Multisensory data fusion with Self-Organizing Maps for robotics

eingereichtes
Hauptseminar
von

cand. ing. Anton Wolf

geb. am 26.06.1987
wohnhaft in:
Scharnhorststraße 11
80992 München
Tel.: 0175 1698297

Lehrstuhl für
STEUERUNGS- und REGELUNGSTECHNIK
Technische Universität München

Univ.-Prof. Dr.-Ing./Univ. Tokio Martin Buss
Univ.-Prof. Dr.-Ing. Sandra Hirche

Betreuer: M.SC. Cristian Axenie
Beginn: 22.10.2012
Abgabe: 18.01.2013

Abstract

In this paper an alternative to the commonly used probabilistic methods for data fusion problems is presented. The alternative algorithm is based on Self-Organizing Maps, which are inspired by the cortical processing and will be applied on a real world sensor fusion scenario. The SOM algorithm is an unsupervised learning method that can find structure in the data without an external teacher. This is useful in handling uncertainty and noise in a data fusion problem. The applied model differs from other SOM algorithms in terms of being able to learn the mapping between different sensors coordinate systems, consider sensor reliabilities over the input space and use the mapping and the reliabilities to fuse in the different data. The results show, that using a SOM a good level of flexibility and a good computational cost can be achieved for a typical sensor fusion scenario of robot localization.

Zusammenfassung

In diesem Artikel wird eine Alternative zu den häufig verwendeten, auf Wahrscheinlichkeiten basierenden Methoden für die Datenfusion beschrieben. Der alternative Algorithmus basiert auf der Methode "Self-Organizing Maps", welche durch die Prozesse in der Kortex inspiriert wurde und hier auf ein reales Sensorfusion-Szenario angewendet wird. Der SOM Algorithmus ist ein unbeaufsichtigter Lernprozess, der ohne äußere Einwirkung Strukturen in den Daten findet. Diese Eigenschaft ist nützlich beim Umgang mit Unsicherheiten und Rauschen bei der Datenfusion. Das verwendete Modell unterscheidet sich von anderen SOM Algorithmen dahin gehend, dass es lernt, verschiedene Sensorkoordinatensysteme zu vereinen, räumlich verteilte Sensorzuverlässigkeiten berücksichtigt und beides für die Sensorfusion benutzt. Die Ergebnisse zeigen, dass in einem typischen Sensorfusion-Szenario zur Lokalisierung eines Roboters durch die Verwedung einer SOM eine hohe Flexibilität und ein geringer Rechenaufwand erreicht wird.

Contents

Table of symbols

Symbol	Description
x_k	state vector
z_k	state observation
υ_k	random variable describing the uncertainty in the evolution of states
u_k	control vector (for state computation)
ω_k	random variable describing the uncertainty in the observation
X^k	history of states
U^k	history of control inputs
Z^k	history of state observations
F_k	contribution of states to state transition
B_k	contribution of controls to state transition
G_k	contribution of noise to state transition
H_k	contribution of states to the observation
D_k	contribution of noise to the observation
w	weight of SOM units
$\Phi(t, k, i)$	neighborhood function
$r(t, i)$	radius
μ	learning rate
σ	standard deviation
d	distance

1 Introduction

During the industrialization engineers developed machines, which took very exhausting and dangerous jobs from the workers. In the 20th century the automotive industry invented the conveyer belt. Nowadays robots are used, which could be programmed to make in a little amount of time a lot of work. The goal of the 21th century industry is to develop autonomously working robots, which take intelligent decisions with increased flexibility. To reach this goal the robots must have a very precise knowledge of their environment. This means they have to be equipped with a bundle of different sensors. Two advantages of using multiple sensors are data redundancy and complementary information, in the sense that more sensors can measure the same quantity allowing a more robust measurement or they can combine their measurements to get a global and more precise representation of the sensed quantities. To handle the different information in a synergistic and efficient way great effort has to be invested in the fusion of data. The state of the art methods in current applications is based on probabilistic reasoning. Although the probabilistic approach is very powerful and widely used it has some drawbacks, that should be considered when designing a system. The first aspect is complexity (the need to specify a large number of probabilities to be able to apply probabilistic reasoning methods correctly). A second aspect is inconsistency (the difficulties involved in specifying a consistent set of beliefs in terms of probability and using these to obtain consistent deductions about states of interest) and the uncertainty and precision of the models (probabilities specification for unknown quantities). An alternative approach comes from neuroscience, mimicking the way our cortex processes and integrates information from the senses. The SOM is an unsupervised learning mechanism, based on competitive learning using a neural network structure that might be used also for data fusion problems due to its interesting capabilities of clustering, dimension reduction while preserving the relationships in the input data (e.g. links between the coordinate systems of the different sensors inputs to the network).

