Chapter 9

Sensor Data Fusion

9.1 Introduction

In the introductory chapter it was discussed how important it is for an autonomous system to be able to sense its own environment. Most often the system cannot rely on a single sensor to provide sufficient information. Typically, sensor measurements contain noise and can be *erroneous* or *incomplete*. In this section we will propose how data from multiple sensors can be *fused* to overcome this problem.

Many sensor measurements are **uncertain**, e.g. an ultrasonic sensor which measures a distance is uncertain about the angle from the sensor at which this distance is measured, as sketched in 9.1 The sonar emits a sound-pulse which is reflected by an obstacle somewhere within the cone. When the reflected pulse is received by the sonar, it can accurately estimate the distance to this object using the "time-of-flight" of the pulse. However, it cannot determine at what angle within the cone the pulse was reflected.

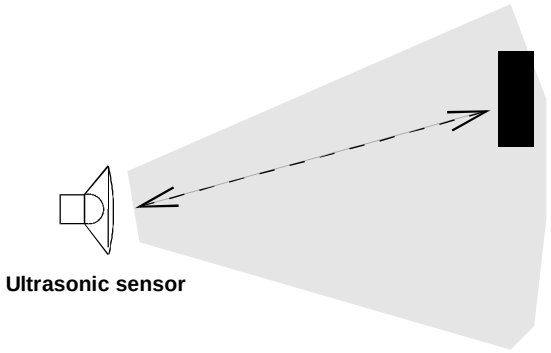


Figure 9.1: *Due to the wide opening angle of the acoustic sensor, the sensor is uncertain about the exact position in the sensor 'cone' of the obstacle to which the distance is measured.*

Thus, there is uncertainty about the angle at which the obstacle was measured which is about as large as the opening angle of the sonar.

In the previous chapter it was discussed that sensor measurements are prone to **measurement errors**. These errors are mainly due to the fact that the true physical sensing device, the "transducer", does usually not operate in practice as we have modeled it. To interpret the measurements from the transducer a model of the sensor is needed; but the true physical operation of the transducer is most often too complex to model. E.g., the systematic error component of the noise in electrical circuits and the fluctuations in air

temperature are typically too complex to include in a model of the sensor. Therefore, the interpretation of the sensor measurements often differs from the true physical value of the parameter that is measured. Let it be noted that this measurement error can also be interpreted as "uncertainty" in the sensor measurement: since the sensor is prone to errors it is uncertain about the true value of the parameter. This explains why the terms "uncertainty" and "erroneous" are often used interchangeably in sensor data fusion texts. In this text, however, we would like to use the two terms separately, since they clearly indicate different causes of uncertainty.

The term "incompleteness" means that a single sensor cannot sense all information, e.g. a single fixed camera cannot view the entire room a mobile autonomous system is moving in. Multiple views are needed to form a complete view of the room. Also, views maybe incomplete because obstacles are *occluding* the view of a sensor. Incompleteness is sometimes treated as total uncertainty but again we wish to make a distinction. In the case of uncertainty, a quantity is measured but the sensor is uncertain about this measurement. In the case of incompleteness a (part of) a quantity is not measured at all.

Because information from sensors is incomplete, erroneous and uncertain, it is essential that the system *fuse* redundant information from multiple sensors. The word 'redundant' means that multiple sensors should measure the *same* information to reduce the errors and uncertainty in this information. In this section we will set forth our approach to *sensor data fusion*, the fusion of information from disparate sensory sources. First an overview of sensor data fusion will be given. We will then introduce an architecture for a sensor data fusion system, which is generally recommended for such a system. We will motivate the choice of this architecture, we will discuss the choice of suitable sensors for such architecture and we will discuss suitable *internal representations* that can be used in the architecture.

Throughout this section, sensors are always taken to be *virtual sensors*, i.e. a sensor is a device, which measures the world of the autonomous system, either the external environment or the internal body of the system, and converts this measurement by giving estimates of the parameters in the internal representation.

The device performing the actual measurement of the robot's environment is called transducer or sensing device. This distinction is best remembered by the following rule of thumb: the transducer (or sensing device) is the part, which you can hold in your hand. In our terminology the sensor also includes lots of software which performs the conversion of the measurements of these transducers to the parameters of the internal model. Note that the translation of a raw measurement to an estimation of a parameter in the internal representation is called "conversion" here. In previous sections, this was called "interpretation" but here we prefer the term conversion since it conveys more clearly the sense that a value must be actively converted from one representation to another and parameter estimation must be performed. Further on in this section we will see that this conversion is an important part of the sensor.

In this section, we will use a sensor data fusion system for an autonomous mobile robot as a running example. In this case the (virtual) sensors are clearly coupled to actual

transducers, such as cameras, acoustic measuring devices or infrared devices. However, in our terminology an acoustic *sensor* does not only include the transducer, but also the *interpretation* component, which converts the measurement by giving estimates of the parameters in the internal representation. This terminology is used to stress the fact that the sensor data fusion system can be used in any autonomous system at any level of abstraction; in that case the same principles of sensor data fusion can be applied to a different set of sensors and a different internal representation.

Furthermore, it is important to notice that the field of sensor data fusion has only very recently begun to attract serious attention (in 1994 the first International Conference on Multisensor Fusion in Intelligent Systems was held). This indicates that the ideas put forth in this section are not standard approaches to sensor data fusion which have been generally acknowledged and used in the field for many years. It is merely an introduction to an approach of sensor data fusion of which we believe it will become the standard approach in the years forthcoming. It is the only structural approach to the problem and therefore fits very well in the structural approach to autonomous systems presented in this course.

9.3 Single sensor data fusion in time

We have already discussed one method of sensor data fusion in chapter 7: the Kalman filter. Kalman filtering enables us to fuse sensor data of a single sensor measured at successive time intervals. In this way we fuse the successive measurements of a single sensor to obtain a more accurate description of the environment of the robot than with a single measurement. This method can be easily applied when we have some parametric description of the environment. In that case we fuse the measurements by using them in the successive updating of the estimation of the parameters of the representation. Kalman filtering for sensor data fusion is described by Kröse et al.[9.3] and Vandorpe[9.4]. We will show the use of the Kalman filter for sensor data fusion by a characteristic example.

We will show in this example how to find the local environment of the robot from an ultrasound sensor mounted on a moving robot. The environment is represented as a two-dimensional map in which obstacles are represented by straight-line segments and points. A robot centered coordinate frame is used. Information about the displacement of the robot is obtained from encoders on the wheels of the vehicle and is used on-line to update the parameters of the lines. Furthermore new measurements from an ultrasonic sensor are fused with the model by updating the parameters. If a new data point falls on or within a certain range from a line segment, the new measurement is considered to be a part of the line and the line parameters (position and orientation) are updated using a Kalman filter. If the new measurement falls outside this range it is added to the set of points. If within the set of points a subset of collinear points is formed, a new line element is estimated and added to the set of line elements.

Conversion of the sensor data to the line representation

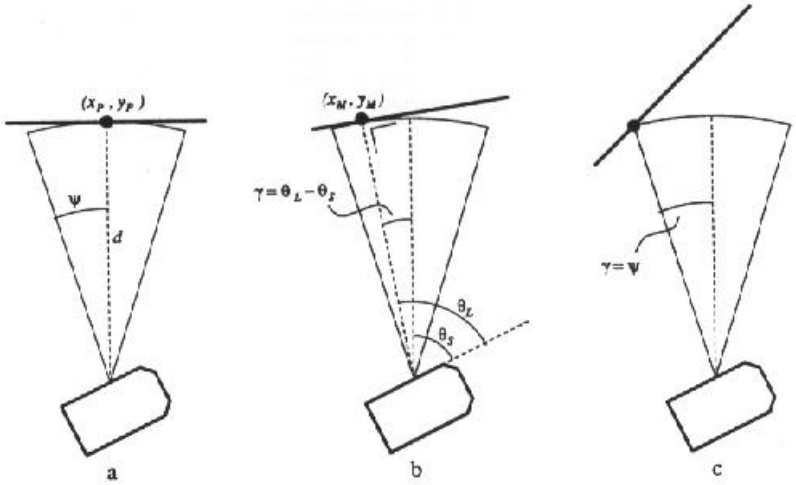


Figure 9.2 : *Relation between surface orientation and location of reflection*

Ultrasonic sensors measure the range to the obstacle point that first reflects their signal. Because of the large beam width of these sensors there is uncertainty about the real direction of the obstacle. Usually the point of reflection is assumed to be in the center of the beam. From the range measurement d and the orientation θ_S of the sensor on the vehicle, the location of reflection (x_P, y_P) can now be derived in the vehicle coordinate system together with an estimate of its variance $\sigma^2_{xy_P}$. This is illustrated in figure 9.2a.

