obtain a more reliable representation. Key issue in sensor fusion is the conversion of the sensor data to a common ‘language’ before the actual fusion is performed ([Shira94]). This conversion involves modeling of the sensors’ error characteristics. At first the sensor’s performance was modeled with standard mathematical models ([Elles89, Sabater91]) but it has been generally acknowledged that these models are not sufficiently accurate. Therefore, several algorithms were devised to learn these models in a calibration phase [Moravec92, Bessiere, Chen93]). In this article we present a neural network approach to the learning of (inverse) sensor models. The main advantages of this approach are that the sensor model remains adaptive both to changes in the error characteristics of the sensor and to changes in the environment, which also influence the performance of a sensor.

The rest of the text is organised as follows. First we define the internal representation we use in our system. This is the ‘language’ all sensor data are converted to before fusion is performed. We will then discuss in more detail traditional methods for converting sensor data. In the subsequent section we discuss our neural network approach to the conversion problem. The network is trained supervisedly; the main problem is how the learning samples for the network can be obtained. We show that the network can be trained with learning samples that do not specify the desired output of the network directly, but instead contain outputs whose expected values equal the desired outputs of the network. Such samples can indeed be provided for by the robot. The text is concluded with test results showing the learned models.
2 Choice of internal representation

An important choice in the design of an autonomous system for a mobile robot is the choice of an internal representation. It is still an open debate which representation is most suitable for which particular applications. In our research, we considered the following:

- there is no a-priori map of the environment,
- the local environment of the robot should be represented in a robot-centered representation,
- for efficient fusion, it must be possible to represent the sensors’ error characteristics.

These considerations led to the choice of a robot-centered occupancy grid. An occupancy grid (see [Elfes89]) $G$ is defined as a tessellation of the environment in a number of cells $c_j$, $j = 1 \ldots N$ where for each cell the probability $g_j$ is given that the cell is occupied. In our approach the grid’s position and orientation coincide with the current position and orientation of the robot.

The remainder of this text is devoted to the subject how an accurate conversion can be performed from sensor measurements to the robot-centered occupancy grids.

3 Conversion of sensor data: traditional methods

In the context of the sensor data fusion system defined in the previous section the problem of the conversion of sensor data can be stated more formally as defining conversion functions $C_j$ such that

$$g_j = P(\text{cell } c_j \text{ is occupied } | \ s) = C_j(s), \quad (3.1)$$

where $s$ is the measurement of the sensor. In this text we restrict ourselves to range measurements $s$, but the theory is easily extended to other types of sensors.

The conditional probability $P(c_j \text{ is occupied } | \ s)$, the inverse sensor model, is hard to obtain directly. Therefore, it is often derived from its inverse using Bayes’ rule:

$$P(c_j \text{ is occupied } | \ s) = \frac{P(s \ | \ c_j \text{ is occupied})P(c_j \text{ is occupied})}{P(s)}. \quad (3.2)$$
The conditional probability $P(s | c_j \text{ is occupied})$ is called the sensor model. The probability $P(s)$ is the a-priori probability for a measurement $s$. The sensor model describes the error characteristics of the sensor.

The prior probability $P(s)$ in (3.2) is obtained by:

$$P(s) = P(s | c_j \text{ is occupied})P(c_j \text{ is occupied}) + P(s | c_j \text{ is empty})(1 - P(c_j \text{ is occupied})). \quad (3.3)$$

Again, this equation uses the sensor model but also the conditional probability $P(s | c_j \text{ is empty})$, which is called the free space hypothesis ([Konolige95]). Since the two are obviously related, the discussion in literature is focused on the definition of sensor models.

Much effort has been devoted to the definition of accurate sensor models. In this section, we will give a short impression of these approaches and in the next section we will introduce a neural network approach to the problem of finding accurate conversion functions $C_j$.

Initially standard mathematical functions were used as sensor models for ease of computation. In [Elles89] Gaussian shaped sensor models were proposed; this has become a common approach in the use of occupancy grids. If such Gaussian shaped sensor models are used, the sensor measurements can be converted to an occupancy grid using (3.2). An example of such a conversion is sketched in fig. 1. The figure clearly shows the free space hypothesis, where the occupancy values

![Figure 1: Example of a range measurement converted to an occupancy grid using Gaussian sensor models. The sensor is positioned in the center of the y-axis on the top left of the figure and is pointed to the lower right corner of the figure. The a-priori occupancy value is 0.5.](image)

are lower than the a-priori occupancy value, and the Gaussian-shaped peak in the figure, indicat-
ing that those cells are probably occupied by an obstacle. This particular conversion shows that
the sensor has determined the distance to the obstacle rather accurately, whereas it is uncertain
about the angle at which the distance is measured. This indicates that this model is probably
derived from an acoustic sensor.