In chapter 2.1 the motivation for using multisensor data fusion for robotics is explained in more detail. Chapter 2.2 describes theoretically the probabilistic methods based on Bayes Network (Kalman Filter and the Sequential Monte Carlo Method) and the Self-Organizing Map as a biologically inspired method. Finally in chapter 2.3 a method based on Bayesian network as a representative of the probabilistic methods and the Self-Organizing Map are applied on a fictive test case scenario. The chapter ends with a comparison of both, where the advantages and disadvantages are highlighted. Chapter 3 completes the paper with a conclusion.

2 Problem description and analysis

2.1 Motivation of using multisensory data fusion for robotics

One of the main research topics in future will be to develop autonomously working robots. Therefore, the robot must have a precise knowledge of the environment. To provide a robust and complete description of the environment a number of different sensors is needed. The following chapter introduces the formalism of probabilistic data fusion methods and of Self-Organising Maps. Furthermore, a test scenario is proposed in which two data fusion methods (Bayesian network and SOM) are used and compared using multiple terms.

2.2 Methods used for multisensory data fusion

The goal of this paper is to discuss the application of a Self-Organizing Map on a data fusion problem. In order to extract the advantages and disadvantages of existing methods with respect to the alternative biologically inspired method a formal description is given.

2.2.1 Probabilistic methods

The general probabilistic problem of data fusion is to find the posterior density given the recorded observations Z^k, the control inputs U^k and the initial state x_0:

$$P(x_k \mid Z^k, U^k, x_0) \tag{2.1}$$

The described probabilistic methods are specific implementations of Bayes' rule. In general the recursive Bayes' rule provides a means to perform inference about an object or environment of interest described by a state x given an observation z. It can be derived from the chain-rule of conditional probabilities:

$$P(x \mid Z^k) = \frac{P(z_k \mid x)P(x \mid Z^{k-1})}{P(z \mid Z^{k-1})} \tag{2.2}$$

Given a sensor model $P(z_k \mid x_k)$ the multisensory data fusion problem of 2.1 can be solved recursively with the observation update step and the time update step:

$$P(x_k \mid Z^k, U^k, x_0) = \frac{P(z_k \mid x_k)P(x_k \mid Z^{k-1}, U^k, x_0)}{P(z_k \mid Z^{k-1}, U^k)} \tag{2.3}$$

$$P(x_k \mid Z^{k-1}, U^k, x_0) = \int P(x_k \mid x_{k-1}, u_k) \times P(x_{k-1} \mid Z^{k-1}, U^{k-1}, x_0) dx_{k-1} \tag{2.4}$$

The Kalman Filter

The Kalman filter minimizes the mean-squared error of the parameter of interest. This means it is rather an average than a most likely value. A state model description and an observation model has to be specified:

$$x_k = F_{k-1}x_{k-1} + B_{k-1}u_{k-1} + G_{k-1}v_{k-1} \tag{2.5}$$

$$z_k = H_{k-1}x_{k-1} + D_{k-1}w_{k-1} \tag{2.6}$$

The algorithm is a recursive loop, where first the actual state is predicted and then the observation is updated. If 2.5 and 2.6 are linear and the noise functions $v(t)$ and $\omega(t)$ are Gaussian distributed, the Kalman filter finds the exact Bayesian filtering distribution [con12].

Advantages and Disadvantages for data fusion applications

+ Wide variety of different sensors due to the explicit description of process and observation.
+ The actual weight of each sensor can be seen easily out of the uncertainty.
- The process and observations noises $v(t)$ and $w(t)$ must not have skewed distributions.
- When filtering nonlinear systems, the covariance and gain matrix must be computed online, which means a great amount of computation.
- When filtering nonlinear systems the models are linearized around a nominal trajectory. The prediction must always be close to the true state, otherwise the filter will become poorly matched.
- Due to the linearization of the model the filter must be accurately initialized to ensure valid results.

Sequential Monte Carlo Methods

The distribution is described as a set of state space values x_i and normalized weights w_i, where $\sum_i w = 1$. The probability density function can be defined in the form

$$P(x) \approx \sum_{i=1}^{N} w_i \delta(x - x_i) \qquad with\ w_i \propto \frac{P(x_i)}{q(x_i)} \tag{2.7}$$

The importance density $q(x)$ defines how many state space values will be chosen. The following steps are taken recursively:

1. **Time Update**: Given the N_k support values x_{ik} and the samples ω_{ik} (drawn from the known probability density $P(\omega_k)$) a new set of support values can be calculated with the process model $x'_{ik} = f(x_i, \omega_i)$. This is repeated N times.