This is correct when the sensor is perpendicular to the surface. However, the location of the first reflection will shift when the sensor is not perpendicularly directed towards the surface. As we have an estimate of the orientation of the line, this perpendicular point of reflection (x_M,y_M) can be calculated. (fig9.2b). In this way we can improve the estimate of the reflection point. When that calculated point should fall outside the center part of the beam, we assume that the point is at the edge of the center part. This corrected measurement (x_M,y_M) has to be fused with the estimates of the line segment we obtained so far.

Fusion through updating the line parameters

In the environment model a line is defined by a redundant set of parameters:

- x_L, y_L : the coordinates of the point on the line of which the uncertainty is minimal;
- θ_L : the surface (or line) normal;
- σ^2_{xyL} : the variance of x_L and y_L ;
- $\sigma^2\theta L$: the variance of θ_L;

Since the coordinate frame is robot centered, the coordinates of the lines and points in the model change when the robot moves. When the robot has moved a distance (Δx, Δy) and

turned over an angle $\Delta\alpha$ since the last measurement, the new values of the parameters x'_L , y'_L and θ'_L are given by :

$$x'_L = x_L \cos(-\Delta\alpha) - y_L \sin(-\Delta\alpha) - \Delta x$$

$$y'_L = x_L \sin(-\Delta\alpha) - y_L \cos(-\Delta\alpha) - \Delta y$$

$$\theta'_L = \theta_L - \Delta\alpha$$

The variance in the line parameters is updated according to:

$$\sigma^{2'}_{xy\,L} = \sigma^2_{xy\,L} + \sigma^2_{\Delta xy} + \max(x_L^2, y_L^2) \cdot \sigma^2_{\Delta\alpha}$$

and

$$\sigma^{2'}_{\theta L} = \sigma^2_{\theta L} + \sigma^2_{\Delta\alpha}$$

in which $\sigma^2_{\Delta xy}$ and $\sigma^2_{\Delta\alpha}$ are derived from the accuracy of the shaft encoders. This means that the variance in the line parameters increases during driving.

If a new measurement point P is a candidate to belong to a line, the corrected location of reflection (x_M ,y_M) is determined. First we have to consider whether the point could belong to the line. When the distance between (x_M ,y_M) and the line is smaller than a threshold, the point is considered to be a new observation of the line. From (x_M ,y_M) and the coordinates (x_L ,y_L) of the line, the new value θ_M of the surface normal according to the new measurement is obtained in two steps. First, the orientation of the line between (x_M ,y_M) and (x_L ,y_L) , β_M, is calculated:

$$\beta_M = \arctan\frac{y_M - y_L}{x_M - x_L}$$

β_M must be rotated over $\pi/2$ in order to obtain the surface normal:

$$\theta_L = \begin{cases} \beta_M + \dfrac{\pi}{2} & \text{if } 0 < L(\beta_M, \theta_S) < \pi \\ \beta_M - \dfrac{\pi}{2} & \text{otherwise} \end{cases}$$

The variance in the new estimates of the line parameters σ^2_{xyM} now has to be calculated (the variance in x_M and y_M is considered the same) and $\sigma^2_{\theta M}$ can be expressed as function of the noise in the measurement and the error in the estimation of the line orientation.

From these variances the Kalman gains can be determined:

$$K_{xy} = \frac{\sigma_{xyL}^2}{\sigma_{xyL}^2 + \sigma_{xyM}^2}$$

$$K_{\theta} = \frac{\sigma_{\theta L}^2}{\sigma_{\theta L}^2 + \sigma_{\theta M}^2}$$

The best estimation of the line parameters is now

$$x'_L = x_L + K_{xy}(x_M - x_L)$$

$$y'_L = y_L + K_{xy}(y_M - y_L)$$

$$\theta'_L = \theta_L + K_{xy}(\theta_M - \theta_L)$$

In this way the successive measurements are fused in the current estimate of the line.

9.4 Data fusion with multiple sensors

As discussed in the section 9.2, it is rarely the case that a single sensor can provide sufficient information for the reasoning component in the autonomous system. In this case, the system must *fuse* information from multiple sensors to obtain more complete and more accurate information about the world. In general three types of sensor data fusion are distinguished (see [9.1]):

- *complementary fusion*: fusion of several disparate sensors which only give *partial* information of the environment, e.g. fusion of several range sensors or camera sensors pointed in different directions. Obviously, this type of fusion resolves **incompleteness** of sensor data.

- *competitive fusion*: fusion of uncertain sensor data from several sources, e.g. a camera and a range sensor pointed at the same obstacle; through sensor data fusion the distance to the object can be obtained more accurately. This type of fusion is predominantly aimed at reducing the effect of **uncertain** and **erroneous** measurements. If a sensor is uncertain about the angle to a certain obstacle it can give a rough estimate of this parameter. This estimate is then refined with other estimates of the *same* parameter, which is a typical example of competitive fusion. Also, since the estimation of parameters is prone to errors, more accurate estimates can be obtained by fusing multiple measurements of the same parameter. Fusion can either be performed on several measurements from different sensors at the same time or on several measurements from the same sensor at different times (typically, fusion of a *series* of measurements from the same sensor). This example of "fusion-in-time" cannot reduce uncertainty or systematic errors, but it can reduce the random error component.

- *cooperative fusion*: fusion of different sensors of which one sensor relies on the observations of another sensor to make its own observations. E.g. when a touch sensor refines the estimated curvature of an object previously sensed by range sensors. This type of fusion is only mentioned here for completeness. It diminishes uncertainty, measurements errors, as well as incompleteness, but at another level: in our terminology, a touch sensing device and a range sensing device cooperating to give one single estimate of an abstract parameter (the curvature of an object) amount to one single virtual sensor! This point will be discussed further in the following sections.

The reduction of uncertainty is sometimes also seen as a form of *complementary* fusion. E.g., if one sensor can accurately estimate the distance to an object but is uncertain about the angle, this estimate can be *complemented* by another sensor that is not as uncertain about the angle (but may be uncertain about the distance to the object). However, since both sensors measure the same parameters, this is strictly speaking a form of competitive fusion.

Most sensor data fusion systems described in literature follow an application specific approach. For the specific task at hand, suitable sensors are sought and a fusion method is applied which is specific to these sensors and to that task only.

We will follow a more general approach because the sensor data fusion (SDF) system is to be applied at every level of abstraction in our hierarchical autonomous system (see figure 2.7), using all different kinds of (virtual) sensors. The SDF system introduced in this section is a *generic* system, which means that it can be to all different kinds of sensors and to all different kinds of tasks.

9.3 A sensor data fusion system

Much of the theory presented in this section is also discussed in the work of Durrant-Whyte, who has written a comprehensive book on the subject of sensor data fusion (9.1). We will first derive an architecture for a generic SDF system and we will then discuss how cooperative, competitive and complementary fusion are performed in this architecture; some examples will be given. We will see that the most important issue in such a system is the choice of a suitable representation. We will discuss several representations that are frequently used in practice and we will show how sensor data fusion can be performed in these representations. We will conclude this section with a discussion of how the virtual sensors for our SDF system should be chosen and we will review the advantages of the proposed architecture.

Introduction of the architecture

An architecture for a generic sensor data fusion system should be able to fuse data from sensors of many different modalities. E.g. a mobile robot would typically be equipped

with infrared measuring devices, acoustic devices and cameras. For all data from these disparate sensors to be fused it is necessary that they 'speak the same language'.

In other words, all sensors should first convert their measurements to a *common* representation before the actual fusion is performed.