It was found, however, that Gaussian shaped sensor models are rather inaccurate and various
alternatives were suggested. E.g., in [Konolige95] an extension to the method is defined which
takes care of specular reflections and redundant readings; in [Sabater91] it is proposed to use
uncertainty sets as sensor models and in [Chen93] a polynomial model is defined. A problem with
the search for the optimal sensor model is the fact that the model does not only depend on the
sensor itself; the error characteristics of the sensor are also influenced by by the environment, such
as the air temperature and the reflection coefficient of the obstacles in the environment.

Therefore, several approaches are described in which the sensor model is learned in a calibration
phase ([Moravec92, Bessiere, Chen93]). The objection to this approach is that the model is not
adaptive to changes in either the sensor or the environment after calibration of the model.
Furthermore, in practice the multiple measurements $s$ taken to estimate the model $P(s \mid c_j \text{ is occupied})$
are correlated and therefore an inaccurate model is obtained. In the next section, we propose a
neural network method to learn the functions $C_j$ for the conversion of sensor data which overcomes
these problems.

4 Converting sensor data with a neural network

In the previous section it functions $C_j$ were defined which perform the conversion of sensor mea-
surements to occupancy grids (see (3.1)). It was shown in (3.2, 3.3) how these conversion functions
can be obtained with the use of a sensor model. In this section we introduce a neural network
which learns the functions $C_j$ from sensor measurements $s$ to occupancy values $g_j$ directly. Thus,
we do not explicitly learn the sensor model and the free space hypothesis, but these are included

\footnote{The sensor model defined in [Konolige95] also includes influence from reflection coefficients.}
in the learned conversion functions.

4.1 Network topology

A neural network is to be defined which learns the conversion functions $g_j = C_j(s)$. To this end a network is defined as sketched in fig. 2:

- input neurons $i$ which are fed with the sensor measurement $s$ (in this case we have only one input unit for the range measurement),
- a number of hidden units $h_j, j = 1, \ldots, N_{\text{hid}}$ with activation function $F_{\text{hid}}$ and biases $\Psi_j$,
- a set of output units $o_i, i = 1, \ldots, N_{\text{out}}$ with activation function $F_{\text{out}}$ and biases $\Theta_i$,
- a set of weights $v_j$, each hidden unit is connected to all input units,
- a set of weights $w_{ij}$, each output unit is connected to all hidden units.

In this application each output unit represents a single cell $c_i$ in the occupancy grid and thus $g_i \equiv o_i$ and $N_{\text{out}} \equiv N$.

We are now faced with the task to adapt the weights and biases in the network such that the function $o_i \equiv C'_i(i)$ learned by the network is given by:

$$C'_i \approx C_i = P(\text{cell } c_j \text{ is occupied } | s).$$  \hspace{1cm} (4.1)
4.2 The learning rule

Suppose we have a number of learning samples \((s, G^*)\) consisting of a sensor measurement \(s\) and the corresponding occupancy values \(G^* = \{g_i^*, \ldots, g_N^*\}\). If we had a set of such samples we could train the network by applying gradient descent to the summed squared error \(\varepsilon\) of the network:

\[
\Delta w_{ij} = -\gamma \cdot \frac{\partial \varepsilon}{\partial w_{ij}} = 2\gamma (g_i^* - o_i) \frac{\partial}{\partial w_{ij}} o_i.
\]  

(4.2)

This learning rule is called the generalised \(\delta\)-rule. The weights to the hidden units can be adjusted similarly by first back-propagating the summed squared error to the hidden units and then applying gradient descent to the back-propagated error (see [Rummelhart86]).

4.3 The learning samples

Unfortunately, it is not possible to obtain such learning samples \((s, G^*)\) in practice. In practice, we can only obtain sensor measurements \(s\) in environments for which we now for certain whether or not the cells in the grid \(G\) are occupied. I.e., we can only obtain samples \((s, G^\text{ins})\): with \(\forall i\)

\[ g_i^{\text{ins}} \in \{0, 1\}. \]

These values are not probabilities, but realisation of a stochastic process following the probability distribution \(g_i^*\). I.e., \(P(g_i^{\text{ins}} = 1) = g_i^*\), or \(E[g_i^{\text{ins}}] = g_i^*\), where the expected value is taken over all possible environments which give the sensor measurement \(s\).

But what if we train the network with the samples \((s, G^\text{ins})\). In this case, we write for the expected change in the weights of the network (using (4.2)):

\[
E[\Delta w_{ij}] = E \left[ 2\gamma (g_i^{\text{ins}} - o_i) \frac{\partial}{\partial w_{ij}} o_i \right] = \ldots = 2\gamma (g_i^* - o_i) \frac{\partial}{\partial w_{ij}} o_i.
\]  

(4.3)

Here, we used the fact that, since the expectation \(E[]\) is taken over all the environments for which the sensor measurement equals \(s\), the expected value of all values in the environment equal the values themselves (e.g., \(E[o_i] = o_i\)).