2. **Observation Update**: Given the observation z_k a likelihood of each support value has to be calculated $\Lambda(x_{ik}) = P(z_k = z_k \mid x_k = x_{ik})$.
3. **Importance Resampling** An estimate of the effective number of particles $N_{eff} = \frac{1}{\sum_i (\omega_{ik})^2}$ must be computed. The Sampling Importance Resampling (SIR) algorithm resamples at every cycle so that the weights are always equal.

Advantages and Disadvantages for data fusion applications

+ Applicable for non-linear state transition and observation models.
– The transition and the observation models must be enumerable and in a simple parametric form.
– Only suitable for state space of low dimensions, because the number of required samples increases exponentially with the dimension.

2.2.2 Biologically inspired method - Self-Organizing Map

General Description From a data processing point of view the method of Self-Organizing Map is an unsupervised learning algorithm, that maps high dimensional data spaces to low dimensional, typically 2 dimensional spaces. Dr. Teuvo Kohonen, a professor at the Academy of Finland, published in 1982 his paper "Self-organized formation of topologically correct feature maps" and counts as the inventor of computational algorithms based on Self-Organizing Maps (SOM) - so called Kohonen networks. The method is inspired by neural networks. A good example to show how neural networks operate is the process of visual impressions [RV96]. The 3-dimensional visual field of the eyes is projected in the 2-dimensional planar cortex of the brain. This "mapping" is not genetically determined, but it is dependent on your visual capability. This means the sizes of the regions in the cortex are chosen respectively to the importance and the accuracy of the input it gets from the eyes. SOMs are working in the same manner. An arbitrarily chosen vector out of an input space has to be presented to the SOM at each time step. The SOM searches for the vertex which resembles the input most and adapts the map, so that the vertex and the surrounding vertices will be spatially closer to the input. An internal map arises, which represents the input space in a reduced dimension and complexity but maintains the topology relation in the input space. But how does the algorithm work in detail? The following paragraph gives a mathematical answer.

Formal Description First we have to introduce the constituent parts of Kohonen networks. Consider a grid consisting of m vertices, called units, and the edges between them (see 2.1). Each unit has a weight vector $w_1, w_2, ...w_m \in \mathbb{R}^n$, where n is the dimension of the input vector. The input vectors $z_k \in \mathbb{R}^n$ are arbitrarily chosen from the input space.

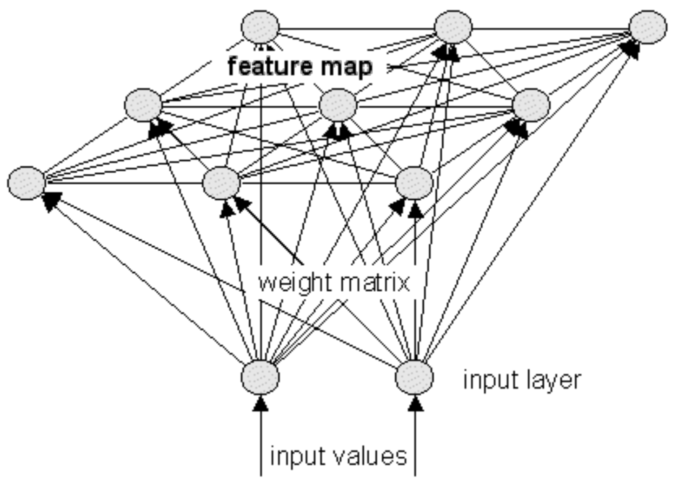

Figure 2.1: Assigning an input vector to the grid units (see [Jev])

Table 2.1: Different types of neighborhood radius functions.

Name	Description	Formula
bubble	r(t,i)= $\begin{matrix} \alpha(t)\ if\ i \in N_c \\ 0 \quad if\ i \notin N_c \end{matrix}$	Where $\alpha(t)$ is some monotonically decreasing function of time. It defines a range around a vertex. Every vertex inside the range is updated in the same way, every vertex outside is ignored.
linear	$r(t) = \frac{A}{B+t}$	Where A and B are constants. It is advisable to use the inverse-time type function with large maps and long training runs, to allow more balanced fine tuning of the reference vectors.
Gaussian	r(t)=r$_0e^{-\frac{t\cdot\log(\Phi_0)}{N}}$	Where N is the total number of iterations and Φ_0 is the initial neighborhood radius.