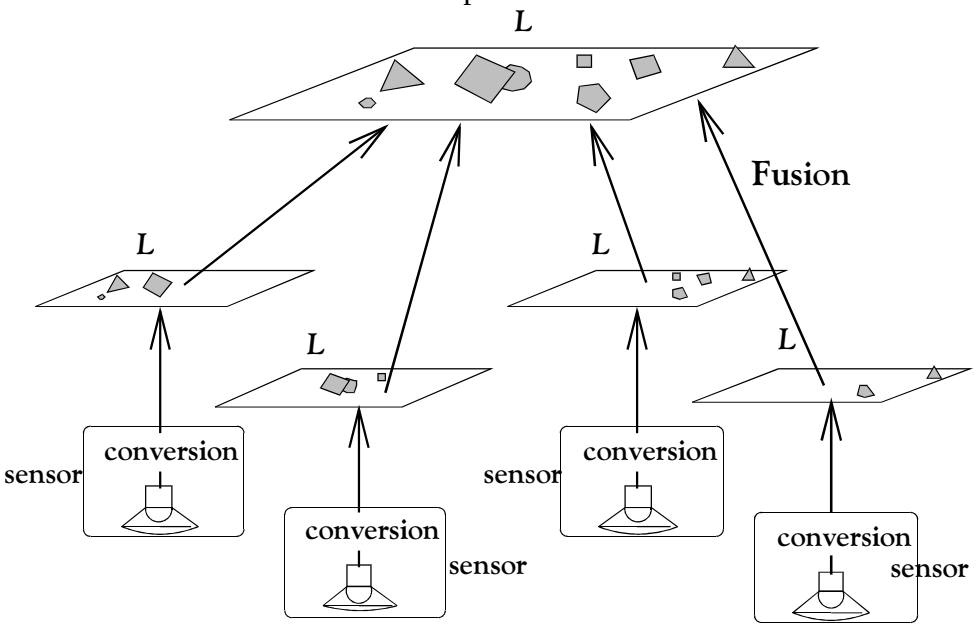


Figure 9.3: *Proposed architecture of a sensor data fusion system. Data from each sensor should first be converted to a common internal representation before the actual fusion of data is performed.*

The main characteristics of this architecture are:

- Data from each measuring device are first converted to an internal representation.
- This internal representation is *common* to all sensors.
- The actual fusion of data is performed in this internal representation.

In this architecture sensors are what we previously called "virtual sensors": the sensor performs both the actual measuring and the conversion to an internal representation. Although this architecture comes very natural in the context of virtual sensors, let it be noted that still very many systems perform the actual fusion before the conversion to the internal model is performed. E.g., distance measurements of acoustic sensors and infrared sensors are fused before an estimation of the wall parameters is given. This means that one sensor relies on another sensor's observations to make a conversion to an internal model. This is what we previously called "cooperative fusion". In short, cooperative fusion is performed within one single virtual sensor and competitive and complementary fusion is performed in the internal representation. This understanding is sketched in figure 9.3.

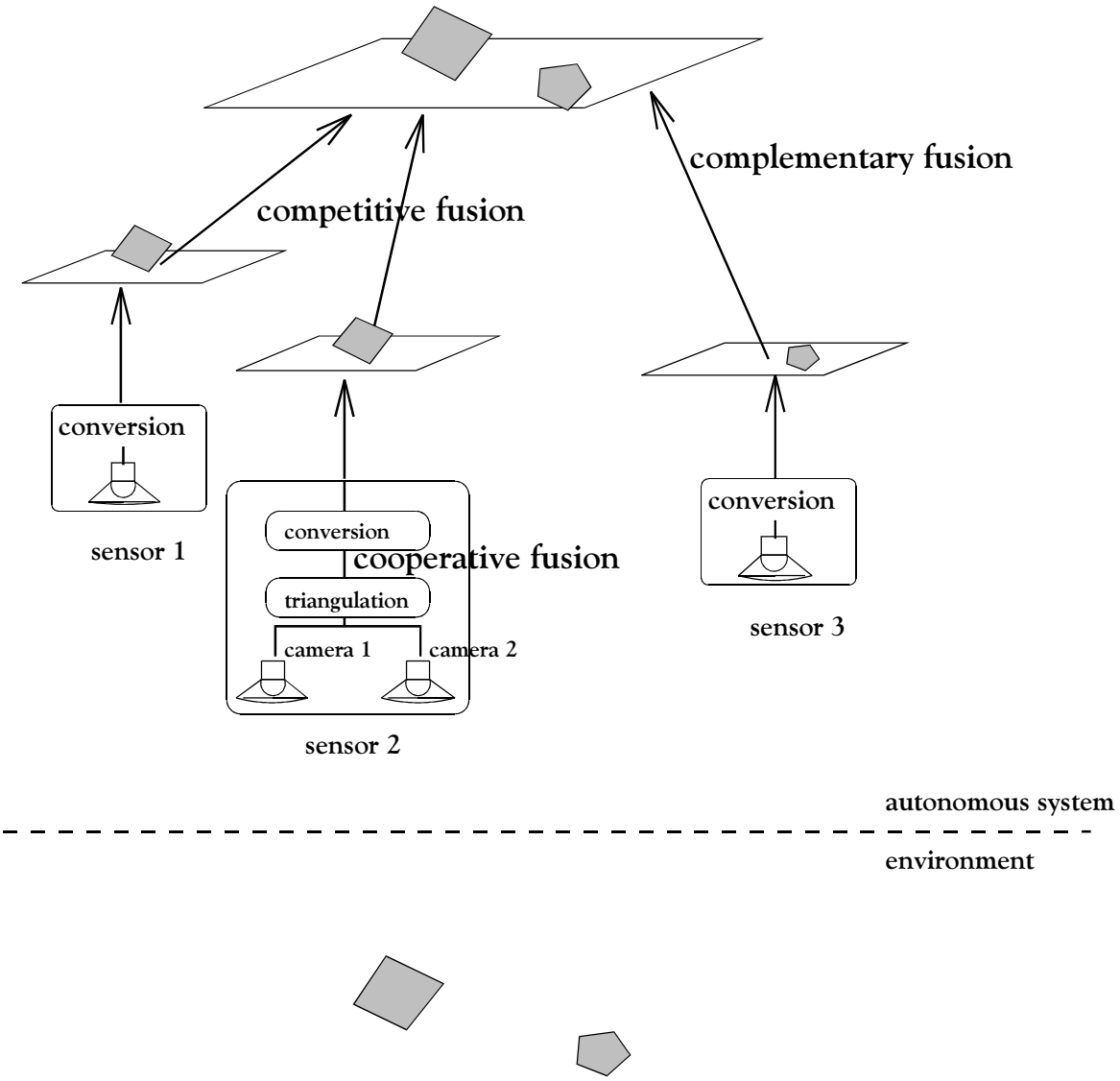


Figure 9.4: *Cooperative fusion of sensor measurements takes place within one single virtual sensor: in sensor 2 two transducers (in this case, two cameras) cooperate to estimate one single object (a cube). Competitive and complementary fusion are performed in the internal model: sensors 1 and 2 compete to give a more accurate estimation of the cube, and sensor 3 complements the internal model by estimating the other object in the scene.*

Further on we will motivate why cooperative fusion should be avoided as much as possible.

Before we move on to examples of competitive and complementary fusion, we will first argue for the need of *certainty values* and we will then show how *higher level fusion* methods use these certainty values to perform competitive and complementary fusion.

Certainty values

If higher-level fusion methods are to perform competitive and complementary fusion successfully, it is important that the sensors not only give an estimate of the parameter in the internal representation but also a *certainty value* for this estimate. This certainty value expresses how certain the sensor is about its estimation of the parameter. These certainty values can then be used in the subsequent fusion process. Consider e.g. an internal model , which represents the distances and angles

$$(d, \alpha)$$

to obstacles i with respect to the current configuration (i.e., position and orientation) of the robot. An acoustic sensor can estimate the distance d to an object i quite accurately while it is less certain about the angle α at which the object is found. Controversially, an infrared sensor could quite accurately determine the angle α', while it would be less certain about the distance d'. If only the estimates (d,α) and (d', α') would be available, we would not be able to use these characteristics of the sensor. All we could do is straightforward averaging of the estimated parameters. However, if also certainty values were available, we would be able to compute, e.g., a weighted average of the estimated parameters which gives more weight to the distance estimate d of the acoustic sensor and the estimate of the angle α' of the infrared sensor. Now that the sensors convert the measurements to the same internal representations and give certainty values for the estimates as well, the higher-level fusion method can be applied.

Higher level fusion

For ease of discussion, let's call the parameters in the internal model "propositions" z. Examples of such propositions are (see also chapter 7):

- $z_{i,\boldsymbol{x}}$ = "the position of the center of cube i is $\boldsymbol{x}$" (in this case, it has already been determined that there is a cube i in the robot's environment).