And thus we have shown that although supervised samples \((s, G^*)\) are not available, the
network can still be trained supervisedly. Instead samples \( (s, G^{\text{true}}) \) are used for which the expected values of the outputs \( g_i^{\text{true}} \) equal the desired output values \( g_i^* \). This method for supervised training of neural networks is also successfully applied in [VanDam94]. It is easily verified that the network converges to \( o_i = P(\text{cell } c_j \text{ is occupied } | s) \) and thus \( C_i^j = C_i \).

### 4.4 The learning algorithm

Learning samples \( (s, G^{\text{true}}) \) are obtained by the robot itself with the “sense-and-drive” algorithm:

- with the robot in configuration \( \phi \) take a sensor measurement \( s \),
- drive in a random direction until collision occurs,
- using the odometry of the robot, represent the path just travelled by the robot in an occupancy grid centered at \( \phi \). Set \( g_i^{\text{true}} = 1 \) for the cell \( c_i \) in which collision occurred and set \( g_j^{\text{true}} = 0 \) for the cells \( c_j \) that were traversed without collision.

The successful application of the sense-and-drive algorithm requires frequent collision of the robot. Obviously, this cannot be done at run time and thus this algorithm can only be applied in a calibration phase. However, at run time the functions \( C_j^i \) for the conversion of sensor data remain adaptive: the network can be trained with learning samples \( (s, G^*) \) where \( s \) are the measurements taken by the robot at run time and where \( G^* \) are the occupancy grids representing the robot’s local environment, obtained by the fusion of the converted sensor measurements.

### 4.5 Experiments and results

The learning algorithm described in the previous sections was tested with a simulator of an ultrasonic range sensor (ASSIM, see [Kröse89]). After the robot takes a sensor measurement \( s \) with the simulated acoustic sensor, it follows a path to explore the (local) environment with touch sensors. This path is then represented in an occupancy grid \( G \) and the network is trained with \( (s, G) \). The path consists of a series of connected straight lines, each at random angle. If a collision occurs, the path is abandoned and a new path is started at a random location. In fig. 3 the environment, the robot, and the size of the robot-centered occupancy grid are sketched on scale.
A network was used as defined in section 4.1 with one input neuron $N_{hid} = 20$, $N_{out} = 16 \times 16$, where the outputs neurons are organised in a regular 2D square lattice. For the activation function $\mathcal{F}_{out}$ we used the identity function and for $\mathcal{F}_{hid}$ we used a Gaussian function. Such network are commonly known as radial basis function networks. The bases $\Psi_j$ were evenly distributed over the input range. The widths $v_j$ were initialised at half of the distance between two adjacent kernels and were updated during training using error back-propagation and the generalised $\delta$-rule. The biases $\Theta_i$ were kept fixed at 0.5, representing ‘unknown’ occupancy. The weights $w_{ij}$ were also updated using the $\delta$-rule. In fig. 4 results are shown of the learned sensor models. The results clearly show the increasing width of the sonar bundle at higher range measurements. This bundle is also wider than one may have expected because the sonar sensor is a point-source on the center of the robot, while the touch sensors are mounted on the sides of the robot. Thus, if an obstacle is measured ‘in front’, a collision may very well occur more to the side. This shows that the network also learns to incorporate the size of the robot in the sensor model.

Furthermore, a comparison of these models with fig. 1 shows that the learned sensor model represents a much larger free space hypothesis. This can be interpreted as follows: if the sensor measures an obstacle ‘in front’ of the robot, the probability of an obstacle being present ‘on the sides’ is almost 0. This is consistent with the learning samples presented to the network: if an obstacle is in front of the robot, there is low probability of an obstacle being present on the sides, i.e. the robot does not often drive into corners. This is an example of how the network incorporates ‘knowledge’ of the environment. Different models would be obtained if the network were trained
Figure 4: The learned sensor model for the acoustic sensor. Shown are the occupancy values grid \( P(\text{cell } i \text{ not empty} | s) \) in the robot-centered grid for various range measurements \( s \). The position of the sensor is in the middle of the grid and it is aimed toward ‘the front’ of the robot, i.e. straight to the right.
while driving through corridors.

5 Future work

We are currently implementing the sense-and-drive algorithm on the MARIE robot, which is the robot modeled in the ASSIM simulator used for our experiments. We expect to report on the results with the real robot soon.

References


[Becker93] P. Becker, B. Dedieu and E. Masri, "Representing object/environment interactions using probabilities the brain in the box experiment".