Table 2.2: Sequence of the SOM algorithm

```
Selection of units' number n, initial neighborhood radius $\Phi_0$, learning rate $\mu_0$
and maximum number of iterations maxiter

Randomly initializing of the weights $w_1, w_2, ..., w_m$

Calculating the time constant:
$\lambda = \frac{N}{\log(\Phi_0)}$

for $(i = 1; i < maxiter; i++)\{$

    Selecting input vector $z_k$

    Finding unit $u_{BMU}$ with minimal euclidean distance to $x_k$:
    $dist(k, i)|_{min} = \sqrt{\sum_{i=0}^{n}(z_k - w_i)^2}|_{min} = dist(k, BMU)$

    for $(j = 1; j < m; j++)\{$

        Calculating neighborhood kernel:
        $\Phi(t, j, BMU) = e^{-\frac{dist(j,BMU)^2}{2r(t)^2}}$

        Updating weighting vectors:
        $w_j(t + 1) = w_j(t) + \mu(t)\Phi(t, j, BMU)(x_k - w_j(t))$
    $\}$

    Updating neighborhood radius and learning rate:
    $r(t + 1) = r_0 e^{-\frac{t}{\lambda}}$
    $\mu(t + 1) = \mu_0 e^{-\frac{t}{\lambda}}$
$\}$
```

The edges between the units are represented by the neighborhood function $\Phi(t, k, i)$, which specifies the strength of coupling between unit i and unit k. The accuracy of the algorithm mainly depends on the choice of the neighborhood radius function. There are different types of function, but all share the feature, that they shrink in distance over time (see table 2.1).
Furthermore, a learning rate μ is needed, which also decreases with time. This parameter changes the time the algorithm needs to finish. The sequence of the algorithm can be seen in table 2.2.

Advantages and Disadvantages for data fusion applications

+ It is easy to **expand the input space**. E.g. in Sensor Data Fusion applications a new sensor can be easily integrated by adding the sensor data to the input space.
+ The algorithm is **unsupervised**. This means no previous knowledge of the system is needed.
– **I**t provides poor classification when it is used for high performance duties without any adjustment.

2.3 Applying Bayesian Network and SOM for sensor fusion

2.3.1 Test Case Scenario - Robot Localization

Let's assume a real world scenario, in which multisensory data fusion methods are necessary in order to extend the environment perception. In an atomic power plant an explosion occurred. Due to danger of radiation the whole plant got evacuated. Now a robot is used to examine the damage and to find survivors. Neither the robot can be controlled via wire, because there is the risk of getting stuck, nor it can be steered via WiFi, because it is out of range. Thus the robot has to act autonomously given a territorial map. The sensors have different (dis-)advantages:

- **Global Position Sensor - GPS:** Under good circumstances it provides a quite exact approximation of the robot's position. But in areas with reflecting obstacles (like trees) the accuracy of the data decreases and inside of buildings the GPS-Module is not available.
- **Camera - CAM:** A camera with auto-focus and the ability to tilt delivers pictures of the environment. But due to the explosion parts of the plant are out of electricity. Without light the camera is useless.
- **Odometry Sensors - ODOM** Wheel encoders provide the position of the robot relative to a starting point. The path through to the plant is wet and slippy, so there is no guarantee of getting correct information. Furthermore, it is sensitive to errors due to the integration of velocity measurements over time.

- **Sonar Sensor - SONAR** Independently of the light conditions this sensor sends exact distances to obstacles. Outside of buildings it might be useless, because the range is very limited. And inside reflections of broken walls might be problematic.
- **Inertial Measurement Unit - IMU** The accelerometer is noisy and filtered with a low-pass. Thus there will be a latency which slows down the response time of the measurement. The gyroscope is sensitive to errors, because it integrates the angle rate. Another problem is that it drifts away from the angle with the temperature.

All sensors are affected by independent noise described by Gaussian **Noise** $N(d, \sigma_k) = \frac{1}{\sqrt{2\pi\sigma_k^2}} e^{-\frac{d^2}{2\sigma_k^2}}$. The standard deviation of the Noise σ_k (and thus also the reliability of the sensor) is variable in input space. A training set consisting of measurements of the environment with the same sensor equipment and the corresponding real robot location is available.
The challenge in this scenario is the localization of the robot. Therefore, an accurate fusion of the sensors' data is needed. In the following paragraphs a Bayesian data fusion method is applied and subsequently the SOM based method. Afterward both methods are compared with each other.