- $z_{d,\ \alpha}$ = "at angle α from the robot's current configuration there is an obstacle present at distance d"

- $z_{i,\ c}$ = "object i in the image belongs to class c"

Each sensor performs the conversion of its measurements of the robot's environment by giving estimates and certainty values for those estimates of the proposition z.

The most widely used approach is the **Bayesian** approach.

9.4 Bayesian approach

In this case the sensor assigns *probabilities* p(z) to the estimates of the parameters. These probabilities are the certainty values. Thus, given the measurement r_A of the sensor A, this

measurement is converted to the common representation z by giving, e.g., the conditional probability density function $p(z \mid r_A)$. This is called the **inverse sensor model**. The inverse sensor model follows from the sensor properties p (r | z) through the Bayes rule :

$$p(z|r) = \frac{p(r|z)}{p(r)} . p(z)$$

In figure 9.4 an example is given of such Bayesian output of a (virtual) sensor, in this case $p(z_{d,\alpha} \mid r_A)$.

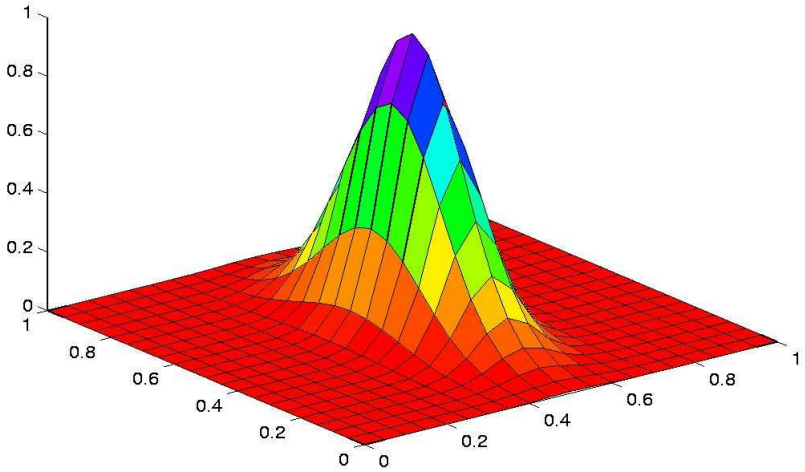


Figure 9.5: *Example of the output of a (virtual) sensor, which has already converted its measurement to the internal representation. In this case the measurement was a distance measurement. The model shows that the sensor is pretty certain about the distance d but that it a rather uncertain about the angle α at which this obstacle is measured.*

Now suppose another sensor B measures the same obstacle and converts its measurements as illustrated in figure 9.6.

This sensor has converted its measurement using another inverse sensor model

$$p'(z \mid r_B)$$

The difference in the inverse sensor models reflects the fact that the sensors have different error characteristics. In this example, the first sensor would typically be an acoustic sensor while the other sensor would typically be an infrared sensor.

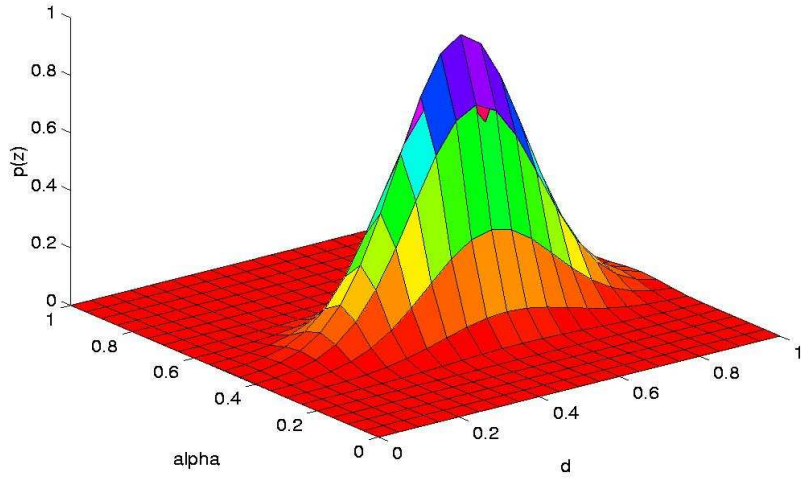


Figure 9.6: *Example of the output of another sensor, which has also converted its measurement to the internal representation. The model shows that this sensor is pretty certain about the angle α but that it a rather uncertain about the distance d at which this obstacle is measured.*

Fusion of the converted measurements of both sensors can now be performed by taking the *product* of the two probability density functions. This way we obtain a joint probability density function p_{joint} (z | r_A r_B)

$$p_{joint}(z|r_A, r_B) = \frac{p(z|r_A)}{p(z)} . p(z|r_B).$$

This can be derived from :

$$p(z|r_A, r_B) = \frac{p(r_A|z, r_B)}{p(r_A|r_B)} . p(z|r_B)$$

Assuming that $p(r_A|z,r_B)$ is **independent** of r_B gives $p(r_A|z,r_B) = p(r_A|z)$and $p(r_A|r_B)$ is independent of r_B results in: $p(r_A|r_B) = p(r_A)$. Applying the Bayes rule again for $p(r_A|z)$ results than in the equation for p_{joint} (z | r_A r_B).

Note that an appropriate normalization with p(z) is needed after taking the product to ensure that the resulting function is a probability density function. This function gives estimates and certainty values over the proposition z_d, α which is obtained by the fusion of the outputs of the two different sensors. The function is shown in figure 9.7.

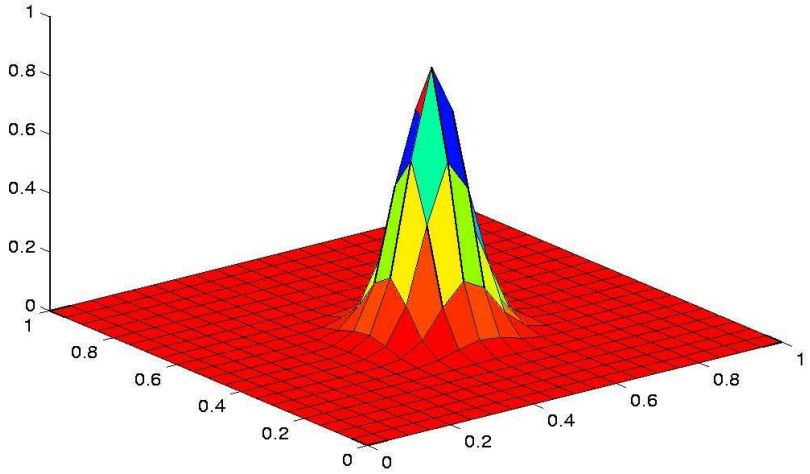


Figure 9.7 *Example of the fusion of the outputs of the two sensors. The figure shows that this type of competitive fusion has indeed led to reduction of uncertainty in the estimated proposition* z_d, α.

The illustrations above are an example of competitive fusion; both sensors measure the same obstacle. In the Bayesian approach, complementary fusion, in which case the sensors measure *different* obstacles, is simply performed by adding the different probability density functions. More details can be found in Durrant-Whyte [9.1].

In the Bayesian approach, the most convenient inverse sensor model p(z | r) is the Gaussian function, since it is easy to work with mathematically. Some fusion algorithms (such as the Kalman filter) implicitly assume a Gaussian shaped inverse sensor model, of which only the widths may vary. However, it has been commonly accepted that these Gaussian models are a rather inaccurate reflection of the true inverse sensor model. Some approaches define other probability density functions while staying in the Bayesian framework, while others break with the Bayesian approach all together.

Example : Occupancy grids (Elfes [9.6])
Let us assume that we will work in a quantized workspace described by a 2D grid of mxn cells. To navigate through this workspace we have to know whether the workspace is occupied by an obstacles or not. The grid cells contains the probability that a cell is occupied: $p(C_{ij}=OCC)$. In the bayesian approach it is assumed that:

$$p(C_{ij}=OCC) + p(C_{ij}=Empty) = 1$$

So when we know the probability that a grid element is occupied, 1 minus that probability is the probability that that element is empty. We will assume that the cell status are independent variables and when there is no information $p(C_{ij}=OCC) = 0.5$. We will start with an initial grid C_{ij} with for which all cells have

C_{ij} : $p(C_{ij}=OCC) = 0.5$.

After a sensor reading r the occupancy grid is updated using the Bayes theorem:

$$p(C_{ij} = OCC|r) = \frac{p(r|C_{ij} = OCC)}{p(r)}.p(C_{ij} = OCC)$$

It will be clear that to update the probabilities in the grid from a measurement r we have to know $p(r| C_{ij}=OCC)$ and $p(r)$.

$p(r|C_{ij}=OCC)$ has to be estimated by applying the sensor model to all possible world configurations, in which $C_{ij}=OCC$, to find the probability of finding measurement: r.
In principle this holds also for p(r). We have to answer the question what the possible worlds are for which we have to find this quantity. We have to make a choice what representative environments the robot will operate in.

These probabilities can be derived using *sensor models*. For an ultrasonic transducer a suitable model for the probability of measurement r is:

$$p(r|z,\theta) = \alpha(z)e^{-\frac{\theta^2}{2\sigma_\theta^2}}e^{-\frac{(r-z)^2}{2\sigma_z^2}}$$

for a hit at distance z and angle θ.

This sensor model has to be applied to all world configurations to come up with the conditional probabilities $p(r| C_{ij}=OCC)$ and $p(r)$. Often this is not done and for z and θ simple the values for cell C_{ij} are taken in the above formula and it is assumed that p(r) is constant.

Updating the grid for a new measurement r_{i+1} when we have had a sequence $\{r_i\}= (r_1,r_2,r_3$ …..$r_i)$, goes in the same way :

$$p(C_{ij} = OCC|r_{i+1}) = \frac{p(r_{i+1}|C_{ij} = OCC)}{p(r_{i+1}|\{r_i\})}.p(C_{ij} = OCC|\{r_i\})$$

In this formula the quantity $p(r_{i+1}|\{r_i\})$ is the hard part. This represents how much the different sensor readings are correlated. When these are measurement at successive times, there should be sufficient time between not almost to repeat the last measurement.

Criticisms to the above approach (Pagnac et al.[9.7], Thrun[9.8], Konolige[9.9])

- There is not taken care of confidence. Baysian theory assumes that

$P(C_{ij} = OCC) + P(C_{ij} = Empty) = 1$

When no information is present:

$P(C_{ij} = OCC) = P(C_{ij} = Empty) = 0.5$.

However the situation is completely different when there is no information present or when in half of the cases the outcome is empty and in half of the cases the outcome is occupied! This can not be represented easily in the Baysian approach

- All **conditional probabilities** must be specified. With a poor sensor model, these conditional probabilities have a large influence in the final results.

- Presence of **redundant readings**. This conflicts with assumptions about independence. In particular the term $\frac{p(r_{i+1}|C_{ij} = OCC)}{p(r_{i+1}|\{r_i\})}$ is 1 for a redundant reading and should not contribute in $p(C_{ij} = OCC|r_{i+1})$

- **Specular reflections** do occur frequently and are not taken care of in a gaussian sensor model. To solve this problem Konolige proposes the use of multiple sensor models.

Alternatives to the Bayesian approach are the possibilistic approach of Dempster-Shafer (Durrant-Whyte) or learning the models and confidence (Thrun[9.8])

9.5 Non-Bayesian approaches

One of the first breaks from Gaussian-shaped inverse sensor models was the use of *uncertainty sets* (see, e.g., 9.2). In this approach a sensor converts its measurements by giving a *set* of possible values $\pi(z \mid r)$ for the parameter, in which π is 0 or 1. Performing fusion on the sensor data then diminishes the widths of these sets. E.g., simply taking the intersection of the sets of the different sensors can perform competitive fusion. This example is sketched in figure 9.8. For the two measuremnets r_A of sensor 1 and r_B of sensor 2, this can be for instance be done by taking the minimum of the two possibility values:

$$\pi(z \mid r_A r_B) = \min_z \{\pi(z \mid r_A), \pi(z \mid r_B)\}$$

Using the minimum to take the intersection, means that we can apply this formula also in

the case that the sets are not crisp but fuzzy.

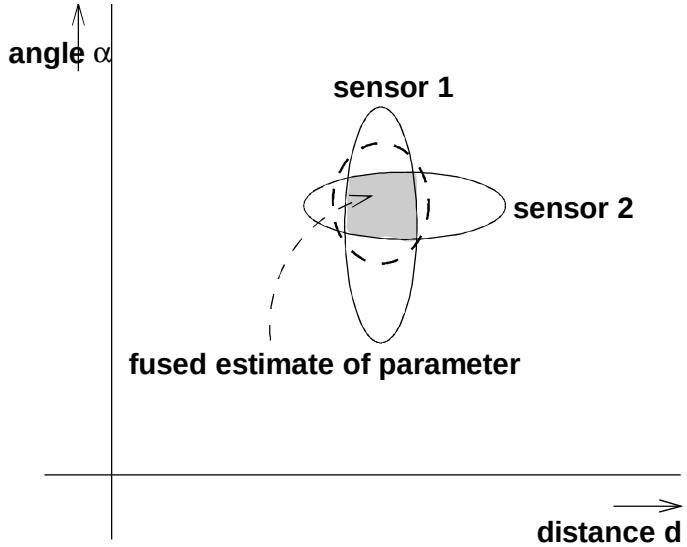


Figure 9.8: *Example of the use of uncertainty sets. Sensor 1 gives a set of possible values for the (d, α) parameter which is rather certain about the distance and more uncertain about the angle (typically an acoustic sensor). Sensor 2 estimates the parameters with a set that is certain about the angle and uncertain about the distance (typically, an infrared sensors). Fusion is performed by taking the intersection of the two sets.*

Another problem with the Bayesian approach is that the distribution of probabilities may not be known. In this case we should not use a *probabilistic* approach but a *possibilistic* approach. The difference is best explained with an example.

Consider an autonomous system, which has to classify various objects on a conveyor belt. At this level of abstraction, the sensors give certainty values for the propositions

z_i, c = "object i in the image belongs to class c"

For this example we take

$c \in$ {cube, cylinder, sphere, prism}

Now a sensor in the system may sense that an object is either a cube or a cylinder, but it cannot discern between the two. Or, a sensor may sense with a certain 'probability' that an object is a cylinder, while it does not want to say it is necessarily a cube, a sphere or a prism otherwise. The formalism known as the *Dempster-Shafer theory*(see [9.5]) is a break from the rather rigid probabilistic framework which accommodates for these flaws. In Dempster Shafer theory certainty values, can also be assigned to a *disjunction* { cube, cylinder } instead of assigning values to the singletons { cube} and { cylinder } only. These certainty values must still sum to 1 but can be assigned to singletons as well as disjunctions of propositions. The disjunctions may have non-empty intersections. Also, certainty may be left uncommitted. A sensor may express

c ∈ { cylinder } with certainty 0.25

and for the other 0.75, I don't know

c ∈ {cube, cylinder, sphere, prism} with certainty 0.75

More formally formulated: we have a finite set θ of mutually exclusive events, which may occur. In this case:

θ= {cube, cylinder, sphere, prism}

The certainty values m( ) can not only be assigned to the events themselves but also to their disjunctions. So we can assign certainty values to all subsets of θ, which form the power set Λ

Λ = 2^{θ} = {∅, cube, cylinder, sphere, prism, {cube, cylinder}, {cube, sphere},….
……..,{cube, cylinder, sphere, prism}}

The label {∅} is the unknown state in which none of the events in the set occurs. In many applications this unknown state can never occur, so its certainty value is 0 or $m(\varnothing) = 0$.

The sum of the certainty values *m* is 1 so:

$$\sum_{A \subset \Lambda} m(A) = m(\varnothing) + m(cube) + \ldots\ldots\ldots = 1$$

The total evidence that is attributed to some event A is the sum of all the certainty values which are assigned to A and all the subsets of which A forms a part. This is called the belief we have in A : Bel (A).

$$Bel(A) = \sum_{\forall B; A \subseteq B} m(B)$$

So we sum up the certainty values of all subsets of Λ in, which A is present, to obtain the total belief in A. As a result the belief we have in the set θ is 1:

Bel (θ) = 1.