2.3.2 Bayesian method

In chapter 2.2.1 two probabilistic methods are introduced. But we use another method, called Bayesian network, because it is less complex. It calculates the probability densities (PDF) of the sensors based on the sensors' observations. Then it merges the sensors' observations to one unified estimation.
The description of the dynamic Bayesian network is mainly out of [SB97] and [Pea88]. Figure 2.4 shows the directed graph of the dynamic Bayesian network suited for our case. The edges represent the conditional probabilities of inferencing of one entity (sensor observation $O_i = [x_i, y_i, \varphi_i]^T$) given the other entity, so called a priori probability. The posterior probability can be computed with Bayes Rule 2.2 (see green shades in figure 2.4). This means, if the knowledge of a few sensors' estimation and reliabilities is present, the current belief in the existence of all entities can be computed. But how do we get the reliability of each sensor's estimation? The answer is supervised learning. Using the given training set contingency tables of the sensors' estimations and the actual position can be drawn. The conditional probability tables can be calculated from the contingency tables and finally the measurements' reliabilities can be derived. Then the observations are merged to get one unified estimation of the robot's actual position $P_{robot} = (x, y, \varphi)^T$:

$$P_{robot} = \frac{\sum_{i=1}^{5}[P(O_i) \cdot O_i]}{\sum_{i=1}^{5}[P(O_i)]} \tag{2.8}$$

For our test case the Bayesian network must be extended. To consider the temporal aspect an history node is added to each sensor (see blue entities in figure 2.4). Furthermore, reasoning strategies must be disregarded by allowing dynamic structure changes. E.g. if the reliability of the GPS sensor is higher than a threshold, the vertex $S = outside$ is added. This means it is very probable that the robot is outside where GPS reception is very good.
This method based on Bayesian Networks should fit our test case. Next a biologically inspired method based on SOM is applied to the same robot localization scenario. Afterward both are compared and conclusions are drawn.

2.3.3 SOM based method

The SOM introduced in 2.2.2 must be modified to fit the test case. The data flow can be seen in figure 2.2. Before the fusion of data can be started, the sensors' raw data must be processed in such way, that all sensors $k = [1; 5]$ provide same units. Table 2.3 gives a brief summarization of the sensors' preprocessing procedures. The output of the preprocessors $z_k \forall k$ are bundled to one SOM input vector $z = [z_1, z_2, z_3, z_4, z_5]^T$.
Referring to [Koh01] a 2 dimensional grid with a rectangular shape will be used. Because the code should be as generic as possible, the number of units is variable (see [LBD+10]). If the distance between the BMU and the input is larger than a predefined limit *min_dist*, a new unit is created. Additionally an upper limit for unit's number *max_no_units* must be defined. If the number is reached, cells will be removed. Figure 2.3 illustrates the procedure. The radius r will be decreased linearly. The learning rate σ_h of the update strength is always a fifth of the radius. The following adaptions are basically out of [BWW12]. The distance function will not only compute the Euclidean distance between input z and unit u, but it will use the knowledge of the noises' intensities (and thus the sensors' reliabilities) to weight the input's components:

$$dist(z, u) = 1 - \prod_{i=1}^{n} N(z_i - w_{uk}, \sigma_{uk}) \tag{2.9}$$

The question is, how to make the SOM units learn the vector σ. The solution is deriving σ from the distances between the sensors' estimates from each other. Therefore, the variance of the difference between estimate of sensor i and sensor j is

$$\nu'_{i,j} = \nu_{i,j} + s'_{ij}[(m_i - m_j)(x_j - x_i)]^2 \tag{2.10}$$

So each SOM unit can compute the ν_{ij} between all sensors' guesses and store them in a matrix (see 2.11).

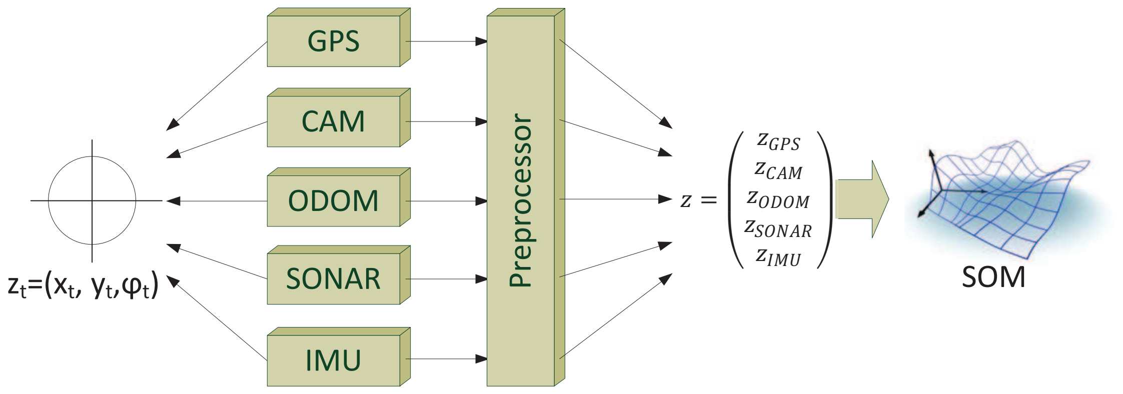
GPS
CAM
ODOM
SONAR
IMU
Preprocessor
$z_t=(x_t, y_t, \varphi_t)$
$z = \begin{pmatrix} z_{GPS} \\ z_{CAM} \\ z_{ODOM} \\ z_{SONAR} \\ z_{IMU} \end{pmatrix}$
SOM