Besides the belief there is the plausibility: the amount of evidence that does not support the negation of an event. It is defined as:

$$Pls(A) = 1 - Bel(\neg A) = 1 - \sum_{\forall B; A \subset B} m(B)$$

Now Bel (A) + Bel (¬A) ≤ 1 (in which the sum smaller than 1 can occur when A is a disjunction). As a result the Belief in A is always equal or smaller than the plausibility of A.

$$Bel(\mathrm{A}) \leq Pls(\mathrm{A})$$

The actual fusion of the beliefs, i.e. the fusion of the converted sensor measurements, is performed with Dempster's "rule of combination". When we have for instance two sensor measurements which resulted in the certainty values m_1 and m_2 for the event A, than the combination is given by:

$$m_1 \oplus m_2(A) = \frac{\sum_{\forall B,C \in \Lambda : B \cap C = A} m_1(B) m_2(C)}{1 - \sum_{\forall B,C \in \Lambda : B \cap C = \varnothing} m_1(B) m_2(C)}$$

So we basically calculate the product of the certainty values of all subsets in which the intersection is exactly A and we divide by 1 minus the product of the certainty values of the subsets that do not intersect. The above formula forms the basis for sensor fusion in a "possibilistic approach".

Example (Pagac et al. 6):
Lets us look now at the same example as before: sensor fusion from data obtained with an ultrasonic sensor. The model that we realize within this framework can both incorporate the evidence we have for the empty space as for the occupied space. In the case we may assume there is evidence in the arc of the ultrasonic sensor that there is an occupied cell. The length of the arc is $2R\beta$, and we suppose that it covers n cells. This results in a certainty value of 1/n for each cell in the arc of being occupied. We have obtained no information whether a cell is empty in the arc.

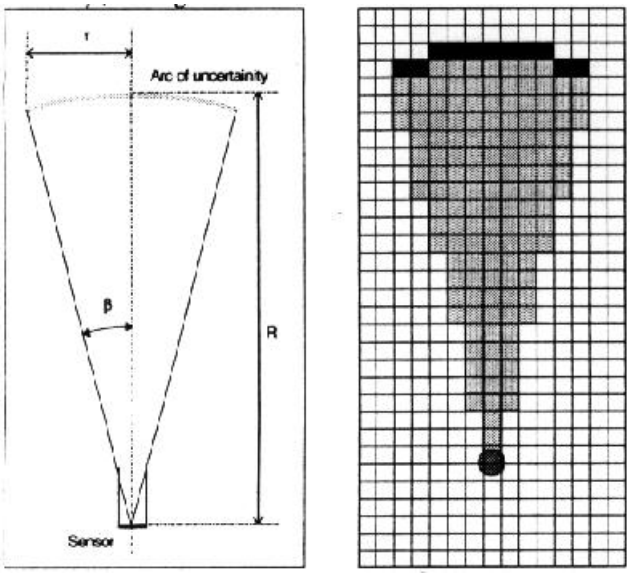


Figure 9.9: *Example of the modeling of an ultra sonic sensor. It consists of an arc of uncertainty for an obstacle being present and a sector where there is an uncertainty of being empty (Pagan et al. [9. 6])*

So the certainty values for a cell in the arc are given by:

$$\left.\begin{aligned} m_{i.j}(F) &= \frac{1}{n} \\ m_{i,j}(E) &= 0 \end{aligned}\right\} \forall cells(i,j) \in arc$$

In the same way for the cells in the sector there is evidence that a cell is empty. This can be modeled by a certainty value of ρ. We have is this case no information whether a cell is occupied. So we obtain:

$$\left.\begin{aligned} m_{i.j}(F) &= 0 \\ m_{i,j}(E) &= \rho \end{aligned}\right\} \forall cells(i,j) \in \text{sec}\, tor$$

And outside the sector and arc:

$$\left.\begin{aligned} m_{i.j}(F) &= 0 \\ m_{i,j}(E) &= 0 \end{aligned}\right\} \forall cells(i,j) \notin arc, \text{sec}\, tor$$

Now we want to fuse the certainty values $m_S(E)$ and $m_S(F)$ resulting from our sensor measurements S with the certainty values $m_M(E)$ and $m_M(F)$ from the current map of the environment M. We will drop the indices i,j for the individual cells for clarity. Applying the rule of combination for each cell we obtain:

$\theta = \{ E , F \}$

$\Lambda = 2^{\theta} = \{ \varnothing, E , F , \{E,F\}\}$
in which $m(\varnothing)=0$

and obtain:

$$m_M \oplus m_S(E) = \frac{m_M(E)m_S(E) + m_M(E)m_S(\{E,F\}) + m_M(\{E,F\})m_S(E)}{1 - m_M(E)m_S(F) + m_M(F)m_S(E)}$$

$$m_M \oplus m_S(F) = \frac{m_M(F)m_S(F) + m_M(F)m_S(\{E,F\}) + m_M(\{E,F\})m_S(F)}{1 - m_M(E)m_S(F) + m_M(F)m_S(E)}$$

The rule of combination is somewhat complex because of the fact that beliefs may be expressed for disjunctions of propositions. Indeed, one of the main disadvantages of Dempster-Shafer theory is that the application of the theory leads to exponential complexity.

Summarizing, the most pronounced features of the theory are:

- the possibility to assign beliefs to disjunctions of propositions without assigning beliefs to the individual propositions (the object is either a cube or a cylinder)

- the possibility to leave belief "uncommitted", which is comparable to assigning a probability to a special "I don't know" proposition (the object is a cylinder with belief 0.25, else I don't know). One of the founders of this theory, Glenn Shafer, has claimed that probability theory is a subset of Dempster-Shafer theory.

Fuzzy logic

Another possibilistic approach, which has attracted much attention in the past few years, is the *fuzzy logic* approach. The method does not assign probabilities to propositions but rather assigns membership values to pre-defined sets (e.g. the set of cylinders or the set of cubes). The essential feature of fuzzy logic is that the boundaries of the sets are not "crisp" but "fuzzy". Crisp sets are sets to which an object either belongs or not. In contrast, fuzzy sets are sets to which an object can belong in a certain degree; the fuzzy boundaries of these sets often overlap. E.g., from a certain point of view an autonomous system may be unable to discern whether a sensed object is a cylinder or a cube. It may then assign membership to this object for the class of "cylinders" as well as the set of "cubes". In this case, the fuzzy boundaries for the set of "cubes" and for the set of "cylinders" overlap. Basically, the fuzzy-logic approach offers the same advantages as the Dempster-Shafer theory (they are both *possibilistic* approaches) whereas the fuzzy logic approach does not suffer from the exponential complexity.

Another attractive side to fuzzy logic is that the membership values can be assigned with intuitively comprehensible values, such as {"possibly", "probably", "certainly", "probably not", "certainly not"}. This makes the fuzzy logic approach especially applicable to capturing knowledge from human experts, which is often hard to represent in precise numerical values. Conversely, it is also much easier for a human operator to comprehend the rules in a fuzzy logic system.

Again, the actual competitive and complementary fusion is performed with special rules of combination of fuzzy values. These rules of combination do not differ much, in effect, from the Bayesian rules of combination. Like Bayesian theory, in fuzzy logic theory also and "AND" and an "OR" rule are defined. Roughly speaking, the "AND"-rule is for competitive fusion and the "OR"-rule for complementary fusion.

There are only general guidelines as to when which higher-level fusion method is preferable. Usually the level of abstraction at which the fusion method is applied determines which method is best applicable. At lower levels, where the fusion method is often applied to numerical values, Bayesian methods and uncertainty sets are more appropriate. Moving up in the hierarchy of the autonomous system, the emphasis shifts from numerical to symbolic processing. As the examples in this section served to show, there are other methods that are better suited for symbolic processing, such as (fuzzy) logic approaches and Dempster-Shafer theory.

9.6 Sensor fusion by learning

A serious problem is how to obtain a sensor model, which is correct for a given situation. The models discussed so far for an ultra sonic sensor both probabilistic and possibilistic are still based on a pretty ideal sensor model. The reality can be much more complex. Realistic responses can be learned using a neural network (Thrun [9.8]), in which a robot moves in a known environment. In that environment it is known which cells are occupied and which are empty, and what the measurements of the ultra sonic sensors are for each situation. By training a neural network with as input the sensor measurements and as output the occupancy values of the cells locally surrounding the robot, the mapping between the sensor measurement and the occupancy of the cells of the local environment of the robot can be learned.

For some cells in the local environment this mapping can be much more reliable than for other cells for a given set of measurements. The confidence we have in these occupancy values can be found from the expected error between the predicted and real occupancy values. Thrun [9.8] suggests to train two networks: one that estimates the occupancy values and one that estimates the confidence we have in these values.
After the training stage we have learned the mapping from the measured sensor values to occupancy values in the local environment and the confidence we have in these values for each cell. Next the occupancy values are fused with the existing map, in which the occupancy values are weighted as function of the confidence.