Figure 2.2: Data flow of SOM based fusion algorithm suited for test case

Table 2.3: Summarization of the sensors' preprocessing steps

Sensors	Raw Data	Preprocessing Steps	Output
GPS	(x, y, z)	The z-coordinate is removed	(x, y, φ)
Camera	frames	Image processing software estimates relative movements of the robot + $(x, y, \varphi)_{init}$	(x, y)
Odometry	v_x, v_y	Integrating robot's longitudinal and lateral speed provides the relative movement of the robot + $(x, y, \varphi)_{init}$	(x, y)
SONAR	$d_{obstacle}$	Robot's relative distance to e.g. a wall can be estimated + obstacle map	(x, y)
IMU	$a_x, a_y, a_z,$ a_Φ, a_Θ, a_Ψ	Integrating robot's accelerations provides the relative movement of the robot + $(x, y, \varphi)_{init}$	(x, y, φ)

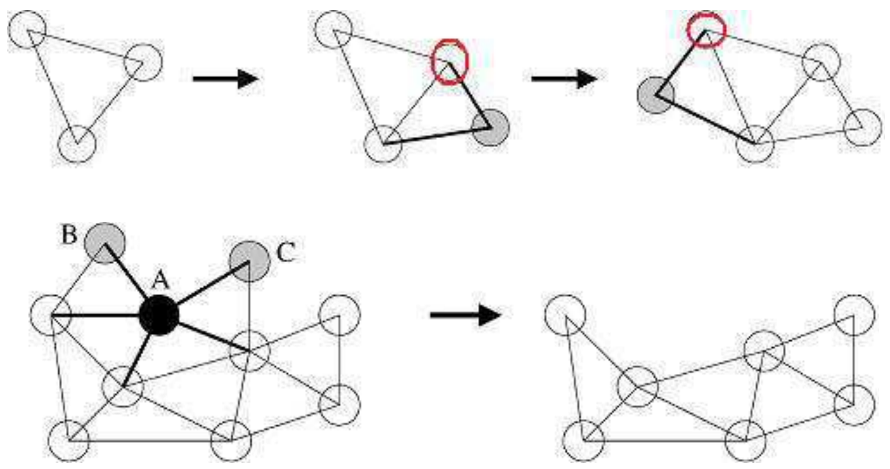

Figure 2.3: Left: The new inserted cell is in gray and the winner cell in red; Right: Cell A is removed. Cells B and C are left behind with only one connection and are therefore also removed (see [LBD+10]).

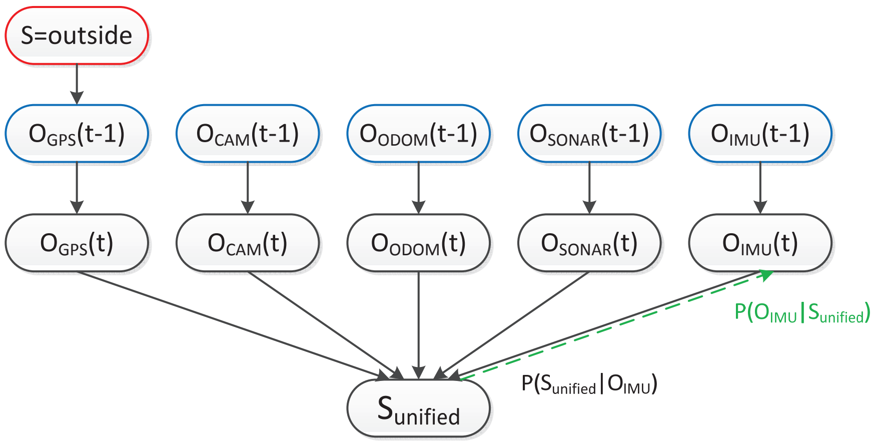

Figure 2.4: Dynamic Bayesian Network suited for test case

Table 2.4: Sequence of the SOM algorithm

```
Selection of units' number $n$, initial neighborhood radius $r_0$, learning rate $\mu_0$ and
maximum number of iterations maxiter

Randomly initializing of the weights $w_1, w_2, w_3, w_4, w_5$ and variances $\nu_{ij}$

for $(i = 1; i < maxiter; i++)\{$
        Selecting input vector $z_k = [z_{k1}, z_{k2}, z_{k3}, z_{k4}, z_{k5}]^T$

        Calculating the standard deviation vector $\sigma_{ui}$ for each unit and sensor

        Finding unit $u_{BMU}$ with minimal distance to $z_k$:
        $dist(z_k, i)|_{min} = 1 - \prod_{i=1}^{n} N(z_k - w_{ui}, \sigma_{ui}) = dist(z_k, u_{BMU})$

        if$(dist(x_i, u_{BMU}) < min\_dist)\{$
            Insert a new unit
        $\}$

        if$(|u| > max\_no\_units)\{$
            Remove unit(s)
        $\}$

        Do for all $n$ units with distance less than radius $r$:
        for $(j = 1; j < n; j++)\{$
            Calculating update strength:
            $s'_{ij} = \frac{1}{\sqrt{2\pi\sigma_h^2}} \cdot e^{-\frac{dist(u_{BMU}, u_j)^2}{2\sigma_h^2}}$

            Incrementing the counter values:
            $c'_{ij} = c_{ij} + s'_{ij}$

            Updating weight vectors:
            $w'_{ij} = \frac{1}{c'_{ij}}(c_{ij} w_{ij} + s'_{ij} x_i)$

            Updating variances:
            $\nu'_{ij} = \nu_{ij} + s'_{ij}[(m_i - m_j)(x_j - x_i)]^2$
        $\}$

        Updating radius $r$ and learning rate $\sigma_h$ of update strength
$\}$
```

$$\nu = \begin{pmatrix} 0 & \nu_{1,2} & \dots & \nu_{1,k} \\ \nu_{2,1} & 0 & \dots & \nu_{2,k} \\ \vdots & \vdots & \ddots & \vdots \\ \nu_{k,1} & \nu_{k,2} & \dots & \nu_{k,k} \end{pmatrix} \quad \mathit{with}\ k = 5 \tag{2.11}$$

Every SOM Unit is now a tuple $u = (w, c, \nu)$ with $w \in \mathbb{R}^{5\times3}$ as the unit's weight vector, $c \in \mathbb{R}$ as a generalized counter and the variances matrix ν.
With these information an estimation of the standard deviations can be made

$$\sigma_{i1}^2 \approx \frac{1}{2c}\nu^s = \frac{1}{2c}(\nu_{i1,i2} - \nu_{i2,i3} + \nu_{i3,i4} - \dots + \nu_{ip,i1}) \tag{2.12}$$

for any sequence of an odd length $p \geq 3$ of integers $s = i_1, i_2, ..., i_p$ between 1 and the number of sensors k, where $i_j \neq i_{j'}, \forall j, j'$. Now an approximation for σ_k can be computed by choosing such a sequence s starting with k and combining an odd number of elements of ν_{ij} as above. Table 2.4 shows the algorithm.
Finally one SOM grid suited for the test case is delivered. The map can be visualized by drawing a coordinate system with the dimensions of the plant‘s territorial map. Each SOM unit can be drawn as an arrow with starting point (x, y), direction φ and length corresponding to the reliabilities of the sensors. Next, a comparison of the two presented methods is given for the considered scenario.

2.3.4 Comparison of Bayesian and SOM based method

- **Computational Cost**
  The calculation time is dependent on the number of SOM units. But the chosen algorithm allows dynamic changes in the number of units, so this problem should not occur.

  In Bayesian networks the problem of reaching exact inference is NP-hard. Thus to get an infinitesimal small error in the result means to spend infinite long time in computation.

- **How generic is the algorithm?**
  One of the biggest advantage of SOMs is, that they are unsupervised. This means it learns autonomously from the input data and thus it is capable to adapt its mapping in case of changing conditions by itself.

  Bayesian methods are supervised algorithms. If the environment is changed, in our case e.g. a sensor is replaced with a more reliable one, the model must be adapted to guarantee the accuracy of the algorithm.

- **Precision in describing the output**
  The accuracy of fusing multiple sensors using a SOM is highly up to the choice of neighborhood function $\Phi(t, k, i)$, radius function $r(t, i)$ and learning rate $\mu(t)$. If the functions are improper the learning phase will probably result in a skewed map (see [RV96], p. 399).

The choice and the precision of the model determines the performance of a Bayesian network most. This means the more effort and the more knowledge is put into the model, the more accurate the predictions are.

- **Number of sensors**
  Due to the computation of the standard deviation the algorithm used here requires 3 sensors at minimum. But it can easily handle changes in the number of sensors. New sensors can be added simply by putting their outputs into the SOM's input space.

  In a Bayesian network and all other probabilistic methods the number of sensors is arbitrary, but the model must suite the environment. This means, when a sensor is added, the model must be adapted and a lot of time must be spent in rewriting the implementation of the algorithm.

- **Ability to embed prior information**
  Since neural network based methods like SOM are unsupervised there is no chance to directly embed empiric knowledge in the algorithm. The entire information comes from the input space.