We like to learn the occupancy values C(X) for a local environment X of the robot, when we have obtained the measurements R from the ultra sonic sensors. When we have K ultrasonic sensors then

$$R=(r_1,r_2,\ldots\ldots r_K)$$

And X could for instance be a local grid of NxN cells, in which the robot is in the middle position.

$$X = \begin{bmatrix} x_{11} & \ldots. & x_{1N} \\ . & & . \\ . & & . \\ . & & . \\ x_{N1} & \ldots. & x_{NN} \end{bmatrix}$$

When we drive through an known environment and measure at successive robot positions i we obtain a sequence of known occupancy values $OCC_i(X)$ together with the measurements R_i

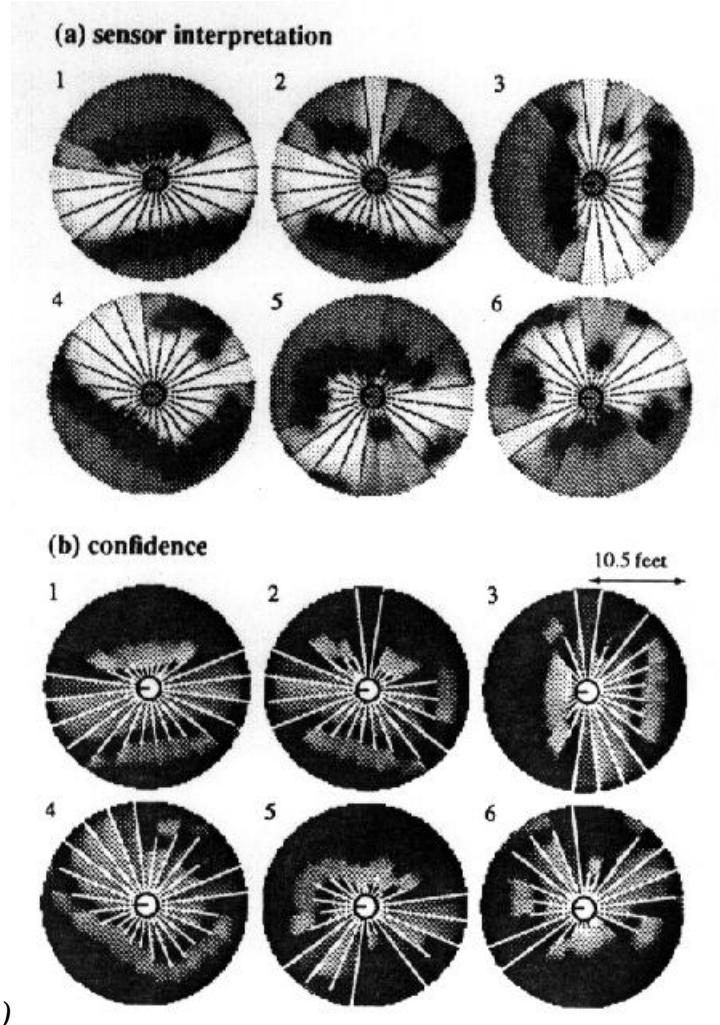
Now a neural network can be trained to find the mapping between OCC_i and R_i: the *sensor interpretation* network. Let us call the output of this network C, then we want to minimize the difference E between the output of the network C and the occupancy values

OCC for the trained positions i.

$$C_i(X) = NN_1 (R_i, X)$$

$$E_i (X) = | C_i(X) – OCC_i(X) |$$

Although the training of the network will minimize this error it will never become completely zero. This error will be different for the different positions in the environment and depend upon the sensor measurements. Some cells in the environment will be not observable or occluded, so information about them cannot be obtained, and it is simply impossible to get a reliable estimation about their occupancy. The larger $E_i(x_{ij})$ for certain cells (i,j) of X, the lower our confidence is in the interpretation of the measurements at those cells.



)

Figure 9.10: *Example of the sensor interpretation of the R network. Lines indicate sonar measurements (distances), and the region darkness represents the occupancy. (0 gives the corresponding confidences C in the interpretation shown.*

A second network NN_2 is trained to give an estimate ε of this error.

$$\varepsilon_i (X) = NN_2 (R_i, X)$$

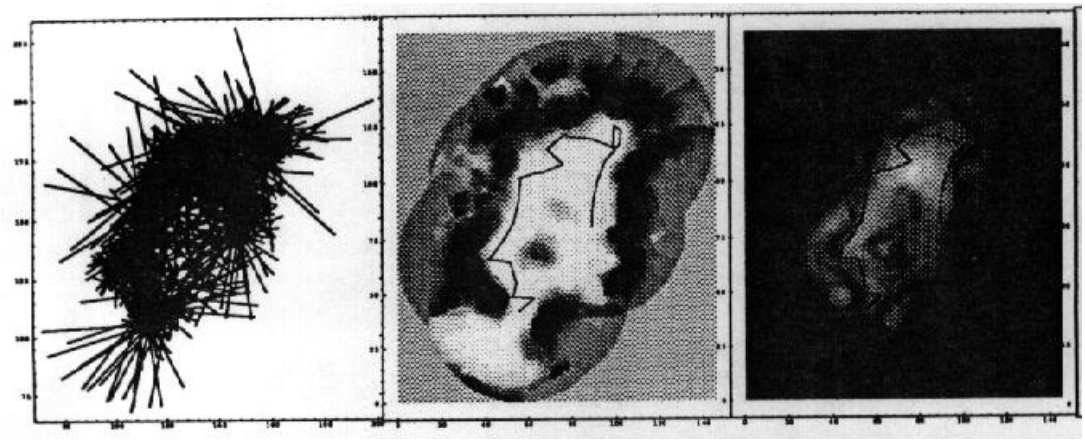
in which the difference between ε_i and E_I is minimized during training. This network is called the *confidence network*. In figure 9.10 the sensor interpretation and confidence values for the local environment X are given for some configurations.

After the training stage the networks are used to create a map of the environment. The sensor interpretation values are fused according to their confidence. Thrun uses as weights : $-\ln \varepsilon_I$, and the Map M is obtained by :

$$M(x) = \frac{\sum_i -\ln \varepsilon_i(x).C_i(x)}{\sum_i -\ln \varepsilon_i(x)}$$

in which the denominator takes care for the normalization, so that the sum of the weights is 1.

In figure 9.11 an example of the integrated measurements is given according to Thrun.



(a) (b) (c)

Figure 9.11: *Integration of several measurements. (a) Raw sensor input. (b) Resulting map, (c) cumulative confidence.*

9.7 Choice of an internal model

At several times throughout this text requirements for the internal representations at the various levels in the autonomous system have been made. In this section we will review these requirements once more and we will give some examples of internal representations that meet these requirements.

Previously, the following requirements for the internal representation have been made:

- it should be *common to all sensors*. Higher level fusion is performed on converted sensor measurements only. Obviously, measurements from different sensors must be converted to a *common* internal representation so that fusion can be performed.

- it must be useful for the task at hand and even more so *only* those aspects from the robot's environment that are *necessary for the task at hand* should be represented. It has been common practice for a long time to use all possibly available sensors to form a comprehensive internal model of the internal environment. As many data as possible are represented in such a representation. The reasoning component of the autonomous system is then fully informed about the environment. In more recent approaches one has become to realize that only those aspects should be present in the internal representation that are necessary to perform the task at hand. This point was already mentioned in chapter 2.

- the internal representation should not only represent estimates for the parameters in this representation but also *certainty values* that express the confidence of the sensor in its own measurements. In the previous sections it was clearly shown how these certainty values are used effectively for competitive fusion.

- it should be possible to apply a *higher-level fusion* method to the estimates of the parameters and the associated certainty values. This suggests that the internal model should contain "propositions", such as given in the previous section.

- the internal representation should be chosen such that the raw sensor measurements can be easily *converted* by the virtual sensor. The conversion of sensor measurements is one of the main problems in sensor data fusion. On the one hand, one should move gradually towards more abstract representations of the environment, for these are better suited for the reasoning component, while on the other hand the model should also be closely related to the actual values measured by the transducer, for else conversion of the data would be too complicated.

A standard example of an internal representation that meets these requirements is the *geometrical representation*, as comprehensively described in 9.1. In this representation, the robot's external environment is modeled by a set of geometrical objects of which the parameters are estimated by the autonomous system. E.g., if at one time a cubic obstacle is detected by the autonomous system, the sensors are used to estimate its 6D position in the environment (position and orientation). The propositions z in this internal representation are, e.g., $z_{i,\mathbf{x}}$ = "the position of the center of cube i is **x**"

Certainty values for estimates of these parameters can be represented using the Bayesian approach or the uncertainty sets approach. Both representations are very well suited for

higher-level fusion methods. The conversion of raw sensor data to this internal geometric representation needs a model of the cube to estimate the parameters and a model of the sensors to give the certainty values.

Closely related is the *partially geometric representation*. Like the previous examples, geometric parameters of objects in a scene are to be estimated by the sensors. But in the partially geometric representations only those geometric objects are represented that will be directly handled by the autonomous system. Consider as an example a conveyor belt, which carries numerous audio tuners into which a cylindrical volume switch is to be inserted by an autonomous system. For this task, the system is only interested in the position of the cylindrical hole in the tuner, regardless of the cubic shape of the tuner itself; and thus only cylinders are represented in the internal representation.

This representation does not basically differ from the previous one, it merely represents only a subset of the geometric objects in the environment. It carries to the extreme the requirement that only those aspects of the system's environment are represented which are necessary to perform the task at hand. This representation is mentioned separately since it is used very often in sensor data fusion systems that are used in practice.

An example of a more abstract internal representation is the set of class labels. These representations contain the propositions: $z_{i,\,c}$ = "object i in the image belongs to class c"

It may very well be used by, e.g., an under-water system, which is to identify large submarine vessels. In this case the classes c can be "friendly vessel", "enemy vessel" or "whale". It is clear that such a language will only be used at higher levels; an acoustic sensor can hardly distinguish between these classes from its reflected signal only. But at a higher level fusion between different sensory systems and even between different under-water detection stations may be required. In this case such an internal representation as mentioned here can be used to fuse the different converted sensor measurements.

Let it also be noted that also other fusion methods are pre-eminently suited for this internal representation. Fuzzy logic approaches or Dempster-Shafer theory may very well be used to represent, e.g., a system's inability to discern between a whale an enemy vessel.

As a final example, let's consider what internal model would be appropriate for an autonomous system for which the task at hand is collision-free navigation. According to the second requirement stated at the beginning of this section only aspects relevant to collision free navigation should be represented. Thus, we are predominantly interested in the representation of free space and occupied space in the robot's external environment. Also, not only should the locations of free and occupied space be represented, but also the associated certainty values.

One choice of representation which meets these requirements is the *occupancy grid* or "certainty grid". This representation divides the robot's external environment in a number of cells and for each cell the probability is given that it is occupied by an obstacle

(occupied space) or not (free space). This approach is described extensively in [9.6]. A simple example of an occupancy grid representation of the robot's external environment is given in figure 9.11.

Actually, this figure was obtained by fusing multiple sensor measurements of a room. Each sensor measurement was converted to the internal representation by calculating occupancy grid representations of these measurements and in this internal model the sensor measurements were subsequently fused using a Bayesian fusion method. Fusion was performed by a what is called the "independent opinion poll method", which comes down to summing the probabilities of the individual cells and performing the appropriate normalization. Details can be found in 9.5.

9.8. Motivation of the architecture

In the previous sections an architecture for a sensor data fusion system was introduced with as most pronounced feature the internal representation which is *common* to all sensors (see figure 9.3). The fact that we want a *generic* SDF system, which is applicable at all possible levels in the hierarchical autonomous system, is not the only motivation for the choice of this architecture. In this section we will shortly review some more advantages of the presented approach.

As the sensor data fusion system requires that each sensor performs its own conversion to the internal model, it is easily *extended* with additional sensors. The only restriction is that not only an additional transducer is required, but also the conversion of the raw data measurements to the internal representation. In contrast, many sensor data fusion systems exist which use the particular strengths of the sensors involved. In these cases, the measurements of the transducer are not explicitly converted before fusion is performed. Instead, the *fusion method* takes the particular strengths of the sensor into account. Obviously, with such approaches an entirely new fusion algorithm must be designed if the system is extended with new sensors. Therefore, such approaches should be limited to cooperative fusion (within a single virtual sensor) only.

In the sensor data fusion community it is still a much debated question how certainty values for sensor measurements should be assigned. As stated above, often fusion methods are used which make *implicit assumptions* about the assignment of certainty values. E.g., the Kalman filter assumes certainty values following a Gaussian distribution centered at the sensor measurement. It is commonly accepted, however, that in many cases the Gaussian certainty function is too inaccurate. In the section on "Higher level fusion" it was shown how *inverse sensor models* can be used in the conversion of sensor measurements to *explicitly define* the assignment of certainty values. In that section we used for the inverse sensor models $p(z \mid r)$ Gaussian functions and uncertainty sets. Obviously, these inverse sensor models can be substituted by other functions if one so desires. The fusion method itself need not be changed. Thus, the architecture proposed here exhibits great flexibility in the use of inverse sensor models. Suppose a sensor in the architecture as sketched in figure 9.3 would malfunction. In this case this particular

sensor would give incorrect estimates of the parameters in the internal representation. But, provided a sufficient number of other sensors are present (redundant information!) with which competitive fusion is performed, the system would still be able to give an acceptable, albeit less accurate, estimation of the parameters. The entire system would still be valid, although with the loss of more and more sensors it becomes gradually worse. This kind of behavior of a system is called *graceful degradation*. It is clear that because of the modular, structural approach to sensor data fusion followed in our architecture our system exhibits this behavior. In contrast, many applications only perform what we defined earlier as "cooperative fusion". In this fusion method, each sensor relies on the other(s) to make its observations. Obviously, if just one sensor in this scheme malfunctions, the whole cooperating pool of sensors is invalidated. This is not consistent with graceful degradation.

As a simple example, in various approaches to mobile robot navigation the reasoning component determines the actions for the robot based on the *sensor vector*. The sensor vector is a concatenation of all measurements made by the transducers. In our terminology, a single observation consists of the whole sensor vector (this is the 'input' to the reasoning component, the "internal representation") and cooperative fusion is performed by concatenation. This is called cooperative fusion since each sensor needs the measurements of all the other sensors to make a single observation, i.e. to produce a single sensor vector. In this case, if a single sensor malfunctions, the entire system is invalidated since observations (sensor vectors) can no longer be made.

9.9. Discussion

In this section an organization for sensor data fusion was introduced that is very interesting from a theoretical point of view. The architecture is easily extendible to new sensors since fusion is performed in an internal representation that is common to all sensors. Also, higher-level fusion methods can be efficiently applied since the inverse sensor model can be explicitly represented in this internal representation. And furthermore, the modular architecture of the system leads to 'graceful degradation'.

However, a key issue remains how to obtain the virtual sensors that can successfully convert their raw measurements to this internal representation. This issue was somewhat hidden in our discussion as it was assumed that at each level the appropriate sensors where available upon request. It should, therefore, be stressed that with the practical application of this theory much effort remains to be put in the design and implementation of the suitable (virtual) sensors. If this design can be performed then the theory in this section provides a generic architecture for sensor data fusion, which can be used at any level in the hierarchical autonomous system.

Bibliography

1. Hugh F. Durrant Whyte Integration, Coordination and Control Multi-Sensor Robot Systems. KluwerAcademic Publishers, 1988
2. A.Sabater. Set membership approach to the propagation of Uncertain Geometric

information, Proc. of 1991 IEEE Int. Conf. On Rob. And Aut. Sacramento, California, April 1991, pp 2718-2723
3. Ben J.A. Krose , Kai M. Compagner, F.C.A. Groen, Accurate Estimation of Environment Parameters from Ultrasonic Data, Robotics and Autonomous Systems,11 (1993), pp 221-230
4. J. Vandorpe, Navigation Techniques for the Mobile Robot Lias, Thesis Catholic University Leuven, January 1997.
5. G.Shafer, A Mathematical Theory of Evidence, Princeton University Press, Princeton 1976
6. Alberto Elfes, Sonar-based real world Mapping and Navigation, IEEE Journal of Robotics and Automation, Vol RA-3,no 3, June 1987
7. D. Pagnac, E.M. Nebot, H.F. Durrant White, An evidential approach to probabilistic map-building. Proc.IEEE conf. On Robotics and Automation, Minneapolis, April 1996, pp 745-750
8. S.B. Thrun, Exploration and Model Building in Mobile Robot Domains, IEEE 1993
9. K.Konolidge A. Refined Method for Occupancy Grid Interpretation. Lectire Notes in Artificial Intelligence 1093, Spinger Verlag, pp. 338-352