  Bayesian networks are supervised methods. So previous known information can be considered in the construction of the model.

- **Handling of shifted and scaled coordinate systems**
  The SOM algorithm is able to learn which coordinates in the combined vector z go together and how to convert between the individual sensors' coordinate systems. Thus it works even if the sensory coordinate system are shifted wrt. each other or scaled moderately.

  The used method is not able to handle sensors with shifted or scaled coordinate systems automatically. It is possible to consider it in the model, but when the shift or scale is not known previously or when it changes during sampling, the method will not provide accurate output.

- **Handling of variable reliabilities**
  The SOM makes an estimation of the noises' intensities based on the differences of the guesses. Using this estimation the algorithm is capable to weight the input considering the actual reliability of the sensors.

  The used Bayesian method is able to derive the reliabilities of the sensors from the training set. But it is not adapting the probabilities during sampling. There are more complex algorithms taking this drawback into account (see [Thr02]).

After formally presenting the chosen method for multisensory fusion and after an extended comparison with existing methods some conclusions must be drawn that will give a hint on when is better to choose one method over the other.

3 Conclusion

This paper shows, that a SOM based model can be applied as an alternative to probabilistic methods on data fusion problems. One benefit is, that it is unsupervised and therefore it is able to learn from the input space by itself. Furthermore, it has less computational cost than a probabilistic method. But the introduced model also has extensions to Kohonen's original model. The number of SOM units can be changed dynamically during the learning phase. The model can deal with sensor coordinate systems, which are shifted and scaled. Moreover, the map considers sensors' reliabilities, which are variable over space and time. All advantages and extensions are described theoretically and implemented in a model, which is applied on a fictive test case scenario for robot localization.

Compared to probabilistic models the SOM has a few drawbacks. One is that a minimum of three sensors is required for learning sensor reliabilities. Another is that previous known information or empiric knowledge can not be implemented in the construction of the map. Especially the last point might be a reason to refuse the usage of a SOM. Bauer et al. already noticed that problem: "Researchers have argued that the brain may and should take into account not only the general reliability of sensory modalities, but also situational cues" ([BWW12], S.8). E.g. the brain may take into account, that in a foggy environment the vision is weighted less. In future further researches in this direction are necessary.

Wan and Fraser deal with another method to implement SOM, known as multiple Self-Organizing Maps [WF99]. Instead of one they use multiple maps. Especially regarding the discrimination and ordering this model provides proceedings. Leivas et al. proposes the use of topological maps in order to provide a method of SLAM feature, that treats better the problem of inaccuracy of the current systems [LBD$^+$10].

The theoretical application of a SOM in our test case identified its advantages and disadvantages. The decision whether to choose a probabilistic method or a SOM highly depends on the problem. When empiric knowledge about the model and the environment is available, then maybe a probabilistic method should be preferred. Otherwise, a SOM based solution should at least be taken into account.

List of Figures

List of Tables

Bibliography

[BWW12] Bauer, J. ; Wecer, C. ; Wermter, S.: A SOM-Based Model for Multi-Sensory Integration in the Superior Colliculus. In: *Neural Networks (IJCNN), The 2012 International Joint Conference on*, 2012, S. 1 –8

[con12] contributors, Wikipedia: *Particle filter.* `%http://en.wikipedia.org/w/index.php?title=Particle_filter&oldid=516942931`. Version: October 2012

[Jev] Jevtic, N.: *Interactive Tutorial on Neural Networks.* `%http://sydney.edu.au/engineering/it/~irena/ai01/nn/som.html`

[Koh01] Kohonen, T.: *Self-Organizing Maps.* Springer, 2001

[LBD+10] Leivas, G. ; Botelho, S. ; Drews, P. ; Figueiredo, M. ; Haeffele, C.: Sensor fusion based on multi-self-organizing maps for SLAM. In: *Multisensor Fusion and Integration for Intelligent Systems (MFI), 2010 IEEE Conference on*, 2010, S. 139 –143

[Pea88] Pearl, J.: *Probabilistic Reasoning in Intelligent Systems: Networks of Plausble Inference.* Morgan Kaufmann Publisher, Inc., 1988

[RV96] Rojas, R. ; Varga, P.: *Neural Networks: A Systematic Introduction.* Springer, 1996

[SB97] Singhal, A. ; Brown, C.R.: *Dynamic Bayes net approach to multimodal sensor fusion.* `http://dx.doi.org/10.1117/12.287628`. Version: 1997

[Thr02] Thrun, S.: *Probabilistic robotics.* ACM, 2002

[WF99] Weijian, W. ; Fraser, D.: Multisource data fusion with multiple self-organizing maps. In: *Geoscience and Remote Sensing, IEEE Transactions on*, 1999, S. 1344 –1349